

MACHINE LEARNING

机器学习

周志华 著

清华大学出版社
北京

内 容 简 介

机器学习是计算机科学的重要分支领域。本书作为该领域的入门教材，在内容上尽可能涵盖机器学习基础知识的各方面。全书共 16 章，大致分为 3 个部分：第 1 部分（第 1~3 章）介绍机器学习的基础知识；第 2 部分（第 4~10 章）讨论一些经典而常用的机器学习方法（决策树、神经网络、支持向量机、贝叶斯分类器、集成学习、聚类、降维与度量学习）；第 3 部分（第 11~16 章）为进阶知识，内容涉及特征选择与稀疏学习、计算学习理论、半监督学习、概率图模型、规则学习以及强化学习等。每章都附有习题并介绍了相关阅读材料，以便有兴趣的读者进一步钻研探索。

本书可作为高等院校计算机、自动化及相关专业的本科生或研究生教材，也可供对机器学习感兴趣的研究人员和工程技术人员阅读参考。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

图书在版编目(CIP)数据

机器学习/周志华著。--北京：清华大学出版社，2016

ISBN 978-7-302-42328-7

I. ①机… II. ①周… III. ①机器学习 IV. ①TP181

中国版本图书馆 CIP 数据核字(2015)第 287090 号

责任编辑：薛 慧

封面设计：何凤霞

责任校对：刘玉霞

责任印制：宋 林

出版发行：清华大学出版社

网 址：<http://www.tup.com.cn>, <http://www.wqbook.com>

地 址：北京清华大学学研大厦 A 座 邮 编：100084

社 总 机：010-62770175 邮 购：010-62786544

投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn

质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn

印 装 者：北京亿浓世纪彩色印刷有限公司

经 销：全国新华书店

开 本：210mm×235mm 印 张：27.75 字 数：626 千字

版 次：2016 年 1 月第 1 版 印 次：2016 年 1 月第 1 次印刷

印 数：1~5000

定 价：88.00 元

产品编号：064027-01

前　　言

这是一本面向中文读者的机器学习教科书,为了使尽可能多的读者通过本书对机器学习有所了解,作者试图尽可能少地使用数学知识。然而,少量的概率、统计、代数、优化、逻辑知识似乎不可避免。因此,本书更适合大学三年级以上理工科本科生和研究生,以及具有类似背景的对机器学习感兴趣的人士。为方便读者,本书附录给出了一些相关数学基础知识简介。

全书共 16 章,大体上可分为 3 个部分:第 1 部分包括第 1~3 章,介绍机器学习基础知识;第 2 部分包括第 4~10 章,介绍一些经典而常用的机器学习方法;第 3 部分包括第 11~16 章,介绍一些进阶知识。前 3 章之外的后续各章均相对独立,读者可根据自己的兴趣和时间情况选择使用。根据课时情况,一个学期的本科生课程可考虑讲授前 9 章或前 10 章;研究生课程则不妨使用全书。

书中除第 1 章外,每章都给出了十道习题。有的习题是帮助读者巩固本章学习,有的是为了引导读者扩展相关知识。一学期的一般课程可使用这些习题,再辅以两到三个针对具体数据集的大作业。带星号的习题则有相当难度,有些并无现成答案,谨供富有进取心的读者启发思考。

本书在内容上尽可能涵盖机器学习基础知识的各方面,但作为机器学习入门读物且因授课时间的考虑,很多重要、前沿的材料未能覆盖,即便覆盖到的部分也仅是管中窥豹,更多的内容留待读者在进阶课程中学习。为便于有兴趣的读者进一步钻研探索,本书每章均介绍了一些阅读材料,谨供读者参考。

笔者以为,对学科相关的重要人物和事件有一定了解,将会增进读者对该学科的认识。本书在每章最后都写了一个与该章内容相关的小故事,希望有助于读者增广见闻,并且在紧张的学习过程中稍微放松调剂一下。

书中不可避免地涉及大量外国人名,若全部译为中文,则读者在日后进一步阅读文献时或许会对不少人名产生陌生感,不利于进一步学习。因此,本书仅对一般读者耳熟能详的名字如“图灵”等加以直接使用,对故事中的一些主要人物给出了译名,其他则保持外文名。

机器学习发展极迅速,目前已成为一个广袤的学科,罕有人士能对其众多分支领域均有精深理解。笔者自认才疏学浅,仅略知皮毛,更兼时间和精力所限,书中错谬之处在所难免,若蒙读者诸君不吝告知,将不胜感激。

周志华

2015 年 6 月

序　　言

在人工智能界有一种说法,认为机器学习是人工智能领域中最能够体现智能的一个分支。从历史来看,机器学习似乎也是人工智能中发展最快的分支之一。在二十世纪八十年代的时候,符号学习可能还是机器学习的主流,而自从二十世纪九十年代以来,就一直是统计机器学习的天下了。不知道是否可以这样认为:从主流为符号机器学习发展到主流为统计机器学习,反映了机器学习从纯粹的理论研究和模型研究发展到以解决现实生活中实际问题为目的的应用研究,这是科学研究的一种进步。有关机器学习的专著国内出版的不是很多。前两年有李航教授的《统计学习方法》出版,以简要的方式介绍了一批重要和常用的机器学习方法。此次周志华教授的鸿篇巨著《机器学习》则全面而详细地介绍了机器学习的各个分支,既可作为教材,又可作为自学用书和科研参考书。

翻阅书稿的过程引起了一些自己的思考,平时由于和机器学习界的朋友接触多了,经常获得一些道听途说的信息以及专家们对机器学习现状及其发展前途的评论。在此过程中,难免会产生一些自己的疑问。我借此机会把它写下来放在这里,算是一种“外行求教机器学习”。

问题一:在人工智能发展早期,机器学习的技术内涵几乎全部是符号学习。可是从二十世纪九十年代开始,统计机器学习犹如一匹黑马横空出世,迅速压倒并取代了符号学习的地位。人们可能会问:在满目的统计学习期刊和会议文章面前,符号学习是否被彻底忽略了?它还能成为机器学习的研究对象吗?它是否将继续在统计学习的阴影里生活并苟延残喘?对这个问题有三种可能的答案:一是告诉符号学习:“你就是该退出历史舞台,认命吧!”二是告诉统计学习:“你的一言堂应该关门了!”单纯的统计学习已经走到了尽头,再想往前走就要把统计学习和符号学习结合起来。三是事物发展总会有“三十年河东,三十年河西”的现象,符号学习还有“翻身”的日子。第一种观点我没有听人明说过,但是我想恐怕有可能已经被许多人默认了。第二种观点我曾听王珏教授多次说过。他并不认为统计学习会衰退,而只是认为机器学习已经到了一个转折点,从今往后,统计学习应该和知识的利用相结合,这是一种“螺旋式上升,进入更高级的形式”,否则,统计学习可能会停留于现状而止步不前。王珏教授还认为:进入转折点的标志就是Koller等的《概率图模型》一书的出版。至于第三种观点,恰好我收到老朋友,美国人工智能资深学者、俄亥俄大学Chandrasekaran教授的来信,他正好谈起符号智能被统计智能“打压”的现象,并且正好表达了河东河西的观点。我请求他允许我把这段话引进正在撰写的序言中,他爽快地同意了,仅仅修改了几处私人通信的口吻。全文如下:“最近几年,人工智能在很大程度上集中于统计学和大数据。我同意由于计算能力的大幅提高,这些技术曾经取得过某些令人印象深刻的成果。但是我们完全有理由相信,虽然这些技术还会继续改进、提高,总有一天这个领域(指AI)会对它们说再见,并转向更加基本的认知科学研究。尽管钟摆的摆回去还需要一段时间,我

相信定有必要把统计技术和对认知结构的深刻理解结合起来。”看来, Chandrasekaran 教授也并不认为若干年以后 AI 真会回到河西, 他的意见和王珏教授的意见基本一致, 但不仅限于机器学习, 而是涉及整个人工智能领域. 只是王珏教授强调知识, 而 Chandrasekaran 教授强调更加基本的“认知”.

问题二: 王珏教授认为统计机器学习不会“一路顺风”的判据是: 统计机器学习算法都是基于样本数据独立同分布的假设. 但是自然界现象千变万化, 王珏教授认为“哪有那么多独立同分布?”这就引来了下一个问题: “独立同分布”条件对于机器学习来讲真是必需的吗? 独立同分布的不存在一定是一个不可逾越的障碍吗? 无独立同分布条件下的机器学习也许只是一个难题, 而不是不可解问题. 我有一个“胡思乱想”, 认为前些时候出现的“迁移学习”也许会对这个问题的解决带来一线曙光. 尽管现在的迁移学习还要求迁移双方具备“独立同分布”条件, 但是不同分布之间的迁移学习, 同分布和异分布之间的迁移学习也许迟早会出现?

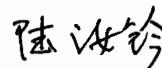
问题三: 近年来出现了一些新的动向, 例如“深度学习”、“无终止学习”等等, 社会上给予了特别关注, 尤其是深度学习. 但它们真的代表了机器学习的新方向吗? 包括本书作者周志华教授在内的一些学者认为: 深度学习掀起的热潮也许大过它本身真正的贡献, 在理论和技术上并没有太多的创新, 只不过是由于硬件技术的革命, 计算机的速度大大提高了, 使得人们有可能采用原来复杂度很高的算法, 从而得到比过去更精细的结果. 当然这对于推动机器学习应用于实践有很大意义. 但我们不禁要斗胆问一句: 深度学习是否又要取代统计学习了? 事实上, 确有专家已经感受到来自深度学习的压力, 指出统计学习正在被深度学习所打压, 正如我们早就看到的符号学习被统计学习所打压. 不过我觉得这种打压还远没有强大到像统计学习打压符号学习的程度. 这是因为深度学习的“理论创新”还不明显; 二是因为目前的深度学习主要适合于神经网络, 在各种机器学习方法百花盛开的今天, 它的应用范围还有限, 还不能直接说是连接主义方法的回归; 三是因为统计学习仍然在机器学习中被有效地普遍采用, “得道多助”, 想抛弃它不容易.

问题四: 机器学习研究出现以来, 我们看到的主要的是从符号方法到统计方法的演变, 用到的数学主要是概率统计. 但是, 数学之大, 就像大海. 难道只有统计方法适合于在机器学习方面应用吗? 当然, 我们也看到了一些其他数学分支在机器学习上的应用的好例子, 例如微分几何在流形学习上的应用, 微分方程在归纳学习上的应用. 但如果和统计方法相比, 它们都只能算是配角. 还有的数学分支如代数可能应用得更广, 但在机器学习中代数一般是作为基础工具来使用, 例如矩阵理论和特征值理论. 又如微分方程求解最终往往归结为代数问题求解. 它们可以算是幕后英雄: “出头露面的是概率和统计, 埋头苦干的是代数和逻辑”. 是否可以想象以数学方法为主角, 以统计方法为配角的机器学习理论呢? 在这方面, 流形学习已经“有点意思”了, 而彭实戈院士的倒排随机微分方程理论之预测金融走势, 也许是用高深数学推动新的机器学习模式的更好例子. 但是从宏观的角度看, 数学理论的介入程度还远远不够. 这里指的主要是深刻的、现代的数学理论, 我们期待着有更多数学家的参与, 开辟机器学习的新模式、新理论、新方向.

问题五：上一个问题的延续：符号机器学习时代主要以离散方法处理问题，统计机器学习时代主要以连续方法处理问题。这两种方法之间应该没有一条鸿沟。流形学习中李群、李代数方法的引入给我们以很好的启示。从微分流形到李群，再从李群到李代数，就是一个沟通连续和离散的过程。然而，现有的方法在数学上并不完美。浏览流形学习的文献可知，许多论文直接把任意数据集看成微分流形，从而就认定测地线的存在并讨论起降维来了。这样的例子也许不是个别的，足可说明数学家介入机器学习研究之必要。

问题六：大数据时代的出现，有没有给机器学习带来本质性的影响？理论上讲，似乎“大数据”给统计机器学习提供了更多的机遇，因为海量的数据更加需要统计、抽样的方法。业界人士估计，大数据的出现将使人工智能的作用更加突出。有人把大数据处理分成三个阶段：收集、分析和预测。收集和分析的工作相对来说已经做得相当好了，现在关注的焦点是要有科学的预测，机器学习技术在这里不可或缺。这一点大概毋庸置疑。然而，同样是使用统计、抽样方法，同样是收集、分析和预测，大数据时代使用这类方法和以前使用这类方法有什么本质的不同吗？量变到质变是辩证法的一个普遍规律。那么，从前大数据时代到大数据时代，数理统计方法有没有发生本质的变化？反映到它们在机器学习上的应用有无本质变化？大数据时代正在呼唤什么样的机器学习方法的产生？哪些机器学习方法又是由于大数据研究的驱动而产生的呢？

以上这些话也许说得远了，我们还是回到本书上来。本书的作者周志华教授在机器学习的许多领域都有出色的贡献，是中国机器学习研究的领军人物之一，在国际学术界有着很高的声誉。他在机器学习的一些重要领域，例如集成学习、半监督学习、多示例和多标记学习等方面都做出了在国际上有重要影响的工作，其中一些可以认为是中国学者在国际上的代表性贡献。除了自身的学术研究以外，他在推动中国的机器学习发展方面也做了许多工作。例如他和不久前刚过世的王珏教授从2002年开始，组织了系列化的“机器学习及其应用”研讨会。初在复旦，后移至南大举行，越办越兴旺，从单一的专家报告发展到专家报告、学生论坛和张贴论文三种方式同时举行，参会者从数十人发展到数百人，活动搞得有声有色，如火如荼。最近更是把研讨会推向全国高校轮流举行。他和王珏教授紧密合作，南北呼应，人称“南周北王”。王珏教授的离去使我们深感悲伤。令我们欣慰的是国内不但有周志华教授这样的机器学习领军人物，而且比周教授更年轻的许多机器学习青年才俊也成长起来了。中国的机器学习大有希望。



中国科学院数学与系统科学研究院

2015年8月于北京

主要符号表

x	标量
\boldsymbol{x}	向量
\mathbf{x}	变量集
\mathbf{A}	矩阵
\mathbf{I}	单位阵
\mathcal{X}	样本空间或状态空间
\mathcal{D}	概率分布
D	数据样本（数据集）
\mathcal{H}	假设空间
H	假设集
\mathfrak{L}	学习算法
(\cdot, \cdot, \cdot)	行向量
$(\cdot; \cdot; \cdot)$	列向量
$(\cdot)^T$	向量或矩阵转置
$\{\dots\}$	集合
$ \{\dots\} $	集合 $\{\dots\}$ 中元素个数
$\ \cdot\ _p$	L_p 范数, p 缺省时为 L_2 范数
$P(\cdot), P(\cdot \cdot)$	概率质量函数, 条件概率质量函数
$p(\cdot), p(\cdot \cdot)$	概率密度函数, 条件概率密度函数
$\mathbb{E}_{\cdot \sim \mathcal{D}}[f(\cdot)]$	函数 $f(\cdot)$ 对 \cdot 在分布 \mathcal{D} 下的数学期望; 意义明确时将省略 \mathcal{D} 和(或) \cdot
$\sup(\cdot)$	上确界
$\mathbb{I}(\cdot)$	指示函数, 在 \cdot 为真和假时分别取值为 1, 0
$\text{sign}(\cdot)$	符号函数, 在 $\cdot < 0, = 0, > 0$ 时分别取值为 $-1, 0, 1$

目 录

第 1 章 绪论	1
1.1 引言	1
1.2 基本术语	2
1.3 假设空间	4
1.4 归纳偏好	6
1.5 发展历程	10
1.6 应用现状	13
1.7 阅读材料	16
习题	19
参考文献	20
休息一会儿	22
第 2 章 模型评估与选择	23
2.1 经验误差与过拟合	23
2.2 评估方法	24
2.3 性能度量	28
2.4 比较检验	37
2.5 偏差与方差	44
2.6 阅读材料	46
习题	48
参考文献	49
休息一会儿	51
第 3 章 线性模型	53
3.1 基本形式	53
3.2 线性回归	53
3.3 对数几率回归	57
3.4 线性判别分析	60
3.5 多分类学习	63

3.6	类别不平衡问题	66
3.7	阅读材料	67
	习题	69
	参考文献	70
	休息一会儿	72
	第 4 章 决策树	73
4.1	基本流程	73
4.2	划分选择	75
4.3	剪枝处理	79
4.4	连续与缺失值	83
4.5	多变量决策树	88
4.6	阅读材料	92
	习题	93
	参考文献	94
	休息一会儿	95
	第 5 章 神经网络	97
5.1	神经元模型	97
5.2	感知机与多层网络	98
5.3	误差逆传播算法	101
5.4	全局最小与局部极小	106
5.5	其他常见神经网络	108
5.6	深度学习	113
5.7	阅读材料	115
	习题	116
	参考文献	117
	休息一会儿	120
	第 6 章 支持向量机	121
6.1	间隔与支持向量	121
6.2	对偶问题	123
6.3	核函数	126
6.4	软间隔与正则化	129
6.5	支持向量回归	133

6.6 核方法	137
6.7 阅读材料	139
习题	141
参考文献	142
休息一会儿	145
第 7 章 贝叶斯分类器	147
7.1 贝叶斯决策论	147
7.2 极大似然估计	149
7.3 朴素贝叶斯分类器	150
7.4 半朴素贝叶斯分类器	154
7.5 贝叶斯网	156
7.6 EM算法	162
7.7 阅读材料	164
习题	166
参考文献	167
休息一会儿	169
第 8 章 集成学习	171
8.1 个体与集成	171
8.2 Boosting	173
8.3 Bagging与随机森林	178
8.4 结合策略	181
8.5 多样性	185
8.6 阅读材料	190
习题	192
参考文献	193
休息一会儿	196
第 9 章 聚类	197
9.1 聚类任务	197
9.2 性能度量	197
9.3 距离计算	199
9.4 原型聚类	202
9.5 密度聚类	211

9.6 层次聚类	214
9.7 阅读材料	217
习题	220
参考文献	221
休息一会儿	224
第 10 章 降维与度量学习	225
10.1 <i>k</i> 近邻学习	225
10.2 低维嵌入	226
10.3 主成分分析	229
10.4 核化线性降维	232
10.5 流形学习	234
10.6 度量学习	237
10.7 阅读材料	240
习题	242
参考文献	243
休息一会儿	246
第 11 章 特征选择与稀疏学习	247
11.1 子集搜索与评价	247
11.2 过滤式选择	249
11.3 包裹式选择	250
11.4 嵌入式选择与 L_1 正则化	252
11.5 稀疏表示与字典学习	254
11.6 压缩感知	257
11.7 阅读材料	260
习题	262
参考文献	263
休息一会儿	266
第 12 章 计算学习理论	267
12.1 基础知识	267
12.2 PAC 学习	268
12.3 有限假设空间	270
12.4 VC 维	273

12.5 Rademacher复杂度	279
12.6 稳定性	284
12.7 阅读材料	287
习题	289
参考文献	290
休息一会儿	292
第 13 章 半监督学习	293
13.1 未标记样本	293
13.2 生成式方法	295
13.3 半监督SVM	298
13.4 图半监督学习	300
13.5 基于分歧的方法	304
13.6 半监督聚类	307
13.7 阅读材料	311
习题	313
参考文献	314
休息一会儿	317
第 14 章 概率图模型	319
14.1 隐马尔可夫模型	319
14.2 马尔可夫随机场	322
14.3 条件随机场	325
14.4 学习与推断	328
14.5 近似推断	331
14.6 话题模型	337
14.7 阅读材料	339
习题	341
参考文献	342
休息一会儿	345
第 15 章 规则学习	347
15.1 基本概念	347
15.2 序贯覆盖	349
15.3 剪枝优化	352

15.4 一阶规则学习	354
15.5 归纳逻辑程序设计	357
15.6 阅读材料	363
习题	365
参考文献	366
休息一会儿	369
第 16 章 强化学习.....	371
16.1 任务与奖赏	371
16.2 <i>K</i> -摇臂赌博机	373
16.3 有模型学习	377
16.4 免模型学习	382
16.5 值函数近似	388
16.6 模仿学习	390
16.7 阅读材料	393
习题	394
参考文献	395
休息一会儿	397
附录.....	399
A 矩阵	399
B 优化	403
C 概率分布	409
后记.....	417
索引.....	419

第1章 緒論

1.1 引言

傍晚小街路面上沁出微雨后的湿润，和煦的细风吹来，抬头看看天边的晚霞，嗯，明天又是一个好天气。走到水果摊旁，挑了个根蒂蜷缩、敲起来声音浊响的青绿西瓜，一边满心期待着皮薄肉厚瓢甜的爽落感，一边愉快地想着，这学期狠下了工夫，基础概念弄得清清楚楚，算法作业也是信手拈来，这门课成绩一定差不了！

希望各位在学期结束时有这样的感觉。作为开场，我们先大致了解一下什么是“机器学习”(machine learning)。

回头看第一段话，我们会发现这里涉及很多基于经验做出的预判。例如，为什么看到微湿路面、感到和风、看到晚霞，就认为明天是好天呢？这是因为在我们的生活经验中已经遇见过很多类似情况，头一天观察到上述特征后，第二天天气通常会很好。为什么色泽青绿、根蒂蜷缩、敲声浊响，就能判断出是正熟的好瓜？因为我们吃过、看过很多西瓜，所以基于色泽、根蒂、敲声这几个特征我们就可以做出相当好的判断。类似的，我们从以往的学习经验知道，下足了工夫、弄清了概念、做好了作业，自然会取得好成绩。可以看出，我们能做出有效的预判，是因为我们已经积累了许多经验，而通过对经验的利用，就能对新情况做出有效的决策。

上面对经验的利用是靠我们人类自身完成的。计算机能帮忙吗？

机器学习正是这样一门学科，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”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”(model)的算法，即“学习算法”(learning algorithm)。有了学习算法，我们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(例如看到一个没剖开的西瓜)，模型会给我们提供相应的判断(例如好瓜)。如果说计算机科学是研究关于“算法”的学问，那么类似的，可以说机器学习是研究关于“学习算法”的学问。

[Mitchell, 1997] 给出了一个更形式化的定义：假设用 P 来评估计算机程序在某任务类 T 上的性能，若一个程序通过利用经验 E 在 T 中任务上获得了性能改善，则我们就说关于 T 和 P ，该程序对 E 进行了学习。

例如[Hand et al., 2001]。

本书用“模型”泛指从数据中学得的结果。有文献用“模型”指全局性结果(例如一棵决策树)，而用“模式”指局部性结果(例如一条规则)。

1.2 基本术语

要进行机器学习,先要有数据。假定我们收集了一批关于西瓜的数据,例如(色泽=青绿;根蒂=蜷缩;敲声=浊响),(色泽=乌黑;根蒂=稍蜷;敲声=沉闷),(色泽=浅白;根蒂=硬挺;敲声=清脆),……,每对括号内是一条记录,“=”意思是“取值为”:

这组记录的集合称为一个“数据集”(data set),其中每条记录是关于一个事件或对象(这里是一个西瓜)的描述,称为一个“示例”(instance)或“样本”(sample)。反映事件或对象在某方面的表现或性质的事项,例如“色泽”“根蒂”“敲声”,称为“属性”(attribute)或“特征”(feature);属性上的取值,例如“青绿”“乌黑”,称为“属性值”(attribute value)。属性张成的空间称为“属性空间”(attribute space)、“样本空间”(sample space)或“输入空间”。例如我们把“色泽”“根蒂”“敲声”作为三个坐标轴,则它们张成一个用于描述西瓜的三维空间,每个西瓜都可在这个空间中找到自己的坐标位置。由于空间中的每个点对应一个坐标向量,因此我们也把一个示例称为一个“特征向量”(feature vector)。

一般地,令 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ 表示包含 m 个示例的数据集,每个示例由 d 个属性描述(例如上面的西瓜数据使用了3个属性),则每个示例 $\mathbf{x}_i = (x_{i1}; x_{i2}; \dots; x_{id})$ 是 d 维样本空间 \mathcal{X} 中的一个向量, $\mathbf{x}_i \in \mathcal{X}$,其中 x_{ij} 是 \mathbf{x}_i 在第 j 个属性上的取值(例如上述第3个西瓜在第2个属性上的值是“硬挺”), d 称为样本 \mathbf{x}_i 的“维数”(dimensionality)。

从数据中学得模型的过程称为“学习”(learning)或“训练”(training),这个过程通过执行某个学习算法来完成。训练过程中使用的数据称为“训练数据”(training data),其中每个样本称为一个“训练样本”(training sample),训练样本组成的集合称为“训练集”(training set)。学得模型对应了关于数据的某种潜在的规律,因此亦称“假设”(hypothesis);这种潜在规律自身,则称为“真相”或“真实”(ground-truth),学习过程就是为了找出或逼近真相。本书有时将模型称为“学习器”(learner),可看作学习算法在给定数据和参数空间上的实例化。

如果希望学得一个能帮助我们判断没剖开的是不是“好瓜”的模型,仅有前面的示例数据显然是不够的。要建立这样的关于“预测”(prediction)的模型,我们需获得训练样本的“结果”信息,例如“((色泽=青绿;根蒂=蜷缩;敲声=浊响),好瓜)”。这里关于示例结果的信息,例如“好瓜”,称为“标记”(label);拥有了标记信息的示例,则称为“样例”(example)。一般地,用

有时整个数据集亦称一个“样本”,因为它可看作对样本空间的一个采样;通过上下文可判断出“样本”是指单个示例还是数据集。

训练样本亦称“训练示例”(training instance)或“训练例”。

学习算法通常有参数需设置,使用不同的参数值和(或)训练数据,将产生不同的结果。

将“label”译为“标记”而非“标签”,是考虑到英文中“label”既可用作名词、也可用作动词。

若将标记看作对象本身的一部分，则“样例”有时也称为“样本”。

亦称“负类”。

亦称“测试示例”
(testing instance) 或“测试例”。

否则标记信息直接形成了簇划分；但也有例外情况，参见 13.6 节。

亦称“有导师学习”和
“无导师学习”。

更确切地说，是“未见
示例”(unseen instance)。

现实任务中样本空间的
规模通常很大(例如 20 个
属性，每个属性有 10 个可
能取值，则样本空间的规
模已达 10^{20})。

(\mathbf{x}_i, y_i) 表示第 i 个样例，其中 $y_i \in \mathcal{Y}$ 是示例 \mathbf{x}_i 的标记， \mathcal{Y} 是所有标记的集合，亦称“标记空间”(label space)或“输出空间”。

若我们欲预测的是离散值，例如“好瓜”“坏瓜”，此类学习任务称为“分类”(classification)；若欲预测的是连续值，例如西瓜成熟度 0.95、0.37，此类学习任务称为“回归”(regression)。对只涉及两个类别的“二分类”(binary classification)任务，通常称其中一个类为“正类”(positive class)，另一个类为“反类”(negative class)；涉及多个类别时，则称为“多分类”(multi-class classification)任务。一般地，预测任务是希望通过训练集 $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ 进行学习，建立一个从输入空间 \mathcal{X} 到输出空间 \mathcal{Y} 的映射 $f : \mathcal{X} \mapsto \mathcal{Y}$ 。对二分类任务，通常令 $\mathcal{Y} = \{-1, +1\}$ 或 $\{0, 1\}$ ；对多分类任务， $|\mathcal{Y}| > 2$ ；对回归任务， $\mathcal{Y} = \mathbb{R}$ ， \mathbb{R} 为实数集。

学得模型后，使用其进行预测的过程称为“测试”(testing)，被预测的样本称为“测试样本”(testing sample)。例如在学得 f 后，对测试例 \mathbf{x} ，可得到其预测标记 $y = f(\mathbf{x})$ 。

我们还可以对西瓜做“聚类”(clustering)，即将训练集中的西瓜分成若干组，每组称为一个“簇”(cluster)；这些自动形成的簇可能对应一些潜在的概念划分，例如“浅色瓜”“深色瓜”，甚至“本地瓜”“外地瓜”。这样的学习过程有助于我们了解数据内在的规律，能为更深入地分析数据建立基础。需说明的是，在聚类学习中，“浅色瓜”“本地瓜”这样的概念我们事先是不知道的，而且学习过程中使用的训练样本通常不拥有标记信息。

根据训练数据是否拥有标记信息，学习任务可大致划分为两大类：“监督学习”(supervised learning) 和“无监督学习”(unsupervised learning)，分类和回归是前者的代表，而聚类则是后者的代表。

需注意的是，机器学习的目标是使学得的模型能很好地适用于“新样本”，而不是仅仅在训练样本上工作得很好；即便对聚类这样的无监督学习任务，我们也希望学得的簇划分能适用于没在训练集中出现的样本。学得模型适用于新样本的能力，称为“泛化”(generalization)能力。具有强泛化能力的模型能很好地适用于整个样本空间。于是，尽管训练集通常只是样本空间的一个很小的采样，我们仍希望它能很好地反映出样本空间的特性，否则就很难期望在训练集上学得的模型能在整个样本空间上都工作得很好。通常假设样本空间中全体样本服从一个未知“分布”(distribution) \mathcal{D} ，我们获得的每个样本都是独立地从这个分布上采样获得的，即“独立同分布”(independent and identically distributed，简称 i.i.d.)。一般而言，训练样本越多，我们得到的关于 \mathcal{D} 的信息

越多,这样就越有可能通过学习获得具有强泛化能力的模型.

1.3 假设空间

归纳(induction)与演绎(deduction)是科学推理的两大基本手段.前者是从特殊到一般的“泛化”(generalization)过程,即从具体的事实在归结出一般性规律;后者则是从一般到特殊的“特化”(specialization)过程,即从基础原理推演出具体状况.例如,在数学公理系统中,基于一组公理和推理规则推导出与之相洽的定理,这是演绎;而“从样例中学习”显然是一个归纳的过程,因此亦称“归纳学习”(inductive learning).

归纳学习有狭义与广义之分,广义的归纳学习大体相当于从样例中学习,而狭义的归纳学习则要求从训练数据中学得概念(concept),因此亦称为“概念学习”或“概念形成”.概念学习技术目前研究、应用都比较少,因为要学得泛化性能好且语义明确的概念实在太困难了,现实常用的技术大多是产生“黑箱”模型.然而,对概念学习有所了解,有助于理解机器学习的一些基础思想.

概念学习中最基本的是布尔概念学习,即对“是”“不是”这样的可表示为0/1布尔值的目标概念的学习.举一个简单的例子,假定我们获得了这样一个训练数据集:

表 1.1 西瓜数据集

编号	色泽	根蒂	敲声	好瓜
1	青绿	蜷缩	浊响	是
2	乌黑	蜷缩	浊响	是
3	青绿	硬挺	清脆	否
4	乌黑	稍蜷	沉闷	否

这里要学习的目标是“好瓜”.暂且假设“好瓜”可由“色泽”“根蒂”“敲声”这三个因素完全确定,换言之,只要某个瓜的这三个属性取值明确了,我们就能判断出它是不是好瓜.于是,我们学得的将是“好瓜是某种色泽、某种根蒂、某种敲声的瓜”这样的概念,用布尔表达式写出来则是“好瓜 \leftrightarrow (色泽=? \wedge 根蒂=? \wedge 敲声=?)”,这里“?”表示尚未确定的取值,而我们的任务就是通过对表 1.1 的训练集进行学习,把“?”确定下来.

更一般的情况是考虑形如 $(A \wedge B) \vee (C \wedge D)$ 的析合范式.

读者可能马上发现,表 1.1 第一行:“(色泽=青绿) \wedge (根蒂=蜷缩) \wedge (敲声=浊响)”不就是好瓜吗?是的,但这是一个已见过的瓜,别忘了我们学习的目的是“泛化”,即通过对训练集中瓜的学习以获得对没见过的瓜进行判断的

“记住”训练样本，就是所谓的“机械学习”[Cohen and Feigenbaum, 1983]，或称“死记硬背式学习”，参见1.5节。

这里我们假定训练样本不含噪声，并且不考虑“非青绿”这样的 $\neg A$ 操作。由于训练集包含正例，因此 \emptyset 假设自然不出现。

能力。如果仅仅把训练集中的瓜“记住”，今后再见到一模一样的瓜当然可判断，但是，对没见过的瓜，例如“(色泽=浅白) \wedge (根蒂=蜷缩) \wedge (敲声=浊响)”怎么办呢？

我们可以把学习过程看作一个在所有假设(hypothesis)组成的空间中进行搜索的过程，搜索目标是找到与训练集“匹配”(fit)的假设，即能够将训练集中的瓜判断正确的假设。假设的表示一旦确定，假设空间及其规模大小就确定了。这里我们的假设空间由形如“(色泽=? \wedge 根蒂=? \wedge 敲声=?)”的可能取值所形成的假设组成。例如色泽有“青绿”“乌黑”“浅白”这三种可能取值；还需考虑到，也许“色泽”无论取什么值都合适，我们用通配符“*”来表示，例如“好瓜 \leftrightarrow (色泽=*) \wedge (根蒂=蜷缩) \wedge (敲声=浊响)”，即“好瓜是根蒂蜷缩、敲声浊响的瓜，什么色泽都行”。此外，还需考虑极端情况：有可能“好瓜”这个概念根本就不成立，世界上没有“好瓜”这种东西；我们用 \emptyset 表示这个假设。这样，若“色泽”“根蒂”“敲声”分别有3、2、2种可能取值，则我们面临的假设空间规模大小为 $4 \times 3 \times 3 + 1 = 37$ 。图1.1直观地显示出了这个西瓜问题假设空间。

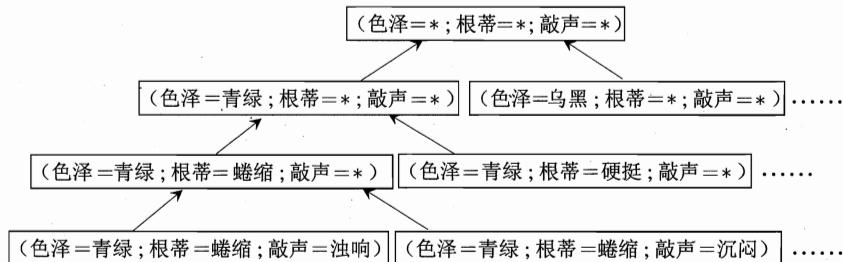


图1.1 西瓜问题的假设空间

可以有许多策略对这个假设空间进行搜索，例如自顶向下、从一般到特殊，或是自底向上、从特殊到一般，搜索过程中可以不断删除与正例不一致的假设、和(或)与反例一致的假设。最终将会获得与训练集一致(即对所有训练样本能够进行正确判断)的假设，这就是我们学得的结果。

需注意的是，现实问题中我们常面临很大的假设空间，但学习过程是基于有限样本训练集进行的，因此，可能有多个假设与训练集一致，即存在着一个与训练集一致的“假设集合”，我们称之为“版本空间”(version space)。例如，在西瓜问题中，与表1.1训练集所对应的版本空间如图1.2所示。

有许多可能的选择，如在路径上自顶向下与自底向上同时进行，在操作上只删除与正例不一致的假设等。

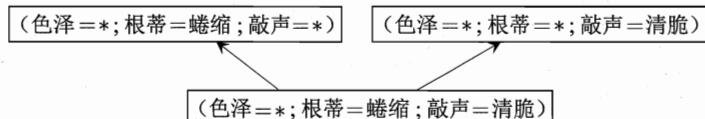


图 1.2 西瓜问题的版本空间

1.4 归纳偏好

通过学习得到的模型对应了假设空间中的一个假设。于是，图 1.2 的西瓜版本空间给我们带来一个麻烦：现在有三个与训练集一致的假设，但与它们对应的模型在面临新样本的时候，却会产生不同的输出。例如，对(色泽=青绿；根蒂=蟠缩；敲声=沉闷)这个新收来的瓜，如果我们采用的是“好瓜 \leftrightarrow (色泽 = *) \wedge (根蒂 = 蟠缩) \wedge (敲声 = *)”，那么将会把新瓜判断为好瓜，而如果采用了另外两个假设，则判断的结果将不是好瓜。那么，应该采用哪一个模型(或假设)呢？

若仅有表 1.1 中的训练样本，则无法断定上述三个假设中哪一个“更好”。然而，对于一个具体的学习算法而言，它必须要产生一个模型。这时，学习算法本身的“偏好”就会起到关键的作用。例如，若我们的算法喜欢“尽可能特殊”的模型，则它会选择“好瓜 \leftrightarrow (色泽 = *) \wedge (根蒂 = 蟠缩) \wedge (敲声 = 浊响)”；但若我们的算法喜欢“尽可能一般”的模型，并且由于某种原因它更“相信”根蒂，则它会选择“好瓜 \leftrightarrow (色泽 = *) \wedge (根蒂 = 蟠缩) \wedge (敲声 = *)”。机器学习算法在学习过程中对某种类型假设的偏好，称为“归纳偏好”(inductive bias)，或简称为“偏好”。

任何一个有效的机器学习算法必有其归纳偏好，否则它将被假设空间中看似在训练集上“等效”的假设所迷惑，而无法产生确定的学习结果。可以想象，如果没有偏好，我们的西瓜学习算法产生的模型每次在进行预测时随机抽选训练集上的等效假设，那么对这个新瓜“(色泽=青绿；根蒂=蟠缩；敲声=沉闷)”，学得模型时而告诉我们它是好的、时而告诉我们它是不好的，这样的学习结果显然没有意义。

归纳偏好的作用在图 1.3 这个回归学习图示中可能更直观。这里的每个训练样本是图中的一个点 (x, y) ，要学得一个与训练集一致的模型，相当于找到一条穿过所有训练样本点的曲线。显然，对有限个样本点组成的训练集，存在着很多条曲线与其一致。我们的学习算法必须有某种偏好，才能产出它认为“正确”的模型。例如，若认为相似的样本应有相似的输出(例如，在各种属性上都

尽可能特殊即“适用情形尽可能少”；尽可能一般即“适用情形尽可能多”。

对“根蒂”还是对“敲声”更重视，看起来和属性选择，亦称“特征选择”(feature selection)有关，但需注意的是，机器学习中的特征选择仍是基于对训练样本的分析进行的，而在此处我们并非基于特征选择做出对“根蒂”的重视；这里对“根蒂”的信赖可视为基于某种领域知识而产生的归纳偏好。关于特征选择方面的内容参见第 11 章。

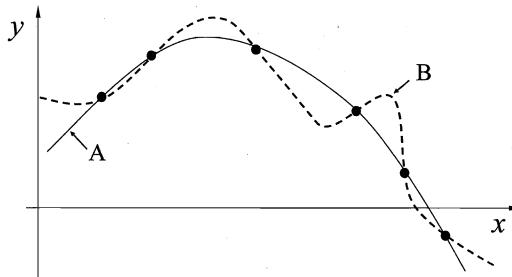


图 1.3 存在多条曲线与有限样本训练集一致

很相像的西瓜，成熟程度应该比较接近），则对应的学习算法可能偏好图 1.3 中比较“平滑”的曲线 A 而不是比较“崎岖”的曲线 B.

归纳偏好可看作学习算法自身在一个可能很庞大的假设空间中对假设进行选择的启发式或“价值观”。那么，有没有一般性的原则来引导算法确立“正确的”偏好呢？“奥卡姆剃刀”（Occam's razor）是一种常用的、自然科学研究中最基本的原则，即“若有多个假设与观察一致，则选最简单的那个”。如果采用这个原则，并且假设我们认为“更平滑”意味着“更简单”（例如曲线 A 更易于描述，其方程式是 $y = -x^2 + 6x + 1$ ，而曲线 B 则要复杂得多），则在图 1.3 中我们会自然地偏好“平滑”的曲线 A.

然而，奥卡姆剃刀并非唯一可行的原则。退一步说，即便假定我们是奥卡姆剃刀的铁杆拥趸，也需注意到，奥卡姆剃刀本身存在不同的诠释，使用奥卡姆剃刀原则并不平凡。例如对我们已经很熟悉的西瓜问题来说，“假设 1：好瓜 \leftrightarrow (色泽 = *) \wedge (根蒂 = 蜷缩) \wedge (敲声 = 浊响)”和假设 2：“好瓜 \leftrightarrow (色泽 = *) \wedge (根蒂 = 蜷缩) \wedge (敲声 = *)”这两个假设，哪一个更“简单”呢？这个问题并不简单，需借助其他机制才能解决。

事实上，归纳偏好对应了学习算法本身所做出的关于“什么样的模型更好”的假设。在具体的现实问题中，这个假设是否成立，即算法的归纳偏好是否与问题本身匹配，大多数时候直接决定了算法能否取得好的性能。

让我们再回头看看图 1.3。假设学习算法 \mathcal{L}_a 基于某种归纳偏好产生了对应于曲线 A 的模型，学习算法 \mathcal{L}_b 基于另一种归纳偏好产生了对应于曲线 B 的模型。基于前面讨论的平滑曲线的某种“描述简单性”，我们满怀信心地期待算法 \mathcal{L}_a 比 \mathcal{L}_b 更好。确实，图 1.4(a) 显示出，与 B 相比，A 与训练集外的样本更一致；换言之，A 的泛化能力比 B 强。

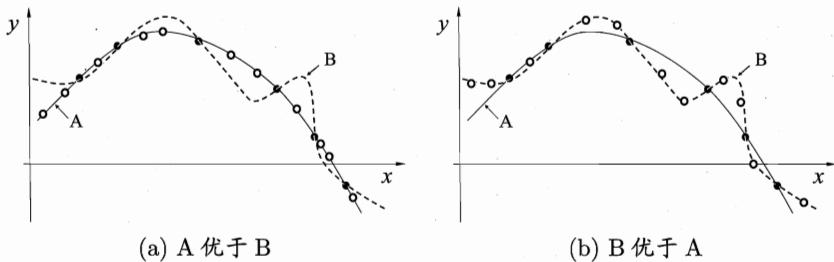


图 1.4 没有免费的午餐. (黑点: 训练样本; 白点: 测试样本)

但是, 且慢! 虽然我们希望并相信 \mathcal{L}_a 比 \mathcal{L}_b 更好, 但会不会出现图 1.4(b) 的情况: 与 A 相比, B 与训练集外的样本更一致?

很遗憾, 这种情况完全可能出现. 换言之, 对于一个学习算法 \mathcal{L}_a , 若它在某些问题上比学习算法 \mathcal{L}_b 好, 则必然存在另一些问题, 在那里 \mathcal{L}_b 比 \mathcal{L}_a 好. 有趣的是, 这个结论对任何算法均成立, 哪怕是把本书后面将要介绍的一些聪明算法作为 \mathcal{L}_a 而将“随机胡猜”这样的笨拙算法作为 \mathcal{L}_b . 惊讶吗? 让我们看看下面这个简短的讨论:

这里只用到一些非常基础的数学知识, 只准备读第 1 章且有“数学恐惧”的读者可以跳过这个部分而不会影响理解, 只需相信, 上面这个看起来“匪夷所思”的结论确实是成立的.

为简单起见, 假设样本空间 \mathcal{X} 和假设空间 \mathcal{H} 都是离散的. 令 $P(h|X, \mathcal{L}_a)$ 代表算法 \mathcal{L}_a 基于训练数据 X 产生假设 h 的概率, 再令 f 代表我们希望学习的真实目标函数. \mathcal{L}_a 的“训练集外误差”, 即 \mathcal{L}_a 在训练集之外的所有样本上的误差为

$$E_{ote}(\mathcal{L}_a|X, f) = \sum_h \sum_{\mathbf{x} \in \mathcal{X}-X} P(\mathbf{x}) \mathbb{I}(h(\mathbf{x}) \neq f(\mathbf{x})) P(h | X, \mathcal{L}_a), \quad (1.1)$$

其中 $\mathbb{I}(\cdot)$ 是指示函数, 若 · 为真则取值 1, 否则取值 0.

考虑二分类问题, 且真实目标函数可以是任何函数 $\mathcal{X} \mapsto \{0, 1\}$, 函数空间为 $\{0, 1\}^{|\mathcal{X}|}$. 对所有可能的 f 按均匀分布对误差求和, 有

$$\begin{aligned} \sum_f E_{ote}(\mathcal{L}_a|X, f) &= \sum_f \sum_{\mathbf{x} \in \mathcal{X}-X} P(\mathbf{x}) \mathbb{I}(h(\mathbf{x}) \neq f(\mathbf{x})) P(h | X, \mathcal{L}_a) \\ &= \sum_{\mathbf{x} \in \mathcal{X}-X} P(\mathbf{x}) \sum_h P(h | X, \mathcal{L}_a) \sum_f \mathbb{I}(h(\mathbf{x}) \neq f(\mathbf{x})) \\ &= \sum_{\mathbf{x} \in \mathcal{X}-X} P(\mathbf{x}) \sum_h P(h | X, \mathcal{L}_a) \frac{1}{2} 2^{|\mathcal{X}|} \\ &= \frac{1}{2} 2^{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}-X} P(\mathbf{x}) \sum_h P(h | X, \mathcal{L}_a) \end{aligned}$$

若 f 均匀分布, 则有一半的 f 对 \mathbf{x} 的预测与 $h(\mathbf{x})$ 不一致.

$$= 2^{|\mathcal{X}|-1} \sum_{\mathbf{x} \in \mathcal{X} - X} P(\mathbf{x}) \cdot 1. \quad (1.2)$$

式(1.2)显示出, 总误差竟然与学习算法无关! 对于任意两个学习算法 \mathcal{L}_a 和 \mathcal{L}_b , 我们都有

$$\sum_f E_{ote}(\mathcal{L}_a|X, f) = \sum_f E_{ote}(\mathcal{L}_b|X, f), \quad (1.3)$$

严格的 NFL 定理证明比这里的简化论述繁难得多.

也就是说, 无论学习算法 \mathcal{L}_a 多聪明、学习算法 \mathcal{L}_b 多笨拙, 它们的期望性能竟然相同! 这就是“没有免费的午餐”定理 (No Free Lunch Theorem, 简称 NFL 定理) [Wolpert, 1996; Wolpert and Macready, 1995].

这下子, 读者对机器学习的热情可能被一盆冷水浇透了: 既然所有学习算法的期望性能都跟随机胡猜差不多, 那还有什么好学的?

我们需注意到, NFL 定理有一个重要前提: 所有“问题”出现的机会相同、或所有问题同等重要. 但实际情形并不是这样. 很多时候, 我们只关注自己正在试图解决的问题(例如某个具体应用任务), 希望为它找到一个解决方案, 至于这个解决方案在别的问题、甚至在相似的问题上是否为好方案, 我们并不关心. 例如, 为了快速从 A 地到达 B 地, 如果我们正在考虑的 A 地是南京鼓楼、B 地是南京新街口, 那么“骑自行车”是很好的解决方案; 这个方案对 A 地是南京鼓楼、B 地是北京新街口的情形显然很糟糕, 但我们对此并不关心.

事实上, 上面 NFL 定理的简短论述过程中假设了 f 的均匀分布, 而实际情形并非如此. 例如, 回到我们熟悉的西瓜问题, 考虑 {假设 1: 好瓜 \leftrightarrow (色泽 = *) \wedge (根蒂 = 蟠缩) \wedge (敲声 = 浊响)} 和 {假设 2: 好瓜 \leftrightarrow (色泽 = *) \wedge (根蒂 = 硬挺) \wedge (敲声 = 清脆)}. 从 NFL 定理可知, 这两个假设同样好. 我们立即会想到符合条件的例子, 对好瓜(色泽 = 青绿; 根蒂 = 蟠缩; 敲声 = 浊响)是假设 1 更好, 而对好瓜(色泽 = 乌黑; 根蒂 = 硬挺; 敲声 = 清脆)则是假设 2 更好. 看上去的確是这样. 然而需注意到, “(根蒂 = 蟠缩; 敲声 = 浊响)”的好瓜很常见, 而“(根蒂 = 硬挺; 敲声 = 清脆)”的好瓜罕见, 甚至不存在.

所以, NFL 定理最重要的寓意, 是让我们清楚地认识到, 脱离具体问题, 空泛地谈论“什么学习算法更好”毫无意义, 因为若考虑所有潜在的问题, 则所有学习算法都一样好. 要谈论算法的相对优劣, 必须要针对具体的学习问题; 在某些问题上表现好的学习算法, 在另一些问题上却可能不尽如人意, 学习算法自身的归纳偏好与问题是否相配, 往往会起到决定性的作用.

1.5 发展历程

机器学习是人工智能(artificial intelligence)研究发展到一定阶段的必然产物。二十世纪五十年代到七十年代初，人工智能研究处于“推理期”，那时人们以为只要能赋予机器逻辑推理能力，机器就能具有智能。这一阶段的代表性工作主要有 A. Newell 和 H. Simon 的“逻辑理论家”(Logic Theorist)程序以及此后的“通用问题求解”(General Problem Solving)程序等，这些工作在当时取得了令人振奋的结果。例如，“逻辑理论家”程序在 1952 年证明了著名数学家罗素和怀特海的名著《数学原理》中的 38 条定理，在 1963 年证明了全部 52 条定理，特别值得一提的是，定理 2.85 甚至比罗素和怀特海证明得更巧妙。A. Newell 和 H. Simon 因为这方面的工作获得了 1975 年图灵奖。然而，随着研究向前发展，人们逐渐认识到，仅具有逻辑推理能力是远远实现不了人工智能的。E. A. Feigenbaum 等人认为，要使机器具有智能，就必须设法使机器拥有知识。在他们的倡导下，从二十世纪七十年代中期开始，人工智能研究进入了“知识期”。在这一时期，大量专家系统问世，在很多应用领域取得了大量成果。E. A. Feigenbaum 作为“知识工程”之父在 1994 年获得图灵奖。但是，人们逐渐认识到，专家系统面临“知识工程瓶颈”，简单地说，就是由人来把知识总结出来再教给计算机是相当困难的。于是，一些学者想到，如果机器自己能够学习知识该多好！

事实上，图灵在 1950 年关于图灵测试的文章中，就曾提到了机器学习的可能；二十世纪五十年代初已有机器学习的相关研究，例如 A. Samuel 著名的跳棋程序。五十年代中后期，基于神经网络的“连接主义”(connectionism)学习开始出现，代表性工作有 F. Rosenblatt 的感知机(Perceptron)、B. Widrow 的 Adaline 等。在六七十年代，基于逻辑表示的“符号主义”(symbolism)学习技术蓬勃发展，代表性工作有 P. Winston 的“结构学习系统”、R. S. Michalski 等人的“基于逻辑的归纳学习系统”、E. B. Hunt 等人的“概念学习系统”等；以决策理论为基础的学习技术以及强化学习技术等也得到发展，代表性工作有 N. J. Nilson 的“学习机器”等；二十多年后红极一时的统计学习理论的一些奠基性结果也是在这个时期取得的。

参见 p.22.

IWML 后来发展为国际
机器学习会议 ICML

1980 年夏，在美国卡耐基梅隆大学举行了第一届机器学习研讨会(IWML)；同年，《策略分析与信息系统》连出三期机器学习专辑；1983 年，Tioga 出版社出版了 R. S. Michalski、J. G. Carbonell 和 T. Mitchell 主编的《机器学习：一种人工智能途径》[Michalski et al., 1983]，对当时的机器学习研究工作进行了总结；1986 年，第一本机器学习专业期刊 *Machine Learning* 创刊；1989 年，人

工智能领域的权威期刊 *Artificial Intelligence* 出版机器学习专辑, 刊发了当时一些比较活跃的研究工作, 其内容后来出现在 J. G. Carbonell 主编、MIT 出版社 1990 年的《机器学习: 范型与方法》[Carbonell, 1990] 一书中。总的来看, 二十世纪八十年代是机器学习成为一个独立的学科领域、各种机器学习技术百花初绽的时期。

R. S. Michalski 等人 [Michalski et al., 1983] 把机器学习研究划分为“从样例中学习”“在问题求解和规划中学习”“通过观察和发现学习”“从指令中学习”等种类; E. A. Feigenbaum 等人在著名的《人工智能手册》(第三卷) [Cohen and Feigenbaum, 1983] 中, 则把机器学习划分为“机械学习”“示教学习”“类比学习”和“归纳学习”。机械学习亦称“死记硬背式学习”, 即把外界输入的信息全部记录下来, 在需要时原封不动地拿出来使用, 这实际上没有进行真正的学习, 仅是在进行信息存储与检索; 示教学习和类比学习类似于 R. S. Michalski 等人所说的“从指令中学习”和“通过观察和发现学习”; 归纳学习相当于“从样例中学习”, 即从训练样例中归纳出学习结果。二十世纪八十年代以来, 被研究最多、应用最广的是“从样例中学习”(也就是广义的归纳学习), 它涵盖了监督学习、无监督学习等, 本书大部分内容均属此范畴。下面我们对这方面主流技术的演进做一个简单回顾。

参见第 4 章。

这时实际是 ILP 的前身。

参见第 15 章。

在二十世纪八十年代, “从样例中学习”的一大主流是符号主义学习, 其代表包括决策树(decision tree)和基于逻辑的学习。典型的决策树学习以信息论为基础, 以信息熵的最小化为目标, 直接模拟了人类对概念进行判定的树形流程。基于逻辑的学习的著名代表是归纳逻辑程序设计(Inductive Logic Programming, 简称 ILP), 可看作机器学习与逻辑程序设计的交叉, 它使用一阶逻辑(即谓词逻辑)来进行知识表示, 通过修改和扩充逻辑表达式(例如 Prolog 表达式)来完成对数据的归纳。符号主义学习占据主流地位与整个人工智能领域的发展历程是分不开的。前面说过, 人工智能在二十世纪五十到八十年代经历了“推理期”和“知识期”, 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成就, 而在“知识期”人们基于符号知识表示、通过获取和利用领域知识来建立专家系统取得了大量成果, 因此, 在“学习期”的开始, 符号知识表示很自然地受到青睐。事实上, 机器学习在二十世纪八十年代正是被视为“解决知识工程瓶颈问题的关键”而走上人工智能主舞台的。决策树学习技术由于简单易用, 到今天仍是最常用的机器学习技术之一。ILP 具有很强的知识表示能力, 可以较容易地表达出复杂数据关系, 而且领域知识通常可方便地通过逻辑表达式进行描述, 因此, ILP 不仅可利用领域知识辅助学习, 还可

通过学习对领域知识进行精化和增强;然而,成也萧何、败也萧何,由于表示能力太强,直接导致学习过程面临的假设空间太大、复杂度极高,因此,问题规模稍大就难以有效进行学习,九十年代中期后这方面的研究相对陷入低潮.

二十世纪九十年代中期之前,“从样例中学习”的另一主流技术是基于神经网络的连接主义学习.连接主义学习在二十世纪五十年代取得了大发展,但因为早期的很多人工智能研究者对符号表示有特别偏爱,例如图灵奖得主 H. Simon 曾断言人工智能是研究“对智能行为的符号化建模”,所以当时连接主义的研究未被纳入主流人工智能研究范畴.尤其是连接主义自身也遇到了很大的障碍,正如图灵奖得主 M. Minsky 和 S. Papert 在1969 年指出,(当时的)神经网络只能处理线性分类,甚至对“异或”这么简单的问题都处理不了.1983 年,J. J. Hopfield 利用神经网络求解“流动推销员问题”这个著名的 NP 难题取得重大进展,使得连接主义重新受到人们关注.1986 年,D. E. Rumelhart 等人重新发明了著名的 BP 算法,产生了深远影响.与符号主义学习能产生明确的概念表示不同,连接主义学习产生的是“黑箱”模型,因此从知识获取的角度来看,连接主义学习技术有明显弱点;然而,由于有 BP 这样有效的算法,使得它可以在很多现实问题上发挥作用.事实上,BP 一直是被应用得最广泛的机器学习算法之一.连接主义学习的最大局限是其“试错性”;简单地说,其学习过程涉及大量参数,而参数的设置缺乏理论指导,主要靠手工“调参”;夸张一点说,参数调节上失之毫厘,学习结果可能谬以千里.

参见第 5 章.

二十世纪九十年代中期,“统计学习”(statistical learning)闪亮登场并迅速占据主流舞台,代表性技术是支持向量机(Support Vector Machine,简称 SVM)以及更一般的“核方法”(kernel methods).这方面的工作早在二十世纪六七十年代就已开始,统计学习理论[Vapnik, 1998]在那个时期也已打下了基础,例如 V. N. Vapnik 在 1963 年提出了“支持向量”概念,他和 A. J. Chervonenkis 在 1968 年提出 VC 维,在 1974 年提出了结构风险最小化原则等.但直到九十年代中期统计学习才开始成为机器学习的主流,一方面是由于有效的支持向量机算法在九十年代初才被提出,其优越性能到九十年代中期在文本分类应用中才得以显现;另一方面,正是在连接主义学习技术的局限性凸显之后,人们才把目光转向了以统计学习理论为直接支撑的统计学习技术.事实上,统计学习与连接主义学习有密切的联系.在支持向量机被普遍接受后,核技巧(kernel trick)被人们用到了机器学习的几乎每一个角落,核方法也逐渐成为机器学习的基本内容之一.

参见第 6 章.

参见习题 6.5.

有趣的是,二十一世纪初,连接主义学习又卷土重来,掀起了以“深度学

参见 5.6 节.

“过拟合” 参见第 2 章.

习” 为名的热潮. 所谓深度学习, 狹义地说就是“很多层”的神经网络. 在若干测试和竞赛上, 尤其是涉及语音、图像等复杂对象的应用中, 深度学习技术取得了优越性能. 以往机器学习技术在应用中要取得好性能, 对使用者的要求较高; 而深度学习技术涉及的模型复杂度非常高, 以至于只要下工夫“调参”, 把参数调节好, 性能往往就好. 因此, 深度学习虽缺乏严格的理论基础, 但它显著降低了机器学习应用者的门槛, 为机器学习技术走向工程实践带来了便利. 那么, 它为什么此时才热起来呢? 有两个基本原因: 数据大了、计算能力强了. 深度学习模型拥有大量参数, 若数据样本少, 则很容易“过拟合”; 如此复杂的模型、如此大的数据样本, 若缺乏强力计算设备, 根本无法求解. 恰由于人类进入了“大数据时代”, 数据储量与计算设备都有了大发展, 才使得连接主义学习技术焕发又一春. 有趣的是, 神经网络在二十世纪八十年代中期走红, 与当时 Intel x86 系列微处理器与内存条技术的广泛应用所造成的计算能力、数据访存效率比七十年代有显著提高不无关联. 深度学习此时的状况, 与彼时的神经网络何其相似.

需说明的是, 机器学习现在已经发展成为一个相当大的学科领域, 本节仅是管中窥豹, 很多重要技术都没有谈及, 耐心的读者在读完本书后会有更全面的了解.

1.6 应用现状

在过去二十年中, 人类收集、存储、传输、处理数据的能力取得了飞速提升, 人类社会的各个角落都积累了大量数据, 亟需能有效地对数据进行分析利用的计算机算法, 而机器学习恰顺应了大时代的这个迫切需求, 因此该学科领域很自然地取得巨大发展、受到广泛关注.

今天, 在计算机科学的诸多分支学科领域中, 无论是多媒体、图形学, 还是网络通信、软件工程, 乃至体系结构、芯片设计, 都能找到机器学习技术的身影, 尤其是在计算机视觉、自然语言处理等“计算机应用技术”领域, 机器学习已成为最重要的技术进步源泉之一.

机器学习还为许多交叉学科提供了重要的技术支撑. 例如, “生物信息学”试图利用信息技术来研究生命现象和规律, 而基因组计划的实施和基因药物的美好愿景让人们为之心潮澎湃. 生物信息学研究涉及从“生命现象”到“规律发现”的整个过程, 其间必然包括数据获取、数据管理、数据分析、仿真实验等环节, 而“数据分析”恰是机器学习技术的舞台, 各种机器学习技术已经在这个舞台上大放异彩.

事实上,随着科学研究的基本手段从传统的“理论+实验”走向现在的“理论+实验+计算”,乃至出现“数据科学”这样的提法,机器学习的重要性日趋显著,因为“计算”的目的往往是数据分析,而数据科学的核心也恰是通过分析数据来获得价值。若要列出目前计算机科学技术中最活跃、最受瞩目的研究分支,那么机器学习必居其中。2001年,美国NASA-JPL的科学家在*Science*杂志上专门撰文[Mjolsness and DeCoste, 2001]指出,机器学习对科学的研究的整个过程正起到越来越大的支撑作用,其进展对科技发展意义重大。2003年,DARPA启动PAL计划,将机器学习的重要性上升到美国国家安全的高度来考虑。众所周知,美国最尖端科技的研究通常是由NASA和DARPA推进的,而这两大机构不约而同地强调机器学习的重要性,其意义不言而喻。

NASA-JPL 的全称是美国航空航天局喷气推进实验室,著名的“勇气”号和“机遇”号火星机器人均是在这个实验室研制的。

DARPA 的全称是美国国防部先进研究计划局,互联网、全球卫星定位系统等都源于 DARPA 启动的研究项目。

机器学习提供数据分析能力,云计算提供数据处理能力,众包提供数据标记能力。

“数据挖掘”这个词很早就在统计学界出现并略带贬义,这是由于传统统计学研究往往醉心于理论的优美而忽视实际效用。但最近情况发生变化,越来越多的统计学家开始关注现实问题,进入机器学习和数据挖掘领域。

2006年,卡耐基梅隆大学宣告成立世界上第一个“机器学习系”,机器学习领域奠基人之一T. Mitchell教授出任首任系主任。2012年3月,美国奥巴马政府启动“大数据研究与发展计划”,美国国家科学基金会旋即在加州大学伯克利分校启动加强计划,强调要深入研究和整合大数据时代的三大关键技术:机器学习、云计算、众包(crowdsourcing)。显然,机器学习在大数据时代是必不可少的核心技术,道理很简单:收集、存储、传输、管理大数据的目的,是为了“利用”大数据,而如果没有机器学习技术分析数据,则“利用”无从谈起。

谈到对数据进行分析利用,很多人会想到“数据挖掘”(data mining),这里简单探讨一下数据挖掘与机器学习的联系。数据挖掘领域在二十世纪九十年代形成,它受到很多学科领域的影响,其中数据库、机器学习、统计学无疑影响最大[Zhou, 2003]。数据挖掘是从海量数据中发掘知识,这就必然涉及对“海量数据”的管理和分析。大体来说,数据库领域的研究为数据挖掘提供数据管理技术,而机器学习和统计学的研究为数据挖掘提供数据分析技术。由于统计学界的研究成果通常需要经由机器学习研究来形成有效的学习算法,之后再进入数据挖掘领域,因此从这个意义上说,统计学主要是通过机器学习对数据挖掘发挥影响,而机器学习领域和数据库领域则是数据挖掘的两大支撑。

今天,机器学习已经与普通人的生活密切相关。例如在天气预报、能源勘探、环境监测等方面,有效地利用机器学习技术对卫星和传感器发回的数据进行分析,是提高预报和检测准确性的重要途径;在商业营销中,有效地利用机器学习技术对销售数据、客户信息进行分析,不仅可帮助商家优化库存降低成本,还有助于针对用户群设计特殊营销策略;……下面再举几例:

众所周知,谷歌、百度等互联网搜索引擎已开始改变人类的生活方式,例如很多人已习惯于在出行前通过互联网搜索来了解目的地信息、寻找合适的

酒店、餐馆等。美国《新闻周刊》曾对谷歌有一句话评论：“它使任何人离任何问题的答案间的距离变得只有点击一下鼠标这么远。”显然，互联网搜索是通过分析网络上的数据来找到用户所需的信息，在这个过程中，用户查询是输入、搜索结果是输出，而要建立输入与输出之间的联系，内核必然需要机器学习技术。事实上，互联网搜索发展至今，机器学习技术的支撑居功至伟。到了今天，搜索的对象、内容日趋复杂，机器学习技术的影响更为明显，例如在进行“图片搜索”时，无论谷歌还是百度都在使用最新潮的机器学习技术。谷歌、百度、脸书、雅虎等公司纷纷成立专攻机器学习技术的研究团队，甚至直接以机器学习技术命名的研究院，充分体现出机器学习技术的发展和应用，甚至在一定程度上影响了互联网产业的走向。

再举一例。车祸是人类最凶险的杀手之一，全世界每年有上百万人丧生车轮，仅我国每年就有约十万人死于车祸。由计算机来实现自动驾驶是一个理想的方案，因为机器上路时可以确保不是新手驾驶、不会疲劳驾驶，更不会酒后驾驶，而且还有重要的军事用途。美国在二十世纪八十年代就开始进行这方面研究。这里最大的困难是无法在汽车厂里事先把汽车上路后所会遇到的所有情况都考虑到、设计出处理规则并加以编程实现，而只能根据上路时遇到的情况即时处理。若把车载传感器接收到的信息作为输入，把方向、刹车、油门的控制行为作为输出，则这里的关键问题恰可抽象为一个机器学习任务。2004年3月，在美国DARPA组织的自动驾驶车比赛中，斯坦福大学机器学习专家S. Thrun的小组研制的参赛车用6小时53分钟成功走完了132英里赛程获得冠军。比赛路段是在内华达州西南部的山区和沙漠中，路况相当复杂，在这样的路段上行车即使对经验丰富的人类司机来说也是一个挑战。S. Thrun后来到谷歌领导自动驾驶车项目团队。值得一提的是，自动驾驶车在近几年取得了飞跃式发展，除谷歌外，通用、奥迪、大众、宝马等传统汽车公司均投入巨资进行研发，目前已开始有产品进入市场。2011年6月，美国内华达州议会通过法案，成为美国第一个认可自动驾驶车的州，此后，夏威夷州和佛罗里达州也先后通过类似法案。自动驾驶汽车可望在不久的将来出现在普通人的生活中，而机器学习技术则起到了“司机”作用。

机器学习技术甚至已影响到人类社会政治生活。2012年美国大选期间，奥巴马麾下有一支机器学习团队，他们对各类选情数据进行分析，为奥巴马提示下一步竞选行动。例如他们使用机器学习技术分析社交网络数据，判断出在总统候选人第一次辩论之后哪些选民会倒戈，并根据分析的结果开发出个性化宣传策略，能为每位选民找出一个最有说服力的挽留理由；他们基于机器学习模

例如著名机器学习教科书 [Mitchell, 1997] 4.2 节介绍了二十世纪九十年代早期利用神经网络学习来控制自动驾驶车的 ALVINN 系统。

型的分析结果提示奥巴马应去何处开展拉票活动, 有些建议甚至让专业竞选顾问大吃一惊, 而结果表明去这些地方大有收获。总统选举需要大量金钱, 机器学习技术在这方面发挥了奇效。例如, 机器学习模型分析出, 某电影明星对某地区某年龄段的特定人群很有吸引力, 而这个群体很愿意出高价与该明星及奥巴马共进晚餐……果然, 这样一次筹资晚宴成功募集到 1500 万美元; 最终, 借助机器学习模型, 奥巴马筹到了创纪录的 10 亿美元竞选经费。机器学习技术不仅有助于竞选经费“开源”, 还可帮助“节流”, 例如机器学习模型通过对不同群体选民进行分析, 建议购买了一些冷门节目的广告时段, 而没有采用在昂贵的黄金时段购买广告的传统做法, 使得广告资金效率相比 2008 年竞选提高了 14%; ……胜选后, 《时代》周刊专门报道了这个被奥巴马称为“竞选核武器”、由半监督学习研究专家 R. Ghani 领导的团队。

值得一提的是, 机器学习备受瞩目当然是由于它已成为智能数据分析技术的创新源泉, 但机器学习研究还有另一个不可忽视的意义, 即通过建立一些关于学习的计算模型来促进我们理解“人类如何学习”。例如, P. Kanerva 在二十世纪八十年代中期提出 SDM (Sparse Distributed Memory) 模型 [Kanerva, 1988] 时并没有刻意模仿脑生理结构, 但后来神经科学的研究发现, SDM 的稀疏编码机制在视觉、听觉、嗅觉功能的脑皮层中广泛存在, 从而为理解脑的某些功能提供了一定的启发。自然科学研究的驱动力归结起来无外是人类对宇宙本源、万物本质、生命本性、自我本识的好奇, 而“人类如何学习”无疑是一个有关自我本识的重大问题。从这个意义上说, 机器学习不仅在信息科学中占有重要地位, 还具有一定的自然科学探索色彩。

1.7 阅读材料

[Mitchell, 1997] 是第一本机器学习专门性教材, [Duda et al., 2001; Alpaydin, 2004; Flach, 2012] 都是出色的入门读物。[Hastie et al., 2009] 是很好的进阶读物, [Bishop, 2006] 也很有参考价值, 尤其适合于贝叶斯学习偏好者。[Shalev-Shwartz and Ben-David, 2014] 则适合于理论偏好者。[Witten et al., 2011] 是基于 WEKA 撰写的入门读物, 有助于初学者通过 WEKA 实践快速掌握常用机器学习算法。

本书 1.5 和 1.6 节主要取材于 [周志华, 2007]。《机器学习: 一种人工智能途径》[Michalski et al., 1983] 汇集了 20 位学者撰写的 16 篇文章, 是机器学习早期最重要的文献。该书出版后产生了很大反响, Morgan Kaufmann 出版社后来分别于 1986 年和 1990 年出版了该书的续篇, 编为第二卷和第三卷。《人工

WEKA 是著名的免费机器学习算法程序库, 由新西兰 Waikato 大学研究人员基于 JAVA 开发:
<http://www.cs.waikato.ac.nz/ml/weka/>

智能手册》系列是图灵奖得主 E. A. Feigenbaum 与不同学者合作编写而成, 该书第三卷 [Cohen and Feigenbaum, 1983] 对机器学习进行了讨论, 是机器学习早期的重要文献. [Dietterich, 1997] 对机器学习领域的发展进行了评述和展望. 早期的很多文献在今天仍值得重视, 一些闪光的思想在相关技术进步后可能焕发新的活力, 例如近来流行的“迁移学习”(transfer learning) [Pan and Yang, 2010], 恰似“类比学习”(learning by analogy) 在统计学习技术大发展后的升级版; 红极一时的“深度学习”(deep learning) 在思想上并未显著超越二十世纪八十年代中后期神经网络学习的研究.

深度学习参见 5.6 节.

机器学习中关于概念学习的研究开始很早, 从中产生的不少思想对整个领域都有深远影响. 例如作为主流学习技术之一的决策树学习, 就起源于关于概念形成的树结构研究 [Hunt and Hovland, 1963]. [Winston, 1970] 在著名的“积木世界”研究中, 将概念学习与基于泛化和特化的搜索过程联系起来. [Simon and Lea, 1974] 较早提出了“学习”是在假设空间中搜索的观点. [Mitchell, 1977] 稍后提出了版本空间的概念. 概念学习中有很多关于规则学习的内容.

规则学习参见第 15 章.

奥卡姆剃刀原则主张选择与经验观察一致的最简单假设, 它在自然科学如物理学、天文学等领域中是一个广为沿用的基础性原则, 例如哥白尼坚持“日心说”的理由之一就是它比托勒密的“地心说”更简单且符合天文观测. 奥卡姆剃刀在机器学习领域也有很多追随者 [Blumer et al., 1996]. 但机器学习中什么是“更简单的”这个问题一直困扰着研究者们, 因此, 对奥卡姆剃刀在机器学习领域的作用一直存在着争议 [Webb, 1996; Domingos, 1999]. 需注意的是, 奥卡姆剃刀并非科学研究所唯一可行的假设选择原则, 例如古希腊哲学家伊壁鸠鲁(公元前341年–前270年)提出的“多释原则”(principle of multiple explanations), 主张保留与经验观察一致的所有假设 [Asmis, 1984], 这与集成学习(ensemble learning)方面的研究更加吻合.

集成学习参见第 8 章.

机器学习领域最重要的国际学术会议是国际机器学习会议(ICML)、国际神经信息处理系统会议(NIPS)和国际学习理论会议(COLT), 重要的区域性会议主要有欧洲机器学习会议(ECML)和亚洲机器学习会议(ACML); 最重要的国际学术期刊是 *Journal of Machine Learning Research* 和 *Machine Learning*. 人工智能领域的重要会议如 IJCAI、AAAI 以及重要期刊如 *Artificial Intelligence*、*Journal of Artificial Intelligence Research*, 数据挖掘领域的重要会议如 KDD、ICDM 以及重要期刊如 *ACM Transactions on Knowledge Discovery from Data*、*Data Mining and Knowledge Discovery*, 计算机视觉与模式识别

领域的重要会议如 CVPR 以及重要期刊如 *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 神经网络领域的重要期刊如 *Neural Computation*、*IEEE Transactions on Neural Networks and Learning Systems* 等也经常发表机器学习方面的论文。此外, 统计学领域的重要期刊如 *Annals of Statistics* 等也常有关于统计学习方面的理论文章发表。

国内不少书籍包含机器学习方面的内容, 例如 [陆汝钤, 1996]. [李航, 2012] 是以统计学习为主题的读物。国内机器学习领域最主要的活动是两年一次的中国机器学习大会(CCML)以及每年举行的“机器学习及其应用”研讨会(MLA); 很多学术刊物都经常刊登有关机器学习的论文。

习题

1.1 表 1.1 中若只包含编号为 1 和 4 的两个样例, 试给出相应的版本空间.

析合范式即多个合取式的析取.

1.2 与使用单个合取式来进行假设表示相比, 使用“析合范式”将使得假设空间具有更强的表示能力. 例如

$$\begin{aligned} \text{好瓜} \leftrightarrow & \left((\text{色泽} = *) \wedge (\text{根蒂} = \text{蜷缩}) \wedge (\text{敲声} = *) \right) \\ \vee & \left((\text{色泽} = \text{乌黑}) \wedge (\text{根蒂} = *) \wedge (\text{敲声} = \text{沉闷}) \right), \end{aligned}$$

会把“(色泽=青绿) \wedge (根蒂=蜷缩) \wedge (敲声=清脆)”以及“(色泽=乌黑) \wedge (根蒂=硬挺) \wedge (敲声=沉闷)”都分类为“好瓜”. 若使用最多包含 k 个合取式的析合范式来表达表 1.1 西瓜分类问题的假设空间, 试估算共有多少种可能的假设.

提示: 注意冗余情况,
如 $(A = a) \vee (A = *)$
与 $(A = *)$ 等价.

即不存在训练错误为 0 的假设.

1.3 若数据包含噪声, 则假设空间中有可能不存在与所有训练样本都一致的假设. 在此情形下, 试设计一种归纳偏好用于假设选择.

1.4* 本章 1.4 节在论述“没有免费的午餐”定理时, 默认使用了“分类错误率”作为性能度量来对分类器进行评估. 若换用其他性能度量 ℓ , 则式(1.1)将改为

$$E_{ote}(\mathcal{L}_a | X, f) = \sum_h \sum_{\mathbf{x} \in \mathcal{X} - X} P(\mathbf{x}) \ell(h(\mathbf{x}), f(\mathbf{x})) P(h | X, \mathcal{L}_a),$$

试证明“没有免费的午餐定理”仍成立.

1.5 试述机器学习能在互联网搜索的哪些环节起什么作用.

参考文献

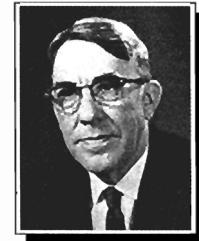
- 陆汝钤. (1996). 人工智能(下册). 科学出版社, 北京.
- 周志华. (2007). “机器学习与数据挖掘.” 中国计算机学会通讯, 3(12):35–44.
- 李航. (2012). 统计学习方法. 清华大学出版社, 北京.
- Alpaydin, E. (2004). *Introduction to Machine Learning*. MIT Press, Cambridge, MA.
- Asmis, E. (1984). *Epicurus' Scientific Method*. Cornell University Press, Ithaca, NY.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, New York, NY.
- Blumer, A., A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. (1996). “Occam's razor.” *Information Processing Letters*, 24(6):377–380.
- Carbonell, J. G., ed. (1990). *Machine Learning: Paradigms and Methods*. MIT Press, Cambridge, MA.
- Cohen, P. R. and E. A. Feigenbaum, eds. (1983). *The Handbook of Artificial Intelligence*, volume 3. William Kaufmann, New York, NY.
- Dietterich, T. G. (1997). “Machine learning research: Four current directions.” *AI Magazine*, 18(4):97–136.
- Domingos, P. (1999). “The role of Occam's razor in knowledge discovery.” *Data Mining and Knowledge Discovery*, 3(4):409–425.
- Duda, R. O., P. E. Hart, and D. G. Stork. (2001). *Pattern Classification*, 2nd edition. John Wiley & Sons, New York, NY.
- Flach, P. (2012). *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, Cambridge, UK.
- Hand, D., H. Mannila, and P. Smyth. (2001). *Principles of Data Mining*. MIT Press, Cambridge, MA.
- Hastie, T., R. Tibshirani, and J. Friedman. (2009). *The Elements of Statistical Learning*, 2nd edition. Springer, New York, NY.
- Hunt, E. G. and D. I. Hovland. (1963). “Programming a model of human concept formation.” In *Computers and Thought* (E. Feigenbaum and J. Feldman, eds.), 310–325, McGraw Hill, New York, NY.

- Kanerva, P. (1988). *Sparse Distributed Memory*. MIT Press, Cambridge, MA.
- Michalski, R. S., J. G. Carbonell, and T. M. Mitchell, eds. (1983). *Machine Learning: An Artificial Intelligence Approach*. Tioga, Palo Alto, CA.
- Mitchell, T. (1997). *Machine Learning*. McGraw Hill, New York, NY.
- Mitchell, T. M. (1977). "Version spaces: A candidate elimination approach to rule learning." In *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI)*, 305–310, Cambridge, MA.
- Mjolsness, E. and D. DeCoste. (2001). "Machine learning for science: State of the art and future prospects." *Science*, 293(5537):2051–2055.
- Pan, S. J. and Q. Yang. (2010). "A survey of transfer learning." *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.
- Shalev-Shwartz, S. and S. Ben-David. (2014). *Understanding Machine Learning*. Cambridge University Press, Cambridge, UK.
- Simon, H. A. and G. Lea. (1974). "Problem solving and rule induction: A unified view." In *Knowledge and Cognition* (L. W. Gregg, ed.), 105–127, Erlbaum, New York, NY.
- Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley, New York, NY.
- Webb, G. I. (1996). "Further experimental evidence against the utility of Occam's razor." *Journal of Artificial Intelligence Research*, 43:397–417.
- Winston, P. H. (1970). "Learning structural descriptions from examples." Technical Report AI-TR-231, AI Lab, MIT, Cambridge, MA.
- Witten, I. H., E. Frank, and M. A. Hall. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd edition. Elsevier, Burlington, MA.
- Wolpert, D. H. (1996). "The lack of a priori distinctions between learning algorithms." *Neural Computation*, 8(7):1341–1390.
- Wolpert, D. H. and W. G. Macready. (1995). "No free lunch theorems for search." Technical Report SFI-TR-05-010, Santa Fe Institute, Sante Fe, NM.
- Zhou, Z.-H. (2003). "Three perspectives of data mining." *Artificial Intelligence*, 143(1):139–146.

休息一会儿

小故事：“机器学习”名字的由来

1952年，阿瑟·萨缪尔 (Arthur Samuel, 1901—1990) 在IBM公司研制了一个西洋跳棋程序，这个程序具有自学习能力，通过对大量棋局的分析逐渐辨识出当前局面下的“好棋”和“坏棋”，从而不断提高弈棋水平，并很快就下赢了萨缪尔自己。1956年，萨缪尔应约翰·麦卡锡 (John McCarthy, “人工智能之父”，1971年图灵奖得主) 之邀，在标志着人工智能学科诞生的达特茅斯会议上介绍这项工作。萨缪尔发明了“机器学习”这个词，将其定义为“不显式编程地赋予计算机能力的研究领域”。他的文章“Some studies in machine learning using the game of checkers”1959年在 *IBM Journal* 正式发表后，爱德华·费根鲍姆 (Edward Feigenbaum, “知识工程之父”，1994年图灵奖得主) 为编写其巨著 *Computers and Thought*，在1961年邀请萨缪尔提供一个该程序最好的对弈实例。于是，萨缪尔借机向康涅狄格州的跳棋冠军、当时全美排名第四的棋手发起了挑战，结果萨缪尔程序获胜，在当时引起轰动。



这个跳棋程序实质上使用了强化学习技术，参见第16章。

事实上，萨缪尔跳棋程序不仅在人工智能领域产生了重大影响，还影响到整个计算机科学的发展。早期计算机科学研究认为，计算机不可能完成事先没有显式编程好的任务，而萨缪尔跳棋程序否证了这个假设。另外，这个程序是最早在计算机上执行非数值计算任务的程序之一，其逻辑指令设计思想极大地影响了IBM计算机的指令集，并很快被其他计算机的设计者采用。

第2章 模型评估与选择

2.1 经验误差与过拟合

精度常写为百分比形式
 $(1 - \frac{a}{m}) \times 100\%$.

这里所说的“误差”均指误差期望.

在后面的章节中将介绍不同的学习算法如何最小化经验误差.

过拟合亦称“过配”
欠拟合亦称“欠配”.

学习能力是否“过于强大”，是由学习算法和数据内涵共同决定的.

通常我们把分类错误的样本数占样本总数的比例称为“错误率”(error rate), 即如果在 m 个样本中有 a 个样本分类错误, 则错误率 $E = a/m$; 相应的, $1 - a/m$ 称为“精度”(accuracy), 即“精度 = 1 – 错误率”. 更一般地, 我们把学习器的实际预测输出与样本的真实输出之间的差异称为“误差”(error), 学习器在训练集上的误差称为“训练误差”(training error)或“经验误差”(empirical error), 在新样本上的误差称为“泛化误差”(generalization error). 显然, 我们希望得到泛化误差小的学习器. 然而, 我们事先并不知道新样本是什么样, 实际能做的是努力使经验误差最小化. 在很多情况下, 我们可以学得一个经验误差很小、在训练集上表现很好的学习器, 例如甚至对所有训练样本都分类正确, 即分类错误率为零, 分类精度为100%, 但这是不是我们想要的学习器呢? 遗憾的是, 这样的学习器在多数情况下都不好.

我们实际希望的, 是在新样本上能表现得很好的学习器. 为了达到这个目的, 应该从训练样本中尽可能学出适用于所有潜在样本的“普遍规律”, 这样才能在遇到新样本时做出正确的判别. 然而, 当学习器把训练样本学得“太好”了的时候, 很可能已经把训练样本自身的一些特点当作了所有潜在样本都会具有一般性质, 这样就会导致泛化性能下降. 这种现象在机器学习中称为“过拟合”(overfitting). 与“过拟合”相对的是“欠拟合”(underfitting), 这是指对训练样本的一般性质尚未学好. 图 2.1 给出了关于过拟合与欠拟合的一个便于直观理解的类比.

有多种因素可能导致过拟合, 其中最常见的情况是由于学习能力过于强大, 以至于把训练样本所包含的不太一般的特性都学到了, 而欠拟合则通常是由学习能力低下而造成的. 欠拟合比较容易克服, 例如在决策树学习中扩展分支、在神经网络学习中增加训练轮数等, 而过拟合则很麻烦. 在后面的学习中我们将看到, 过拟合是机器学习面临的关键障碍, 各类学习算法都必然带有一些针对过拟合的措施; 然而必须认识到, 过拟合是无法彻底避免的, 我们所能做的只是“缓解”, 或者说减小其风险. 关于这一点, 可大致这样理解: 机器学习面临的问题通常是 NP 难甚至更难, 而有效的学习算法必然是在多项式时间内

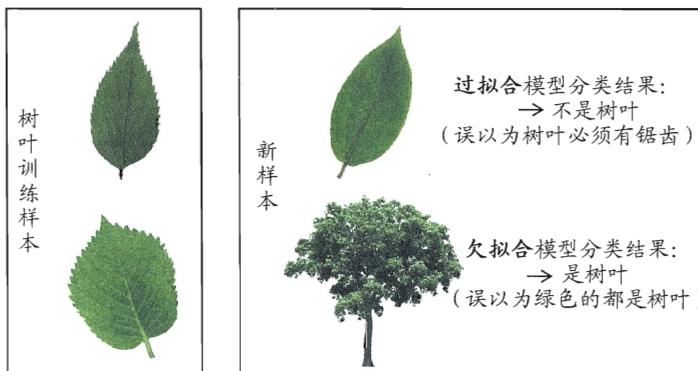


图 2.1 过拟合、欠拟合的直观类比

运行完成, 若可彻底避免过拟合, 则通过经验误差最小化就能获最优解, 这就意味着我们构造性地证明了 “ $P=NP$ ” ; 因此, 只要相信 “ $P \neq NP$ ”, 过拟合就不可避免.

在现实任务中, 我们往往有多种学习算法可供选择, 甚至对同一个学习算法, 当使用不同的参数配置时, 也会产生不同的模型. 那么, 我们该选用哪一个学习算法、使用哪一种参数配置呢? 这就是机器学习中的 “模型选择” (model selection) 问题. 理想的解决方案当然是对候选模型的泛化误差进行评估, 然后选择泛化误差最小的那个模型. 然而如上面所讨论的, 我们无法直接获得泛化误差, 而训练误差又由于过拟合现象的存在而不适合作为标准, 那么, 在现实中如何进行模型评估与选择呢?

2.2 评估方法

在现实任务中往往还会考虑时间开销、存储开销、可解释性等因素, 这里暂且只考虑泛化误差.

通常, 我们可通过实验测试来对学习器的泛化误差进行评估并进而做出选择. 为此, 需使用一个 “测试集” (testing set) 来测试学习器对新样本的判别能力, 然后以测试集上的 “测试误差” (testing error) 作为泛化误差的近似. 通常我们假设测试样本也是从样本真实分布中独立同分布采样而得. 但需注意的是, 测试集应该尽可能与训练集互斥, 即测试样本尽量不在训练集中出现、未在训练过程中使用过.

测试样本为什么要尽可能不出现在训练集中呢? 为理解这一点, 不妨考虑这样一个场景: 老师出了 10 道习题供同学们练习, 考试时老师又用同样的这 10 道题作为试题, 这个考试成绩能否有效反映出同学们学得好不好呢? 答案是否定的, 可能有的同学只会做这 10 道题却能得高分. 回到我们的问题上来, 我们

希望得到泛化性能强的模型, 好比是希望同学们对课程学得很好、获得了对所学知识“举一反三”的能力; 训练样本相当于给同学们练习的习题, 测试过程则相当于考试。显然, 若测试样本被用作训练了, 则得到的将是过于“乐观”的估计结果。

可是, 我们只有一个包含 m 个样例的数据集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$, 既要训练, 又要测试, 怎样才能做到呢? 答案是: 通过对 D 进行适当的处理, 从中产生出训练集 S 和测试集 T 。下面介绍几种常见的做法。

2.2.1 留出法

“留出法”(hold-out)直接将数据集 D 划分为两个互斥的集合, 其中一个集合作为训练集 S , 另一个作为测试集 T , 即 $D = S \cup T$, $S \cap T = \emptyset$ 。在 S 上训练出模型后, 用 T 来评估其测试误差, 作为对泛化误差的估计。

以二分类任务为例, 假定 D 包含 1000 个样本, 将其划分为 S 包含 700 个样本, T 包含 300 个样本, 用 S 进行训练后, 如果模型在 T 上有 90 个样本分类错误, 那么其错误率为 $(90/300) \times 100\% = 30\%$, 相应的, 精度为 $1 - 30\% = 70\%$ 。

需注意的是, 训练/测试集的划分要尽可能保持数据分布的一致性, 避免因数据划分过程引入额外的偏差而对最终结果产生影响, 例如在分类任务中至少要保持样本的类别比例相似。如果从采样(sampling)的角度来看待数据集的划分过程, 则保留类别比例的采样方式通常称为“分层采样”(stratified sampling)。例如通过对 D 进行分层采样而获得含 70% 样本的训练集 S 和含 30% 样本的测试集 T , 若 D 包含 500 个正例、500 个反例, 则分层采样得到的 S 应包含 350 个正例、350 个反例, 而 T 则包含 150 个正例和 150 个反例; 若 S 、 T 中样本类别比例差别很大, 则误差估计将由于训练/测试数据分布的差异而产生偏差。

参见习题 2.1.

同时可得估计结果的
标准差。

另一个需注意的问题是, 即便在给定训练/测试集的样本比例后, 仍存在多种划分方式对初始数据集 D 进行分割。例如在上面的例子中, 可以把 D 中的样本排序, 然后把前 350 个正例放到训练集中, 也可以把最后 350 个正例放到训练集中, ……这些不同的划分将导致不同的训练/测试集, 相应的, 模型评估的结果也会有差别。因此, 单次使用留出法得到的估计结果往往不够稳定可靠, 在使用留出法时, 一般要采用若干次随机划分、重复进行实验评估后取平均值作为留出法的评估结果。例如进行 100 次随机划分, 每次产生一个训练/测试集用于实验评估, 100 次后就得到 100 个结果, 而留出法返回的则是这 100 个结果的平均。

此外, 我们希望评估的是用 D 训练出的模型的性能, 但留出法需划分训

可从“偏差-方差”(参见2.6节)的角度来理解: 测试集小时, 评估结果的方差较大; 训练集小时, 评估结果的偏差较大.

一般而言, 测试集至少应含30个样例 [Mitchell, 1997].

亦称“ k 倍交叉验证”.

训练/测试集, 这就会导致一个窘境: 若令训练集 S 包含绝大多数样本, 则训练出的模型可能更接近于用 D 训练出的模型, 但由于 T 比较小, 评估结果可能不够稳定准确; 若令测试集 T 多包含一些样本, 则训练集 S 与 D 差别更大了, 被评估的模型与用 D 训练出的模型相比可能有较大差别, 从而降低了评估结果的保真性(fidelity). 这个问题没有完美的解决方案, 常见做法是将大约 $2/3 \sim 4/5$ 的样本用于训练, 剩余样本用于测试.

2.2.2 交叉验证法

“交叉验证法”(cross validation)先将数据集 D 划分为 k 个大小相似的互斥子集, 即 $D = D_1 \cup D_2 \cup \dots \cup D_k$, $D_i \cap D_j = \emptyset$ ($i \neq j$). 每个子集 D_i 都尽可能保持数据分布的一致性, 即从 D 中通过分层采样得到. 然后, 每次用 $k - 1$ 个子集的并集作为训练集, 余下的那个子集作为测试集; 这样就可获得 k 组训练/测试集, 从而可进行 k 次训练和测试, 最终返回的是这 k 个测试结果的均值. 显然, 交叉验证法评估结果的稳定性和保真性在很大程度上取决于 k 的取值, 为强调这一点, 通常把交叉验证法称为“ k 折交叉验证”(k -fold cross validation). k 最常用的取值是 10, 此时称为 10 折交叉验证; 其他常用的 k 值有 5、20 等. 图 2.2 给出了 10 折交叉验证的示意图.

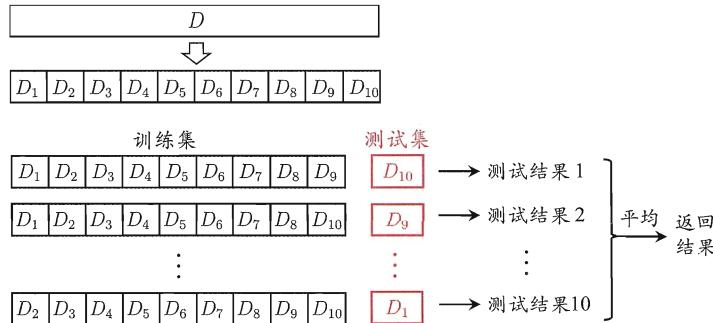


图 2.2 10 折交叉验证示意图

与留出法相似, 将数据集 D 划分为 k 个子集同样存在多种划分方式. 为减小因样本划分不同而引入的差别, k 折交叉验证通常要随机使用不同的划分重复 p 次, 最终的评估结果是这 p 次 k 折交叉验证结果的均值, 例如常见的有“10 次 10 折交叉验证”.

假定数据集 D 中包含 m 个样本, 若令 $k = m$, 则得到了交叉验证法的一个特例: 留一法(Leave-One-Out, 简称 LOO). 显然, 留一法不受随机样本划分

“10 次 10 折交叉验证法”与“100 次留出法”都是进行了 100 次训练/测试.

参见习题 2.2.

NFL 定理参见 1.4 节。

关于样本复杂度与泛化性能之间的关系，参见第 12 章。

Bootstrap 本意是“解靴带”；这里是在使用德国 18 世纪文学作品《吹牛大王历险记》中解靴带自助的典故，因此本书译为“自助法”。自助采样亦称“可重复采样”或“有放回采样”。

e 是自然常数。

\ 表示集合减法。

集成学习参见第 8 章。

方式的影响，因为 m 个样本只有唯一的方式划分为 m 个子集——每个子集包含一个样本；留一法使用的训练集与初始数据集相比只少了一个样本，这就使得在绝大多数情况下，留一法中被实际评估的模型与期望评估的用 D 训练出的模型很相似。因此，留一法的评估结果往往被认为比较准确。然而，留一法也有其缺陷：在数据集比较大时，训练 m 个模型的计算开销可能是难以忍受的（例如数据集包含 1 百万个样本，则需训练 1 百万个模型），而这还是在未考虑算法调参的情况下。另外，留一法的估计结果也未必永远比其他评估方法准确；“没有免费的午餐”定理对实验评估方法同样适用。

2.2.3 自助法

我们希望评估的是用 D 训练出的模型。但在留出法和交叉验证法中，由于保留了一部分样本用于测试，因此实际评估的模型所使用的训练集比 D 小，这必然会引入一些因训练样本规模不同而导致的估计偏差。留一法受训练样本规模变化的影响较小，但计算复杂度又太高了。有没有什么办法可以减少训练样本规模不同造成的影响，同时还能比较高效地进行实验估计呢？

“自助法”（bootstrapping）是一个比较好的解决方案，它直接以自助采样法（bootstrap sampling）为基础 [Efron and Tibshirani, 1993]。给定包含 m 个样本的数据集 D ，我们对它进行采样产生数据集 D' ：每次随机从 D 中挑选一个样本，将其拷贝放入 D' ，然后再将该样本放回初始数据集 D 中，使得该样本在下次采样时仍有可能被采到；这个过程重复执行 m 次后，我们就得到了包含 m 个样本的数据集 D' ，这就是自助采样的结果。显然， D 中有一部分样本会在 D' 中多次出现，而另一部分样本不出现。可以做一个简单的估计，样本在 m 次采样中始终不被采到的概率是 $(1 - \frac{1}{m})^m$ ，取极限得到

$$\lim_{m \rightarrow \infty} \left(1 - \frac{1}{m}\right)^m \mapsto \frac{1}{e} \approx 0.368, \quad (2.1)$$

即通过自助采样，初始数据集 D 中约有 36.8% 的样本未出现在采样数据集 D' 中。于是我们可将 D' 用作训练集， $D \setminus D'$ 用作测试集；这样，实际评估的模型与期望评估的模型都使用 m 个训练样本，而我们仍有数据总量约 $1/3$ 的、没在训练集中出现的样本用于测试。这样的测试结果，亦称“包外估计”（out-of-bag estimate）。

自助法在数据集较小、难以有效划分训练/测试集时很有用；此外，自助法能从初始数据集中产生多个不同的训练集，这对集成学习等方法有很大的好处。然而，自助法产生的数据集改变了初始数据集的分布，这会引入估计偏差。因

此, 在初始数据量足够时, 留出法和交叉验证法更常用一些.

2.2.4 调参与最终模型

大多数学习算法都有些参数(parameter)需要设定, 参数配置不同, 学得模型的性能往往有显著差别. 因此, 在进行模型评估与选择时, 除了要对适用学习算法进行选择, 还需对算法参数进行设定, 这就是通常所说的“参数调节”或简称“调参”(parameter tuning).

读者可能马上想到, 调参和算法选择没什么本质区别: 对每种参数配置都训练出模型, 然后把对应最好模型的参数作为结果. 这样的考虑基本是正确的, 但有一点需注意: 学习算法的很多参数是在实数范围内取值, 因此, 对每种参数配置都训练出模型来是不可行的. 现实中常用的做法, 是对每个参数选定一个范围和变化步长, 例如在 $[0, 0.2]$ 范围内以 0.05 为步长, 则实际要评估的候选参数值有 5 个, 最终是从这 5 个候选值中产生选定值. 显然, 这样选定的参数值往往不是“最佳”值, 但这是在计算开销和性能估计之间进行折中的结果, 通过这个折中, 学习过程才变得可行. 事实上, 即便在进行这样的折中后, 调参往往仍很困难. 可以简单估算一下: 假定算法有 3 个参数, 每个参数仅考虑 5 个候选值, 这样对每一组训练/测试集就有 $5^3 = 125$ 个模型需考察; 很多强大的学习算法有大量参数需设定, 这将导致极大的调参工程量, 以至于在不少应用任务中, 参数调得好不好往往对最终模型性能有关键性影响.

例如大型“深度学习”模型甚至有上百亿个参数.

给定包含 m 个样本的数据集 D , 在模型评估与选择过程中由于需要留出一部分数据进行评估测试, 事实上我们只使用了一部分数据训练模型. 因此, 在模型选择完成后, 学习算法和参数配置已选定, 此时应该用数据集 D 重新训练模型. 这个模型在训练过程中使用了所有 m 个样本, 这才是我们最终提交给用户的模型.

另外, 需注意的是, 我们通常把学得模型在实际使用中遇到的数据称为测试数据, 为了加以区分, 模型评估与选择中用于评估测试的数据集常称为“验证集”(validation set). 例如, 在研究对比不同算法的泛化性能时, 我们用测试集上的判别效果来估计模型在实际使用时的泛化能力, 而把训练数据另外划分为训练集和验证集, 基于验证集上的性能来进行模型选择和调参.

2.3 性能度量

对学习器的泛化性能进行评估, 不仅需要有效可行的实验估计方法, 还需要有衡量模型泛化能力的评价标准, 这就是性能度量(performance measure).

性能度量反映了任务需求, 在对比不同模型的能力时, 使用不同的性能度量往往会导致不同的评判结果; 这意味着模型的“好坏”是相对的, 什么样的模型是好的, 不仅取决于算法和数据, 还决定于任务需求.

聚类的性能度量参见第9章.

在预测任务中, 给定样例集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$, 其中 y_i 是示例 \mathbf{x}_i 的真实标记. 要评估学习器 f 的性能, 就要把学习器预测结果 $f(\mathbf{x})$ 与真实标记 y 进行比较.

回归任务最常用的性能度量是“均方误差”(mean squared error)

$$E(f; D) = \frac{1}{m} \sum_{i=1}^m (f(\mathbf{x}_i) - y_i)^2. \quad (2.2)$$

更一般的, 对于数据分布 \mathcal{D} 和概率密度函数 $p(\cdot)$, 均方误差可描述为

$$E(f; \mathcal{D}) = \int_{\mathbf{x} \sim \mathcal{D}} (f(\mathbf{x}) - y)^2 p(\mathbf{x}) d\mathbf{x}. \quad (2.3)$$

本节下面主要介绍分类任务中常用的性能度量.

2.3.1 错误率与精度

本章开头提到了错误率和精度, 这是分类任务中最常用的两种性能度量, 既适用于二分类任务, 也适用于多分类任务. 错误率是分类错误的样本数占样本总数的比例, 精度则是分类正确的样本数占样本总数的比例. 对样例集 D , 分类错误率定义为

$$E(f; D) = \frac{1}{m} \sum_{i=1}^m \mathbb{I}(f(\mathbf{x}_i) \neq y_i). \quad (2.4)$$

精度则定义为

$$\begin{aligned} \text{acc}(f; D) &= \frac{1}{m} \sum_{i=1}^m \mathbb{I}(f(\mathbf{x}_i) = y_i) \\ &= 1 - E(f; D). \end{aligned} \quad (2.5)$$

更一般的, 对于数据分布 \mathcal{D} 和概率密度函数 $p(\cdot)$, 错误率与精度可分别描述为

$$E(f; \mathcal{D}) = \int_{\mathbf{x} \sim \mathcal{D}} \mathbb{I}(f(\mathbf{x}) \neq y) p(\mathbf{x}) d\mathbf{x}, \quad (2.6)$$

$$\begin{aligned}\text{acc}(f; \mathcal{D}) &= \int_{\mathbf{x} \sim \mathcal{D}} \mathbb{I}(f(\mathbf{x}) = y) p(\mathbf{x}) d\mathbf{x} \\ &= 1 - E(f; \mathcal{D}).\end{aligned}\quad (2.7)$$

2.3.2 查准率、查全率与F1

错误率和精度虽常用, 但并不能满足所有任务需求。以西瓜问题为例, 假定瓜农拉来一车西瓜, 我们用训练好的模型对这些西瓜进行判别, 显然, 错误率衡量了有多少比例的瓜被判别错误。但是若我们关心的是“挑出的西瓜中有多少比例是好瓜”, 或者“所有好瓜中有多少比例被挑了出来”, 那么错误率显然就不够用了, 这时需要使用其他的性能度量。

类似的需求在信息检索、Web搜索等应用中经常出现, 例如在信息检索中, 我们经常会关心“检索出的信息中有多少比例是用户感兴趣的”“用户感兴趣的信息中有多少被检索出来了”。“查准率”(precision)与“查全率”(recall)是更为适用于此类需求的性能度量。

查准率亦称“准确率”
查全率亦称“召回率”

对于二分类问题, 可将样例根据其真实类别与学习器预测类别的组合划分为真正例(true positive)、假正例(false positive)、真反例(true negative)、假反例(false negative)四种情形, 令 TP 、 FP 、 TN 、 FN 分别表示其对应的样例数, 则显然有 $TP + FP + TN + FN =$ 样例总数。分类结果的“混淆矩阵”(confusion matrix)如表 2.1 所示。

表 2.1 分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

查准率 P 与查全率 R 分别定义为

$$P = \frac{TP}{TP + FP}, \quad (2.8)$$

$$R = \frac{TP}{TP + FN}. \quad (2.9)$$

查准率和查全率是一对矛盾的度量。一般来说, 查准率高时, 查全率往往偏低; 而查全率高时, 查准率往往偏低。例如, 若希望将好瓜尽可能多地选出来, 则可通过增加选瓜的数量来实现, 如果将所有西瓜都选上, 那么所有的好瓜也

必然都被选上了, 但这样查准率就会较低; 若希望选出的瓜中好瓜比例尽可能高, 则可只挑选最有把握的瓜, 但这样就难免会漏掉不少好瓜, 使得查全率较低. 通常只有在一些简单任务中, 才可能使查全率和查准率都很高.

在很多情形下, 我们可根据学习器的预测结果对样例进行排序, 排在前面的是学习器认为“最可能”是正例的样本, 排在最后的是学习器认为“最不可能”是正例的样本. 按此顺序逐个把样本作为正例进行预测, 则每次可以计算出当前的查全率、查准率. 以查准率为纵轴、查全率为横轴作图, 就得到了查准率-查全率曲线, 简称“P-R曲线”, 显示该曲线的图称为“P-R图”. 图 2.3 给出了一个示意图.

以信息检索应用为例, 逐条向用户反馈其可能感兴趣的信息, 即可计算出查全率、查准率.

亦称“PR 曲线”或“PR 图”.

为绘图方便和美观, 示意图显示出单调平滑曲线; 但现实任务中的 P-R 曲线常是非单调、不平滑的, 在很多局部有上下波动.

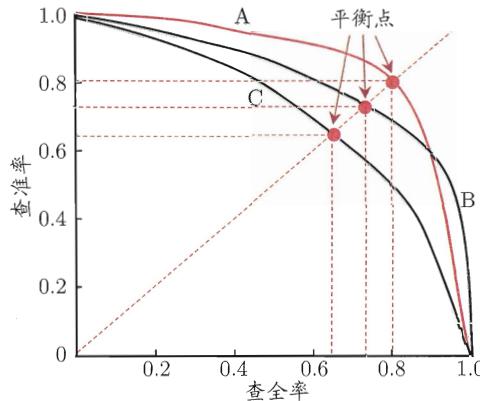


图 2.3 P-R 曲线与平衡点示意图

P-R 图直观地显示出学习器在样本总体上的查全率、查准率. 在进行比较时, 若一个学习器的 P-R 曲线被另一个学习器的曲线完全“包住”, 则可断言后者的性能优于前者, 例如图 2.3 中学习器 A 的性能优于学习器 C; 如果两个学习器的 P-R 曲线发生了交叉, 例如图 2.3 中的 A 与 B, 则难以一般性地断言两者孰优孰劣, 只能在具体的查准率或查全率条件下进行比较. 然而, 在很多情形下, 人们往往仍希望把学习器 A 与 B 比出个高低. 这时一个比较合理的判据是比较 P-R 曲线下面积的大小, 它在一定程度上表征了学习器在查准率和查全率上取得相对“双高”的比例. 但这个值不太容易估算, 因此, 人们设计了一些综合考虑查准率、查全率的性能度量.

“平衡点”(Break-Event Point, 简称 BEP)就是这样一个度量, 它是“查准率=查全率”时的取值, 例如图 2.3 中学习器 C 的 BEP 是 0.64, 而基于 BEP 的比较, 可认为学习器 A 优于 B.

但 BEP 还是过于简化了些, 更常用的是 $F1$ 度量:

$$F1 = \frac{2 \times P \times R}{P + R} = \frac{2 \times TP}{\text{样例总数} + TP - TN} . \quad (2.10)$$

$F1$ 是基于查准率与查全率的调和平均(harmonic mean)定义的:

$$\frac{1}{F1} = \frac{1}{2} \cdot \left(\frac{1}{P} + \frac{1}{R} \right) .$$

F_β 则是加权调和平均:

$$\frac{1}{F_\beta} = \frac{1}{1 + \beta^2} \cdot \left(\frac{1}{P} + \frac{\beta^2}{R} \right) .$$

与算术平均($\frac{P+R}{2}$)和几何平均($\sqrt{P \times R}$)相比, 调和平均更重视较小值.

在一些应用中, 对查准率和查全率的重视程度有所不同. 例如在商品推荐系统中, 为了尽可能少打扰用户, 更希望推荐内容确是用户感兴趣的, 此时查准率更重要; 而在逃犯信息检索系统中, 更希望尽可能少漏掉逃犯, 此时查全率更重要. $F1$ 度量的一般形式—— F_β , 能让我们表达出对查准率/查全率的不同偏好, 它定义为

$$F_\beta = \frac{(1 + \beta^2) \times P \times R}{(\beta^2 \times P) + R} , \quad (2.11)$$

其中 $\beta > 0$ 度量了查全率对查准率的相对重要性 [Van Rijsbergen, 1979]. $\beta = 1$ 时退化为标准的 $F1$; $\beta > 1$ 时查全率有更大影响; $\beta < 1$ 时查准率有更大影响.

很多时候我们有多个二分类混淆矩阵, 例如进行多次训练/测试, 每次得到一个混淆矩阵; 或是在多个数据集上进行训练/测试, 希望估计算法的“全局”性能; 甚或是执行多分类任务, 每两两类别的组合都对应一个混淆矩阵; …… 总之, 我们希望在 n 个二分类混淆矩阵上综合考察查准率和查全率.

一种直接的做法是先在各混淆矩阵上分别计算出查准率和查全率, 记为 $(P_1, R_1), (P_2, R_2), \dots, (P_n, R_n)$, 再计算平均值, 这样就得到“宏查准率”(macro- P)、“宏查全率”(macro- R), 以及相应的“宏 $F1$ ”(macro- $F1$):

$$\text{macro-}P = \frac{1}{n} \sum_{i=1}^n P_i , \quad (2.12)$$

$$\text{macro-}R = \frac{1}{n} \sum_{i=1}^n R_i , \quad (2.13)$$

$$\text{macro-}F1 = \frac{2 \times \text{macro-}P \times \text{macro-}R}{\text{macro-}P + \text{macro-}R} . \quad (2.14)$$

还可先将各混淆矩阵的对应元素进行平均, 得到 TP 、 FP 、 TN 、 FN 的平均值, 分别记为 \overline{TP} 、 \overline{FP} 、 \overline{TN} 、 \overline{FN} , 再基于这些平均值计算出“微查准率”(micro- P)、“微查全率”(micro- R)和“微 $F1$ ”(micro- $F1$):

$$\text{micro-}P = \frac{\overline{TP}}{\overline{TP} + \overline{FP}} , \quad (2.15)$$

$$\text{micro-}R = \frac{\overline{TP}}{\overline{TP} + \overline{FN}}, \quad (2.16)$$

$$\text{micro-}F1 = \frac{2 \times \text{micro-}P \times \text{micro-}R}{\text{micro-}P + \text{micro-}R}. \quad (2.17)$$

2.3.3 ROC 与 AUC

神经网络参见第 5 章。

很多学习器是为测试样本产生一个实值或概率预测，然后将这个预测值与一个分类阈值(threshold)进行比较，若大于阈值则分为正类，否则为反类。例如，神经网络在一般情形下是对每个测试样本预测出一个 [0.0, 1.0] 之间的实值，然后将这个值与 0.5 进行比较，大于 0.5 则判为正例，否则为反例。这个实值或概率预测结果的好坏，直接决定了学习器的泛化能力。实际上，根据这个实值或概率预测结果，我们可将测试样本进行排序，“最可能”是正例的排在最前面，“最不可能”是正例的排在最后面。这样，分类过程就相当于在这个排序中以某个“截断点”(cut point)将样本分为两部分，前一部分判作正例，后一部分则判作反例。

在不同的应用任务中，我们可根据任务需求来采用不同的截断点，例如若我们更重视“查准率”，则可选择排序中靠前的位置进行截断；若更重视“查全率”，则可选择靠后的位置进行截断。因此，排序本身的质量好坏，体现了综合考虑学习器在不同任务下的“期望泛化性能”的好坏，或者说，“一般情况下”泛化性能的好坏。ROC 曲线则是从这个角度出发来研究学习器泛化性能的有力工具。

ROC 全称是“受试者工作特征”(Receiver Operating Characteristic)曲线，它源于“二战”中用于敌机检测的雷达信号分析技术，二十世纪六七十年代开始被用于一些心理学、医学检测应用中，此后被引入机器学习领域 [Spackman, 1989]。与 2.3.2 节中介绍的 P-R 曲线相似，我们根据学习器的预测结果对样例进行排序，按此顺序逐个把样本作为正例进行预测，每次计算出两个重要量的值，分别以它们为横、纵坐标作图，就得到了“ROC 曲线”。与 P-R 曲线使用查准率、查全率为纵、横轴不同，ROC 曲线的纵轴是“真正例率”(True Positive Rate, 简称 TPR)，横轴是“假正例率”(False Positive Rate, 简称 FPR)，基于表 2.1 中的符号，两者分别定义为

$$\text{TPR} = \frac{TP}{TP + FN}, \quad (2.18)$$

$$\text{FPR} = \frac{FP}{TN + FP}. \quad (2.19)$$

显示 ROC 曲线的图称为“ROC 图”. 图 2.4(a)给出了一个示意图, 显然, 对角线对应于“随机猜测”模型, 而点 $(0, 1)$ 则对应于将所有正例排在所有反例之前的“理想模型”.

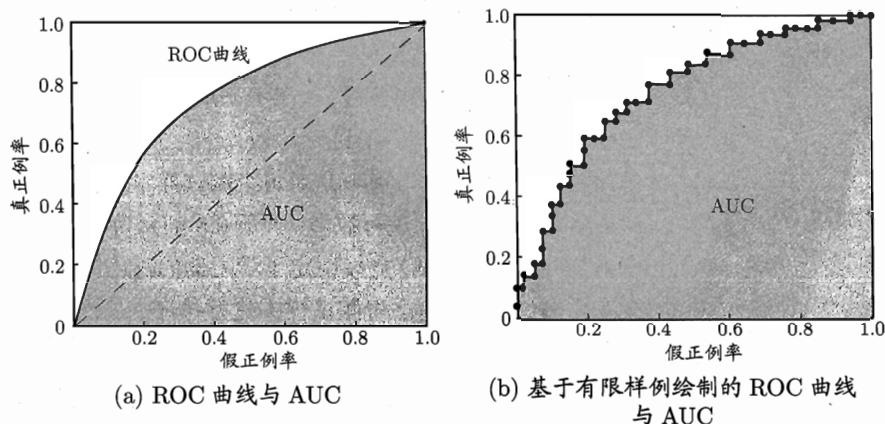


图 2.4 ROC 曲线与 AUC 示意图

基于有限个测试样例绘制 P-R 图时有同样问题. 本书到这里才介绍近似曲线的绘制, 是为了便于下面介绍 AUC 的计算.

现实任务中通常是利用有限个测试样例来绘制 ROC 图, 此时仅能获得有限个(真正例率, 假正例率)坐标对, 无法产生图 2.4(a)中的光滑 ROC 曲线, 只能绘制出如图 2.4(b)所示的近似 ROC 曲线. 绘图过程很简单: 给定 m^+ 个正例和 m^- 个反例, 根据学习器预测结果对样例进行排序, 然后把分类阈值设为最大, 即把所有样例均预测为反例, 此时真正例率和假正例率均为 0, 在坐标 $(0, 0)$ 处标记一个点. 然后, 将分类阈值依次设为每个样例的预测值, 即依次将每个样例划分为正例. 设前一个标记点坐标为 (x, y) , 当前若为真正例, 则对应标记点的坐标为 $(x, y + \frac{1}{m^+})$; 当前若为假正例, 则对应标记点的坐标为 $(x + \frac{1}{m^-}, y)$, 然后用线段连接相邻点即得.

进行学习器的比较时, 与 P-R 图相似, 若一个学习器的 ROC 曲线被另一个学习器的曲线完全“包住”, 则可断言后者的性能优于前者; 若两个学习器的 ROC 曲线发生交叉, 则难以一般性地断言两者孰优孰劣. 此时如果一定要进行比较, 则较为合理的判据是比较 ROC 曲线下的面积, 即 AUC (Area Under ROC Curve), 如图 2.4 所示.

从定义可知, AUC 可通过对 ROC 曲线下各部分的面积求和而得. 假定 ROC 曲线是由坐标为 $\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ 的点按序连接而形成 $(x_1 = 0, x_m = 1)$, 参见图 2.4(b), 则 AUC 可估算为

$$\text{AUC} = \frac{1}{2} \sum_{i=1}^{m-1} (x_{i+1} - x_i) \cdot (y_i + y_{i+1}) . \quad (2.20)$$

形式化地看, AUC 考虑的是样本预测的排序质量, 因此它与排序误差有紧密联系. 给定 m^+ 个正例和 m^- 个反例, 令 D^+ 和 D^- 分别表示正、反例集合, 则排序“损失”(loss)定义为

$$\ell_{rank} = \frac{1}{m^+ m^-} \sum_{\mathbf{x}^+ \in D^+} \sum_{\mathbf{x}^- \in D^-} \left(\mathbb{I}(f(\mathbf{x}^+) < f(\mathbf{x}^-)) + \frac{1}{2} \mathbb{I}(f(\mathbf{x}^+) = f(\mathbf{x}^-)) \right) , \quad (2.21)$$

即考虑每一对正、反例, 若正例的预测值小于反例, 则记一个“罚分”, 若相等, 则记 0.5 个“罚分”. 容易看出, ℓ_{rank} 对应的是 ROC 曲线之上的面积: 若一个正例在 ROC 曲线上对应标记点的坐标为 (x, y) , 则 x 恰是排序在其之前的反例所占的比例, 即假正例率. 因此有

$$\text{AUC} = 1 - \ell_{rank} . \quad (2.22)$$

2.3.4 代价敏感错误率与代价曲线

在现实任务中常会遇到这样的情况: 不同类型的错误所造成的后果不同. 例如在医疗诊断中, 错误地把患者诊断为健康人与错误地把健康人诊断为患者, 看起来都是犯了“一次错误”, 但后者的影响是增加了进一步检查的麻烦, 前者的后果却可能是丧失了拯救生命的最佳时机; 再如, 门禁系统错误地把可通行人员拦在门外, 将使得用户体验不佳, 但错误地把陌生人放进门内, 则会造成严重的安全事故. 为权衡不同类型错误所造成的不同损失, 可为错误赋予“非均等代价”(unequal cost).

以二分类任务为例, 我们可根据任务的领域知识设定一个“代价矩阵”(cost matrix), 如表 2.2 所示, 其中 $cost_{ij}$ 表示将第 i 类样本预测为第 j 类样本的代价. 一般来说, $cost_{ii} = 0$; 若将第 0 类判别为第 1 类所造成的损失更大, 则 $cost_{01} > cost_{10}$; 损失程度相差越大, $cost_{01}$ 与 $cost_{10}$ 值的差别越大.

一般情况下, 重要的是代价比值而非绝对值, 例如 $cost_{01} : cost_{10} = 5 : 1$ 与 $50 : 10$ 所起效果相当.

表 2.2 二分类代价矩阵

真实类别	预测类别	
	第 0 类	第 1 类
第 0 类	0	$cost_{01}$
第 1 类	$cost_{10}$	0

回顾前面介绍的一些性能度量可看出, 它们大都隐式地假设了均等代价, 例如式(2.4)所定义的错误率是直接计算“错误次数”, 并没有考虑不同错误会造成不同的后果. 在非均等代价下, 我们所希望的不再是简单地最小化错误次数, 而是希望最小化“总体代价”(total cost). 若将表 2.2 中的第 0 类作为正类、第 1 类作为反类, 令 D^+ 与 D^- 分别代表样例集 D 的正例子集和反例子集, 则“代价敏感”(cost-sensitive)错误率为

$$\begin{aligned} E(f; D; \text{cost}) = & \frac{1}{m} \left(\sum_{\mathbf{x}_i \in D^+} \mathbb{I}(f(\mathbf{x}_i) \neq y_i) \times \text{cost}_{01} \right. \\ & \left. + \sum_{\mathbf{x}_i \in D^-} \mathbb{I}(f(\mathbf{x}_i) \neq y_i) \times \text{cost}_{10} \right). \end{aligned} \quad (2.23)$$

类似的, 可给出基于分布定义的代价敏感错误率, 以及其他一些性能度量如精度的代价敏感版本. 若令 cost_{ij} 中的 i, j 取值不限于 0、1, 则可定义出多分类任务的代价敏感性能度量.

在非均等代价下, ROC 曲线不能直接反映出学习器的期望总体代价, 而“代价曲线”(cost curve)则可达到该目的. 代价曲线图的横轴是取值为 $[0, 1]$ 的正例概率代价

$$P(+)\text{cost} = \frac{p \times \text{cost}_{01}}{p \times \text{cost}_{01} + (1-p) \times \text{cost}_{10}}, \quad (2.24)$$

“规范化”(normalization)是将不同变化范围的值映射到相同的固定范围内, 常见的是 $[0, 1]$, 此时亦称“归一化”. 参见习题 2.8.

其中 p 是样例为正例的概率; 纵轴是取值为 $[0, 1]$ 的归一化代价

$$\text{cost}_{norm} = \frac{\text{FNR} \times p \times \text{cost}_{01} + \text{FPR} \times (1-p) \times \text{cost}_{10}}{p \times \text{cost}_{01} + (1-p) \times \text{cost}_{10}}, \quad (2.25)$$

其中 FPR 是式(2.19)定义的假正例率, $\text{FNR} = 1 - \text{TPR}$ 是假反例率. 代价曲线的绘制很简单: ROC 曲线上每一点对应了代价平面上的一条线段, 设 ROC 曲线上点的坐标为 (TPR, FPR) , 则可相应计算出 FNR , 然后在代价平面上绘制一条从 $(0, \text{FPR})$ 到 $(1, \text{FNR})$ 的线段, 线段下的面积即表示了该条件下的期望总体代价; 如此将 ROC 曲线上的每个点转化为代价平面上的一条线段, 然后取所有线段的下界, 围成的面积即为在所有条件下学习器的期望总体代价, 如图 2.5 所示.

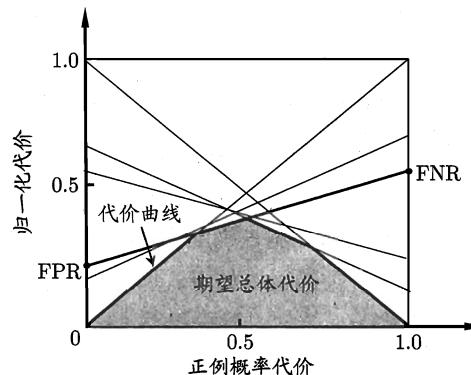


图 2.5 代价曲线与期望总体代价

2.4 比较检验

有了实验评估方法和性能度量, 看起来就能对学习器的性能进行评估比较了: 先使用某种实验评估方法测得学习器的某个性能度量结果, 然后对这些结果进行比较. 但怎么来做这个“比较”呢? 是直接取得性能度量的值然后“比大小”吗? 实际上, 机器学习中性能比较这件事要比大家想象的复杂得多. 这里面涉及几个重要因素: 首先, 我们希望比较的是泛化性能, 然而通过实验评估方法我们获得的是测试集上的性能, 两者的对比结果可能未必相同; 第二, 测试集上的性能与测试集本身的选择有很大关系, 且不论使用不同大小的测试集会得到不同的结果, 即便用相同大小的测试集, 若包含的测试样例不同, 测试结果也会有不同; 第三, 很多机器学习算法本身有一定的随机性, 即便用相同的参数设置在同一个测试集上多次运行, 其结果也会有不同. 那么, 有没有适当的方法对学习器的性能进行比较呢?

统计假设检验(hypothesis test)为我们进行学习器性能比较提供了重要依据. 基于假设检验结果我们可推断出, 若在测试集上观察到学习器 A 比 B 好, 则 A 的泛化性能是否在统计意义上优于 B, 以及这个结论的把握有多大. 下面我们先介绍两种最基本的假设检验, 然后介绍几种常用的机器学习性能比较方法. 为便于讨论, 本节默认以错误率为性能度量, 用 ϵ 表示.

2.4.1 假设检验

假设检验中的“假设”是对学习器泛化错误率分布的某种判断或猜想, 例如“ $\epsilon = \epsilon_0$ ”. 现实任务中我们并不知道学习器的泛化错误率, 只能获知其测试错误率 $\hat{\epsilon}$. 泛化错误率与测试错误率未必相同, 但直观上, 二者接近的可能性应比

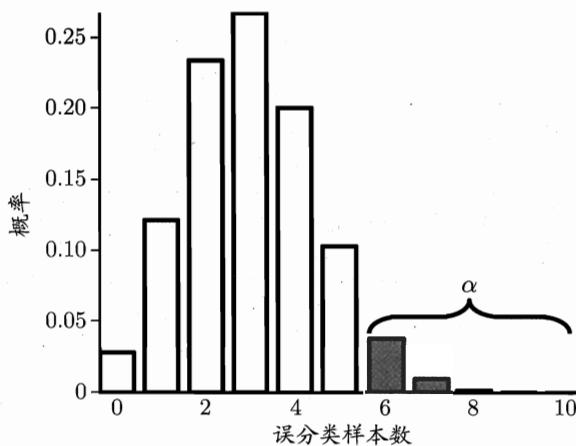
更多关于假设检验的介绍可参见[Wellek, 2010].

较大, 相差很远的可能性比较小. 因此, 可根据测试错误率估推出泛化错误率的分布.

泛化错误率为 ϵ 的学习器在一个样本上犯错的概率是 ϵ ; 测试错误率 $\hat{\epsilon}$ 意味着在 m 个测试样本中恰有 $\hat{\epsilon} \times m$ 个被误分类. 假定测试样本是从样本总体分布中独立采样而得, 那么泛化错误率为 ϵ 的学习器将其中 m' 个样本误分类、其余样本全都分类正确的概率是 $\epsilon^{m'}(1 - \epsilon)^{m-m'}$; 由此可估算出其恰将 $\hat{\epsilon} \times m$ 个样本误分类的概率如下式所示, 这也表达了在包含 m 个样本的测试集上, 泛化错误率为 ϵ 的学习器被测得测试错误率为 $\hat{\epsilon}$ 的概率:

$$P(\hat{\epsilon}; \epsilon) = \binom{m}{\hat{\epsilon} \times m} \epsilon^{\hat{\epsilon} \times m} (1 - \epsilon)^{m - \hat{\epsilon} \times m}. \quad (2.26)$$

给定测试错误率, 则解 $\partial P(\hat{\epsilon}; \epsilon) / \partial \epsilon = 0$ 可知, $P(\hat{\epsilon}; \epsilon)$ 在 $\epsilon = \hat{\epsilon}$ 时最大, $|\epsilon - \hat{\epsilon}|$ 增大时 $P(\hat{\epsilon}; \epsilon)$ 减小. 这符合二项(binomial)分布, 如图 2.6 所示, 若 $\epsilon = 0.3$, 则 10 个样本中测得 3 个被误分类的概率最大.



α 的常用取值有 0.05、0.1, 图 2.6 中 α 较大是为了绘图方便.

图 2.6 二项分布示意图($m = 10, \epsilon = 0.3$)

我们可使用“二项检验”(binomial test)来对“ $\epsilon \leq 0.3$ ”(即“泛化错误率是否不大于 0.3”)这样的假设进行检验. 更一般的, 考虑假设“ $\epsilon \leq \epsilon_0$ ”, 则在 $1 - \alpha$ 的概率内所能观测到的最大错误率如下式计算. 这里 $1 - \alpha$ 反映了结论的“置信度”(confidence), 直观地来看, 相应于图 2.6 中非阴影部分的范围.

s.t. 是 “subject to”的
简写, 使左边式子在右边
条件满足时成立.

$$\bar{\epsilon} = \max \epsilon \quad \text{s.t.} \quad \sum_{i=\epsilon_0 \times m+1}^m \binom{m}{i} \epsilon^i (1 - \epsilon)^{m-i} < \alpha. \quad (2.27)$$

二项检验的临界值在 R 语言中可通过 `qbinom(1 - alpha, m, epsilon_0)` 计算, 在 Matlab 中是 `icdf('Binomial', 1 - alpha, m, epsilon_0)`.

R 语言是面向统计计算的开源脚本语言, 参见 www.r-project.org.

此时若测试错误率 $\hat{\epsilon}$ 小于临界值 $\bar{\epsilon}$, 则根据二项检验可得出结论: 在 α 的显著度下, 假设 “ $\epsilon \leq \epsilon_0$ ” 不能被拒绝, 即能以 $1 - \alpha$ 的置信度认为, 学习器的泛化错误率不大于 ϵ_0 ; 否则该假设可被拒绝, 即在 α 的显著度下可认为学习器的泛化错误率大于 ϵ_0 .

在很多时候我们并非仅做一次留出法估计, 而是通过多次重复留出法或是交叉验证法等进行多次训练/测试, 这样会得到多个测试错误率, 此时可使用

“*t* 检验” (*t*-test). 假定我们得到了 k 个测试错误率, $\hat{\epsilon}_1, \hat{\epsilon}_2, \dots, \hat{\epsilon}_k$, 则平均测试错误率 μ 和方差 σ^2 为

$$\mu = \frac{1}{k} \sum_{i=1}^k \hat{\epsilon}_i, \quad (2.28)$$

$$\sigma^2 = \frac{1}{k-1} \sum_{i=1}^k (\hat{\epsilon}_i - \mu)^2. \quad (2.29)$$

考虑到这 k 个测试错误率可看作泛化错误率 ϵ_0 的独立采样, 则变量

$$\tau_t = \frac{\sqrt{k}(\mu - \epsilon_0)}{\sigma} \quad (2.30)$$

服从自由度为 $k - 1$ 的 *t* 分布, 如图 2.7 所示.

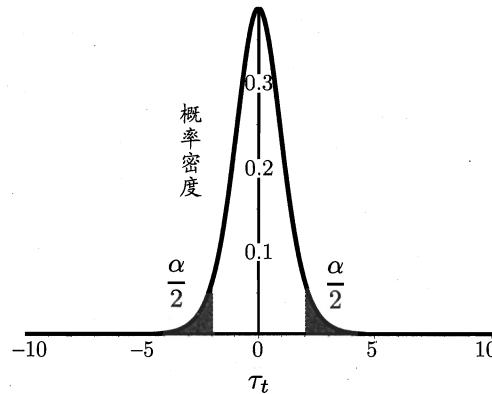


图 2.7 *t* 分布示意图($k = 10$)

对假设 “ $\mu = \epsilon_0$ ” 和显著度 α , 我们可计算出当测试错误率均值为 ϵ_0 时, 在 $1 - \alpha$ 概率内能观测到的最大错误率, 即临界值. 这里考虑双边(two-tailed)假设, 如图 2.7 所示, 两边阴影部分各有 $\alpha/2$ 的面积; 假定阴影部分范围分别为 $[-\infty, t_{-\alpha/2}]$ 和 $[t_{\alpha/2}, \infty]$. 若平均错误率 μ 与 ϵ_0 之差 $|\mu - \epsilon_0|$ 位于临界值范围

$[t_{-\alpha/2}, t_{\alpha/2}]$ 内, 则不能拒绝假设 “ $\mu = \epsilon_0$ ”, 即可认为泛化错误率为 ϵ_0 , 置信度为 $1 - \alpha$; 否则可拒绝该假设, 即在该显著度下可认为泛化错误率与 ϵ_0 有显著不同. α 常用取值有 0.05 和 0.1. 表 2.3 给出了一些常用临界值.

临界值 $t_{\alpha/2}$ 在 R 语言中可通过 $qt(1 - \alpha/2, k - 1)$ 计算, 在 Matlab 中是 $icdf('T', 1 - \alpha/2, k - 1)$.

表 2.3 双边 t 检验的常用临界值

α	k				
	2	5	10	20	30
0.05	12.706	2.776	2.262	2.093	2.045
0.10	6.314	2.132	1.833	1.729	1.699

上面介绍的两种方法都是对关于单个学习器泛化性能的假设进行检验, 而在现实任务中, 更多时候我们需对不同学习器的性能进行比较, 下面将介绍适用于此类情况的假设检验方法.

2.4.2 交叉验证 t 检验

对两个学习器 A 和 B, 若我们使用 k 折交叉验证法得到的测试错误率分别为 $\epsilon_1^A, \epsilon_2^A, \dots, \epsilon_k^A$ 和 $\epsilon_1^B, \epsilon_2^B, \dots, \epsilon_k^B$, 其中 ϵ_i^A 和 ϵ_i^B 是在相同的第 i 折训练/测试集上得到的结果, 则可用 k 折交叉验证 “成对 t 检验” (paired t -tests) 来进行比较检验. 这里的基本思想是若两个学习器的性能相同, 则它们使用相同的训练/测试集得到的测试错误率应相同, 即 $\epsilon_i^A = \epsilon_i^B$.

具体来说, 对 k 折交叉验证产生的 k 对测试错误率: 先对每对结果求差, $\Delta_i = \epsilon_i^A - \epsilon_i^B$; 若两个学习器性能相同, 则差值均值应为零. 因此, 可根据差值 $\Delta_1, \Delta_2, \dots, \Delta_k$ 来对 “学习器 A 与 B 性能相同” 这个假设做 t 检验, 计算出差值的均值 μ 和方差 σ^2 , 在显著度 α 下, 若变量

$$\tau_t = \left| \frac{\sqrt{k}\mu}{\sigma} \right| \quad (2.31)$$

小于临界值 $t_{\alpha/2, k-1}$, 则假设不能被拒绝, 即认为两个学习器的性能没有显著差别; 否则可认为两个学习器的性能有显著差别, 且平均错误率较小的那个学习器性能较优. 这里 $t_{\alpha/2, k-1}$ 是自由度为 $k - 1$ 的 t 分布上尾部累积分布为 $\alpha/2$ 的临界值.

欲进行有效的假设检验, 一个重要前提是测试错误率均为泛化错误率的独立采样. 然而, 通常情况下由于样本有限, 在使用交叉验证等实验估计方法时, 不同轮次的训练集会有一定程度的重叠, 这就使得测试错误率实际上并不独立, 会导致过高估计假设成立的概率. 为缓解这一问题, 可采用 “ 5×2 交叉验证”

法 [Dietterich, 1998].

5×2 交叉验证是做 5 次 2 折交叉验证, 在每次 2 折交叉验证之前随机将数据打乱, 使得 5 次交叉验证中的数据划分不重复. 对两个学习器 A 和 B, 第 i 次 2 折交叉验证将产生两对测试错误率, 我们对它们分别求差, 得到第 1 折上的差值 Δ_i^1 和第 2 折上的差值 Δ_i^2 . 为缓解测试错误率的非独立性, 我们仅计算第 1 次 2 折交叉验证的两个结果的平均值 $\mu = 0.5(\Delta_1^1 + \Delta_1^2)$, 但对每次 2 折实验的结果都计算出其方差 $\sigma_i^2 = \left(\Delta_i^1 - \frac{\Delta_1^1 + \Delta_1^2}{2}\right)^2 + \left(\Delta_i^2 - \frac{\Delta_1^1 + \Delta_1^2}{2}\right)^2$. 变量

$$\tau_t = \frac{\mu}{\sqrt{0.2 \sum_{i=1}^5 \sigma_i^2}} \quad (2.32)$$

服从自由度为 5 的 t 分布, 其双边检验的临界值 $t_{\alpha/2, 5}$ 当 $\alpha = 0.05$ 时为 2.5706, $\alpha = 0.1$ 时为 2.0150.

2.4.3 McNemar 检验

对二分类问题, 使用留出法不仅可估计出学习器 A 和 B 的测试错误率, 还可获得两学习器分类结果的差别, 即两者都正确、都错误、一个正确另一个错误的样本数, 如“列联表”(contingency table) 2.4 所示.

表 2.4 两学习器分类差别列联表

算法 B	算法 A	
	正确	错误
正确	e_{00}	e_{01}
错误	e_{10}	e_{11}

若我们做的假设是两学习器性能相同, 则应有 $e_{01} = e_{10}$, 那么变量 $|e_{01} - e_{10}|$ 应当服从正态分布, 且均值为 1, 方差为 $e_{01} + e_{10}$. 因此变量

$$\tau_{\chi^2} = \frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}} \quad (2.33)$$

中文称为“卡方分布”.

临界值 χ_{α}^2 在 R 语言中可通过 `qchisq(1 - alpha, k - 1)` 计算, 在 Matlab 中是 `icdf('Chisquare', 1 - alpha, k - 1)`. 这里的 $k = 2$ 是进行比较的算法个数.

服从自由度为 1 的 χ^2 分布, 即标准正态分布变量的平方. 给定显著度 α , 当以上变量值小于临界值 χ_{α}^2 时, 不能拒绝假设, 即认为两学习器的性能没有显著差别; 否则拒绝假设, 即认为两者性能有显著差别, 且平均错误率较小的那个学习器性能较优. 自由度为 1 的 χ^2 检验的临界值当 $\alpha = 0.05$ 时为 3.8415, $\alpha = 0.1$ 时为 2.7055.

2.4.4 Friedman 检验与 Nemenyi 后续检验

交叉验证 t 检验和 McNemar 检验都是在一个数据集上比较两个算法的性能, 而在很多时候, 我们会在一组数据集上对多个算法进行比较. 当有多个算法参与比较时, 一种做法是在每个数据集上分别列出两两比较的结果, 而在两两比较时可使用前述方法; 另一种方法更为直接, 即使用基于算法排序的 Friedman 检验.

假定我们用 D_1 、 D_2 、 D_3 和 D_4 四个数据集对算法 A、B、C 进行比较. 首先, 使用留出法或交叉验证法得到每个算法在每个数据集上的测试结果, 然后在每个数据集上根据测试性能由好到坏排序, 并赋予序值 1, 2, …; 若算法的测试性能相同, 则平分序值. 例如, 在 D_1 和 D_3 上, A 最好、B 其次、C 最差, 而在 D_2 上, A 最好、B 与 C 性能相同, ……, 则可列出表 2.5, 其中最后一行通过对每一列的序值求平均, 得到平均序值.

表 2.5 算法比较序值表

数据集	算法 A	算法 B	算法 C
D_1	1	2	3
D_2	1	2.5	2.5
D_3	1	2	3
D_4	1	2	3
平均序值	1	2.125	2.875

然后, 使用 Friedman 检验来判断这些算法是否性能都相同. 若相同, 则它们的平均序值应当相同. 假定我们在 N 个数据集上比较 k 个算法, 令 r_i 表示第 i 个算法的平均序值, 为简化讨论, 暂不考虑平分序值的情况, 则 r_i 服从正态分布, 其均值和方差分别为 $(k+1)/2$ 和 $(k^2-1)/12$. 变量

$$\begin{aligned} \tau_{\chi^2} &= \frac{k-1}{k} \cdot \frac{12N}{k^2-1} \sum_{i=1}^k \left(r_i - \frac{k+1}{2} \right)^2 \\ &= \frac{12N}{k(k+1)} \left(\sum_{i=1}^k r_i^2 - \frac{k(k+1)^2}{4} \right) \end{aligned} \quad (2.34)$$

在 k 和 N 都较大时, 服从自由度为 $k-1$ 的 χ^2 分布.

然而, 上述这样的“原始 Friedman 检验”过于保守, 现在通常使用变量

$$\tau_F = \frac{(N-1)\tau_{\chi^2}}{N(k-1) - \tau_{\chi^2}}, \quad (2.35)$$

其中 τ_{χ^2} 由式(2.34)得到. τ_F 服从自由度为 $k - 1$ 和 $(k - 1)(N - 1)$ 的 F 分布, 表 2.6 给出了一些常用临界值.

表 2.6 F 检验的常用临界值

$\alpha = 0.05$										
数据集		算法个数 k								
个数 N		2	3	4	5	6	7	8	9	10
	4	10.128	5.143	3.863	3.259	2.901	2.661	2.488	2.355	2.250
	5	7.709	4.459	3.490	3.007	2.711	2.508	2.359	2.244	2.153
	8	5.591	3.739	3.072	2.714	2.485	2.324	2.203	2.109	2.032
	10	5.117	3.555	2.960	2.634	2.422	2.272	2.159	2.070	1.998
	15	4.600	3.340	2.827	2.537	2.346	2.209	2.104	2.022	1.955
	20	4.381	3.245	2.766	2.492	2.310	2.179	2.079	2.000	1.935

$\alpha = 0.1$										
数据集		算法个数 k								
个数 N		2	3	4	5	6	7	8	9	10
	4	5.538	3.463	2.813	2.480	2.273	2.130	2.023	1.940	1.874
	5	4.545	3.113	2.606	2.333	2.158	2.035	1.943	1.870	1.811
	8	3.589	2.726	2.365	2.157	2.019	1.919	1.843	1.782	1.733
	10	3.360	2.624	2.299	2.108	1.980	1.886	1.814	1.757	1.710
	15	3.102	2.503	2.219	2.048	1.931	1.845	1.779	1.726	1.682
	20	2.990	2.448	2.182	2.020	1.909	1.826	1.762	1.711	1.668

若“所有算法的性能相同”这个假设被拒绝, 则说明算法的性能显著不同. 这时需进行“后续检验”(post-hoc test)来进一步区分各算法. 常用的有 Nemenyi 后续检验.

Nemenyi 检验计算出平均序值差别的临界值域

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \quad (2.36)$$

q_α 是 Tukey 分布的临界值, 在 R 语言中可通过 `qtukey(1 - alpha, k, Inf) / sqrt(2)` 计算.

表 2.7 给出了 $\alpha = 0.05$ 和 0.1 时常用的 q_α 值. 若两个算法的平均序值之差超出了临界值域 CD , 则以相应的置信度拒绝“两个算法性能相同”这一假设.

表 2.7 Nemenyi 检验中常用的 q_α 值

α	算法个数 k									
	2	3	4	5	6	7	8	9	10	
0.05	1.960	2.344	2.569	2.728	2.850	2.949	3.031	3.102	3.164	
0.1	1.645	2.052	2.291	2.459	2.589	2.693	2.780	2.855	2.920	

以表 2.5 中的数据为例, 先根据式(2.34)和(2.35)计算出 $\tau_F = 24.429$, 由表 2.6 可知, 它大于 $\alpha = 0.05$ 时的 F 检验临界值 5.143, 因此拒绝“所有算法性能相同”这个假设。然后使用 Nemenyi 后续检验, 在表 2.7 中找到 $k = 3$ 时 $q_{0.05} = 2.344$, 根据式(2.36)计算出临界值域 $CD = 1.657$, 由表 2.5 中的平均序值可知, 算法 A 与 B 的差距, 以及算法 B 与 C 的差距均未超过临界值域, 而算法 A 与 C 的差距超过临界值域, 因此检验结果认为算法 A 与 C 的性能显著不同, 而算法 A 与 B、以及算法 B 与 C 的性能没有显著差别。

上述检验比较可以直观地用 Friedman 检验图显示。例如根据表 2.5 的序值结果可绘制出图 2.8, 图中纵轴显示各个算法, 横轴是平均序值。对每个算法, 用一个圆点显示其平均序值, 以圆点为中心的横线段表示临界值域的大小。然后就可从图中观察, 若两个算法的横线段有交叠, 则说明这两个算法没有显著差别, 否则即说明有显著差别。从图 2.8 中可容易地看出, 算法 A 与 B 没有显著差别, 因为它们的横线段有交叠区域, 而算法 A 显著优于算法 C, 因为它们的横线段没有交叠区域。

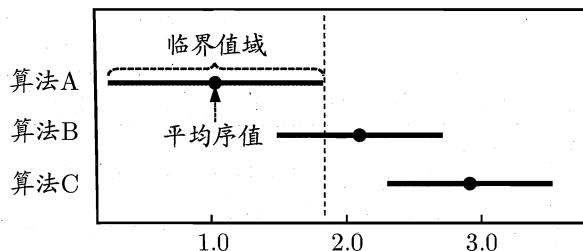


图 2.8 Friedman 检验图

2.5 偏差与方差

对学习算法除了通过实验估计其泛化性能, 人们往往还希望了解它“为什么”具有这样的性能。“偏差-方差分解”(bias-variance decomposition)是解释学习算法泛化性能的一种重要工具。

偏差-方差分解试图对学习算法的期望泛化错误率进行拆解。我们知道, 算法在不同训练集上学得的结果很可能不同, 即便这些训练集是来自同一个分布。对测试样本 x , 令 y_D 为 x 在数据集中的标记, y 为 x 的真实标记, $f(x; D)$ 为训练集 D 上学得模型 f 在 x 上的预测输出。以回归任务为例, 学习算法的期望预

有可能出现噪声使得
 $y_D \neq y$

测为

$$\bar{f}(\mathbf{x}) = \mathbb{E}_D[f(\mathbf{x}; D)], \quad (2.37)$$

使用样本数相同的不同训练集产生的方差为

$$var(\mathbf{x}) = \mathbb{E}_D [(f(\mathbf{x}; D) - \bar{f}(\mathbf{x}))^2], \quad (2.38)$$

噪声为

$$\varepsilon^2 = \mathbb{E}_D [(y_D - y)^2]. \quad (2.39)$$

期望输出与真实标记的差别称为偏差(bias), 即

$$bias^2(\mathbf{x}) = (\bar{f}(\mathbf{x}) - y)^2. \quad (2.40)$$

为便于讨论, 假定噪声期望为零, 即 $\mathbb{E}_D[y_D - y] = 0$. 通过简单的多项式展开合并, 可对算法的期望泛化误差进行分解:

$$\begin{aligned} E(f; D) &= \mathbb{E}_D [(f(\mathbf{x}; D) - y_D)^2] \\ &= \mathbb{E}_D [(f(\mathbf{x}; D) - \bar{f}(\mathbf{x}) + \bar{f}(\mathbf{x}) - y_D)^2] \\ &= \mathbb{E}_D [(f(\mathbf{x}; D) - \bar{f}(\mathbf{x}))^2] + \mathbb{E}_D [(\bar{f}(\mathbf{x}) - y_D)^2] \\ &\quad + \mathbb{E}_D [2(f(\mathbf{x}; D) - \bar{f}(\mathbf{x}))(\bar{f}(\mathbf{x}) - y_D)] \\ &= \mathbb{E}_D [(f(\mathbf{x}; D) - \bar{f}(\mathbf{x}))^2] + \mathbb{E}_D [(\bar{f}(\mathbf{x}) - y_D)^2] \\ &= \mathbb{E}_D [(f(\mathbf{x}; D) - \bar{f}(\mathbf{x}))^2] + \mathbb{E}_D [(\bar{f}(\mathbf{x}) - y + y - y_D)^2] \\ &= \mathbb{E}_D [(f(\mathbf{x}; D) - \bar{f}(\mathbf{x}))^2] + \mathbb{E}_D [(\bar{f}(\mathbf{x}) - y)^2] + \mathbb{E}_D [(y - y_D)^2] \\ &\quad + 2\mathbb{E}_D [(\bar{f}(\mathbf{x}) - y)(y - y_D)] \\ &= \mathbb{E}_D [(f(\mathbf{x}; D) - \bar{f}(\mathbf{x}))^2] + (\bar{f}(\mathbf{x}) - y)^2 + \mathbb{E}_D [(y_D - y)^2], \end{aligned} \quad (2.41)$$

由式(2.37), 最后项为0.

噪声期望为0,
因此最后项为0.

于是,

$$E(f; D) = bias^2(\mathbf{x}) + var(\mathbf{x}) + \varepsilon^2, \quad (2.42)$$

也就是说, 泛化误差可分解为偏差、方差与噪声之和.

回顾偏差、方差、噪声的含义: 偏差(2.40)度量了学习算法的期望预测与

真实结果的偏离程度, 即刻画了学习算法本身的拟合能力; 方差(2.38)度量了同样大小的训练集的变动所导致的学习性能的变化, 即刻画了数据扰动所造成的影响; 噪声(2.39)则表达了在当前任务上任何学习算法所能达到的期望泛化误差的下界, 即刻画了学习问题本身的难度. 偏差-方差分解说明, 泛化性能是由学习算法的能力、数据的充分性以及学习任务本身的难度所共同决定的. 给定学习任务, 为了取得好的泛化性能, 则需使偏差较小, 即能够充分拟合数据, 并且使方差较小, 即使得数据扰动产生的影响小.

一般来说, 偏差与方差是有冲突的, 这称为偏差-方差窘境(bias-variance dilemma). 图 2.9 给出了一个示意图. 给定学习任务, 假定我们能控制学习算法的训练程度, 则在训练不足时, 学习器的拟合能力不够强, 训练数据的扰动不足以使学习器产生显著变化, 此时偏差主导了泛化错误率; 随着训练程度的加深, 学习器的拟合能力逐渐增强, 训练数据发生的扰动渐渐能被学习器学到, 方差逐渐主导了泛化错误率; 在训练程度充足后, 学习器的拟合能力已非常强, 训练数据发生的轻微扰动都会导致学习器发生显著变化, 若训练数据自身的、非全局的特性被学习器学到了, 则将发生过拟合.

很多学习算法都可控制训练程度, 例如决策树可控制层数, 神经网络可控制训练轮数, 集成学习方法可控制基学习器个数.

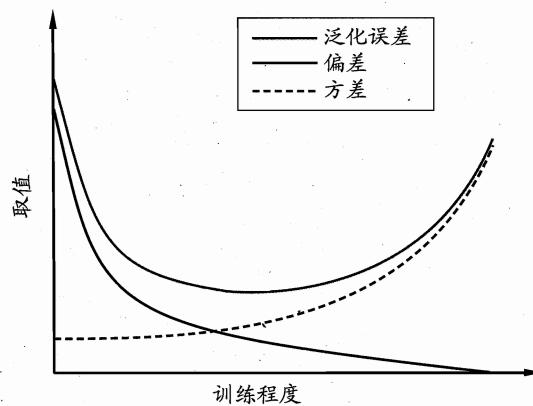


图 2.9 泛化误差与偏差、方差的关系示意图

2.6 阅读材料

自助采样法在机器学习中有重要用途, [Efron and Tibshirani, 1993] 对此进行了详细的讨论.

ROC 曲线在二十世纪八十年代后期被引入机器学习 [Spackman, 1989], AUC 则是从九十年代中期起在机器学习领域广为使用 [Bradley, 1997], 但利用

ROC 曲线下面积来评价模型期望性能的做法在医疗检测中早已有之 [Hanley and McNeil, 1983]. [Hand and Till, 2001] 将 ROC 曲线从二分类任务推广到多分类任务. [Fawcett, 2006] 综述了 ROC 曲线的用途.

2.3.4 节仅讨论了基于类别的误分类代价.

[Drummond and Holte, 2006] 发明了代价曲线. 需说明的是, 机器学习过程涉及许多类型的代价, 除了误分类代价, 还有测试代价、标记代价、属性代价等, 即便仅考虑误分类代价, 仍可进一步划分为基于类别的误分类代价以及基于样本的误分类代价. 代价敏感学习(cost-sensitive learning) [Elkan, 2001; Zhou and Liu, 2006] 专门研究非均等代价下的学习.

[Dietterich, 1998] 指出了常规 k 折交叉验证法存在的风险, 并提出了 5×2 交叉验证法. [Demšar, 2006] 讨论了对多个算法进行比较检验的方法.

[Geman et al., 1992] 针对回归任务给出了偏差-方差-协方差分解(bias-variance-covariance decomposition), 后来被简称为偏差-方差分解. 虽然偏差和方差确实反映了各类学习任务内在的误差决定因素, 但式(2.42)这样优美的形式仅在基于均方误差的回归任务中得以推导出. 对分类任务, 由于 0/1 损失函数的跳变性, 理论上推导出偏差-方差分解很困难. 已有多种方法可通过实验对偏差和方差进行估计 [Kong and Dietterich, 1995; Kohavi and Wolpert, 1996; Breiman, 1996; Friedman, 1997; Domingos, 2000].

习题

- 2.1** 数据集包含 1000 个样本, 其中 500 个正例、500 个反例, 将其划分为包含 70% 样本的训练集和 30% 样本的测试集用于留出法评估, 试估算共有多少种划分方式.
- 2.2** 数据集包含 100 个样本, 其中正、反例各一半, 假定学习算法所产生的模型是将新样本预测为训练样本数较多的类别(训练样本数相同时进行随机猜测), 试给出用 10 折交叉验证法和留一法分别对错误率进行评估所得的结果.
- 2.3** 若学习器 A 的 $F1$ 值比学习器 B 高, 试析 A 的 BEP 值是否也比 B 高.
- 2.4** 试述真正例率(TPR)、假正例率(FPR)与查准率(P)、查全率(R)之间的联系.
- 2.5** 试证明式(2.22).
- 2.6** 试述错误率与 ROC 曲线的联系.
- 2.7** 试证明任意一条 ROC 曲线都有一条代价曲线与之对应, 反之亦然.
- 2.8** Min-max 规范化和 z -score 规范化是两种常用的规范化方法. 令 x 和 x' 分别表示变量在规范化前后的取值, 相应的, 令 x_{min} 和 x_{max} 表示规范化前的最小值和最大值, x'_{min} 和 x'_{max} 表示规范化后的最小值和最大值, \bar{x} 和 σ_x 分别表示规范化前的均值和标准差, 则 min-max 规范化、 z -score 规范化分别如式(2.43)和(2.44)所示. 试析二者的优缺点.
- $$x' = x'_{min} + \frac{x - x_{min}}{x_{max} - x_{min}} \times (x'_{max} - x'_{min}), \quad (2.43)$$
- $$x' = \frac{x - \bar{x}}{\sigma_x}. \quad (2.44)$$
- 2.9** 试述 χ^2 检验过程.
- 2.10*** 试述在 Friedman 检验中使用式(2.34)与(2.35)的区别.

参考文献

- Bradley, A. P. (1997). "The use of the area under the ROC curve in the evaluation of machine learning algorithms." *Pattern Recognition*, 30(7):1145–1159.
- Breiman, L. (1996). "Bias, variance, and arcing classifiers." Technical Report 460, Statistics Department, University of California, Berkeley, CA.
- Demsar, J. (2006). "Statistical comparison of classifiers over multiple data sets." *Journal of Machine Learning Research*, 7:1–30.
- Dietterich, T. G. (1998). "Approximate statistical tests for comparing supervised classification learning algorithms." *Neural Computation*, 10(7):1895–1923.
- Domingos, P. (2000). "A unified bias-variance decomposition." In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, 231–238, Stanford, CA.
- Drummond, C. and R. C. Holte. (2006). "Cost curves: An improved method for visualizing classifier performance." *Machine Learning*, 65(1):95–130.
- Efron, B. and R. Tibshirani. (1993). *An Introduction to the Bootstrap*. Chapman & Hall, New York, NY.
- Elkan, C. (2001). "The foundations of cost-sensitive learning." In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 973–978, Seattle, WA.
- Fawcett, T. (2006). "An introduction to ROC analysis." *Pattern Recognition Letters*, 27(8):861–874.
- Friedman, J. H. (1997). "On bias, variance, 0/1-loss, and the curse-of-dimensionality." *Data Mining and Knowledge Discovery*, 1(1):55–77.
- Geman, S., E. Bienenstock, and R. Doursat. (1992). "Neural networks and the bias/variance dilemma." *Neural Computation*, 4(1):1–58.
- Hand, D. J. and R. J. Till. (2001). "A simple generalisation of the area under the ROC curve for multiple class classification problems." *Machine Learning*, 45(2):171–186.
- Hanley, J. A. and B. J. McNeil. (1983). "A method of comparing the areas under receiver operating characteristic curves derived from the same cases." *Radiology*, 148(3):839–843.

- Kohavi, R. and D. H. Wolpert. (1996). "Bias plus variance decomposition for zero-one loss functions." In *Proceeding of the 13th International Conference on Machine Learning (ICML)*, 275–283, Bari, Italy.
- Kong, E. B. and T. G. Dietterich. (1995). "Error-correcting output coding corrects bias and variance." In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, 313–321, Tahoe City, CA.
- Mitchell, T. (1997). *Machine Learning*. McGraw Hill, New York, NY.
- Spackman, K. A. (1989). "Signal detection theory: Valuable tools for evaluating inductive learning." In *Proceedings of the 6th International Workshop on Machine Learning (IWML)*, 160–163, Ithaca, NY.
- Van Rijsbergen, C. J. (1979). *Information Retrieval*, 2nd edition. Butterworths, London, UK.
- Wellek, S. (2010). *Testing Statistical Hypotheses of Equivalence and Noninferiority*, 2nd edition. Chapman & Hall/CRC, Boca Raton, FL.
- Zhou, Z.-H. and X.-Y. Liu. (2006). "On multi-class cost-sensitive learning." In *Proceeding of the 21st National Conference on Artificial Intelligence (AAAI)*, 567–572, Boston, WA.

休息一会儿

小故事： t 检验、啤酒、“学生”与威廉·戈瑟特

1954 年该厂开始出版
《吉尼斯世界纪录大全》。

1899 年，由于爱尔兰都柏林的吉尼斯啤酒厂热衷于聘用剑桥、牛津的优秀毕业生，学化学的牛津毕业生威廉·戈瑟特 (William Gosset, 1876—1937) 到该厂就职，希望将他的生物化学知识用于啤酒生产过程。为降低啤酒质量监控的成本，戈瑟特发明了 t 检验法，1908 年在 *Biometrika* 发表。为防止泄漏商业机密，戈瑟特发表文章时用了笔名“学生”，于是该方法被称为“学生氏 t 检验” (Student's t -test)。



吉尼斯啤酒厂是一家很有远见的企业，为保持技术人员的高水准，该厂像高校一样给予技术人员“学术假”，1906—1907 年戈瑟特得以到“统计学之父”卡尔·皮尔逊 (Karl Pearson, 1857—1936) 教授在伦敦大学学院 (University College London, 简称 UCL) 的实验室访问学习。因此，很难说 t 检验法是戈瑟特在啤酒厂还是在 UCL 访学期间提出的，但“学生”与戈瑟特之间的联系是被 UCL 的统计学家们发现的，尤其因为皮尔逊教授恰是 *Biometrika* 的主编。

第3章 线性模型

3.1 基本形式

给定由 d 个属性描述的示例 $\mathbf{x} = (x_1; x_2; \dots; x_d)$, 其中 x_i 是 \mathbf{x} 在第 i 个属性上的取值, 线性模型(linear model)试图学得一个通过属性的线性组合来进行预测的函数, 即

$$f(\mathbf{x}) = w_1x_1 + w_2x_2 + \dots + w_dx_d + b, \quad (3.1)$$

一般用向量形式写成

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b, \quad (3.2)$$

其中 $\mathbf{w} = (w_1; w_2; \dots; w_d)$. \mathbf{w} 和 b 学得之后, 模型就得以确定.

亦称“可理解性”(understandability).

线性模型形式简单、易于建模, 但却蕴涵着机器学习中一些重要的基本思想. 许多功能更为强大的非线性模型(nonlinear model)可在线性模型的基础上通过引入层级结构或高维映射而得. 此外, 由于 \mathbf{w} 直观表达了各属性在预测中的重要性, 因此线性模型有很好的可解释性(comprehensibility). 例如若在西瓜问题中学得 “ $f_{\text{好瓜}}(\mathbf{x}) = 0.2 \cdot x_{\text{色泽}} + 0.5 \cdot x_{\text{根蒂}} + 0.3 \cdot x_{\text{敲声}} + 1$ ”, 则意味着可通过综合考虑色泽、根蒂和敲声来判断瓜好不好, 其中根蒂最要紧, 而敲声比色泽更重要.

本章介绍几种经典的线性模型. 我们先从回归任务开始, 然后讨论二分类和多分类任务.

3.2 线性回归

给定数据集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$, 其中 $\mathbf{x}_i = (x_{i1}; x_{i2}; \dots; x_{id})$, $y_i \in \mathbb{R}$. “线性回归”(linear regression)试图学得一个线性模型以尽可能准确地预测实值输出标记.

我们先考虑一种最简单的情形: 输入属性的数目只有一个. 为便于讨论, 此时我们忽略关于属性的下标, 即 $D = \{(x_i, y_i)\}_{i=1}^m$, 其中 $x_i \in \mathbb{R}$. 对离散属性, 若属性值间存在“序”(order)关系, 可通过连续化将其转化为连续值, 例如二

若将无序属性连续化, 则会不恰当地引入序关系, 对后续处理如距离计算等造成误导, 参见 9.3 节.

值属性“身高”的取值“高”“矮”可转化为 {1.0, 0.0}, 三值属性“高度”的取值“高”“中”“低”可转化为 {1.0, 0.5, 0.0}; 若属性值间不存在序关系, 假定有 k 个属性值, 则通常转化为 k 维向量, 例如属性“瓜类”的取值“西瓜”“南瓜”“黄瓜”可转化为 (0, 0, 1), (0, 1, 0), (1, 0, 0).

线性回归试图学得

$$f(x_i) = wx_i + b, \text{ 使得 } f(x_i) \simeq y_i. \quad (3.3)$$

均方误差亦称平方损失 (square loss).

如何确定 w 和 b 呢? 显然, 关键在于如何衡量 $f(x)$ 与 y 之间的差别. 2.3 节介绍过, 均方误差 (2.2) 是回归任务中最常用的性能度量, 因此我们可试图让均方误差最小化, 即

w^*, b^* 表示 w 和 b 的解.

$$\begin{aligned} (w^*, b^*) &= \arg \min_{(w,b)} \sum_{i=1}^m (f(x_i) - y_i)^2 \\ &= \arg \min_{(w,b)} \sum_{i=1}^m (y_i - wx_i - b)^2. \end{aligned} \quad (3.4)$$

最小二乘法用途很广, 不仅限于线性回归.

这里 $E_{(w,b)}$ 是关于 w 和 b 的凸函数, 当它关于 w 和 b 的导数均为零时, 得到 w 和 b 的最优解.

对区间 $[a, b]$ 上定义的函数 f , 若它对区间中任意两点 x_1, x_2 均有 $f\left(\frac{x_1+x_2}{2}\right) \leq \frac{f(x_1)+f(x_2)}{2}$, 则称 f 为区间 $[a, b]$ 上的凸函数.

U 形曲线的函数如 $f(x) = x^2$, 通常是凸函数.

对实数集上的函数, 可通过求二阶导数来判别: 若二阶导数在区间上非负, 则称为凸函数; 若二阶导数在区间上恒大于 0, 则称为严格凸函数.

均方误差有非常好的几何意义, 它对应了常用的欧几里得距离或简称“欧式距离” (Euclidean distance). 基于均方误差最小化来进行模型求解的方法称为“最小二乘法” (least square method). 在线性回归中, 最小二乘法就是试图找到一条直线, 使所有样本到直线上的欧式距离之和最小.

求解 w 和 b 使 $E_{(w,b)} = \sum_{i=1}^m (y_i - wx_i - b)^2$ 最小化的过程, 称为线性回归模型的最小二乘“参数估计” (parameter estimation). 我们可将 $E_{(w,b)}$ 分别对 w 和 b 求导, 得到

$$\frac{\partial E_{(w,b)}}{\partial w} = 2 \left(w \sum_{i=1}^m x_i^2 - \sum_{i=1}^m (y_i - b) x_i \right), \quad (3.5)$$

$$\frac{\partial E_{(w,b)}}{\partial b} = 2 \left(mb - \sum_{i=1}^m (y_i - wx_i) \right), \quad (3.6)$$

然后令式(3.5)和(3.6)为零可得到 w 和 b 最优解的闭式(closed-form)解

$$w = \frac{\sum_{i=1}^m y_i (x_i - \bar{x})}{\sum_{i=1}^m x_i^2 - \frac{1}{m} \left(\sum_{i=1}^m x_i \right)^2}, \quad (3.7)$$

$$b = \frac{1}{m} \sum_{i=1}^m (y_i - wx_i), \quad (3.8)$$

其中 $\bar{x} = \frac{1}{m} \sum_{i=1}^m x_i$ 为 x 的均值.

更一般的情形是如本节开头的数据集 D , 样本由 d 个属性描述. 此时我们试图学得

$$f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i + b, \text{ 使得 } f(\mathbf{x}_i) \simeq y_i,$$

亦称“多变量线性回归”.

这称为“多元线性回归”(multivariate linear regression).

类似的, 可利用最小二乘法来对 \mathbf{w} 和 b 进行估计. 为便于讨论, 我们把 \mathbf{w} 和 b 吸收入向量形式 $\hat{\mathbf{w}} = (\mathbf{w}; b)$, 相应的, 把数据集 D 表示为一个 $m \times (d+1)$ 大小的矩阵 \mathbf{X} , 其中每行对应于一个示例, 该行前 d 个元素对应于示例的 d 个属性值, 最后一个元素恒置为 1, 即

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1d} & 1 \\ x_{21} & x_{22} & \dots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{md} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1^T & 1 \\ \mathbf{x}_2^T & 1 \\ \vdots & \vdots \\ \mathbf{x}_m^T & 1 \end{pmatrix},$$

再把标记也写成向量形式 $\mathbf{y} = (y_1; y_2; \dots; y_m)$, 则类似于式(3.4), 有

$$\hat{\mathbf{w}}^* = \arg \min_{\hat{\mathbf{w}}} (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})^T (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}). \quad (3.9)$$

令 $E_{\hat{\mathbf{w}}} = (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})^T (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})$, 对 $\hat{\mathbf{w}}$ 求导得到

$$\frac{\partial E_{\hat{\mathbf{w}}}}{\partial \hat{\mathbf{w}}} = 2 \mathbf{X}^T (\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}). \quad (3.10)$$

令上式为零可得 $\hat{\mathbf{w}}$ 最优解的闭式解, 但由于涉及矩阵逆的计算, 比单变量情形要复杂一些. 下面我们做一个简单的讨论.

当 $\mathbf{X}^T \mathbf{X}$ 为满秩矩阵(full-rank matrix)或正定矩阵(positive definite matrix)时, 令式(3.10)为零可得

$$\hat{\mathbf{w}}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}, \quad (3.11)$$

其中 $(\mathbf{X}^T \mathbf{X})^{-1}$ 是矩阵 $(\mathbf{X}^T \mathbf{X})$ 的逆矩阵. 令 $\hat{\mathbf{x}}_i = (\mathbf{x}_i, 1)$, 则最终学得的多元

线性回归模型为

$$f(\hat{x}_i) = \hat{x}_i^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} . \quad (3.12)$$

例如, 生物信息学的基本芯片数据中常有成千上万个属性, 但往往只有几十、上百个样例.

回忆一下: 解线性方程组时, 若因变量过多, 则会解出多组解.

归纳偏好参见 1.4 节; 正则化参见 6.4、11.4 节.

然而, 现实任务中 $\mathbf{X}^T \mathbf{X}$ 往往不是满秩矩阵. 例如在许多任务中我们会遇到大量的变量, 其数目甚至超过样例数, 导致 \mathbf{X} 的列数多于行数, $\mathbf{X}^T \mathbf{X}$ 显然不满秩. 此时可解出多个 $\hat{\mathbf{w}}$, 它们都能使均方误差最小化. 选择哪一个解作为输出, 将由学习算法的归纳偏好决定, 常见的做法是引入正则化 (regularization) 项.

线性模型虽简单, 却有丰富的变化. 例如对于样例 (\mathbf{x}, y) , $y \in \mathbb{R}$, 当我们希望线性模型(3.2) 的预测值逼近真实标记 y 时, 就得到了线性回归模型. 为便于观察, 我们把线性回归模型简写为

$$y = \mathbf{w}^T \mathbf{x} + b . \quad (3.13)$$

可否令模型预测值逼近 y 的衍生物呢? 譬如说, 假设我们认为示例所对应的输出标记是在指数尺度上变化, 那就可将输出标记的对数作为线性模型逼近的目标, 即

$$\ln y = \mathbf{w}^T \mathbf{x} + b . \quad (3.14)$$

这就是“对数线性回归” (log-linear regression), 它实际上是在试图让 $e^{\mathbf{w}^T \mathbf{x} + b}$ 逼近 y . 式(3.14)在形式上仍是线性回归, 但实质上已是在求取输入空间到输出空间的非线性函数映射, 如图 3.1 所示. 这里的对数函数起到了将线性回归模型的预测值与真实标记联系起来的作用.

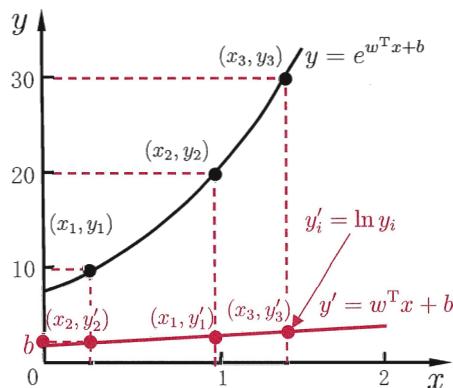


图 3.1 对数线性回归示意图

$g(\cdot)$ 连续且充分光滑.

更一般地, 考虑单调可微函数 $g(\cdot)$, 令

$$y = g^{-1}(\mathbf{w}^T \mathbf{x} + b), \quad (3.15)$$

广义线性模型的参数估计常通过加权最小二乘法或极大似然法进行.

这样得到的模型称为“广义线性模型”(generalized linear model), 其中函数 $g(\cdot)$ 称为“联系函数”(link function). 显然, 对数线性回归是广义线性模型在 $g(\cdot) = \ln(\cdot)$ 时的特例.

3.3 对数几率回归

上一节讨论了如何使用线性模型进行回归学习, 但若要做的是分类任务该怎么办? 答案蕴涵在式(3.15)的广义线性模型中: 只需找一个单调可微函数将分类任务的真实标记 y 与线性回归模型的预测值联系起来.

亦称 Heaviside 函数.

考虑二分类任务, 其输出标记 $y \in \{0, 1\}$, 而线性回归模型产生的预测值 $z = \mathbf{w}^T \mathbf{x} + b$ 是实值, 于是, 我们需将实值 z 转换为 0/1 值. 最理想的是“单位阶跃函数”(unit-step function)

$$y = \begin{cases} 0, & z < 0; \\ 0.5, & z = 0; \\ 1, & z > 0, \end{cases} \quad (3.16)$$

即若预测值 z 大于零就判为正例, 小于零则判为反例, 预测值为临界值零则可任意判别, 如图 3.2 所示.

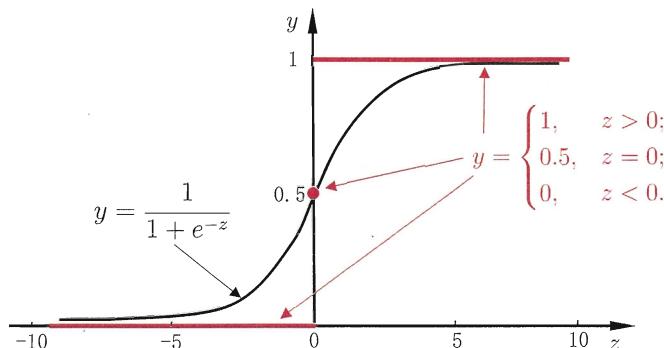


图 3.2 单位阶跃函数与对数几率函数

简称“对率函数”。

注意对数几率函数与
“对数函数” $\ln(\cdot)$ 不同。

Sigmoid 函数即形似 S 的函数。对率函数是 Sigmoid 函数最重要的代表，在第 5 章将看到它在神经网络中的重要作用。

但从图 3.2 可看出，单位阶跃函数不连续，因此不能直接用式(3.15)中的 $g^-(\cdot)$ 。于是我们希望找到能在一定程度上近似单位阶跃函数的“替代函数”(surrogate function)，并希望它单调可微。对数几率函数(logistic function)正是这样一个常用的替代函数：

$$y = \frac{1}{1 + e^{-z}}. \quad (3.17)$$

从图 3.2 可看出，对数几率函数是一种“Sigmoid 函数”，它将 z 值转化为一个接近 0 或 1 的 y 值，并且其输出值在 $z = 0$ 附近变化很陡。将对数几率函数作为 $g^-(\cdot)$ 代入式(3.15)，得到

$$y = \frac{1}{1 + e^{-(w^T x + b)}}. \quad (3.18)$$

类似于式(3.14)，式(3.18)可变化为

$$\ln \frac{y}{1 - y} = w^T x + b. \quad (3.19)$$

若将 y 视为样本 x 作为正例的可能性，则 $1 - y$ 是其反例可能性，两者的比值

$$\frac{y}{1 - y} \quad (3.20)$$

称为“几率”(odds)，反映了 x 作为正例的相对可能性。对几率取对数则得到“对数几率”(log odds，亦称 logit)

$$\ln \frac{y}{1 - y}. \quad (3.21)$$

由此可看出，式(3.18)实际上是在用线性回归模型的预测结果去逼近真实标记的对数几率，因此，其对应的模型称为“对数几率回归”(logistic regression，亦称 logit regression)。特别需注意到，虽然它的名字是“回归”，但实际却是一种分类学习方法。这种方法有很多优点，例如它是直接对分类可能性进行建模，无需事先假设数据分布，这样就避免了假设分布不准确所带来的问题；它不是仅预测出“类别”，而是可得到近似概率预测，这对许多需利用概率辅助决策的任务很有用；此外，对率函数是任意阶可导的凸函数，有很好的数学性质，现有的许多数值优化算法都可直接用于求取最优解。

有文献译为“逻辑回归”，但中文“逻辑”与 logistic 和 logit 的含义相去甚远，因此本书意译为“对数几率回归”，简称“对率回归”。

下面我们来看看如何确定式(3.18)中的 \mathbf{w} 和 b . 若将式(3.18)中的 y 视为类后验概率估计 $p(y = 1 | \mathbf{x})$, 则式(3.19)可重写为

$$\ln \frac{p(y = 1 | \mathbf{x})}{p(y = 0 | \mathbf{x})} = \mathbf{w}^T \mathbf{x} + b . \quad (3.22)$$

显然有

$$p(y = 1 | \mathbf{x}) = \frac{e^{\mathbf{w}^T \mathbf{x} + b}}{1 + e^{\mathbf{w}^T \mathbf{x} + b}} , \quad (3.23)$$

$$p(y = 0 | \mathbf{x}) = \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x} + b}} . \quad (3.24)$$

于是, 我们可通过“极大似然法”(maximum likelihood method)来估计
极大似然法参见 7.2 节. \mathbf{w} 和 b . 给定数据集 $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, 对率回归模型最大化“对数似然”(log-likelihood)

$$\ell(\mathbf{w}, b) = \sum_{i=1}^m \ln p(y_i | \mathbf{x}_i; \mathbf{w}, b) , \quad (3.25)$$

即令每个样本属于其真实标记的概率越大越好. 为便于讨论, 令 $\boldsymbol{\beta} = (\mathbf{w}; b)$, $\hat{\mathbf{x}} = (\mathbf{x}; 1)$, 则 $\mathbf{w}^T \mathbf{x} + b$ 可简写为 $\boldsymbol{\beta}^T \hat{\mathbf{x}}$. 再令 $p_1(\hat{\mathbf{x}}; \boldsymbol{\beta}) = p(y = 1 | \hat{\mathbf{x}}; \boldsymbol{\beta})$, $p_0(\hat{\mathbf{x}}; \boldsymbol{\beta}) = p(y = 0 | \hat{\mathbf{x}}; \boldsymbol{\beta}) = 1 - p_1(\hat{\mathbf{x}}; \boldsymbol{\beta})$, 则式(3.25) 中的似然项可重写为

$$p(y_i | \mathbf{x}_i; \mathbf{w}, b) = y_i p_1(\hat{\mathbf{x}}_i; \boldsymbol{\beta}) + (1 - y_i) p_0(\hat{\mathbf{x}}_i; \boldsymbol{\beta}) . \quad (3.26)$$

将式(3.26)代入(3.25), 并根据式(3.23)和(3.24)可知, 最大化式(3.25)等价于最小化

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^m \left(-y_i \boldsymbol{\beta}^T \hat{\mathbf{x}}_i + \ln \left(1 + e^{\boldsymbol{\beta}^T \hat{\mathbf{x}}_i} \right) \right) . \quad (3.27)$$

式(3.27)是关于 $\boldsymbol{\beta}$ 的高阶可导连续凸函数, 根据凸优化理论 [Boyd and Vandenberghe, 2004], 经典的数值优化算法如梯度下降法(gradient descent method)、牛顿法(Newton method)等都可求得其最优解, 于是就得到

$$\boldsymbol{\beta}^* = \arg \min_{\boldsymbol{\beta}} \ell(\boldsymbol{\beta}) . \quad (3.28)$$

以牛顿法为例, 其第 $t + 1$ 轮迭代解的更新公式为

$$\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t - \left(\frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} \right)^{-1} \frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} , \quad (3.29)$$

其中关于 β 的一阶、二阶导数分别为

$$\frac{\partial \ell(\beta)}{\partial \beta} = -\sum_{i=1}^m \hat{x}_i(y_i - p_1(\hat{x}_i; \beta)), \quad (3.30)$$

$$\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = \sum_{i=1}^m \hat{x}_i \hat{x}_i^T p_1(\hat{x}_i; \beta)(1 - p_1(\hat{x}_i; \beta)). \quad (3.31)$$

3.4 线性判别分析

严格说来 LDA 与 Fisher 判别分析稍有不同，前者假设了各类样本的协方差矩阵相同且满秩。

线性判别分析(Linear Discriminant Analysis, 简称 LDA)是一种经典线性学习方法，在二分类问题上因为最早由 [Fisher, 1936] 提出，亦称“Fisher 判别分析”。

LDA 的思想非常朴素：给定训练样例集，设法将样例投影到一条直线上，使得同类样例的投影点尽可能接近、异类样例的投影点尽可能远离；在对新样本进行分类时，将其投影到同样的这条直线上，再根据投影点的位置来确定新样本的类别。图 3.3 给出了一个二维示意图。

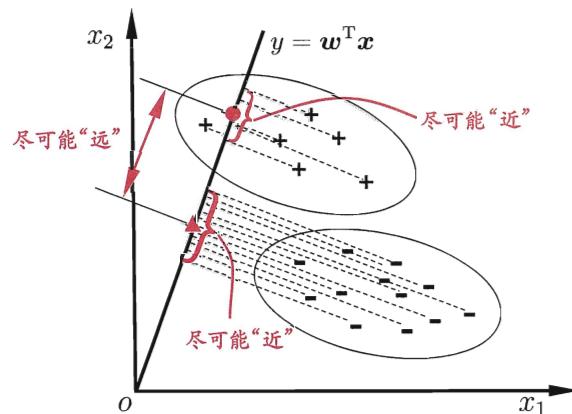


图 3.3 LDA 的二维示意图。“+”、“-”分别代表正例和反例，椭圆表示数据簇的外轮廓，虚线表示投影，红色实心圆和实心三角形分别表示两类样本投影后的中心点。

给定数据集 $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, $y_i \in \{0, 1\}$, 令 X_i 、 μ_i 、 Σ_i 分别表示第 $i \in \{0, 1\}$ 类示例的集合、均值向量、协方差矩阵。若将数据投影到直线 \mathbf{w} 上，则两类样本的中心在直线上的投影分别为 $\mathbf{w}^T \mu_0$ 和 $\mathbf{w}^T \mu_1$ ；若将所有样本点都投影到直线上，则两类样本的协方差分别为 $\mathbf{w}^T \Sigma_0 \mathbf{w}$ 和 $\mathbf{w}^T \Sigma_1 \mathbf{w}$ 。由于直线是

一维空间, 因此 $w^T \mu_0$ 、 $w^T \mu_1$ 、 $w^T \Sigma_0 w$ 和 $w^T \Sigma_1 w$ 均为实数.

欲使同类样例的投影点尽可能接近, 可以让同类样例投影点的协方差尽可能小, 即 $w^T \Sigma_0 w + w^T \Sigma_1 w$ 尽可能小; 而欲使异类样例的投影点尽可能远离, 可以让类中心之间的距离尽可能大, 即 $\|w^T \mu_0 - w^T \mu_1\|_2^2$ 尽可能大. 同时考虑二者, 则可得到欲最大化的目标

$$\begin{aligned} J &= \frac{\|w^T \mu_0 - w^T \mu_1\|_2^2}{w^T \Sigma_0 w + w^T \Sigma_1 w} \\ &= \frac{w^T (\mu_0 - \mu_1) (\mu_0 - \mu_1)^T w}{w^T (\Sigma_0 + \Sigma_1) w}. \end{aligned} \quad (3.32)$$

定义“类内散度矩阵”(within-class scatter matrix)

$$\begin{aligned} S_w &= \Sigma_0 + \Sigma_1 \\ &= \sum_{x \in X_0} (x - \mu_0)(x - \mu_0)^T + \sum_{x \in X_1} (x - \mu_1)(x - \mu_1)^T \end{aligned} \quad (3.33)$$

以及“类间散度矩阵”(between-class scatter matrix)

$$S_b = (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T, \quad (3.34)$$

则式(3.32)可重写为

$$J = \frac{w^T S_b w}{w^T S_w w}. \quad (3.35)$$

这就是 LDA 欲最大化的目标, 即 S_b 与 S_w 的“广义瑞利商”(generalized Rayleigh quotient).

若 w 是一个解, 则对于任意常数 α , αw 也是式(3.35)的解.

如何确定 w 呢? 注意到式(3.35)的分子和分母都是关于 w 的二次项, 因此式(3.35)的解与 w 的长度无关, 只与其方向有关. 不失一般性, 令 $w^T S_w w = 1$, 则式(3.35)等价于

$$\begin{aligned} \min_w \quad &-w^T S_b w \\ \text{s.t.} \quad &w^T S_w w = 1. \end{aligned} \quad (3.36)$$

拉格朗日乘子法参见附录 B.1.

由拉格朗日乘子法, 上式等价于

$$S_b w = \lambda S_w w, \quad (3.37)$$

其中 λ 是拉格朗日乘子。注意到 $\mathbf{S}_b \mathbf{w}$ 的方向恒为 $\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1$, 不妨令

$$\mathbf{S}_b \mathbf{w} = \lambda(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1), \quad (3.38)$$

代入式(3.37)即得

$$\mathbf{w} = \mathbf{S}_w^{-1}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1). \quad (3.39)$$

奇异值分解参见附录
A.3.

考虑到数值解的稳定性, 在实践中通常是对 \mathbf{S}_w 进行奇异值分解, 即 $\mathbf{S}_w = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, 这里 $\boldsymbol{\Sigma}$ 是一个实对角矩阵, 其对角线上的元素是 \mathbf{S}_w 的奇异值, 然后再由 $\mathbf{S}_w^{-1} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^T$ 得到 \mathbf{S}_w^{-1} .

参见习题 7.5.

值得一提的是, LDA 可从贝叶斯决策理论的角度来阐释, 并可证明, 当两类数据同先验、满足高斯分布且协方差相等时, LDA 可达到最优分类。

可以将 LDA 推广到多分类任务中。假定存在 N 个类, 且第 i 类示例数为 m_i 。我们先定义“全局散度矩阵”

$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_b + \mathbf{S}_w \\ &= \sum_{i=1}^m (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T, \end{aligned} \quad (3.40)$$

其中 $\boldsymbol{\mu}$ 是所有示例的均值向量。将类内散度矩阵 \mathbf{S}_w 重定义为每个类别的散度矩阵之和, 即

$$\mathbf{S}_w = \sum_{i=1}^N \mathbf{S}_{w_i}, \quad (3.41)$$

其中

$$\mathbf{S}_{w_i} = \sum_{\mathbf{x} \in X_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T. \quad (3.42)$$

由式(3.40)~(3.42)可得

$$\begin{aligned} \mathbf{S}_b &= \mathbf{S}_t - \mathbf{S}_w \\ &= \sum_{i=1}^N m_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T. \end{aligned} \quad (3.43)$$

显然, 多分类 LDA 可以有多种实现方法: 使用 \mathbf{S}_b , \mathbf{S}_w , \mathbf{S}_t 三者中的任何两个即可。常见的一种实现是采用优化目标

$$\max_{\mathbf{W}} \frac{\text{tr}(\mathbf{W}^T \mathbf{S}_b \mathbf{W})}{\text{tr}(\mathbf{W}^T \mathbf{S}_w \mathbf{W})}, \quad (3.44)$$

其中 $\mathbf{W} \in \mathbb{R}^{d \times (N-1)}$, $\text{tr}(\cdot)$ 表示矩阵的迹(trace). 式(3.44)可通过如下广义特征值问题求解:

$$\mathbf{S}_b \mathbf{W} = \lambda \mathbf{S}_w \mathbf{W}. \quad (3.45)$$

\mathbf{W} 的闭式解则是 $\mathbf{S}_w^{-1} \mathbf{S}_b$ 的 $N - 1$ 个最大广义特征值所对应的特征向量组成的矩阵.

若将 \mathbf{W} 视为一个投影矩阵, 则多分类 LDA 将样本投影到 $N - 1$ 维空间, $N - 1$ 通常远小于数据原有的属性数. 于是, 可通过这个投影来减小样本点的维数, 且投影过程中使用了类别信息, 因此 LDA 也常被视为一种经典的监督降维技术.

降维参见第 10 章.

3.5 多分类学习

例如上一节最后介绍的
LDA 推广.

通常称分类学习器为
“分类器” (classifier).

关于多个分类器的集成,
参见第 8 章.

OvR 亦称 OvA (One vs.
All), 但 OvA 这个说法不严
格, 因为不可能把 “所有
类” 作为反类.

亦可根据各分类器的预
测置信度等信息进行集成,
参见 8.4 节.

现实中常遇到多分类学习任务. 有些二分类学习方法可直接推广到多分类, 但在更多情形下, 我们是基于一些基本策略, 利用二分类学习器来解决多分类问题.

不失一般性, 考虑 N 个类别 C_1, C_2, \dots, C_N , 多分类学习的基本思路是 “拆解法”, 即将多分类任务拆为若干个二分类任务求解. 具体来说, 先对问题进行拆分, 然后为拆出的每个二分类任务训练一个分类器; 在测试时, 对这些分类器的预测结果进行集成以获得最终的多分类结果. 这里的关键是如何对多分类任务进行拆分, 以及如何对多个分类器进行集成. 本节主要介绍拆分策略.

最经典的拆分策略有三种: “一对一” (One vs. One, 简称 OvO)、 “一对其余” (One vs. Rest, 简称 OvR) 和 “多对多” (Many vs. Many, 简称 MvM).

给定数据集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$, $y_i \in \{C_1, C_2, \dots, C_N\}$. OvO 将这 N 个类别两两配对, 从而产生 $N(N - 1)/2$ 个二分类任务, 例如 OvO 将为区分类别 C_i 和 C_j 训练一个分类器, 该分类器把 D 中的 C_i 类样例作为正例, C_j 类样例作为反例. 在测试阶段, 新样本将同时提交给所有分类器, 于是我们将得到 $N(N - 1)/2$ 个分类结果, 最终结果可通过投票产生: 即把被预测得最多的类别作为最终分类结果. 图 3.4 给出了一个示意图.

OvR 则是每次将一个类的样例作为正例、所有其他类的样例作为反例来训练 N 个分类器. 在测试时若仅有一个分类器预测为正类, 则对应的类别标记作为最终分类结果, 如图 3.4 所示. 若有多个分类器预测为正类, 则通常考虑各

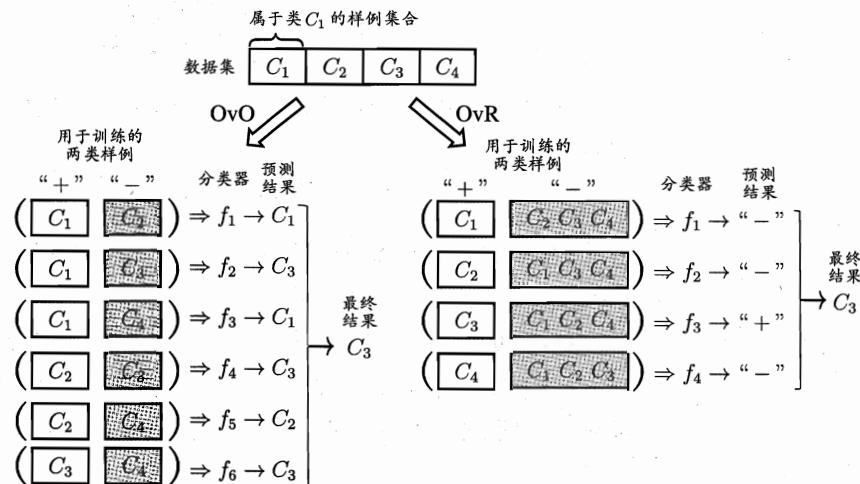


图 3.4 OvO 与 OvR 示意图

分类器的预测置信度，选择置信度最大的类别标记作为分类结果。

容易看出，OvR 只需训练 N 个分类器，而 OvO 需训练 $N(N - 1)/2$ 个分类器，因此，OvO 的存储开销和测试时间开销通常比 OvR 更大。但在训练时，OvR 的每个分类器均使用全部训练样例，而 OvO 的每个分类器仅用到两个类的样例，因此，在类别很多时，OvO 的训练时间开销通常比 OvR 更小。至于预测性能，则取决于具体的数据分布，在多数情形下两者差不多。

MvM 是每次将若干个类作为正类，若干个其他类作为反类。显然，OvO 和 OvR 是 MvM 的特例。MvM 的正、反类构造必须有特殊的设计，不能随意选取。这里我们介绍一种最常用的 MvM 技术：“纠错输出码”(Error Correcting Output Codes, 简称 ECOC)。

ECOC [Dietterich and Bakiri, 1995] 是将编码的思想引入类别拆分，并尽可能在解码过程中具有容错性。ECOC 工作过程主要分为两步：

- 编码：对 N 个类别做 M 次划分，每次划分将一部分类别划为正类，一部分划为反类，从而形成一个二分类训练集；这样一共产生 M 个训练集，可训练出 M 个分类器。
- 解码： M 个分类器分别对测试样本进行预测，这些预测标记组成一个编码。将这个预测编码与每个类别各自的编码进行比较，返回其中距离最小的类别作为最终预测结果。

类别划分通过“编码矩阵”(coding matrix)指定。编码矩阵有多种形式，常见的主要有二元码 [Dietterich and Bakiri, 1995] 和三元码 [Allwein et al., 2000]。前者将每个类别分别指定为正类和反类，后者在正、反类之外，还可指定“停用类”。图 3.5 给出了一个示意图，在图 3.5(a)中，分类器 f_2 将 C_1 类和 C_3 类的样例作为正例， C_2 类和 C_4 类的样例作为反例；在图 3.5(b)中，分类器 f_4 将 C_1 类和 C_4 类的样例作为正例， C_3 类的样例作为反例。在解码阶段，各分类器的预测结果联合起来形成了测试示例的编码，该编码与各类所对应的编码进行比较，将距离最小的编码所对应的类别作为预测结果。例如在图 3.5(a)中，若基于欧氏距离，预测结果将是 C_3 。

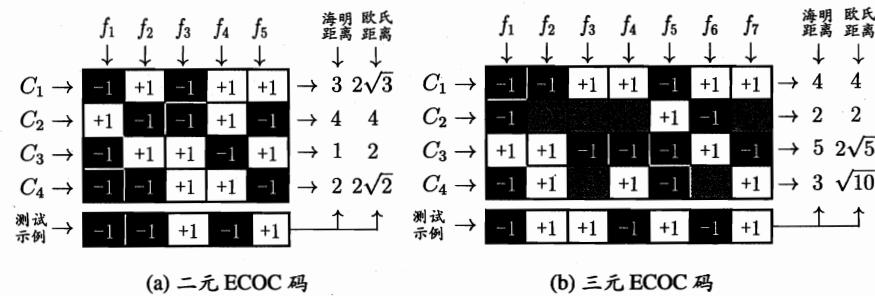


图 3.5 ECOC 编码示意图。“+1”、“-1”分别表示学习器 f_i 将该类样本作为正、反例；三元码中“0”表示 f_i 不使用该类样本

为什么称为“纠错输出码”呢？这是因为在测试阶段，ECOC 编码对分类器的错误有一定的容忍和修正能力。例如图 3.5(a)中对测试示例的正确预测编码是 $(-1, +1, +1, -1, +1)$ ，假设在预测时某个分类器出错了，例如 f_2 出错从而导致了错误编码 $(-1, -1, +1, -1, +1)$ ，但基于这个编码仍能产生正确的最终分类结果 C_3 。一般来说，对同一个学习任务，ECOC 编码越长，纠错能力越强。然而，编码越长，意味着所需训练的分类器越多，计算、存储开销都会增大；另一方面，对有限类别数，可能的组合数目是有限的，码长超过一定范围后就失去了意义。

对同等长度的编码，理论上来说，任意两个类别之间的编码距离越远，则纠错能力越强。因此，在码长较小时可根据这个原则计算出理论最优编码。然而，码长稍大一些就难以有效地确定最优编码，事实上这是 NP 难问题。不过，通常我们并不需获得理论最优编码，因为非最优编码在实践中往往已能产生足够好的分类器。另一方面，并不是编码的理论性质越好，分类性能就越好，因为机器

学习问题涉及很多因素,例如将多个类拆解为两个“类别子集”,不同拆解方式所形成的两个类别子集的区分难度往往不同,即其导致的二分类问题的难度不同;于是,一个理论纠错性质很好、但导致的二分类问题较难的编码,与另一个理论纠错性质差一些、但导致的二分类问题较简单的编码,最终产生的模型性能孰强孰弱很难说。

3.6 类别不平衡问题

前面介绍的分类学习方法都有一个共同的基本假设,即不同类别的训练样例数目相当。如果不同类别的训练样例数目稍有差别,通常影响不大,但若差别很大,则会对学习过程造成困扰。例如有998个反例,但正例只有2个,那么学习方法只需返回一个永远将新样本预测为反例的学习器,就能达到99.8%的精度;然而这样的学习器往往没有价值,因为它不能预测出任何正例。

类别不平衡(class-imbalance)就是指分类任务中不同类别的训练样例数目差别很大的情况。不失一般性,本节假定正类样例较少,反类样例较多。在现实的分类学习任务中,我们经常会遇到类别不平衡,例如在通过拆分法解决多分类问题时,即使原始问题中不同类别的训练样例数目相当,在使用OvR、MvM策略后产生的二分类任务仍可能出现类别不平衡现象,因此有必要了解类别不平衡性处理的基本方法。

从线性分类器的角度讨论容易理解,在我们用 $y = \mathbf{w}^T \mathbf{x} + b$ 对新样本 \mathbf{x} 进行分类时,事实上是在用预测出的 y 值与一个阈值进行比较,例如通常在 $y > 0.5$ 时判别为正例,否则为反例。 y 实际上表达了正例的可能性,几率 $\frac{y}{1-y}$ 则反映了正例可能性与反例可能性之比值,阈值设置为0.5恰表明分类器认为真实正、反例可能性相同,即分类器决策规则为

$$\text{若 } \frac{y}{1-y} > 1 \text{ 则 预测为正例.} \quad (3.46)$$

然而,当训练集中正、反例的数目不同时,令 m^+ 表示正例数目, m^- 表示反例数目,则观测几率是 $\frac{m^+}{m^-}$,由于我们通常假设训练集是真实样本总体的无偏采样,因此观测几率就代表了真实几率。于是,只要分类器的预测几率高于观测几率就应判定为正例,即

$$\text{若 } \frac{y}{1-y} > \frac{m^+}{m^-} \text{ 则 预测为正例.} \quad (3.47)$$

无偏采样意味着真实样本总体的类别比例在训练集中得以保持。

但是, 我们的分类器是基于式(3.46)进行决策, 因此, 需对其预测值进行调整, 使其在基于式(3.46)决策时, 实际是在执行式(3.47). 要做到这一点很容易, 只需令

$$\frac{y'}{1-y'} = \frac{y}{1-y} \times \frac{m^-}{m^+}. \quad (3.48)$$

亦称“再平衡” (rebalance).

这就是类别不平衡学习的一个基本策略——“再缩放” (rescaling).

欠采样亦称“下采样” (downsampling), 过采样亦称“上采样” (oversampling).

再缩放的思想虽简单, 但实际操作却并不平凡, 主要因为“训练集是真实样本总体的无偏采样”这个假设往往并不成立, 也就是说, 我们未必能有效地基于训练集观测几率来推断出真实几率. 现有技术大体上有三类做法: 第一类是直接对训练集里的反类样例进行“欠采样” (undersampling), 即去除一些反例使得正、反例数目接近, 然后再进行学习; 第二类是对训练集里的正类样例进行“过采样” (oversampling), 即增加一些正例使得正、反例数目接近, 然后再进行学习; 第三类则是直接基于原始训练集进行学习, 但在用训练好的分类器进行预测时, 将式(3.48)嵌入到其决策过程中, 称为“阈值移动” (threshold-moving).

欠采样法的时间开销通常远小于过采样法, 因为前者丢弃了很多反例, 使得分类器训练集远小于初始训练集, 而过采样法增加了很多正例, 其训练集大于初始训练集. 需注意的是, 过采样法不能简单地对初始正例样本进行重复采样, 否则会招致严重的过拟合; 过采样法的代表性算法 SMOTE [Chawla et al., 2002] 是通过对训练集里的正例进行插值来产生额外的正例. 另一方面, 欠采样法若随机丢弃反例, 可能丢失一些重要信息; 欠采样法的代表性算法 EasyEnsemble [Liu et al., 2009] 则是利用集成学习机制, 将反例划分为若干个集合供不同学习器使用, 这样对每个学习器来看都进行了欠采样, 但在全局来看却不会丢失重要信息.

代价敏感学习研究非均等代价下的学习. 参见 2.3.4 节.

值得一提的是, “再缩放” 也是“代价敏感学习” (cost-sensitive learning) 的基础. 在代价敏感学习中将式(3.48)中的 m^-/m^+ 用 $cost^+/cost^-$ 代替即可, 其中 $cost^+$ 是将正例误分为反例的代价, $cost^-$ 是将反例误分为正例的代价.

3.7 阅读材料

参见第 11 章.

“稀疏表示” (sparse representation) 近年来很受关注, 但即便对多元线性回归这样简单的模型, 获得具有最优“稀疏性” (sparsity) 的解也并不容易. 稀疏性问题本质上对应了 L_0 范数的优化, 这在通常条件下是 NP 难问题. LASSO [Tibshirani, 1996] 通过 L_1 范数来近似 L_0 范数, 是求取稀疏解的重要技术.

可以证明, OvO 和 OvR 都是 ECOC 的特例 [Allwein et al., 2000]. 人们以往希望设计通用的编码法, [Crammer and Singer, 2002] 提出要考虑问题本身的特点, 设计“问题依赖”的编码法, 并证明寻找最优的离散编码矩阵是一个 NP 完全问题. 此后, 有多种问题依赖的 ECOC 编码法被提出, 通常是通过找出具有代表性的二分类问题来进行编码 [Pujol et al., 2006, 2008]. [Escalera et al., 2010] 开发了一个开源 ECOC 库.

MvM 除了 ECOC 还可有其他实现方式, 例如 DAG (Directed Acyclic Graph) 拆分法 [Platt et al., 2000] 将类别划分表达成树形结构, 每个结点对应于一个二类分类器. 还有一些工作是致力于直接求解多分类问题, 例如多类支持向量机方面的一些研究 [Crammer and Singer, 2001; Lee et al., 2004].

代价敏感学习中研究得最多的是基于类别的“误分类代价” (misclassification cost), 代价矩阵如表 2.2 所示; 本书在提及代价敏感学习时, 默认指此类情形. 已经证明, 对二分类任务可通过“再缩放”获得理论最优解 [Elkan, 2001], 但对多分类任务, 仅在某些特殊情形下存在闭式解 [Zhou and Liu, 2006a]. 非均等代价和类别不平衡性虽然都可借助“再缩放”技术, 但两者本质不同 [Zhou and Liu, 2006b]. 需注意的是, 类别不平衡学习中通常是较小类的代价更高, 否则无需进行特殊处理.

多分类学习中虽然有多个类别, 但每个样本仅属于一个类别. 如果希望为一个样本同时预测出多个类别标记, 例如一幅图像可同时标注为“蓝天”、“白云”、“羊群”、“自然场景”, 这样的任务就不再是多分类学习, 而是“多标记学习” (multi-label learning), 这是机器学习中近年来相当活跃的一个研究领域. 对多标记学习感兴趣的读者可参阅 [Zhang and Zhou, 2014].

习题

西瓜数据集 3.0 α 见 p.89
的表 4.5.

UCI 数据集见
<http://archive.ics.uci.edu/ml/>.

线性可分是指存在线性
超平面能将不同类的样本
点分开。参见 6.3 节。

- 3.1 试析在什么情形下式(3.2)中不必考虑偏置项 b .
- 3.2 试证明, 对于参数 w , 对率回归的目标函数(3.18)是非凸的, 但其对数似然函数(3.27)是凸的.
- 3.3 编程实现对率回归, 并给出西瓜数据集 3.0 α 上的结果.
- 3.4 选择两个 UCI 数据集, 比较 10 折交叉验证法和留一法所估计出的对率回归的错误率.
- 3.5 编程实现线性判别分析, 并给出西瓜数据集 3.0 α 上的结果.
- 3.6 线性判别分析仅在线性可分数据上能获得理想结果, 试设计一个改进方法, 使其能较好地用于非线性可分数据
- 3.7 令码长为 9, 类别数为 4, 试给出海明距离意义下理论最优的 ECOC 二元码并证明之.
- 3.8* ECOC 编码能起到理想纠错作用的重要条件是: 在每一位编码上出错的概率相当且独立. 试析多分类任务经 ECOC 编码后产生的二类分类器满足该条件的可能性及由此产生的影响.
- 3.9 使用 OvR 和 MvM 将多分类任务分解为二分类任务求解时, 试述为何无需专门针对类别不平衡性进行处理.
- 3.10* 试推导出多分类代价敏感学习(仅考虑基于类别的误分类代价)使用“再缩放”能获得理论最优解的条件.

参考文献

- Allwein, E. L., R. E. Schapire, and Y. Singer. (2000). "Reducing multiclass to binary: A unifying approach for margin classifiers." *Journal of Machine Learning Research*, 1:113–141.
- Boyd, S. and L. Vandenberghe. (2004). *Convex Optimization*. Cambridge University Press, Cambridge, UK.
- Chawla, N. V., K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. (2002). "SMOTE: Synthetic minority over-sampling technique." *Journal of Artificial Intelligence Research*, 16:321–357.
- Crammer, K. and Y. Singer. (2001). "On the algorithmic implementation of multiclass kernel-based vector machines." *Journal of Machine Learning Research*, 2:265–292.
- Crammer, K. and Y. Singer. (2002). "On the learnability and design of output codes for multiclass problems." *Machine Learning*, 47(2-3):201–233.
- Dietterich, T. G. and G. Bakiri. (1995). "Solving multiclass learning problems via error-correcting output codes." *Journal of Artificial Intelligence Research*, 2:263–286.
- Elkan, C. (2001). "The foundations of cost-sensitive learning." In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 973–978, Seattle, WA.
- Escalera, S., O. Pujol, and P. Radeva. (2010). "Error-correcting ouput codes library." *Journal of Machine Learning Research*, 11:661–664.
- Fisher, R. A. (1936). "The use of multiple measurements in taxonomic problems." *Annals of Eugenics*, 7(2):179–188.
- Lee, Y., Y. Lin, and G. Wahba. (2004). "Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data." *Journal of the American Statistical Association*, 99 (465):67–81.
- Liu, X.-Y., J. Wu, and Z.-H. Zhou. (2009). "Exploratory undersampling for class-imbalance learning." *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 39(2):539–550.
- Platt, J. C., N. Cristianini, and J. Shawe-Taylor. (2000). "Large margin DAGs

- for multiclass classification.” In *Advances in Neural Information Processing Systems 12 (NIPS)* (S. A. Solla, T. K. Leen, and K.-R. Müller, eds.), MIT Press, Cambridge, MA.
- Pujol, O., S. Escalera, and P. Radeva. (2008). “An incremental node embedding technique for error correcting output codes.” *Pattern Recognition*, 41(2):713–725.
- Pujol, O., P. Radeva, and J. Vitrià. (2006). “Discriminant ECOC: A heuristic method for application dependent design of error correcting output codes.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):1007–1012.
- Tibshirani, R. (1996). “Regression shrinkage and selection via the LASSO.” *Journal of the Royal Statistical Society: Series B*, 58(1):267–288.
- Zhang, M.-L. and Z.-H. Zhou. (2014). “A review on multi-label learning algorithms.” *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837.
- Zhou, Z.-H. and X.-Y. Liu. (2006a). “On multi-class cost-sensitive learning.” In *Proceeding of the 21st National Conference on Artificial Intelligence (AAAI)*, 567–572, Boston, WA.
- Zhou, Z.-H. and X.-Y. Liu. (2006b). “Training cost-sensitive neural networks with methods addressing the class imbalance problem.” *IEEE Transactions on Knowledge and Data Engineering*, 18(1):63–77.

休息一会儿

小故事：关于“最小二乘法”

1801年，意大利天文学家皮亚齐发现了1号小行星“谷神星”，但在跟踪观测了40天后，因谷神星转至太阳的背后，皮亚齐失去了谷神星的位置。许多天文学家试图重新找到谷神星，但都徒劳无获。这引起了伟大的德国数学家高斯(1777—1855)的注意，他发明了一种方法，根据皮亚齐的观测数据计算出了谷神星的轨道，后来德国天文学家奥伯斯在高斯预言的时间和星空领域重新找到了谷神星。1809年，高斯在他的著作《天体运动论》中发表了这种方法，即最小二乘法。



(1993年版德国10马克纸币上的高斯像)

另两位是拉格朗日和拉普拉斯，三人姓氏首字母相同，时称“3L”。

1805年，在椭圆积分、数论和几何方面都有重大贡献的法国大数学家勒让德(1752—1833)发表了《计算彗星轨道的新方法》，其附录中描述了最小二乘法。勒让德是法国18—19世纪数学界的三驾马车之一，早已是法国科学院院士。但勒让德的书中没有涉及最小二乘法的误差分析，高斯1809年的著作中包括了这方面的内容，这对最小二乘法用于数理统计、乃至今天的机器学习有极为重要的意义。由于高斯的这一重大贡献，以及他声称自己1799年就已开始使用这个方法，因此很多人将最小二乘法的发明优先权归之为高斯。当时这两位大数学家发生了著名的优先权之争，此后有许多数学史家专门进行研究，但至今也没弄清到底是谁最先发明了最小二乘法。

第4章 决策树

4.1 基本流程

亦称“判定树”。根据上下文，本书中的“决策树”有时是指学习方法，有时是指学得的树。

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，我们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，我们要对“这是好瓜吗？”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”：我们先看“它是什么颜色？”，如果是“青绿色”，则我们再看“它的根蒂是什么形态？”，如果是“蜷缩”，我们再判断“它敲起来是什么声音？”，最后，我们得出最终决策：这是个好瓜。这个决策过程如图 4.1 所示。

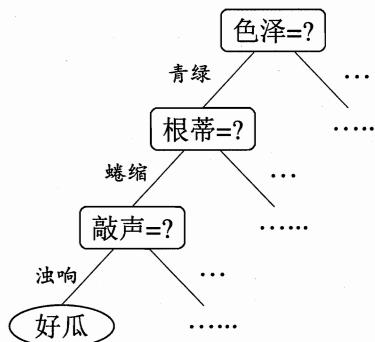


图 4.1 西瓜问题的一棵决策树

显然，决策过程的最终结论对应了我们所希望的判定结果，例如“是”或“不是”好瓜；决策过程中提出的每个判定问题都是对某个属性的“测试”，例如“色澤=?”“根蒂=?”；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，其考虑范围是在上次决策结果的限定范围之内，例如若在“色澤=青绿”之后再判断“根蒂=?”，则仅在考虑青绿色瓜的根蒂。

一般的，一棵决策树包含一个根结点、若干个内部结点和若干个叶结点；

叶结点对应于决策结果, 其他每个结点则对应于一个属性测试; 每个结点包含的样本集合根据属性测试的结果被划分到子结点中; 根结点包含样本全集. 从根结点到每个叶结点的路径对应了一个判定测试序列. 决策树学习的目的是为了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, 如图 4.2 所示.

递归返回, 情形(1).

递归返回, 情形(2).

我们将在下一节讨论如何获得最优划分属性.

递归返回, 情形(3).

从 A 中去掉 a_* .

输入: 训练集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$;
 属性集 $A = \{a_1, a_2, \dots, a_d\}$.
过程: 函数 TreeGenerate(D, A)
 1: 生成结点 node;
 2: **if** D 中样本全属于同一类别 C **then**
 3: 将 node 标记为 C 类叶结点; **return**
 4: **end if**
 5: **if** $A = \emptyset$ OR D 中样本在 A 上取值相同 **then**
 6: 将 node 标记为叶结点, 其类别标记为 D 中样本数最多的类; **return**
 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 中在 a_* 上取值为 a_*^v 的样本子集;
 11: **if** D_v 为空 **then**
 12: 将分支结点标记为叶结点, 其类别标记为 D 中样本最多的类; **return**
 13: **else**
 14: 以 TreeGenerate($D_v, A \setminus \{a_*\}$) 为分支结点
 15: **end if**
 16: **end for**
输出: 以 node 为根结点的一棵决策树

图 4.2 决策树学习基本算法

显然, 决策树的生成是一个递归过程. 在决策树基本算法中, 有三种情形会导致递归返回: (1) 当前结点包含的样本全属于同一类别, 无需划分; (2) 当前属性集为空, 或是所有样本在所有属性上取值相同, 无法划分; (3) 当前结点包含的样本集合为空, 不能划分.

在第(2)种情形下, 我们把当前结点标记为叶结点, 并将其类别设定为该结点所含样本最多的类别; 在第(3)种情形下, 同样把当前结点标记为叶结点, 但将其类别设定为其父结点所含样本最多的类别. 注意这两种情形的处理实质不同: 情形(2)是在利用当前结点的后验分布, 而情形(3)则是把父结点的样本分布作为当前结点的先验分布.

4.2 划分选择

由算法 4.2 可看出, 决策树学习的关键是第 8 行, 即如何选择最优划分属性。一般而言, 随着划分过程不断进行, 我们希望决策树的分支结点所包含的样本尽可能属于同一类别, 即结点的“纯度”(purity)越来越高。

4.2.1 信息增益

“信息熵”(information entropy)是度量样本集合纯度最常用的一种指标。假定当前样本集合 D 中第 k 类样本所占的比例为 p_k ($k = 1, 2, \dots, |\mathcal{Y}|$), 则 D 的信息熵定义为

计算信息熵时约定: 若
 $p = 0$, 则 $p \log_2 p = 0$.

$$\text{Ent}(D) = - \sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k. \quad (4.1)$$

$\text{Ent}(D)$ 的最小值为 0, $\text{Ent}(D)$ 的值越小, 则 D 的纯度越高。
 最大值为 $\log_2 |\mathcal{Y}|$.

假定离散属性 a 有 V 个可能的取值 $\{a^1, a^2, \dots, a^V\}$, 若使用 a 来对样本集 D 进行划分, 则会产生 V 个分支结点, 其中第 v 个分支结点包含了 D 中所有在属性 a 上取值为 a^v 的样本, 记为 D^v . 我们可根据式(4.1)计算出 D^v 的信息熵, 再考虑到不同的分支结点所包含的样本数不同, 给分支结点赋予权重 $|D^v|/|D|$, 即样本数越多的分支结点的影响越大, 于是可计算出用属性 a 对样本集 D 进行划分所获得的“信息增益”(information gain)

$$\text{Gain}(D, a) = \text{Ent}(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Ent}(D^v). \quad (4.2)$$

一般而言, 信息增益越大, 则意味着使用属性 a 来进行划分所获得的“纯度提升”越大。因此, 我们可用信息增益来进行决策树的划分属性选择, 即在图 4.2 算法第 8 行选择属性 $a_* = \arg \max_{a \in A} \text{Gain}(D, a)$. 著名的 ID3 决策树学习算

ID3 名字中的 ID 是 Iterative Dichotomiser (迭代二分器)的简称。

法 [Quinlan, 1986] 就是以信息增益为准则来选择划分属性。

以表 4.1 中的西瓜数据集 2.0 为例, 该数据集包含 17 个训练样例, 用以学习一棵能预测没剖开的是不是好瓜的决策树。显然, $|\mathcal{Y}| = 2$ 。在决策树学习开始时, 根结点包含 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(4.1)可计算出根结点的信息熵为

$$\text{Ent}(D) = - \sum_{k=1}^2 p_k \log_2 p_k = - \left(\frac{8}{17} \log_2 \frac{8}{17} + \frac{9}{17} \log_2 \frac{9}{17} \right) = 0.998.$$

表 4.1 西瓜数据集 2.0

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

然后, 我们要计算出当前属性集合 {色泽, 根蒂, 敲声, 纹理, 脐部, 触感} 中每个属性的信息增益。以属性“色泽”为例, 它有 3 个可能的取值: {青绿, 乌黑, 浅白}。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 (色泽=青绿), D^2 (色泽=乌黑), D^3 (色泽=浅白)。

子集 D^1 包含编号为 {1, 4, 6, 10, 13, 17} 的 6 个样例, 其中正例占 $p_1 = \frac{3}{6}$, 反例占 $p_2 = \frac{3}{6}$; D^2 包含编号为 {2, 3, 7, 8, 9, 15} 的 6 个样例, 其中正、反例分别占 $p_1 = \frac{4}{6}$, $p_2 = \frac{2}{6}$; D^3 包含编号为 {5, 11, 12, 14, 16} 的 5 个样例, 其中正、反例分别占 $p_1 = \frac{1}{5}$, $p_2 = \frac{4}{5}$ 。根据式(4.1)可计算出用“色泽”划分之后所获得的 3 个分支结点的信息熵为

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

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

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

于是, 根据式(4.2)可计算出属性“色泽”的信息增益为

$$\begin{aligned}
 \text{Gain}(D, \text{色泽}) &= \text{Ent}(D) - \sum_{v=1}^3 \frac{|D^v|}{|D|} \text{Ent}(D^v) \\
 &= 0.998 - \left(\frac{6}{17} \times 1.000 + \frac{6}{17} \times 0.918 + \frac{5}{17} \times 0.722 \right) \\
 &= 0.109 .
 \end{aligned}$$

类似的，我们可计算出其他属性的信息增益：

$$\text{Gain}(D, \text{根蒂}) = 0.143; \quad \text{Gain}(D, \text{敲声}) = 0.141;$$

$$\text{Gain}(D, \text{纹理}) = 0.381; \quad \text{Gain}(D, \text{脐部}) = 0.289;$$

$$\text{Gain}(D, \text{触感}) = 0.006.$$

显然，属性“纹理”的信息增益最大，于是它被选为划分属性。图 4.3 给出了基于“纹理”对根结点进行划分的结果，各分支结点所包含的样例子集显示在结点中。

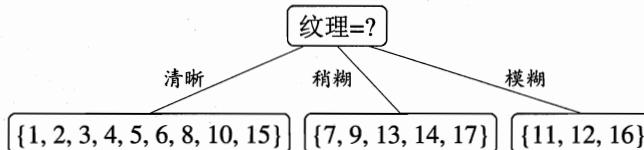


图 4.3 基于“纹理”属性对根结点划分

“纹理”不再作为候选
划分属性。

然后，决策树学习算法将对每个分支结点做进一步划分。以图 4.3 中第一个分支结点（“纹理=清晰”）为例，该结点包含的样例集合 D^1 中有编号为 {1, 2, 3, 4, 5, 6, 8, 10, 15} 的 9 个样例，可用属性集合为 {色泽, 根蒂, 敲声, 脐部, 触感}。基于 D^1 计算出各属性的信息增益：

$$\text{Gain}(D^1, \text{色泽}) = 0.043; \quad \text{Gain}(D^1, \text{根蒂}) = 0.458;$$

$$\text{Gain}(D^1, \text{敲声}) = 0.331; \quad \text{Gain}(D^1, \text{脐部}) = 0.458;$$

$$\text{Gain}(D^1, \text{触感}) = 0.458.$$

“根蒂”、“脐部”、“触感”3 个属性均取得了最大的信息增益，可任选其中之一作为划分属性。类似的，对每个分支结点进行上述操作，最终得到的决策树如图 4.4 所示。

4.2.2 增益率

在上面的介绍中，我们有意忽略了表 4.1 中的“编号”这一列。若把“编

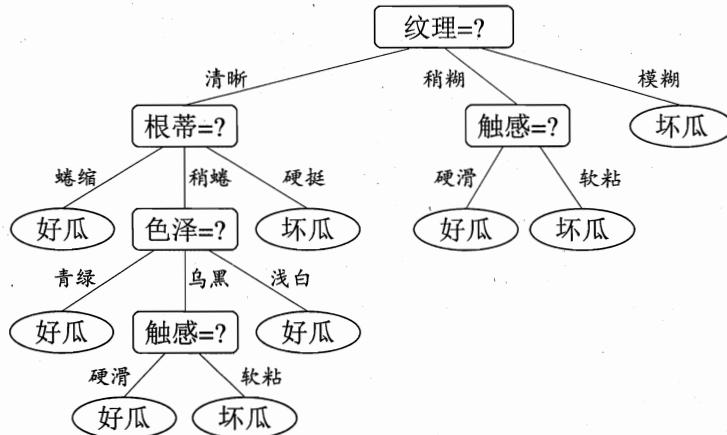


图 4.4 在西瓜数据集 2.0 上基于信息增益生成的决策树

号”也作为一个候选划分属性，则根据式(4.2)可计算出它的信息增益为 0.998，远大于其他候选划分属性。这很容易理解：“编号”将产生 17 个分支，每个分支结点仅包含一个样本，这些分支结点的纯度已达最大。然而，这样的决策树显然不具有泛化能力，无法对新样本进行有效预测。

实际上，信息增益准则对可取值数目较多的属性有所偏好，为减少这种偏好可能带来的不利影响，著名的 C4.5 决策树算法 [Quinlan, 1993] 不直接使用信息增益，而是使用“增益率”(gain ratio) 来选择最优划分属性。采用与式(4.2)相同的符号表示，增益率定义为

$$\text{Gain_ratio}(D, a) = \frac{\text{Gain}(D, a)}{\text{IV}(a)}, \quad (4.3)$$

其中

$$\text{IV}(a) = - \sum_{v=1}^V \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|} \quad (4.4)$$

称为属性 a 的“固有值”(intrinsic value) [Quinlan, 1993]。属性 a 的可能取值数目越多(即 V 越大)，则 $\text{IV}(a)$ 的值通常会越大。例如，对表 4.1 的西瓜数据集 2.0，有 $\text{IV}(\text{触感}) = 0.874$ ($V = 2$)， $\text{IV}(\text{色泽}) = 1.580$ ($V = 3$)， $\text{IV}(\text{编号}) = 4.088$ ($V = 17$)。

需注意的是，增益率准则对可取值数目较少的属性有所偏好，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

[Quinlan, 1993]: 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.

CART 是 Classification and Regression Tree 的简称, 这是一种著名的决策树学习算法, 分类和回归任务都可用.

4.2.3 基尼指数

CART 决策树 [Breiman et al., 1984] 使用“基尼指数”(Gini index)来选择划分属性. 采用与式(4.1)相同的符号, 数据集 D 的纯度可用基尼值来度量:

$$\begin{aligned} \text{Gini}(D) &= \sum_{k=1}^{|Y|} \sum_{k' \neq k} p_k p_{k'} \\ &= 1 - \sum_{k=1}^{|Y|} p_k^2. \end{aligned} \quad (4.5)$$

直观来说, $\text{Gini}(D)$ 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, $\text{Gini}(D)$ 越小, 则数据集 D 的纯度越高.

采用与式(4.2)相同的符号表示, 属性 a 的基尼指数定义为

$$\text{Gini_index}(D, a) = \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Gini}(D^v). \quad (4.6)$$

于是, 我们在候选属性集合 A 中, 选择那个使得划分后基尼指数最小的属性作为最优划分属性, 即 $a_* = \arg \min_{a \in A} \text{Gini_index}(D, a)$.

4.3 剪枝处理

关于过拟合, 参见2.1节.

剪枝(pruning)是决策树学习算法对付“过拟合”的主要手段. 在决策树学习中, 为了尽可能正确分类训练样本, 结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”了, 以致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”(prepruning)和“后剪枝”(post-pruning) [Quinlan, 1993]. 预剪枝是指在决策树生成过程中, 对每个结点在划分前先进行估计, 若当前结点的划分不能带来决策树泛化性能提升, 则停止划分并将当前结点标记为叶结点; 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结点进行考察, 若将该结点对应的子树替换为叶结点能

带来决策树泛化性能提升，则将该子树替换为叶结点。

如何判断决策树泛化性能是否提升呢？这可使用2.2节介绍的性能评估方法。本节假定采用留出法，即预留一部分数据用作“验证集”以进行性能评估。例如对表4.1的西瓜数据集2.0，我们将其随机划分为两部分，如表4.2所示，编号为{1, 2, 3, 6, 7, 10, 14, 15, 16, 17}的样例组成训练集，编号为{4, 5, 8, 9, 11, 12, 13}的样例组成验证集。

表4.2 西瓜数据集2.0划分出的训练集(双线上部)与验证集(双线下部)

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否
编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否

假定我们采用4.2.1节的信息增益准则来进行划分属性选择，则从表4.2的训练集将会生成一棵如图4.5所示的决策树。为便于讨论，我们对图中的部分结点做了编号。

4.3.1 预剪枝

我们先讨论预剪枝。基于信息增益准则，我们会选取属性“脐部”来对训练集进行划分，并产生3个分支，如图4.6所示。然而，是否应该进行这个划分呢？预剪枝要对划分前后的泛化性能进行估计。

在划分之前，所有样例集中在根结点。若不进行划分，则根据算法4.2第6行，该结点将被标记为叶结点，其类别标记为训练样例数最多的类别，假设我们

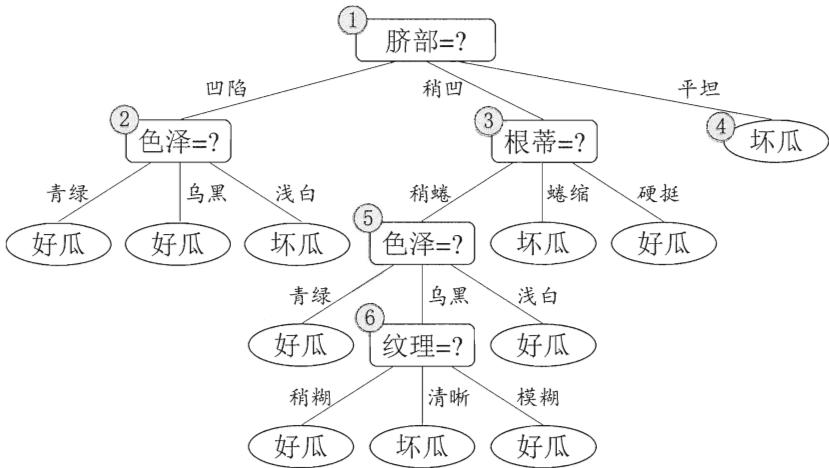


图 4.5 基于表 4.2 生成的未剪枝决策树

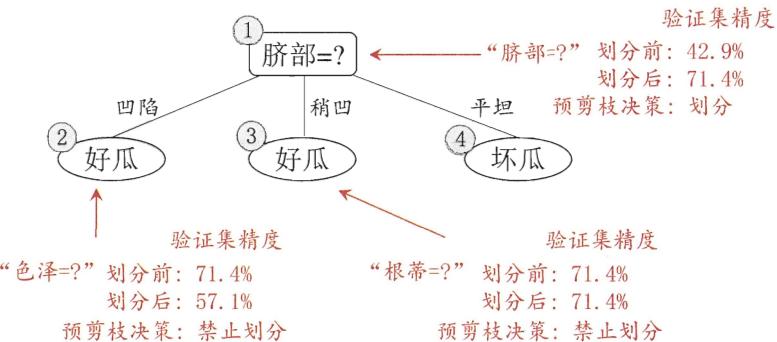


图 4.6 基于表 4.2 生成的预剪枝决策树

当样例最多的类不唯一时，可任选其中一类。

将这个叶结点标记为“好瓜”。用表 4.2 的验证集对这个单结点决策树进行评估，则编号为 {4,5,8} 的样例被分类正确，另外 4 个样例分类错误，于是，验证集精度为 $\frac{3}{7} \times 100\% = 42.9\%$ 。

在用属性“脐部”划分之后，图 4.6 中的结点②、③、④ 分别包含编号为 {1,2,3,14}、{6,7,15,17}、{10,16} 的训练样例，因此这 3 个结点分别被标记为叶结点“好瓜”、“好瓜”、“坏瓜”。此时，验证集中编号为 {4,5,8,11,12} 的样例被分类正确，验证集精度为 $\frac{5}{7} \times 100\% = 71.4\% > 42.9\%$ 。于是，用“脐部”进行划分得以确定。

然后, 决策树算法应该对结点②进行划分, 基于信息增益准则将挑选出划分属性“色泽”. 然而, 在使用“色泽”划分后, 编号为{5}的验证集样本分类结果会由正确转为错误, 使得验证集精度下降为 57.1%. 于是, 预剪枝策略将禁止结点②被划分.

对结点③, 最优划分属性为“根蒂”, 划分后验证集精度仍为 71.4%. 这个划分不能提升验证集精度, 于是, 预剪枝策略禁止结点③被划分.

对结点④, 其所含训练样例已属于同一类, 不再进行划分.

于是, 基于预剪枝策略从表 4.2 数据所生成的决策树如图 4.6 所示, 其验证集精度为 71.4%. 这是一棵仅有一层划分的决策树, 亦称“决策树桩”(decision stump).

对比图 4.6 和图 4.5 可看出, 预剪枝使得决策树的很多分支都没有“展开”, 这不仅降低了过拟合的风险, 还显著减少了决策树的训练时间开销和测试时间开销. 但另一方面, 有些分支的当前划分虽不能提升泛化性能、甚至可能导致泛化性能暂时下降, 但在其基础上进行的后续划分却有可能导致性能显著提高; 预剪枝基于“贪心”本质禁止这些分支展开, 给预剪枝决策树带来了欠拟合的风险.

4.3.2 后剪枝

后剪枝先从训练集生成一棵完整决策树, 例如基于表 4.2 的数据我们得到如图 4.5 所示的决策树. 易知, 该决策树的验证集精度为 42.9%.

后剪枝首先考察图 4.5 中的结点⑥. 若将其领衔的分支剪除, 则相当于把⑥ 替换为叶结点. 替换后的叶结点包含编号为{7, 15}的训练样本, 于是, 该叶结点的类别标记为“好瓜”, 此时决策树的验证集精度提高至 57.1%. 于是, 后剪枝策略决定剪枝, 如图 4.7 所示.

然后考察结点⑤, 若将其领衔的子树替换为叶结点, 则替换后的叶结点包含编号为{6, 7, 15}的训练样例, 叶结点类别标记为“好瓜”, 此时决策树验证集精度仍为 57.1%. 于是, 可以不进行剪枝.

对结点②, 若将其领衔的子树替换为叶结点, 则替换后的叶结点包含编号为{1, 2, 3, 14}的训练样例, 叶结点标记为“好瓜”. 此时决策树的验证集精度提高至 71.4%. 于是, 后剪枝策略决定剪枝.

对结点③和①, 若将其领衔的子树替换为叶结点, 则所得决策树的验证集精度分别为 71.4% 与 42.9%, 均未得到提高. 于是它们被保留.

此种情形下验证集精度虽无提高, 但根据奥卡姆剃刀准则, 剪枝后的模型更好. 因此, 实际的决策树算法在此种情形下通常要进行剪枝. 本书为绘图的方便, 采取了不剪枝的保守策略.

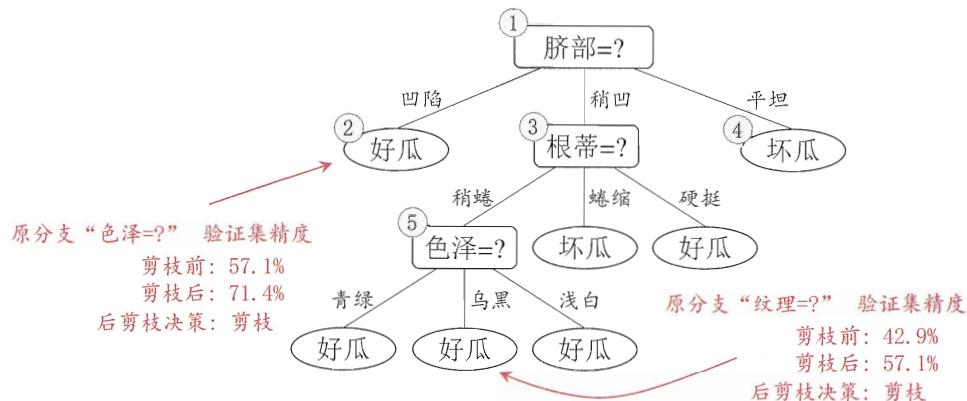


图 4.7 基于表 4.2 生成的后剪枝决策树

最终, 基于后剪枝策略从表 4.2 数据所生成的决策树如图 4.7 所示, 其验证集精度为 71.4%.

对比图 4.7 和图 4.6 可看出, 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的所有非叶结点进行逐一考察, 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.

4.4 连续与缺失值

4.4.1 连续值处理

到目前为止我们仅讨论了基于离散属性来生成决策树. 现实学习任务中常会遇到连续属性, 有必要讨论如何在决策树学习中使用连续属性.

由于连续属性的可取值数目不再有限, 因此, 不能直接根据连续属性的可取值来对结点进行划分. 此时, 连续属性离散化技术可派上用场. 最简单的策略是采用二分法(bi-partition)对连续属性进行处理, 这正是 C4.5 决策树算法中采用的机制 [Quinlan, 1993].

给定样本集 D 和连续属性 a , 假定 a 在 D 上出现了 n 个不同的取值, 将这些值从小到大进行排序, 记为 $\{a^1, a^2, \dots, a^n\}$. 基于划分点 t 可将 D 分为子集 D_t^- 和 D_t^+ , 其中 D_t^- 包含那些在属性 a 上取值不大于 t 的样本, 而 D_t^+ 则包含那些在属性 a 上取值大于 t 的样本. 显然, 对相邻的属性取值 a^i 与 a^{i+1} 来说, t

在区间 $[a^i, a^{i+1})$ 中取任意值所产生的划分结果相同。因此，对连续属性 a ，我们可考察包含 $n - 1$ 个元素的候选划分点集合

$$T_a = \left\{ \frac{a^i + a^{i+1}}{2} \mid 1 \leq i \leq n - 1 \right\}, \quad (4.7)$$

可将划分点设为该属性在训练集中出现的不大于中位点的最大值，从而使得最终决策树使用的划分点都在训练集中出现过 [Quinlan, 1993]。

即把区间 $[a^i, a^{i+1})$ 的中位点 $\frac{a^i + a^{i+1}}{2}$ 作为候选划分点。然后，我们就可像离散属性值一样来考察这些划分点，选取最优的划分点进行样本集合的划分。例如，可对式(4.2)稍加改造：

$$\begin{aligned} \text{Gain}(D, a) &= \max_{t \in T_a} \text{Gain}(D, a, t) \\ &= \max_{t \in T_a} \text{Ent}(D) - \sum_{\lambda \in \{-, +\}} \frac{|D_t^\lambda|}{|D|} \text{Ent}(D_t^\lambda), \end{aligned} \quad (4.8)$$

其中 $\text{Gain}(D, a, t)$ 是样本集 D 基于划分点 t 二分后的信息增益。于是，我们就可选择使 $\text{Gain}(D, a, t)$ 最大化的划分点。

作为一个例子，我们在表 4.1 的西瓜数据集 2.0 上增加两个连续属性“密度”和“含糖率”，得到表 4.3 所示的西瓜数据集 3.0。下面我们用这个数据集来生成一棵决策树。

表 4.3 西瓜数据集 3.0

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	0.774	0.376	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	0.634	0.264	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	0.608	0.318	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	0.556	0.215	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	0.403	0.237	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	0.360	0.370	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	0.719	0.103	否

对属性“密度”，在决策树学习开始时，根结点包含的 17 个训练样本在该属性上取值均不同。根据式(4.7)，该属性的候选划分点集合包含 16 个候选值： $T_{\text{密度}} = \{0.244, 0.294, 0.351, 0.381, 0.420, 0.459, 0.518, 0.574, 0.600, 0.621, 0.636, 0.648, 0.661, 0.681, 0.708, 0.746\}$ 。由式(4.8)可计算出属性“密度”的信息增益为 0.262，对应于划分点 0.381。

对属性“含糖率”，其候选划分点集合也包含 16 个候选值： $T_{\text{含糖率}} = \{0.049, 0.074, 0.095, 0.101, 0.126, 0.155, 0.179, 0.204, 0.213, 0.226, 0.250, 0.265, 0.292, 0.344, 0.373, 0.418\}$ 。类似的，根据式(4.8)可计算出其信息增益为 0.349，对应于划分点 0.126。

再由 4.2.1 节可知，表 4.3 的数据上各属性的信息增益为

$$\text{Gain}(D, \text{色泽}) = 0.109; \quad \text{Gain}(D, \text{根蒂}) = 0.143;$$

$$\text{Gain}(D, \text{敲声}) = 0.141; \quad \text{Gain}(D, \text{纹理}) = 0.381;$$

$$\text{Gain}(D, \text{脐部}) = 0.289; \quad \text{Gain}(D, \text{触感}) = 0.006;$$

$$\text{Gain}(D, \text{密度}) = 0.262; \quad \text{Gain}(D, \text{含糖率}) = 0.349.$$

于是，“纹理”被选作根结点划分属性，此后结点划分过程递归进行，最终生成如图 4.8 所示的决策树。

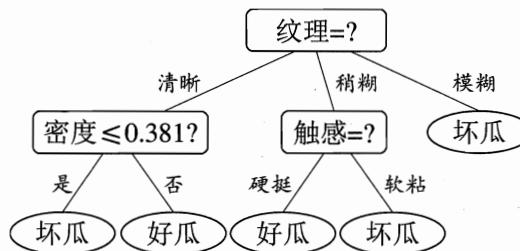


图 4.8 在西瓜数据集 3.0 上基于信息增益生成的决策树

例如在父结点上使用了“密度≤0.381”，不会禁止在子结点上使用“密度≤0.294”。

4.4.2 缺失值处理

现实任务中常会遇到不完整样本，即样本的某些属性值缺失。例如由于检测成本、隐私保护等因素，患者的医疗数据在某些属性上的取值（如 HIV 测试结果）未知；尤其是在属性数目较多的情况下，往往会有大量样本出现缺失值。如果简单地放弃不完整样本，仅使用无缺失值的样本来进行学习，显然是对数

据信息极大的浪费。例如，表 4.4 是表 4.1 中的西瓜数据集 2.0 出现缺失值的版本，如果放弃不完整样本，则仅有编号 {4, 7, 14, 16} 的 4 个样本能被使用。显然，有必要考虑利用有缺失属性值的训练样例来进行学习。

表 4.4 西瓜数据集 2.0 α

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	-	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	-	是
3	乌黑	蜷缩	-	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	-	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	-	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	-	稍凹	硬滑	是
9	乌黑	-	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	-	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	-	否
12	浅白	蜷缩	-	模糊	平坦	软粘	否
13	-	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	-	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	-	沉闷	稍糊	稍凹	硬滑	否

- 我们需解决两个问题：(1) 如何在属性值缺失的情况下进行划分属性选择？
(2) 给定划分属性，若样本在该属性上的值缺失，如何对样本进行划分？

给定训练集 D 和属性 a ，令 \tilde{D} 表示 D 中在属性 a 上没有缺失值的样本子集。对问题(1)，显然我们仅可根据 \tilde{D} 来判断属性 a 的优劣。假定属性 a 有 V 个可取值 $\{a^1, a^2, \dots, a^V\}$ ，令 \tilde{D}^v 表示 \tilde{D} 中在属性 a 上取值为 a^v 的样本子集， \tilde{D}_k 表示 \tilde{D} 中属于第 k 类 ($k = 1, 2, \dots, |\mathcal{Y}|$) 的样本子集，则显然有 $\tilde{D} = \bigcup_{k=1}^{|\mathcal{Y}|} \tilde{D}_k$ ， $\tilde{D} = \bigcup_{v=1}^V \tilde{D}^v$ 。假定我们为每个样本 x 赋予一个权重 w_x ，并定义

$$\rho = \frac{\sum_{x \in \tilde{D}} w_x}{\sum_{x \in D} w_x}, \quad (4.9)$$

$$\tilde{p}_k = \frac{\sum_{x \in \tilde{D}_k} w_x}{\sum_{x \in \tilde{D}} w_x} \quad (1 \leq k \leq |\mathcal{Y}|), \quad (4.10)$$

$$\tilde{r}_v = \frac{\sum_{x \in \tilde{D}^v} w_x}{\sum_{x \in \tilde{D}} w_x} \quad (1 \leq v \leq V). \quad (4.11)$$

在决策树学习开始阶段，
根结点中各样本的权重初
始化为 1。

直观地看, 对属性 a , ρ 表示无缺失值样本所占的比例, \tilde{p}_k 表示无缺失值样本中第 k 类所占的比例, \tilde{r}_v 则表示无缺失值样本中在属性 a 上取值 a^v 的样本所占的比例。显然, $\sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k = 1$, $\sum_{v=1}^V \tilde{r}_v = 1$ 。

基于上述定义, 我们可将信息增益的计算式(4.2)推广为

$$\begin{aligned}\text{Gain}(D, a) &= \rho \times \text{Gain}(\tilde{D}, a) \\ &= \rho \times \left(\text{Ent}(\tilde{D}) - \sum_{v=1}^V \tilde{r}_v \text{Ent}(\tilde{D}^v) \right),\end{aligned}\quad (4.12)$$

其中由式(4.1), 有

$$\text{Ent}(\tilde{D}) = - \sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k \log_2 \tilde{p}_k.$$

对问题(2), 若样本 x 在划分属性 a 上的取值已知, 则将 x 划入与其取值对应的子结点, 且样本权值在子结点中保持为 w_x 。若样本 x 在划分属性 a 上的取值未知, 则将 x 同时划入所有子结点, 且样本权值在与属性值 a^v 对应的子结点中调整为 $\tilde{r}_v \cdot w_x$; 直观地看, 这就是让同一个样本以不同的概率划入到不同的子结点中去。

C4.5 算法使用了上述解决方案 [Quinlan, 1993]。下面我们以表 4.4 的数据集为例来生成一棵决策树。

在学习开始时, 根结点包含样本集 D 中全部 17 个样例, 各样例的权值均为 1。以属性“色泽”为例, 该属性上无缺失值的样例子集 \tilde{D} 包含编号为 {2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17} 的 14 个样例。显然, \tilde{D} 的信息熵为

$$\begin{aligned}\text{Ent}(\tilde{D}) &= - \sum_{k=1}^2 \tilde{p}_k \log_2 \tilde{p}_k \\ &= - \left(\frac{6}{14} \log_2 \frac{6}{14} + \frac{8}{14} \log_2 \frac{8}{14} \right) = 0.985.\end{aligned}$$

令 \tilde{D}^1 , \tilde{D}^2 与 \tilde{D}^3 分别表示在属性“色泽”上取值为“青绿”“乌黑”以及“浅白”的样本子集, 有

$$\begin{aligned}\text{Ent}(\tilde{D}^1) &= - \left(\frac{2}{4} \log_2 \frac{2}{4} + \frac{2}{4} \log_2 \frac{2}{4} \right) = 1.000, \\ \text{Ent}(\tilde{D}^2) &= - \left(\frac{4}{6} \log_2 \frac{4}{6} + \frac{2}{6} \log_2 \frac{2}{6} \right) = 0.918,\end{aligned}$$

$$\text{Ent}(\tilde{D}^3) = -\left(\frac{0}{4} \log_2 \frac{0}{4} + \frac{4}{4} \log_2 \frac{4}{4}\right) = 0.000,$$

因此, 样本子集 \tilde{D} 上属性 “色泽” 的信息增益为

$$\begin{aligned}\text{Gain}(\tilde{D}, \text{色泽}) &= \text{Ent}(\tilde{D}) - \sum_{v=1}^3 \tilde{r}_v \text{Ent}(\tilde{D}^v) \\ &= 0.985 - \left(\frac{4}{14} \times 1.000 + \frac{6}{14} \times 0.918 + \frac{4}{14} \times 0.000 \right) \\ &= 0.306.\end{aligned}$$

于是, 样本集 D 上属性 “色泽” 的信息增益为

$$\text{Gain}(D, \text{色泽}) = \rho \times \text{Gain}(\tilde{D}, \text{色泽}) = \frac{14}{17} \times 0.306 = 0.252.$$

类似地可计算出所有属性在 D 上的信息增益:

$$\text{Gain}(D, \text{色泽}) = 0.252; \quad \text{Gain}(D, \text{根蒂}) = 0.171;$$

$$\text{Gain}(D, \text{敲声}) = 0.145; \quad \text{Gain}(D, \text{纹理}) = 0.424;$$

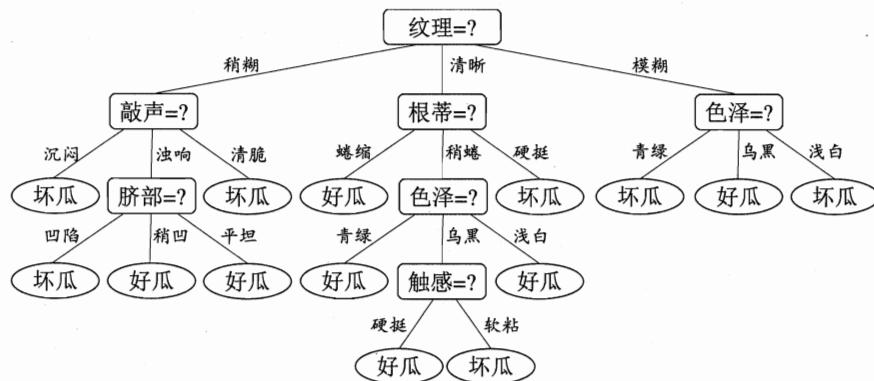
$$\text{Gain}(D, \text{脐部}) = 0.289; \quad \text{Gain}(D, \text{触感}) = 0.006.$$

“纹理” 在所有属性中取得了最大的信息增益, 被用于对根结点进行划分. 划分结果是使编号为 $\{1, 2, 3, 4, 5, 6, 15\}$ 的样本进入 “纹理=清晰” 分支, 编号为 $\{7, 9, 13, 14, 17\}$ 的样本进入 “纹理=稍糊” 分支, 而编号为 $\{11, 12, 16\}$ 的样本进入 “纹理=模糊” 分支, 且样本在各子结点中的权重保持为 1. 需注意的是, 编号为 $\{8\}$ 的样本在属性 “纹理” 上出现了缺失值, 因此它将同时进入三个分支中, 但权重在三个子结点中分别调整为 $\frac{7}{15}$ 、 $\frac{5}{15}$ 和 $\frac{3}{15}$. 编号为 $\{10\}$ 的样本有类似划分结果.

上述结点划分过程递归执行, 最终生成的决策树如图 4.9 所示.

4.5 多变量决策树

若我们把每个属性视为坐标空间中的一个坐标轴, 则 d 个属性描述的样本就对应了 d 维空间中的一个数据点, 对样本分类则意味着在这个坐标空间中寻找不同类样本之间的分类边界. 决策树所形成的分类边界有一个明显的特点: 轴平行(axis-parallel), 即它的分类边界由若干个与坐标轴平行的分段组成.

图 4.9 在西瓜数据集 2.0α 上基于信息增益生成的决策树

以表 4.5 中的西瓜数据 3.0α 为例, 将它作为训练集可学得图 4.10 所示的决策树, 这棵树所对应的分类边界如图 4.11 所示.

西瓜数据集 3.0α 是由表 4.3 的西瓜数据集 3.0 忽略离散属性而得.

表 4.5 西瓜数据集 3.0α

编号	密度	含糖率	好瓜
1	0.697	0.460	是
2	0.774	0.376	是
3	0.634	0.264	是
4	0.608	0.318	是
5	0.556	0.215	是
6	0.403	0.237	是
7	0.481	0.149	是
8	0.437	0.211	是
9	0.666	0.091	否
10	0.243	0.267	否
11	0.245	0.057	否
12	0.343	0.099	否
13	0.639	0.161	否
14	0.657	0.198	否
15	0.360	0.370	否
16	0.593	0.042	否
17	0.719	0.103	否

显然, 分类边界的每一段都是与坐标轴平行的. 这样的分类边界使得学习结果有较好的可解释性, 因为每一段划分都直接对应了某个属性取值. 但在学习任务的真实分类边界比较复杂时, 必须使用很多段划分才能获得较好的近似,

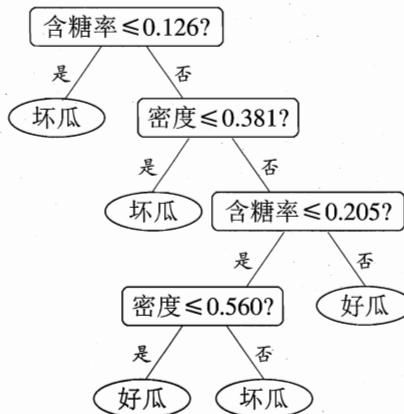
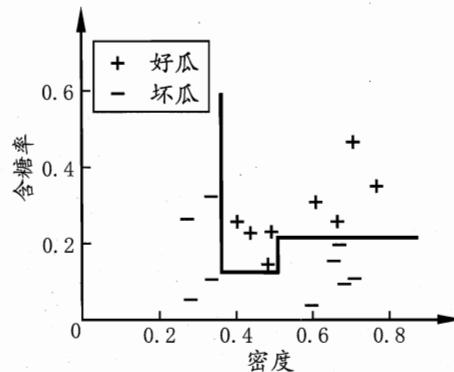
图 4.10 在西瓜数据集 3.0 α 上生成的决策树

图 4.11 图 4.10 决策树对应的分类边界

如图 4.12 所示；此时的决策树会相当复杂，由于要进行大量的属性测试，预测时间开销会很大。

若能使用斜的划分边界，如图 4.12 中红色线段所示，则决策树模型将大为简化。“多变量决策树” (multivariate decision tree) 就是能实现这样的“斜划分”甚至更复杂划分的决策树。以实现斜划分的多变量决策树为例，在此类决策树中，非叶结点不再是仅对某个属性，而是对属性的线性组合进行测试；换言之，每个非叶结点是一个形如 $\sum_{i=1}^d w_i a_i = t$ 的线性分类器，其中 w_i 是属性 a_i 的权重， w_i 和 t 可在该结点所含的样本集和属性集上学得。于是，与传统的“单变量决策树” (univariate decision tree) 不同，在多变量决策树的学习过程中，不是为每个非叶结点寻找一个最优划分属性，而是试图建立一个合适的线性分

这样的多变量决策树亦称“斜决策树” (oblique decision tree)。

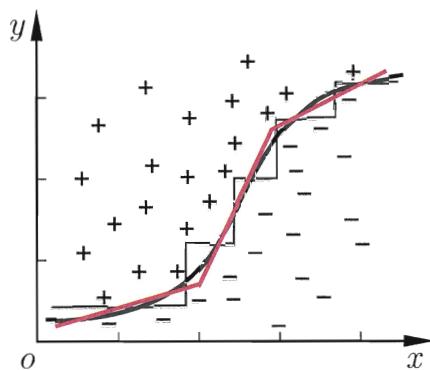


图 4.12 决策树对复杂分类边界的分段近似

线性分类器参见第 3 章. 类器. 例如对西瓜数据 3.0α , 我们可学得图 4.13 这样的多变量决策树, 其分类边界如图 4.14 所示.

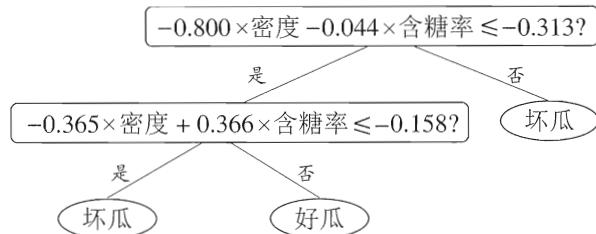
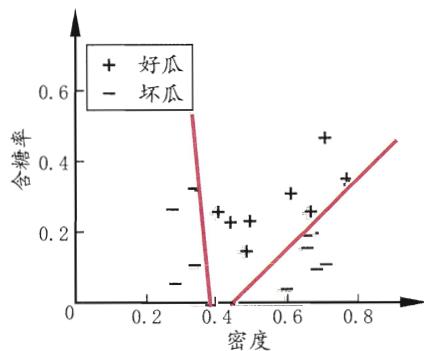
图 4.13 在西瓜数据集 3.0α 上生成的多变量决策树

图 4.14 图 4.13 多变量决策树对应的分类边界

4.6 阅读材料

决策树学习算法最著名的代表是 ID3 [Quinlan, 1979, 1986]、C4.5 [Quinlan, 1993] 和 CART [Breiman et al., 1984]. [Murthy, 1998] 提供了一个关于决策树文献的阅读指南. C4.5Rule 是一个将 C4.5 决策树转化为符号规则的算法 [Quinlan, 1993], 决策树的每个分支可以容易地重写为一条规则, 但 C4.5Rule 算法在转化过程中会进行规则前件合并、删减等操作, 因此最终规则集的泛化性能甚至可能优于原决策树.

本质上, 各种特征选择方法均可用于决策树的划分属性选择. 特征选择参见第 11 章.

在信息增益、增益率、基尼指数之外, 人们还设计了许多其他的准则用于决策树划分选择, 然而有实验研究表明 [Mingers, 1989b], 这些准则虽然对决策树的尺寸有较大影响, 但对泛化性能的影响很有限. [Raileanu and Stoffel, 2004] 对信息增益和基尼指数进行的理论分析也显示出, 它们仅在 2% 的情况下会有所不同. 4.3 节介绍了决策树剪枝的基本策略; 剪枝方法和程度对决策树泛化性能的影响相当显著, 有实验研究表明 [Mingers, 1989a], 在数据带有噪声时通过剪枝甚至可将决策树的泛化性能提高 25%.

多变量决策树算法主要有 OC1 [Murthy et al., 1994] 和 [Brodley and Utgoff, 1995] 提出的一系列算法. OC1 先贪心地寻找每个属性的最优权值, 在局部优化的基础上再对分类边界进行随机扰动以试图找到更好的边界; [Brodley and Utgoff, 1995] 则直接引入了线性分类器学习的最小二乘法. 还有一些算法试图在决策树的叶结点上嵌入神经网络, 以结合这两种学习机制的优势, 例如“感知机树” (Perceptron tree) [Utgoff, 1989b] 在决策树的每个叶结点上训练一个感知机, 而 [Guo and Gelfand, 1992] 则直接在叶结点上嵌入多层神经网络.

关于感知机和神经网络,
参见第 5 章.

有一些决策树学习算法可进行“增量学习” (incremental learning), 即在接收到新样本后可对已学得的模型进行调整, 而不用完全重新学习. 主要机制是通过调整分支路径上的划分属性次序来对树进行部分重构, 代表性算法有 ID4 [Schlimmer and Fisher, 1986]、ID5R [Utgoff, 1989a]、ITI [Utgoff et al., 1997] 等. 增量学习可有效地降低每次接收到新样本后的训练时间开销, 但多步增量学习后的模型会与基于全部数据训练而得的模型有较大差别.

习题

- 4.1 试证明对于不含冲突数据(即特征向量完全相同但标记不同)的训练集, 必存在与训练集一致(即训练误差为 0)的决策树.
- 4.2 试析使用“最小训练误差”作为决策树划分选择准则的缺陷.
- 4.3 试编程实现基于信息熵进行划分选择的决策树算法, 并为表 4.3 中数据生成一棵决策树.
- 4.4 试编程实现基于基尼指数进行划分选择的决策树算法, 为表 4.2 中数据生成预剪枝、后剪枝决策树, 并与未剪枝决策树进行比较.
- 4.5 试编程实现基于对率回归进行划分选择的决策树算法, 并为表 4.3 中数据生成一棵决策树.

UCI 数据集见
<http://archive.ics.uci.edu/ml/>. 4.6 试选择 4 个 UCI 数据集, 对上述 3 种算法所产生的未剪枝、预剪枝、后剪枝决策树进行实验比较, 并进行适当的统计显著性检验.
统计显著性检验参见
2.4 节.

- 4.7 图 4.2 是一个递归算法, 若面临巨量数据, 则决策树的层数会很深, 使用递归方法易导致“栈”溢出. 试使用“队列”数据结构, 以参数 $MaxDepth$ 控制树的最大深度, 写出与图 4.2 等价、但不使用递归的决策树生成算法.
- 4.8* 试将决策树生成的深度优先搜索过程修改为广度优先搜索, 以参数 $MaxNode$ 控制树的最大结点数, 将题 4.7 中基于队列的决策树算法进行改写. 对比题 4.7 中的算法, 试析哪种方式更易于控制决策树所需存储不超出内存.
- 4.9 试将 4.4.2 节对缺失值的处理机制推广到基尼指数的计算中去.
- 4.10 从网上下载或自己编程实现任意一种多变量决策树算法, 并观察其在西瓜数据集 3.0 上产生的结果.

西瓜数据集 3.0 见 p.84
的表 4.3.

参考文献

- Breiman, L., J. Friedman, C. J. Stone, and R. A. Olshen. (1984). *Classification and Regression Trees*. Chapman & Hall/CRC, Boca Raton, FL.
- Brodley, C. E. and P. E. Utgoff. (1995). "Multivariate decision trees." *Machine Learning*, 19(1):45–77.
- Guo, H. and S. B. Gelfand. (1992). "Classification trees with neural network feature extraction." *IEEE Transactions on Neural Networks*, 3(6):923–933.
- Mingers, J. (1989a). "An empirical comparison of pruning methods for decision tree induction." *Machine Learning*, 4(2):227–243.
- Mingers, J. (1989b). "An empirical comparison of selection measures for decision-tree induction." *Machine Learning*, 3(4):319–342.
- Murthy, S. K. (1998). "Automatic construction of decision trees from data: A multi-disciplinary survey." *Data Mining and Knowledge Discovery*, 2(4): 345–389.
- Murthy, S. K., S. Kasif, and S. Salzberg. (1994). "A system for induction of oblique decision trees." *Journal of Artificial Intelligence Research*, 2:1–32.
- Quinlan, J. R. (1979). "Discovering rules by induction from large collections of examples." In *Expert Systems in the Micro-electronic Age* (D. Michie, ed.), 168–201, Edinburgh University Press, Edinburgh, UK.
- Quinlan, J. R. (1986). "Induction of decision trees." *Machine Learning*, 1(1): 81–106.
- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.
- Raileanu, L. E. and K. Stoffel. (2004). "Theoretical comparison between the Gini index and information gain criteria." *Annals of Mathematics and Artificial Intelligence*, 41(1):77–93.
- Schlittner, J. C. and D. Fisher. (1986). "A case study of incremental concept induction." In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI)*, 495–501, Philadelphia, PA.
- Utgoff, P. E. (1989a). "Incremental induction of decision trees." *Machine Learning*, 4(2):161–186.

- Utgoff, P. E. (1989b). "Perceptron trees: A case study in hybrid concept representations." *Connection Science*, 1(4):377–391.
- Utgoff, P. E., N. C. Berkman, and J. A. Clouse. (1997). "Decision tree induction based on efficient tree restructuring." *Machine Learning*, 29(1):5–44.

休息一会儿

小故事：决策树与罗斯·昆兰

说起决策树学习，就必然要谈到澳大利亚计算机科学家罗斯·昆兰 (J. Ross Quinlan, 1943—)。

最初的决策树算法是心理学家兼计算机科学家 E. B. Hunt 1962 年在研究人类的概念学习过程时提出的 CLS (Concept Learning System)，这个算法确立了决策树“分而治之”的学习策略。罗斯·昆兰在 Hunt 的指导下于 1968 年在美国华盛顿大学获得计算机博士学位，然后到悉尼大学任教。1978 年他在学术假时到斯坦福大学访问，选修了图灵的助手 D. Michie 开设的一门研究生课程。课上有一个大作业，要求写程序来学习出完备正确的规则，以判断国际象棋残局中一方是否会在两步棋后被将死。昆兰写了一个类似于 CLS 的程序来完成作业，其中最重要的改进是引入了信息增益准则。后来他把这个工作整理出来在 1979 年发表，这就是 ID3 算法。



1986 年 *Machine Learning* 杂志创刊，昆兰应邀在创刊号上重新发表了 ID3 算法，掀起了决策树研究的热潮。短短几年间众多决策树算法问世，ID4、ID5 等名字迅速被其他研究者提出的算法占用，昆兰只好将自己的 ID3 后继算法命名为 C4.0，在此基础上进一步提出了著名的 C4.5。有趣的是，昆兰自称 C4.5 仅是对 C4.0 做了些小改进，因此将它命名为“第 4.5 代分类器”，而将后续的商业化版本称为 C5.0。

C4.0 是 Classifier 4.0 的简称。

C4.5 在 WEKA 中的实现称为 J4.8。

第5章 神经网络

5.1 神经元模型

本书所谈的是“人工神经网络”，不是生物学意义上的神经网络。

这是 T. Kohonen 1988 年在 *Neural Networks* 创刊号上给出的定义。

neuron 亦称 unit.

亦称 bias。注意不是“阈值”，虽然其含义的确类似于“阀门”。

神经网络(neural networks)方面的研究很早就已出现，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，本书采用目前使用得最广泛的一种，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作出的交互反应”[Kohonen, 1988]。我们在机器学习中谈论神经网络时指的是“神经网络学习”，或者说，是机器学习与神经网络这两个学科领域的交叉部分。

神经网络中最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，当它“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”(threshold)，那么它就会被激活，即“兴奋”起来，向其他神经元发送化学物质。

1943年，[McCulloch and Pitts, 1943] 将上述情形抽象为图 5.1 所示的简单模型，这就是一直沿用至今的“M-P 神经元模型”。在这个模型中，神经元接收到来自 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(connection)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

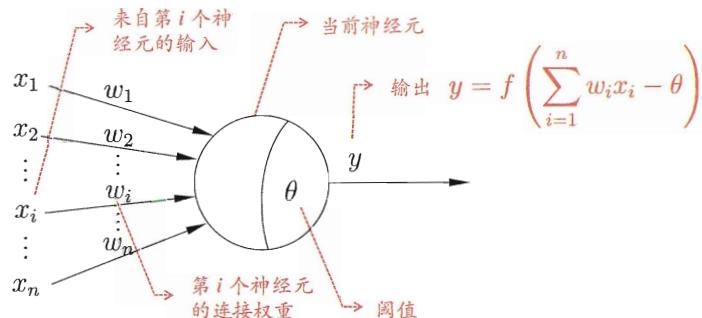


图 5.1 M-P 神经元模型

亦称“响应函数”.

这里的阶跃函数是单位阶跃函数的变体; 对数几率函数则是 Sigmoid 函数的典型代表. 参见 3.3 节.

较, 然后通过“激活函数”(activation function) 处理以产生神经元的输出.

理想中的激活函数是图 5.2(a) 所示的阶跃函数, 它将输入值映射为输出值“0”或“1”, 显然“1”对应于神经元兴奋, “0”对应于神经元抑制. 然而, 阶跃函数具有不连续、不光滑等不太好的性质, 因此实际常用 Sigmoid 函数作为激活函数. 典型的 Sigmoid 函数如图 5.2(b) 所示, 它把可能在较大范围内变化的输入值挤压到 $(0, 1)$ 输出值范围内, 因此有时也称为“挤压函数”(squashing function).

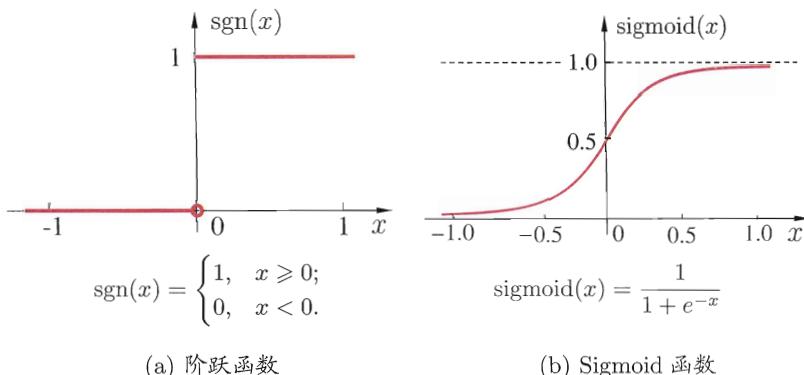


图 5.2 典型的神经元激活函数

把许多个这样的神经元按一定的层次结构连接起来, 就得到了神经网络.

“模拟生物神经网络”是认知科学家对神经网络所做的一个类比阐释.

例如 10 个神经元两两连接, 则有 100 个参数: 90 个连接权和 10 个阈值.

事实上, 从计算机科学的角度看, 我们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, 这个模型是若干个函数, 例如 $y_j = f(\sum_i w_{ij}x_i - \theta_j)$ 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.

5.2 感知机与多层网络

感知机(Perceptron)由两层神经元组成, 如图 5.3 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, 亦称“阈值逻辑单元”(threshold logic unit).

感知机能容易地实现逻辑与、或、非运算. 注意到 $y = f(\sum_i w_i x_i - \theta)$, 假定 f 是图 5.2 中的阶跃函数, 有

- “与”($x_1 \wedge x_2$): 令 $w_1 = w_2 = 1, \theta = 2$, 则 $y = f(1 \cdot x_1 + 1 \cdot x_2 - 2)$, 仅

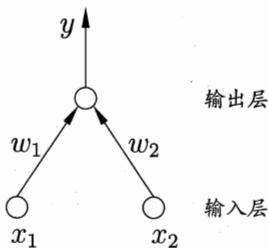


图 5.3 两个输入神经元的感知机网络结构示意图

在 $x_1 = x_2 = 1$ 时, $y = 1$;

- “或” ($x_1 \vee x_2$): 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, $y = 1$;
- “非” ($\neg x_1$): 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, $y = 1$.

更一般地, 给定训练数据集, 权重 w_i ($i = 1, 2, \dots, n$) 以及阈值 θ 可通过学习得到. 阈值 θ 可看作一个固定输入为 -1.0 的“哑结点” (dummy node) 所对应的连接权重 w_{n+1} , 这样, 权重和阈值的学习就可统一为权重的学习. 感知机学习规则非常简单, 对训练样例 (\mathbf{x}, y) , 若当前感知机的输出为 \hat{y} , 则感知机权重将这样调整:

x_i 是 \mathbf{x} 对应于第 i 个输入神经元的分量.

$$w_i \leftarrow w_i + \Delta w_i, \quad (5.1)$$

$$\Delta w_i = \eta(y - \hat{y})x_i, \quad (5.2)$$

η 通常设置为一个小正数, 例如 0.1.

其中 $\eta \in (0, 1)$ 称为学习率 (learning rate). 从式(5.1) 可看出, 若感知机对训练样例 (\mathbf{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.

需注意的是, 感知机只有输出层神经元进行激活函数处理, 即只拥有一层功能神经元 (functional neuron), 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 (linearly separable) 的问题. 可以证明 [Minsky and Papert, 1969], 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, 如图 5.4(a)-(c) 所示, 则感知机的学习过程一定会收敛 (converge) 而求得适当的权向量 $\mathbf{w} = (w_1; w_2; \dots; w_{n+1})$; 否则感知机学习过程将会发生振荡 (fluctuation), \mathbf{w} 难以稳定下来, 不能求得合适解, 例如感知机甚至不能解决如图 5.4(d) 所示的异或这样简单的非线性可分问题.

“非线性可分” 意味着用线性超平面无法划分.

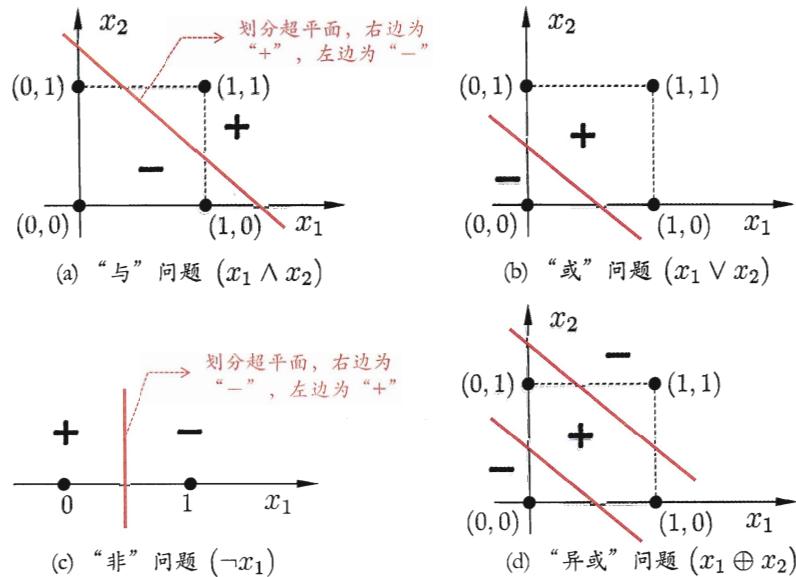


图 5.4 线性可分的“与”“或”“非”问题与非线性可分的“异或”问题

要解决非线性可分问题，需考虑使用多层功能神经元。例如图 5.5 中这个简单的两层感知机就能解决异或问题。在图 5.5(a)中，输出层与输入层之间的一层神经元，被称为隐层或隐含层(hidden layer)，隐含层和输出层神经元都是拥有激活函数的功能神经元。

更一般的，常见的神经网络是形如图 5.6 所示的层级结构，每层神经元与下一层神经元全互连，神经元之间不存在同层连接，也不存在跨层连接。这样的神经网络结构通常称为“多层前馈神经网络”(multi-layer feedforward neural

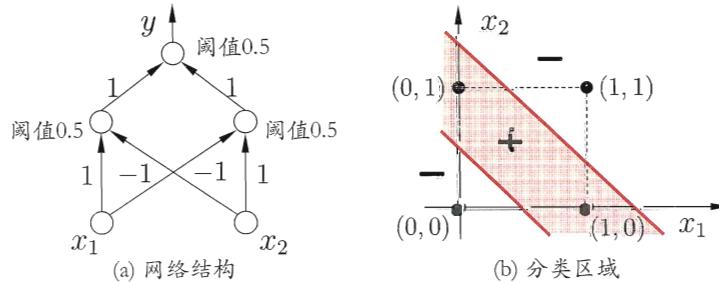


图 5.5 能解决异或问题的两层感知机

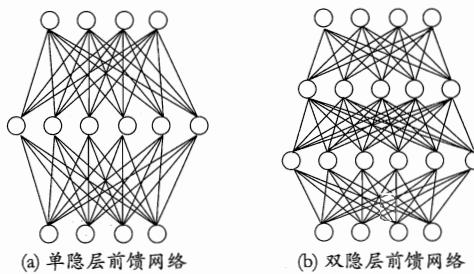


图 5.6 多层前馈神经网络结构示意图

“前馈”并不意味着网络中信号不能向后传，而是指网络拓扑结构上不存在环或回路；参见 5.5.5 节。

即神经元连接的权重。

networks)，其中输入层神经元接收外界输入，隐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，输入层神经元仅是接受输入，不进行函数处理，隐层与输出层包含功能神经元。因此，图 5.6(a) 通常被称为“两层网络”。为避免歧义，本书称其为“单隐层网络”。只需包含隐层，即可称为多层网络。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”(connection weight) 以及每个功能神经元的阈值；换言之，神经网络“学”到的东西，蕴涵在连接权与阈值中。

5.3 误差逆传播算法

亦称“反向传播算法”。

多层网络的学习能力比单层感知机强得多。欲训练多层网络，式(5.1)的简单感知机学习规则显然不够了，需要更强大的学习算法。误差逆传播(error BackPropagation，简称 BP)算法就是其中最杰出的代表，它是迄今最成功的神经网络学习算法。现实任务中使用神经网络时，大多是在使用 BP 算法进行训练。值得指出的是，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，例如训练递归神经网络 [Pineda, 1987]。但通常说“BP 网络”时，一般是指用 BP 算法训练的多层前馈神经网络。

离散属性需先进行处理：若属性值间存在“序”关系则可进行连续化；否则通常转化为 k 维向量， k 为属性值数。参见 3.2 节。

下面我们来看看 BP 算法究竟是什么样。给定训练集 $D = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_m, \mathbf{y}_m)\}$ ， $\mathbf{x}_i \in \mathbb{R}^d$, $\mathbf{y}_i \in \mathbb{R}^l$ ，即输入示例由 d 个属性描述，输出 l 维实值向量。为便于讨论，图 5.7 给出了一个拥有 d 个输入神经元、 l 个输出神经元、 q 个隐层神经元的多层前馈网络结构，其中输出层第 j 个神经元的阈值用 θ_j 表示，隐层第 h 个神经元的阈值用 γ_h 表示。输入层第 i 个神经元与隐层第 h 个神经元之间的连接权为 v_{ih} ，隐层第 h 个神经元与输出层第 j 个神经元之间的连接权为 w_{hj} 。记隐层第 h 个神经元接收到的输入为 $\alpha_h = \sum_{i=1}^d v_{ih} x_i$ ，输出层第 j 个神经元接收到的输入为 $\beta_j = \sum_{h=1}^q w_{hj} b_h$ ，其中 b_h 为隐层第 h 个神经

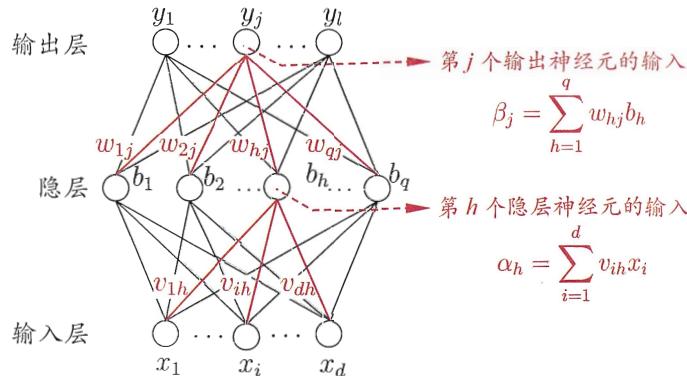


图 5.7 BP 网络及算法中的变量符号

实际是对率函数，参见 3.3 节。元的输出。假设隐层和输出层神经元都使用图 5.2(b) 中的 Sigmoid 函数。

对训练例 $(\mathbf{x}_k, \mathbf{y}_k)$ ，假定神经网络的输出为 $\hat{\mathbf{y}}_k = (\hat{y}_1^k, \hat{y}_2^k, \dots, \hat{y}_l^k)$ ，即

$$\hat{y}_j^k = f(\beta_j - \theta_j), \quad (5.3)$$

则网络在 $(\mathbf{x}_k, \mathbf{y}_k)$ 上的均方误差为

这里的 $1/2$ 是为了后续求导的便利。

$$E_k = \frac{1}{2} \sum_{j=1}^l (\hat{y}_j^k - y_j^k)^2. \quad (5.4)$$

图 5.7 的网络中有 $(d + l + 1)q + l$ 个参数需确定：输入层到隐层的 $d \times q$ 个权值、隐层到输出层的 $q \times l$ 个权值、 q 个隐层神经元的阈值、 l 个输出层神经元的阈值。BP 是一个迭代学习算法，在迭代的每一轮中采用广义的感知机学习规则对参数进行更新估计，即与式(5.1)类似，任意参数 v 的更新估计式为

$$v \leftarrow v + \Delta v. \quad (5.5)$$

下面我们以图 5.7 中隐层到输出层的连接权 w_{hj} 为例来进行推导。

梯度下降参见附录 B.4。

BP 算法基于梯度下降(gradient descent)策略，以目标的负梯度方向对参数进行调整。对式(5.4)的误差 E_k ，给定学习率 η ，有

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}. \quad (5.6)$$

注意到 w_{hj} 先影响到第 j 个输出层神经元的输入值 β_j , 再影响到其输出值 \hat{y}_j^k , 然后影响到 E_k , 有

这就是“链式法则”.

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}. \quad (5.7)$$

根据 β_j 的定义, 显然有

$$\frac{\partial \beta_j}{\partial w_{hj}} = b_h. \quad (5.8)$$

图 5.2 中的 Sigmoid 函数有一个很好的性质:

$$f'(x) = f(x)(1 - f(x)), \quad (5.9)$$

于是根据式(5.4)和(5.3), 有

$$\begin{aligned} g_j &= -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \\ &= -(\hat{y}_j^k - y_j^k)f'(\beta_j - \theta_j) \\ &= \hat{y}_j^k(1 - \hat{y}_j^k)(y_j^k - \hat{y}_j^k). \end{aligned} \quad (5.10)$$

将式(5.10)和(5.8)代入式(5.7), 再代入式(5.6), 就得到了BP 算法中关于 w_{hj} 的更新公式

$$\Delta w_{hj} = \eta g_j b_h. \quad (5.11)$$

类似可得

$$\Delta \theta_j = -\eta g_j, \quad (5.12)$$

$$\Delta v_{ih} = \eta e_h x_i, \quad (5.13)$$

$$\Delta \gamma_h = -\eta e_h, \quad (5.14)$$

式(5.13)和(5.14)中

$$\begin{aligned} e_h &= -\frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \\ &= -\sum_{j=1}^l \frac{\partial E_k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} f'(\alpha_h - \gamma_h) \end{aligned}$$

$$\begin{aligned}
 &= \sum_{j=1}^l w_{hj} g_j f'(\alpha_h - \gamma_h) \\
 &= b_h(1 - b_h) \sum_{j=1}^l w_{hj} g_j .
 \end{aligned} \tag{5.15}$$

学习率 $\eta \in (0, 1)$ 控制着算法每一轮迭代中的更新步长, 若太大则容易振荡, 太小则收敛速度又会过慢. 有时为了做精细调节, 可令式(5.11)与(5.12)使用 η_1 , 式(5.13)与(5.14)使用 η_2 , 两者未必相等.

图 5.8 给出了 BP 算法的工作流程. 对每个训练样例, BP 算法执行以下操作: 先将输入示例提供给输入层神经元, 然后逐层将信号前传, 直到产生输出层的结果; 然后计算输出层的误差(第 4~5 行), 再将误差逆向传播至隐层神经元(第 6 行), 最后根据隐层神经元的误差来对连接权和阈值进行调整(第 7 行). 该迭代过程循环进行, 直到达到某些停止条件为止, 例如训练误差已达到一个很小的值. 图 5.9 给出了在 2 个属性、5 个样本的西瓜数据上, 随着训练轮数的增加, 网络参数和分类边界的变化情况.

停止条件与缓解 BP 过拟合的策略有关.

输入: 训练集 $D = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^m$;
学习率 η .

过程:

- 1: 在 $(0, 1)$ 范围内随机初始化网络中所有连接权和阈值
- 2: **repeat**
- 3: **for all** $(\mathbf{x}_k, \mathbf{y}_k) \in D$ **do**
- 4: 根据当前参数和式(5.3)计算当前样本的输出 $\hat{\mathbf{y}}_k$;
- 5: 根据式(5.10)计算输出层神经元的梯度项 g_j ;
- 6: 根据式(5.15)计算隐层神经元的梯度项 e_h ;
- 7: 根据式(5.11)~(5.14)更新连接权 w_{hj} , v_{ih} 与阈值 θ_j , γ_h
- 8: **end for**
- 9: **until** 达到停止条件

输出: 连接权与阈值确定的多层前馈神经网络

图 5.8 误差逆传播算法

需注意的是, BP 算法的目标是要最小化训练集 D 上的累积误差

$$E = \frac{1}{m} \sum_{k=1}^m E_k , \tag{5.16}$$

但我们上面介绍的“标准 BP 算法”每次仅针对一个训练样例更新连接权和阈值, 也就是说, 图 5.8 中算法的更新规则是基于单个的 E_k 推导而得. 如

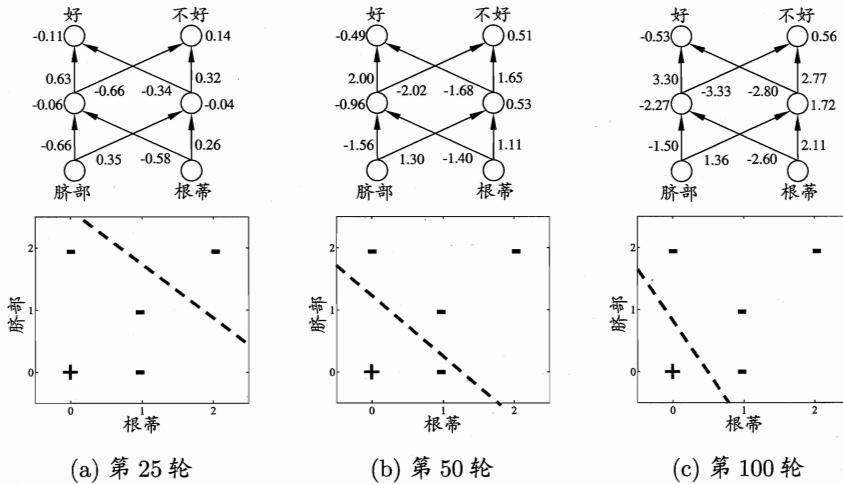


图 5.9 在2个属性、5个样本的西瓜数据上, BP网络参数更新和分类边界的变化情况

果类似地推导出基于累积误差最小化的更新规则, 就得到了累积误差逆传播(accumulated error backpropagation) 算法。累积 BP 算法与标准 BP 算法都很常用。一般来说, 标准 BP 算法每次更新只针对单个样例, 参数更新得非常频繁, 而且对不同样例进行更新的效果可能出现“抵消”现象。因此, 为了达到同样的累积误差极小点, 标准 BP 算法往往需进行更多次数的迭代。累积 BP 算法直接针对累积误差最小化, 它在读取整个训练集 D 一遍后才对参数进行更新, 其参数更新的频率低得多。但在很多任务中, 累积误差下降到一定程度之后, 进一步下降会非常缓慢, 这时标准 BP 往往会更快获得较好的解, 尤其是在训练集 D 非常大时更明显。

读取训练集一遍称为进行了“一轮”(one round, 亦称 one epoch)学习。

标准 BP 算法和累积 BP 算法的区别类似于随机梯度下降(stochastic gradient descent, 简称 SGD)与标准梯度下降之间的区别。

[Hornik et al., 1989] 证明, 只需一个包含足够多神经元的隐层, 多层前馈网络就能以任意精度逼近任意复杂度的连续函数。然而, 如何设置隐层神经元的个数仍是个未决问题, 实际应用中通常靠“试错法”(trial-by-error)调整。

正是由于其强大的表示能力, BP 神经网络经常遭遇过拟合, 其训练误差持续降低, 但测试误差却可能上升。有两种策略常用来缓解BP网络的过拟合。第一种策略是“早停”(early stopping): 将数据分成训练集和验证集, 训练集用来计算梯度、更新连接权和阈值, 验证集用来估计误差, 若训练集误差降低但验证集误差升高, 则停止训练, 同时返回具有最小验证集误差的连接权和阈值。第二种策略是“正则化”(regularization) [Barron, 1991; Girosi et al., 1995], 其基本思想是在误差目标函数中增加一个用于描述网络复杂度的部分, 例如连接

引入正则化策略的神经网络与第 6 章的 SVM 已非常相似。

增加连接权与阈值平方和这一项后, 训练过程将会偏好比较小的连接权和阈值, 使网络输出更加“光滑”, 从而对过拟合有所缓解.

权与阈值的平方和. 仍令 E_k 表示第 k 个训练样例上的误差, w_i 表示连接权和阈值, 则误差目标函数(5.16) 改变为

$$E = \lambda \frac{1}{m} \sum_{k=1}^m E_k + (1 - \lambda) \sum_i w_i^2, \quad (5.17)$$

其中 $\lambda \in (0, 1)$ 用于对经验误差与网络复杂度这两项进行折中, 常通过交叉验证法来估计.

5.4 全局最小与局部极小

若用 E 表示神经网络在训练集上的误差, 则它显然是关于连接权 w 和阈值 θ 的函数. 此时, 神经网络的训练过程可看作一个参数寻优过程, 即在参数空间中, 寻找一组最优参数使得 E 最小.

我们常会谈到两种“最优”: “局部极小” (local minimum) 和“全局最小” (global minimum). 对 w^* 和 θ^* , 若存在 $\epsilon > 0$ 使得

$$\forall (w; \theta) \in \{(w; \theta) \mid \|(w; \theta) - (w^*; \theta^*)\| \leq \epsilon\},$$

都有 $E(w; \theta) \geq E(w^*; \theta^*)$ 成立, 则 $(w^*; \theta^*)$ 为局部极小解; 若对参数空间中的任意 $(w; \theta)$ 都有 $E(w; \theta) \geq E(w^*, \theta^*)$, 则 $(w^*; \theta^*)$ 为全局最小解. 直观地看, 局部极小解是参数空间中的某个点, 其邻域点的误差函数值均不小于该点的函数值; 全局最小解则是指参数空间中所有点的误差函数值均不小于该点的误差函数值. 两者对应的 $E(w^*; \theta^*)$ 分别称为误差函数的局部极小值和全局最小值.

显然, 参数空间内梯度为零的点, 只要其误差函数值小于邻点的误差函数值, 就是局部极小点; 可能存在多个局部极小值, 但却只会有一个全局最小值. 也就是说, “全局最小” 一定是“局部极小”, 反之则不成立. 例如, 图 5.10 中有两个局部极小, 但只有其中之一是全局最小. 显然, 我们在参数寻优过程中是希望找到全局最小.

基于梯度的搜索是使用最为广泛的参数寻优方法. 在此类方法中, 我们从某些初始解出发, 迭代寻找最优参数值. 每次迭代中, 我们先计算误差函数在当前点的梯度, 然后根据梯度确定搜索方向. 例如, 由于负梯度方向是函数值下降最快的方向, 因此梯度下降法就是沿着负梯度方向搜索最优解. 若误差函数在当前点的梯度为零, 则已达到局部极小, 更新量将为零, 这意味着参数的迭代更新将在此停止. 显然, 如果误差函数仅有一个局部极小, 那么此时找到的局部极

这里的讨论对其他机器学习模型同样适用.

感知机更新规则式(5.1)和BP更新规则式(5.11)-(5.14)都是基于梯度下降.

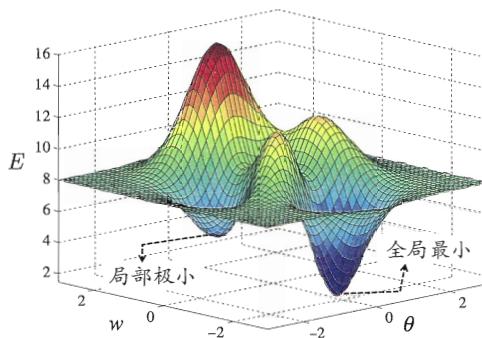


图 5.10 全局最小与局部极小

小就是全局最小; 然而, 如果误差函数具有多个局部极小, 则不能保证找到的解是全局最小. 对后一种情形, 我们称参数寻优陷入了局部极小, 这显然不是我们所希望的.

在现实任务中, 人们常采用以下策略来试图“跳出”局部极小, 从而进一步接近全局最小:

- 以多组不同参数值初始化多个神经网络, 按标准方法训练后, 取其中误差最小的解作为最终参数. 这相当于从多个不同的初始点开始搜索, 这样就可能陷入不同的局部极小, 从中进行选择有可能获得更接近全局最小的结果.
- 使用“模拟退火”(simulated annealing)技术[Aarts and Korst, 1989]. 模拟退火在每一步都以一定的概率接受比当前解更差的结果, 从而有助于“跳出”局部极小. 在每步迭代过程中, 接受“次优解”的概率要随着时间的推移而逐渐降低, 从而保证算法稳定.
- 使用随机梯度下降. 与标准梯度下降法精确计算梯度不同, 随机梯度下降法在计算梯度时加入了随机因素. 于是, 即便陷入局部极小点, 它计算出的梯度仍可能不为零, 这样就有机会跳出局部极小继续搜索.

但是也会造成“跳出”全局最小.

此外, 遗传算法(genetic algorithms) [Goldberg, 1989] 也常用来训练神经网络以更好地逼近全局最小. 需注意的是, 上述用于跳出局部极小的技术大多是启发式, 理论上尚缺乏保障.

5.5 其他常见神经网络

神经网络模型、算法繁多, 本节不能详尽描述, 只对特别常见的几种网络稍作简介.

5.5.1 RBF网络

理论上来说可使用多个隐层, 但常见的 RBF 设置是单隐层.

RBF(Radial Basis Function, 径向基函数)网络 [Broomhead and Lowe, 1988] 是一种单隐层前馈神经网络, 它使用径向基函数作为隐层神经元激活函数, 而输出层则是对隐层神经元输出的线性组合. 假定输入为 d 维向量 \mathbf{x} , 输出为实值, 则 RBF 网络可表示为

$$\varphi(\mathbf{x}) = \sum_{i=1}^q w_i \rho(\mathbf{x}, \mathbf{c}_i), \quad (5.18)$$

其中 q 为隐层神经元个数, \mathbf{c}_i 和 w_i 分别是第 i 个隐层神经元所对应的中心和权重, $\rho(\mathbf{x}, \mathbf{c}_i)$ 是径向基函数, 这是某种沿径向对称的标量函数, 通常定义为样本 \mathbf{x} 到数据中心 \mathbf{c}_i 之间欧氏距离的单调函数. 常用的高斯径向基函数形如

$$\rho(\mathbf{x}, \mathbf{c}_i) = e^{-\beta_i \|\mathbf{x} - \mathbf{c}_i\|^2}. \quad (5.19)$$

[Park and Sandberg, 1991] 证明, 具有足够的隐层神经元的 RBF 网络能以任意精度逼近任意连续函数.

通常采用两步过程来训练 RBF 网络: 第一步, 确定神经元中心 \mathbf{c}_i , 常用的方式包括随机采样、聚类等; 第二步, 利用 BP 算法等来确定参数 w_i 和 β_i .

5.5.2 ART网络

竞争型学习(competitive learning)是神经网络中一种常用的无监督学习策略, 在使用该策略时, 网络的输出神经元相互竞争, 每一时刻仅有一个竞争获胜的神经元被激活, 其他神经元的状态被抑制. 这种机制亦称“胜者通吃”(winner-take-all)原则.

ART(Adaptive Resonance Theory, 自适应谐振理论)网络 [Carpenter and Grossberg, 1987] 是竞争型学习的重要代表. 该网络由比较层、识别层、识别阈值和重置模块构成. 其中, 比较层负责接收输入样本, 并将其传递给识别层神经元. 识别层每个神经元对应一个模式类, 神经元数目可在训练过程中动态增长以增加新的模式类.

在接收到比较层的输入信号后, 识别层神经元之间相互竞争以产生获胜神

模式类可认为是某类别
的“子类”.

这就是“胜者通吃”原则的体现。

经元。竞争的最简单方式是，计算输入向量与每个识别层神经元所对应的模式类的代表向量之间的距离，距离最小者胜。获胜神经元将向其他识别层神经元发送信号，抑制其激活。若输入向量与获胜神经元所对应的代表向量之间的相似度大于识别阈值，则当前输入样本将被归为该代表向量所属类别，同时，网络连接权将会更新，使得以后在接收到相似输入样本时该模式类会计算出更大的相似度，从而使该获胜神经元有更大可能获胜；若相似度不大于识别阈值，则重置模块将在识别层增设一个新的神经元，其代表向量就设置为当前输入向量。

显然，识别阈值对ART网络的性能有重要影响。当识别阈值较高时，输入样本将会被分成比较多、比较精细的模式类，而如果识别阈值较低，则会产生比较少、比较粗略的模式类。

ART比较好地缓解了竞争型学习中的“可塑性-稳定性窘境”(stability-plasticity dilemma)，可塑性是指神经网络要有学习新知识的能力，而稳定性则是指神经网络在学习新知识时要保持对旧知识的记忆。这就使得ART网络具有一个很重要的优点：可进行增量学习(incremental learning)或在线学习(online learning)。

早期的ART网络只能处理布尔型输入数据，此后ART发展成了一个算法族，包括能处理实值输入的ART2网络、结合模糊处理的FuzzyART网络，以及可进行监督学习的ARTMAP网络等。

5.5.3 SOM网络

亦称“自组织特征映射”(Self-Organizing Feature Map)、Kohonen网络。

SOM(Self-Organizing Map，自组织映射)网络[Kohonen, 1982]是一种竞争学习型的无监督神经网络，它能将高维输入数据映射到低维空间(通常为二维)，同时保持输入数据在高维空间的拓扑结构，即将高维空间中相似的样本点映射到网络输出层中的邻近神经元。

如图5.11所示，SOM网络中的输出层神经元以矩阵方式排列在二维空间中，每个神经元都拥有一个权向量，网络在接收输入向量后，将会确定输出层获胜神经元，它决定了该输入向量在低维空间中的位置。SOM的训练目标就是为每个输出层神经元找到合适的权向量，以达到保持拓扑结构的目的。

SOM的训练过程很简单：在接收到一个训练样本后，每个输出层神经元会计算该样本与自身携带的权向量之间的距离，距离最近的神经元成为竞争获胜者，称为最佳匹配单元(best matching unit)。然后，最佳匹配单元及其邻近神经元的权向量将被调整，以使得这些权向量与当前输入样本的距离缩小。这个过程不断迭代，直至收敛。

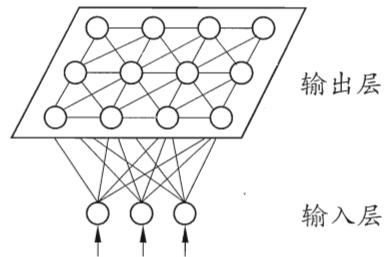


图 5.11 SOM 网络结构

5.5.4 级联相关网络

结构自适应神经网络亦称“构造性”(constructive)神经网络。

5.5.2 节介绍的 ART 网络由于隐层神经元数目可在训练过程中增长，因此也是一种结构自适应神经网络。

一般的神经网络模型通常假定网络结构是事先固定的，训练的目的是利用训练样本来确定合适的连接权、阈值等参数。与此不同，结构自适应网络则将网络结构也当作学习的目标之一，并希望能在训练过程中找到最符合数据特点的网络结构。级联相关(Cascade-Correlation)网络 [Fahlman and Lebiere, 1990] 是结构自适应网络的重要代表。

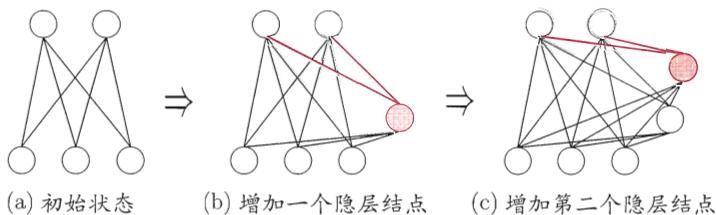


图 5.12 级联相关网络的训练过程。新的隐结点加入时，红色连接权通过最大化新结点的输出与网络误差之间的相关性来进行训练。

级联相关网络有两个主要成分：“级联”和“相关”。级联是指建立层次连接的层级结构。在开始训练时，网络只有输入层和输出层，处于最小拓扑结构；随着训练的进行，如图 5.12 所示，新的隐层神经元逐渐加入，从而创建起层级结构。当新的隐层神经元加入时，其输入端连接权值是冻结固定的。相关是指通过最大化新神经元的输出与网络误差之间的相关性(correlation)来训练相关的参数。

与一般的前馈神经网络相比，级联相关网络无需设置网络层数、隐层神经元数目，且训练速度较快，但其在数据较小时易陷入过拟合。

5.5.5 Elman 网络

亦称“recursive neural networks”.

与前馈神经网络不同，“递归神经网络”(recurrent neural networks)允许网络中出现环形结构，从而可让一些神经元的输出反馈回来作为输入信号。这样的结构与信息反馈过程，使得网络在 t 时刻的输出状态不仅与 t 时刻的输入有关，还与 $t - 1$ 时刻的网络状态有关，从而能处理与时间有关的动态变化。

Elman 网络 [Elman, 1990] 是最常用的递归神经网络之一，其结构如图 5.13 所示，它的结构与多层前馈网络很相似，但隐层神经元的输出被反馈回来，与下一时刻输入层神经元提供的信号一起，作为隐层神经元在下一时刻的输入。隐层神经元通常采用 Sigmoid 激活函数，而网络的训练则常通过推广的 BP 算法进行 [Pineda, 1987]。

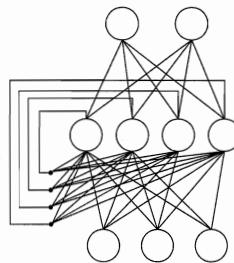


图 5.13 Elman 网络结构

5.5.6 Boltzmann 机

从图 5.14(a) 可看出，Boltzmann 机是一种递归神经网络。

神经网络中有一类模型是为网络状态定义一个“能量”(energy)，能量最小化时网络达到理想状态，而网络的训练就是在最小化这个能量函数。Boltzmann 机 [Ackley et al., 1985] 就是一种“基于能量的模型”(energy-based model)，常见结构如图 5.14(a) 所示，其神经元分为两层：显层与隐层。显层用于表示数据的输入与输出，隐层则被理解为数据的内在表达。Boltzmann 机中的神经元都是布尔型的，即只能取 0、1 两种状态，状态 1 表示激活，状态 0 表示抑制。令向量 $s \in \{0, 1\}^n$ 表示 n 个神经元的状态， w_{ij} 表示神经元 i 与 j 之间的连接权， θ_i 表示神经元 i 的阈值，则状态向量 s 所对应的 Boltzmann 机能量定义为

$$E(s) = - \sum_{i=1}^{n-1} \sum_{j=i+1}^n w_{ij} s_i s_j - \sum_{i=1}^n \theta_i s_i. \quad (5.20)$$

Boltzmann 分布亦称“平衡态”(equilibrium)或“平稳分布”(stationary distribution)。

若网络中的神经元以任意不依赖于输入值的顺序进行更新，则网络最终将达到 Boltzmann 分布，此时状态向量 s 出现的概率将仅由其能量与所有可能状

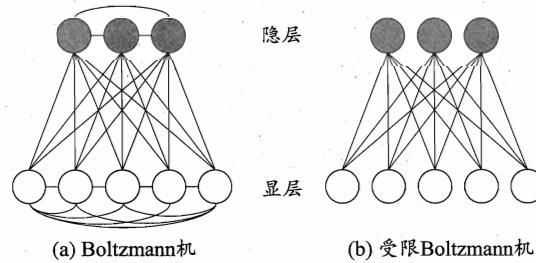


图 5.14 Boltzmann 机与受限 Boltzmann 机

态向量的能量确定:

$$P(\mathbf{s}) = \frac{e^{-E(\mathbf{s})}}{\sum_t e^{-E(t)}} \quad (5.21)$$

Boltzmann 机的训练过程就是将每个训练样本视为一个状态向量, 使其出现的概率尽可能大. 标准的 Boltzmann 机是一个全连接图, 训练网络的复杂度很高, 这使其难以用于解决现实任务. 现实中常采用受限 Boltzmann 机(Restricted Boltzmann Machine, 简称 RBM). 如图 5.14(b) 所示, 受限 Boltzmann 机仅保留显层与隐层之间的连接, 从而将 Boltzmann 机结构由完全图简化为二部图.

受限 Boltzmann 机常用“对比散度”(Contrastive Divergence, 简称 CD)算法 [Hinton, 2010] 来进行训练. 假定网络中有 d 个显层神经元和 q 个隐层神经元, 令 \mathbf{v} 和 \mathbf{h} 分别表示显层与隐层的状态向量, 则由于同一层内不存在连接, 有

$$P(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^d P(v_i | \mathbf{h}), \quad (5.22)$$

$$P(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^q P(h_j | \mathbf{v}). \quad (5.23)$$

CD 算法对每个训练样本 \mathbf{v} , 先根据式(5.23)计算出隐层神经元状态的概率分布, 然后根据这个概率分布采样得到 \mathbf{h} ; 此后, 类似地根据式(5.22)从 \mathbf{h} 产生 \mathbf{v}' , 再从 \mathbf{v}' 产生 \mathbf{h}' ; 连接权的更新公式为

阈值的更新公式可类似
获得.

$$\Delta w = \eta (\mathbf{v}\mathbf{h}^\top - \mathbf{v}'\mathbf{h}'^\top) . \quad (5.24)$$

5.6 深度学习

关于学习器容量，参见第 12 章。

大型深度学习模型中甚至有上百亿个参数。

这里所说的“多隐层”是指三个以上隐层；深度学习模型通常有八九层甚至更多隐层。

理论上来说，参数越多的模型复杂度越高、“容量”(capacity)越大，这意味着它能完成更复杂的学习任务。但一般情形下，复杂模型的训练效率低，易陷入过拟合，因此难以受到人们青睐。而随着云计算、大数据时代的到来，计算能力的大幅提高可缓解训练低效性，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。

典型的深度学习模型就是很深层的神经网络。显然，对神经网络模型，提高容量的一个简单办法是增加隐层的数目。隐层多了，相应的神经元连接权、阈值等参数就会更多。模型复杂度也可通过单纯增加隐层神经元的数目来实现，前面我们谈到过，单隐层的多层前馈网络已具有很强大的学习能力；但从增加模型复杂度的角度来看，增加隐层的数目显然比增加隐层神经元的数目更有效，因为增加隐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的层数。然而，多隐层神经网络难以直接用经典算法(例如标准 BP 算法)进行训练，因为误差在多隐层内逆传播时，往往会“发散”(diverge)而不能收敛到稳定状态。

无监督逐层训练(unsupervised layer-wise training)是多隐层网络训练的有效手段，其基本思想是每次训练一层隐结点，训练时将上一层隐结点的输出作为输入，而本层隐结点的输出作为下一层隐结点的输入，这称为“预训练”(pre-training)；在预训练全部完成后，再对整个网络进行“微调”(fine-tuning)训练。例如，在深度信念网络(deep belief network，简称DBN) [Hinton et al., 2006] 中，每层都是一个受限 Boltzmann 机，即整个网络可视为若干个 RBM 堆叠而得。在使用无监督逐层训练时，首先训练第一层，这是关于训练样本的RBM模型，可按标准的 RBM 训练；然后，将第一层预训练好的隐结点视为第二层的输入结点，对第二层进行预训练；……各层预训练完成后，再利用 BP 算法等对整个网络进行训练。

事实上，“预训练+微调”的做法可视为将大量参数分组，对每组先找到局部看来比较好的设置，然后再基于这些局部较优的结果联合起来进行全局寻优。这样就在利用了模型大量参数所提供的自由度的同时，有效地节省了训练开销。

另一种节省训练开销的策略是“权共享”(weight sharing)，即让一组神经元使用相同的连接权。这个策略在卷积神经网络(Convolutional Neural Network，简称 CNN) [LeCun and Bengio, 1995; LeCun et al., 1998] 中发挥了重要作用。以 CNN 进行手写数字识别任务为例 [LeCun et al., 1998]，如图 5.15

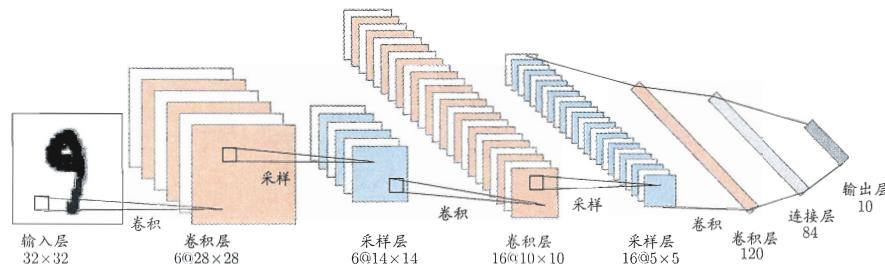


图 5.15 卷积神经网络用于手写数字识别 [LeCun et al., 1998]

近来人们在使用 CNN 时常将 Sigmoid 激活函数替换为修正线性函数

$$f(x) = \begin{cases} 0, & \text{if } x < 0, \\ x, & \text{otherwise,} \end{cases}$$

这样的神经元称为 ReLU(Rectified Linear Unit);此外, 汇合层的操作常采用“最大”或“平均”, 这更接近于集成学习中的一些操作, 参见 8.4 节。

若将网络中前若干层处理都看作是在进行特征表示, 只把最后一层处理看作是在进行“分类”, 则分类使用的就是一个简单模型。

所示, 网络输入是一个 32×32 的手写数字图像, 输出是其识别结果, CNN 复合多个“卷积层”和“采样层”对输入信号进行加工, 然后在连接层实现与输出目标之间的映射。每个卷积层都包含多个特征映射(feature map), 每个特征映射是一个由多个神经元构成的“平面”, 通过一种卷积滤波器提取输入的一种特征。例如, 图 5.15 中第一个卷积层由 6 个特征映射构成, 每个特征映射是一个 28×28 的神经元阵列, 其中每个神经元负责从 5×5 的区域通过卷积滤波器提取局部特征。采样层亦称为“汇合”(pooling)层, 其作用是基于局部相关性原理进行亚采样, 从而在减少数据量的同时保留有用信息。例如图 5.15 中第一个采样层有 6 个 14×14 的特征映射, 其中每个神经元与上一层中对应特征映射的 2×2 邻域相连, 并据此计算输出。通过复合卷积层和采样层, 图 5.15 中的 CNN 将原始图像映射成 120 维特征向量, 最后通过一个由 84 个神经元构成的连接层和输出层连接完成识别任务。CNN 可用 BP 算法进行训练, 但在训练中, 无论是卷积层还是采样层, 其每一组神经元(即图 5.15 中的每个“平面”)都是用相同的连接权, 从而大幅减少了需要训练的参数数目。

我们可以从另一个角度来理解深度学习。无论是 DBN 还是 CNN, 其多隐层堆叠、每层对上一层的输出进行处理的机制, 可看作是在对输入信号进行逐层加工, 从而把初始的、与输出目标之间联系不太密切的输入表示, 转化成与输出目标联系更密切的表示, 使得原来仅基于最后一层输出映射难以完成的任务成为可能。换言之, 通过多层处理, 逐渐将初始的“低层”特征表示转化为“高层”特征表示后, 用“简单模型”即可完成复杂的分类等学习任务。由此可将深度学习理解为进行“特征学习”(feature learning)或“表示学习”(representation learning)。

以往在机器学习用于现实任务时, 描述样本的特征通常需由人类专家来设计, 这称为“特征工程”(feature engineering)。众所周知, 特征的好坏对泛化性

能有至关重要的影响，人类专家设计出好特征也并非易事；特征学习则通过机器学习技术自身来产生好特征，这使机器学习向“全自动数据分析”又前进了一步。

5.7 阅读材料

2012 年前的名称是
IEEE Transactions on Neural Networks.

近来 NIPS 更偏重于机器学习。

LMS 亦称 Widrow-Hoff 规则或 δ 规则。

[Haykin, 1998] 是很好的神经网络教科书，[Bishop, 1995] 则偏重于机器学习和模式识别。神经网络领域的主流学术期刊有 *Neural Computation*、*Neural Networks*、*IEEE Transactions on Neural Networks and Learning Systems*；主要国际学术会议有国际神经信息处理系统会议(NIPS) 和国际神经网络联合会议(IJCNN)，区域性国际会议主要有欧洲神经网络会议(ICANN)和亚太神经网络会议(ICONIP)。

M-P 神经元模型使用最为广泛，但还有一些神经元模型也受到关注，如考虑了电位脉冲发放时间而不仅是累积电位的脉冲神经元(spiking neuron)模型 [Gerstner and Kistler, 2002]。

BP 算法由 [Werbos, 1974] 首先提出，此后 [Rumelhart et al., 1986a,b] 重新发明。BP 算法实质是 LMS (Least Mean Square) 算法的推广。LMS 试图使网络的输出均方误差最小化，可用于神经元激活函数可微的感知机学习；将 LMS 推广到由非线性可微神经元组成的多层前馈网络，就得到 BP 算法，因此 BP 算法亦称广义 δ 规则 [Chauvin and Rumelhart, 1995]。

[MacKay, 1992] 在贝叶斯框架下提出了自动确定神经网络正则化参数的方法。[Gori and Tesi, 1992] 对 BP 网络的局部极小问题进行了详细讨论。[Yao, 1999] 综述了利用以遗传算法为代表的演化计算(evolutionary computation)技术来生成神经网络的研究工作。对 BP 算法的改进有大量研究，例如为了提速，可在训练过程中自适应缩小学习率，即先使用较大的学习率然后逐步缩小，更多“窍门”(trick) 可参阅 [Reed and Marks, 1998; Orr and Müller, 1998]。

关于 RBF 网络训练过程可参阅 [Schwenker et al., 2001]。[Carpenter and Grossberg, 1991] 介绍了 ART 族算法。SOM 网络在聚类、高维数据可视化、图像分割等方面有广泛应用，可参阅 [Kohonen, 2001]。[Bengio et al., 2013] 综述了深度学习方面的研究进展。

神经网络是一种难解释的“黑箱模型”，但已有一些工作尝试改善神经网络的可解释性，主要途径是从神经网络中抽取易于理解的符号规则，可参阅 [Tickle et al., 1998; Zhou, 2004]。

习题

- 5.1** 试述将线性函数 $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ 用作神经元激活函数的缺陷.
- 5.2** 试述使用图 5.2(b) 激活函数的神经元与对率回归的联系.
- 5.3** 对于图 5.7 中的 v_{ih} , 试推导出 BP 算法中的更新公式(5.13).
- 5.4** 试述式(5.6)中学习率的取值对神经网络训练的影响.
- 5.5** 试编程实现标准 BP 算法和累积 BP 算法, 在西瓜数据集 3.0 上分别用这两个算法训练一个单隐层网络, 并进行比较.
- 5.6** 试设计一个 BP 改进算法, 能通过动态调整学习率显著提升收敛速度. 编程实现该算法, 并选择两个 UCI 数据集与标准 BP 算法进行实验比较.
- 5.7** 根据式(5.18)和(5.19), 试构造一个能解决异或问题的单层 RBF 神经网络.
- 5.8** 从网上下载或自己编程实现 SOM 网络, 并观察其在西瓜数据集 3.0 α 上产生的结果.
- 5.9*** 试推导用于 Elman 网络的 BP 算法.
- 5.10** 从网上下载或自己编程实现一个卷积神经网络, 并在手写字符识别数据 MNIST 上进行实验测试.

西瓜数据集 3.0 见 p.84
的表 4.3.

UCI 数据集见
<http://archive.ics.uci.edu/ml/>.

西瓜数据集 3.0 α 见 p.89
的表 4.5.

MNIST 数据集见
<http://yann.lecun.com/exdb/mnist/>.

参考文献

- Aarts, E. and J. Korst. (1989). *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley & Sons, New York, NY.
- Ackley, D. H., G. E. Hinton, and T. J. Sejnowski. (1985). “A learning algorithm for Boltzmann machines.” *Cognitive Science*, 9(1):147–169.
- Barron, A. R. (1991). “Complexity regularization with application to artificial neural networks.” In *Nonparametric Functional Estimation and Related Topics; NATO ASI Series Volume 335* (G. Roussas, ed.), 561–576, Kluwer, Amsterdam, The Netherlands.
- Bengio, Y., A. Courville, and P. Vincent. (2013). “Representation learning: A review and new perspectives.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.
- Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, New York, NY.
- Broomhead, D. S. and D. Lowe. (1988). “Multivariate functional interpolation and adaptive networks.” *Complex Systems*, 2(3):321–355.
- Carpenter, G. A. and S. Grossberg. (1987). “A massively parallel architecture for a self-organizing neural pattern recognition machine.” *Computer Vision, Graphics, and Image Processing*, 37(1):54–115.
- Carpenter, G. A. and S. Grossberg, eds. (1991). *Pattern Recognition by Self-Organizing Neural Networks*. MIT Press, Cambridge, MA.
- Chauvin, Y. and D. E. Rumelhart, eds. (1995). *Backpropagation: Theory, Architecture, and Applications*. Lawrence Erlbaum Associates, Hillsdale, NJ.
- Elman, J. L. (1990). “Finding structure in time.” *Cognitive Science*, 14(2): 179–211.
- Fahlman, S. E. and C. Lebiere. (1990). “The cascade-correlation learning architecture.” Technical Report CMU-CS-90-100, School of Computer Sciences, Carnegie Mellon University, Pittsburgh, PA.
- Gerstner, W. and W. Kistler. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, Cambridge, UK.
- Girosi, F., M. Jones, and T. Poggio. (1995). “Regularization theory and neural

- networks architectures.” *Neural Computation*, 7(2):219–269.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Boston, MA.
- Gori, M. and A. Tesi. (1992). “On the problem of local minima in backpropagation.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86.
- Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*, 2nd edition. Prentice-Hall, Upper Saddle River, NJ.
- Hinton, G. (2010). “A practical guide to training restricted Boltzmann machines.” Technical Report UML TR 2010-003, Department of Computer Science, University of Toronto.
- Hinton, G., S. Osindero, and Y.-W. Teh. (2006). “A fast learning algorithm for deep belief nets.” *Neural Computation*, 18(7):1527–1554.
- Hornik, K., M. Stinchcombe, and H. White. (1989). “Multilayer feedforward networks are universal approximators.” *Neural Networks*, 2(5):359–366.
- Kohonen, T. (1982). “Self-organized formation of topologically correct feature maps.” *Biological Cybernetics*, 43(1):59–69.
- Kohonen, T. (1988). “An introduction to neural computing.” *Neural Networks*, 1(1):3–16.
- Kohonen, T. (2001). *Self-Organizing Maps*, 3rd edition. Springer, Berlin.
- LeCun, Y. and Y. Bengio. (1995). “Convolutional networks for images, speech, and time-series.” In *The Handbook of Brain Theory and Neural Networks* (M. A. Arbib, ed.), MIT Press, Cambridge, MA.
- LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. (1998). “Gradient-based learning applied to document recognition.” *Proceedings of the IEEE*, 86(11): 2278–2324.
- MacKay, D. J. C. (1992). “A practical Bayesian framework for backpropagation networks.” *Neural Computation*, 4(3):448–472.
- McCulloch, W. S. and W. Pitts. (1943). “A logical calculus of the ideas immanent in nervous activity.” *Bulletin of Mathematical Biophysics*, 5(4):115–133.
- Minsky, M. and S. Papert. (1969). *Perceptrons*. MIT Press, Cambridge, MA.

- Orr, G. B. and K.-R. Müller, eds. (1998). *Neural Networks: Tricks of the Trade*. Springer, London, UK.
- Park, J. and I. W. Sandberg. (1991). “Universal approximation using radial-basis-function networks.” *Neural Computation*, 3(2):246–257.
- Pineda, F. J. (1987). “Generalization of Back-Propagation to recurrent neural networks.” *Physical Review Letters*, 59(19):2229–2232.
- Reed, R. D. and R. J. Marks. (1998). *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press, Cambridge, MA.
- Rumelhart, D. E., G. E. Hinton, and R. J. Williams. (1986a). “Learning internal representations by error propagation.” In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (D. E. Rumelhart and J. L. McClelland, eds.), volume 1, 318–362, MIT Press, Cambridge, MA.
- Rumelhart, D. E., G. E. Hinton, and R. J. Williams. (1986b). “Learning representations by backpropagating errors.” *Nature*, 323(9):318–362.
- Schwenker, F., H.A. Kestler, and G. Palm. (2001). “Three learning phases for radial-basis-function networks.” *Neural Networks*, 14(4-5):439–458.
- Tickle, A. B., R. Andrews, M. Golea, and J. Diederich. (1998). “The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks.” *IEEE Transactions on Neural Networks*, 9(6):1057–1067.
- Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavior science*. Ph.D. thesis, Harvard University, Cambridge, MA.
- Yao, X. (1999). “Evolving artificial neural networks.” *Proceedings of the IEEE*, 87(9):1423–1447.
- Zhou, Z.-H. (2004). “Rule extraction: Using neural networks or for neural networks?” *Journal of Computer Science and Technology*, 19(2):249–253.

休息一会儿

小故事：神经网络的几起几落

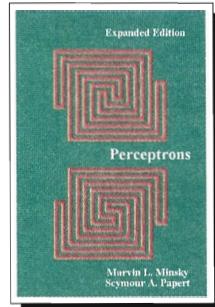
闵斯基于 1969 年获图灵奖。

此书中有不少关于神经网络的真知灼见，但其重要论断所导致的后果，对神经网络乃至人工智能整体的研究产生了极为残酷的影响，因此在神经网络重又兴起后，该书受到很多批判。1988 年再版时，闵斯基专门增加了一章以作辩护。

二十世纪四十年代 M-P 神经元模型、Hebb 学习律出现后，五十年代出现了以感知机、Adaline 为代表的一系列成果，这是神经网络发展的第一个高潮期。不幸的是，MIT 计算机科学研究的奠基人马文·闵斯基 (Marvin Minsky, 1927—) 与 Seymour Papert 在 1969 年出版了《感知机》一书，书中指出，单层神经网络无法解决非线性问题，而多层网络的训练算法尚看不到希望。这个论断直接使神经网络研究进入了“冰河期”，美国和苏联均停止了对神经网络研究的资助，全球该领域研究人员纷纷转行，仅剩极少数人坚持下来。哈佛大学的 Paul Werbos 在 1974 年发明 BP 算法时，正值神经网络冰河期，因此未受到应有的重视。

1983 年，加州理工学院的物理学家 John Hopfield 利用神经网络，在旅行商问题这个 NP 完全问题的求解上获得当时最好结果，引起了轰动。稍后，UCSD 的 David Rumelhart 与 James McClelland 领导的 PDP 小组出版了《并行分布处理：认知微结构的探索》一书，Rumelhart 等人重新发明了 BP 算法，由于当时正处于 Hopfield 带来的兴奋之中，BP 算法迅速走红。这掀起了神经网络的第二次高潮。二十世纪九十年代中期，随着统计学习理论和支持向量机的兴起，神经网络学习的理论性质不够清楚、试错性强、在使用中充斥大量“窍门”(trick)的弱点更为明显，于是神经网络研究又进入低谷，NIPS 会议甚至多年不接受以神经网络为主题的论文。

2010 年前后，随着计算能力的迅猛提升和大数据的涌现，神经网络研究在“深度学习”的名义下又重新崛起，先是在 ImageNet 等若干竞赛上以大优势夺冠，此后谷歌、百度、脸书等公司纷纷投入巨资进行研发，神经网络迎来了第三次高潮。



第6章 支持向量机

6.1 间隔与支持向量

给定训练样本集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$, $y_i \in \{-1, +1\}$, 分类学习最基本的想法就是基于训练集 D 在样本空间中找到一个划分超平面, 将不同类别的样本分开. 但能将训练样本分开的划分超平面可能有很多, 如图 6.1 所示, 我们应该努力去找到哪一个呢?

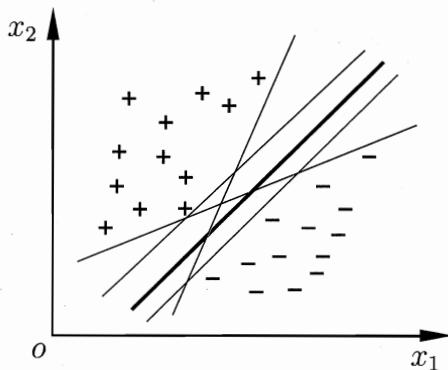


图 6.1 存在多个划分超平面将两类训练样本分开

直观上看, 应该去找位于两类训练样本“正中间”的划分超平面, 即图 6.1 中红色的那个, 因为该划分超平面对训练样本局部扰动的“容忍”性最好. 例如, 由于训练集的局限性或噪声的因素, 训练集外的样本可能比图 6.1 中的训练样本更接近两个类的分隔界, 这将使许多划分超平面出现错误, 而红色的超平面受影响最小. 换言之, 这个划分超平面所产生的分类结果是最鲁棒的, 对未见示例的泛化能力最强.

在样本空间中, 划分超平面可通过如下线性方程来描述:

$$\mathbf{w}^T \mathbf{x} + b = 0, \quad (6.1)$$

其中 $\mathbf{w} = (w_1; w_2; \dots; w_d)$ 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之间的距离. 显然, 划分超平面可被法向量 \mathbf{w} 和位移 b 确定,

参见习题 6.1.

下面我们将其记为 (\mathbf{w}, b) . 样本空间中任意点 \mathbf{x} 到超平面 (\mathbf{w}, b) 的距离可写为

$$r = \frac{|\mathbf{w}^T \mathbf{x} + b|}{\|\mathbf{w}\|}. \quad (6.2)$$

若超平面 (\mathbf{w}', b') 能将训练样本正确分类, 即对于 $(\mathbf{x}_i, y_i) \in D$, 若 $y_i = +1$, 则有 $\mathbf{w}'^T \mathbf{x}_i + b' > 0$; 若 $y_i = -1$, 则有 $\mathbf{w}'^T \mathbf{x}_i + b' < 0$. 令
训练样本正确分类, 则总
存在缩放变换 $\zeta \mathbf{w} \mapsto \mathbf{w}'$
和 $\zeta b \mapsto b'$ 使式(6.3)成立.

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq +1, & y_i = +1; \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1, & y_i = -1. \end{cases} \quad (6.3)$$

如图 6.2 所示, 距离超平面最近的这几个训练样本点使式(6.3)的等号成立, 每个样本点对应一个特征向量. 它们被称为“支持向量” (support vector), 两个异类支持向量到超平面的距离之和为

$$\gamma = \frac{2}{\|\mathbf{w}\|}, \quad (6.4)$$

它被称为“间隔” (margin).

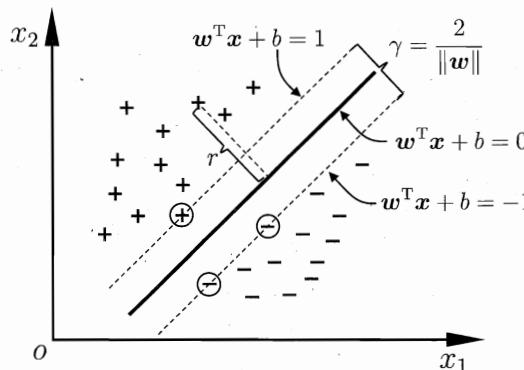


图 6.2 支持向量与间隔

欲找到具有“最大间隔” (maximum margin)的划分超平面, 也就是要找到能满足式(6.3)中约束的参数 \mathbf{w} 和 b , 使得 γ 最大, 即

$$\begin{aligned} \max_{\mathbf{w}, b} \quad & \frac{2}{\|\mathbf{w}\|} \\ \text{s.t. } & y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, m. \end{aligned} \quad (6.5)$$

间隔貌似仅与 \mathbf{w} 有关, 但事实上 b 通过约束隐式地影响着 \mathbf{w} 的取值, 进而对间隔产生影响.

显然, 为了最大化间隔, 仅需最大化 $\|\mathbf{w}\|^{-1}$, 这等价于最小化 $\|\mathbf{w}\|^2$. 于是, 式(6.5)可重写为

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad (6.6)$$

$$\text{s.t. } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, m.$$

这就是支持向量机(Support Vector Machine, 简称 SVM)的基本型.

6.2 对偶问题

我们希望求解式(6.6)来得到大间隔划分超平面所对应的模型

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b, \quad (6.7)$$

其中 \mathbf{w} 和 b 是模型参数. 注意到式(6.6)本身是一个凸二次规划(convex quadratic programming)问题, 能直接用现成的优化计算包求解, 但我们可以有更高效的办法.

参见附录 B.1.

对式(6.6)使用拉格朗日乘子法可得到其“对偶问题”(dual problem). 具体来说, 对式(6.6)的每条约束添加拉格朗日乘子 $\alpha_i \geq 0$, 则该问题的拉格朗日函数可写为

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^m \alpha_i (1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)), \quad (6.8)$$

其中 $\boldsymbol{\alpha} = (\alpha_1; \alpha_2; \dots; \alpha_m)$. 令 $L(\mathbf{w}, b, \boldsymbol{\alpha})$ 对 \mathbf{w} 和 b 的偏导为零可得

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i, \quad (6.9)$$

$$0 = \sum_{i=1}^m \alpha_i y_i. \quad (6.10)$$

将式(6.9)代入(6.8), 即可将 $L(\mathbf{w}, b, \boldsymbol{\alpha})$ 中的 \mathbf{w} 和 b 消去, 再考虑式(6.10)的约束, 就得到式(6.6)的对偶问题

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \quad (6.11)$$

$$\text{s.t. } \sum_{i=1}^m \alpha_i y_i = 0, \\ \alpha_i \geq 0, \quad i = 1, 2, \dots, m.$$

解出 α 后, 求出 w 与 b 即可得到模型

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + b \\ &= \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b. \end{aligned} \quad (6.12)$$

从对偶问题(6.11)解出的 α_i 是式(6.8)中的拉格朗日乘子, 它恰对应着训练样本 (\mathbf{x}_i, y_i) . 注意到式(6.6)中有不等式约束, 因此上述过程需满足 KKT (Karush-Kuhn-Tucker) 条件, 即要求

$$\left\{ \begin{array}{l} \alpha_i \geq 0; \\ y_i f(\mathbf{x}_i) - 1 \geq 0; \\ \alpha_i (y_i f(\mathbf{x}_i) - 1) = 0. \end{array} \right. \quad (6.13)$$

于是, 对任意训练样本 (\mathbf{x}_i, y_i) , 总有 $\alpha_i = 0$ 或 $y_i f(\mathbf{x}_i) = 1$. 若 $\alpha_i = 0$, 则该样本将不会在式(6.12) 的求和中出现, 也就不会对 $f(\mathbf{x})$ 有任何影响; 若 $\alpha_i > 0$, 则必有 $y_i f(\mathbf{x}_i) = 1$, 所对应的样本点位于最大间隔边界上, 是一个支持向量. 这显示出支持向量机的一个重要性质: 训练完成后, 大部分的训练样本都不需保留, 最终模型仅与支持向量有关.

如 [Vapnik, 1999] 所述, 支持向量机这个名字强调了此类学习器的关键是如何从支持向量构建出解; 同时也暗示着其复杂度主要与支持向量的数目有关.

二次规划参见附录 B.2.

那么, 如何求解式(6.11) 呢? 不难发现, 这是一个二次规划问题, 可使用通用的二次规划算法来求解; 然而, 该问题的规模正比于训练样本数, 这会在实际任务中造成很大的开销. 为了避开这个障碍, 人们通过利用问题本身的特性, 提出了很多高效算法, SMO (Sequential Minimal Optimization) 是其中一个著名的代表 [Platt, 1998].

SMO 的基本思路是先固定 α_i 之外的所有参数, 然后求 α_i 上的极值. 由于存在约束 $\sum_{i=1}^m \alpha_i y_i = 0$, 若固定 α_i 之外的其他变量, 则 α_i 可由其他变量导出. 于是, SMO 每次选择两个变量 α_i 和 α_j , 并固定其他参数. 这样, 在参数初始化后, SMO 不断执行如下两个步骤直至收敛:

- 选取一对需更新的变量 α_i 和 α_j ;

- 固定 α_i 和 α_j 以外的参数, 求解式(6.11)获得更新后的 α_i 和 α_j .

注意到只需选取的 α_i 和 α_j 中有一个不满足 KKT 条件(6.13), 目标函数就会在迭代后减小 [Osuna et al., 1997]. 直观来看, KKT 条件违背的程度越大, 则变量更新后可能导致的目标函数值减幅越大. 于是, SMO 先选取违背 KKT 条件程度最大的变量. 第二个变量应选择一个使目标函数值减小最快的变量, 但由于比较各变量所对应的目标函数值减幅的复杂度过高, 因此 SMO 采用了一个启发式: 使选取的两变量所对应样本之间的间隔最大. 一种直观的解释是, 这样的两个变量有很大的差别, 与对两个相似的变量进行更新相比, 对它们进行更新会带给目标函数值更大的变化.

SMO 算法之所以高效, 恰由于在固定其他参数后, 仅优化两个参数的过程能做到非常高效. 具体来说, 仅考虑 α_i 和 α_j 时, 式(6.11)中的约束可重写为

$$\alpha_i y_i + \alpha_j y_j = c, \quad \alpha_i \geq 0, \quad \alpha_j \geq 0, \quad (6.14)$$

其中

$$c = - \sum_{k \neq i, j} \alpha_k y_k \quad (6.15)$$

是使 $\sum_{i=1}^m \alpha_i y_i = 0$ 成立的常数. 用

$$\alpha_i y_i + \alpha_j y_j = c \quad (6.16)$$

消去式(6.11)中的变量 α_j , 则得到一个关于 α_i 的单变量二次规划问题, 仅有的约束是 $\alpha_i \geq 0$. 不难发现, 这样的二次规划问题具有闭式解, 于是不必调用数值优化算法即可高效地计算出更新后的 α_i 和 α_j .

如何确定偏移项 b 呢? 注意到对任意支持向量 (\mathbf{x}_s, y_s) 都有 $y_s f(\mathbf{x}_s) = 1$, 即

$$y_s \left(\sum_{i \in S} \alpha_i y_i \mathbf{x}_i^\top \mathbf{x}_s + b \right) = 1, \quad (6.17)$$

其中 $S = \{i \mid \alpha_i > 0, i = 1, 2, \dots, m\}$ 为所有支持向量的下标集. 理论上, 可选取任意支持向量并通过求解式(6.17)获得 b , 但现实任务中常采用一种更鲁棒的做法: 使用所有支持向量求解的平均值

$$b = \frac{1}{|S|} \sum_{s \in S} \left(y_s - \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^\top \mathbf{x}_s \right). \quad (6.18)$$

6.3 核函数

在本章前面的讨论中, 我们假设训练样本是线性可分的, 即存在一个划分超平面能将训练样本正确分类. 然而在现实任务中, 原始样本空间内也许并不存在一个能正确划分两类样本的超平面. 例如图 6.3 中的“异或”问题就不是线性可分的.

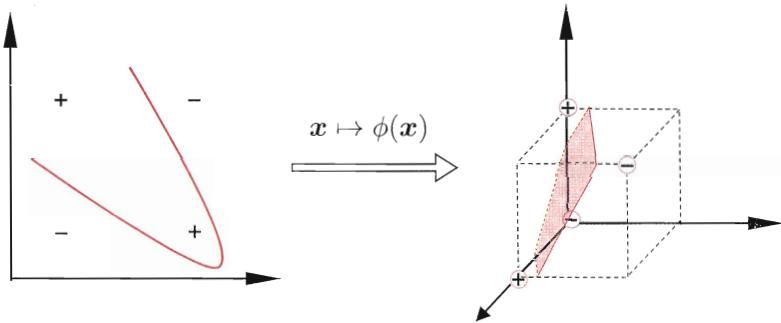


图 6.3 异或问题与非线性映射

对这样的问题, 可将样本从原始空间映射到一个更高维的特征空间, 使得样本在这个特征空间内线性可分. 例如在图 6.3 中, 若将原始的二维空间映射到一个合适的三维空间, 就能找到一个合适的划分超平面. 幸运的是, 如果原始空间是有限维, 即属性数有限, 那么一定存在一个高维特征空间使样本可分.

参见第 12 章.

令 $\phi(\mathbf{x})$ 表示将 \mathbf{x} 映射后的特征向量, 于是, 在特征空间中划分超平面所对应的模型可表示为

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b, \quad (6.19)$$

其中 \mathbf{w} 和 b 是模型参数. 类似式(6.6), 有

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{s.t. } & y_i (\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1, \quad i = 1, 2, \dots, m. \end{aligned} \quad (6.20)$$

其对偶问题是

$$\max_{\alpha} \quad \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \quad (6.21)$$

$$\text{s.t. } \sum_{i=1}^m \alpha_i y_i = 0, \\ \alpha_i \geq 0, \quad i = 1, 2, \dots, m.$$

求解式(6.21)涉及到计算 $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$, 这是样本 \mathbf{x}_i 与 \mathbf{x}_j 映射到特征空间之后的内积. 由于特征空间维数可能很高, 甚至可能是无穷维, 因此直接计算 $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ 通常是困难的. 为了避开这个障碍, 可以设想这样一个函数:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j), \quad (6.22)$$

即 \mathbf{x}_i 与 \mathbf{x}_j 在特征空间的内积等于它们在原始样本空间中通过函数 $\kappa(\cdot, \cdot)$ 计算的结果. 有了这样的函数, 我们就不必直接去计算高维甚至无穷维特征空间中的内积, 于是式(6.21)可重写为

$$\begin{aligned} \max_{\alpha} & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} & \sum_{i=1}^m \alpha_i y_i = 0, \\ & \alpha_i \geq 0, \quad i = 1, 2, \dots, m. \end{aligned} \quad (6.23)$$

求解后即可得到

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}^T \phi(\mathbf{x}) + b \\ &= \sum_{i=1}^m \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b \\ &= \sum_{i=1}^m \alpha_i y_i \kappa(\mathbf{x}, \mathbf{x}_i) + b. \end{aligned} \quad (6.24)$$

这里的函数 $\kappa(\cdot, \cdot)$ 就是“核函数”(kernel function). 式(6.24)显示出模型最优解可通过训练样本的核函数展开, 这一展式亦称“支持向量展式”(support vector expansion).

显然, 若已知合适映射 $\phi(\cdot)$ 的具体形式, 则可写出核函数 $\kappa(\cdot, \cdot)$. 但在现实任务中我们通常不知道 $\phi(\cdot)$ 是什么形式, 那么, 合适的核函数是否一定存在呢? 什么样的函数能做核函数呢? 我们有下面的定理:

证明可参阅 [Schölkopf and Smola, 2002].

定理 6.1 (核函数) 令 \mathcal{X} 为输入空间, $\kappa(\cdot, \cdot)$ 是定义在 $\mathcal{X} \times \mathcal{X}$ 上的对称函数, 则 κ 是核函数当且仅当对于任意数据 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, “核矩阵” (kernel matrix) \mathbf{K} 总是半正定的:

$$\mathbf{K} = \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_j) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_m) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_i, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_i, \mathbf{x}_j) & \cdots & \kappa(\mathbf{x}_i, \mathbf{x}_m) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_m, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_m, \mathbf{x}_j) & \cdots & \kappa(\mathbf{x}_m, \mathbf{x}_m) \end{bmatrix}.$$

定理 6.1 表明, 只要一个对称函数所对应的核矩阵半正定, 它就能作为核函数使用. 事实上, 对于一个半正定核矩阵, 总能找到一个与之对应的映射 ϕ . 换言之, 任何一个核函数都隐式地定义了一个称为“再生核希尔伯特空间” (Reproducing Kernel Hilbert Space, 简称 RKHS) 的特征空间.

通过前面的讨论可知, 我们希望样本在特征空间内线性可分, 因此特征空间的好坏对支持向量机的性能至关重要. 需注意的是, 在不知道特征映射的形式时, 我们并不知道什么样的核函数是合适的, 而核函数也仅是隐式地定义了这个特征空间. 于是, “核函数选择” 成为支持向量机的最大变数. 若核函数选择不合适, 则意味着将样本映射到了一个不合适的特征空间, 很可能导致性能不佳.

这方面有一些基本的经验, 例如对文本数据通常采用线性核, 情况不明时可先尝试高斯核.

表 6.1 列出了几种常用的核函数.

表 6.1 常用核函数

名称	表达式	参数
线性核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$	
多项式核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^d$	$d \geq 1$ 为多项式的次数
高斯核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\ \mathbf{x}_i - \mathbf{x}_j\ ^2}{2\sigma^2}\right)$	$\sigma > 0$ 为高斯核的带宽 (width)
拉普拉斯核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\ \mathbf{x}_i - \mathbf{x}_j\ }{\sigma}\right)$	$\sigma > 0$
Sigmoid 核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta \mathbf{x}_i^T \mathbf{x}_j + \theta)$	\tanh 为双曲正切函数, $\beta > 0, \theta < 0$

$d = 1$ 时退化为线性核.
高斯核亦称 RBF 核.

此外, 还可通过函数组合得到, 例如:

- 若 κ_1 和 κ_2 为核函数, 则对于任意正数 γ_1, γ_2 , 其线性组合

$$\gamma_1 \kappa_1 + \gamma_2 \kappa_2 \quad (6.25)$$

也是核函数;

- 若 κ_1 和 κ_2 为核函数, 则核函数的直积

$$\kappa_1 \otimes \kappa_2(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z})\kappa_2(\mathbf{x}, \mathbf{z}) \quad (6.26)$$

也是核函数;

- 若 κ_1 为核函数, 则对于任意函数 $g(\mathbf{x})$,

$$\kappa(\mathbf{x}, \mathbf{z}) = g(\mathbf{x})\kappa_1(\mathbf{x}, \mathbf{z})g(\mathbf{z}) \quad (6.27)$$

也是核函数.

6.4 软间隔与正则化

在前面的讨论中, 我们一直假定训练样本在样本空间或特征空间中是线性可分的, 即存在一个超平面能将不同类的样本完全划分开. 然而, 在现实任务中往往很难确定合适的核函数使得训练样本在特征空间中线性可分; 退一步说, 即便恰好找到了某个核函数使训练集在特征空间中线性可分, 也很难断定这个貌似线性可分的结果不是由于过拟合所造成的.

缓解该问题的一个办法是允许支持向量机在一些样本上出错. 为此, 要引入“软间隔”(soft margin)的概念, 如图 6.4 所示.

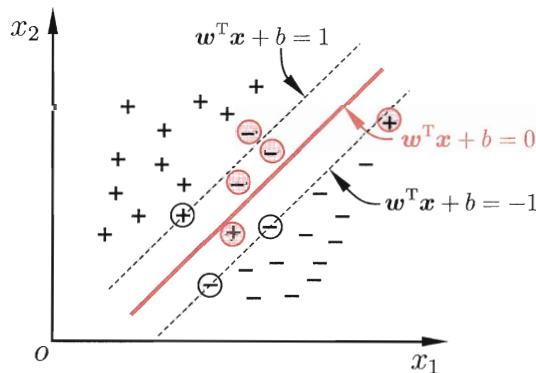


图 6.4 软间隔示意图. 红色圈出了一些不满足约束的样本.

具体来说, 前面介绍的支持向量机形式是要求所有样本均满足约束(6.3), 即所有样本都必须划分正确, 这称为“硬间隔”(hard margin), 而软间隔则是

允许某些样本不满足约束

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1. \quad (6.28)$$

当然, 在最大化间隔的同时, 不满足约束的样本应尽可能少。于是, 优化目标可写为

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \ell_{0/1}(y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1), \quad (6.29)$$

其中 $C > 0$ 是一个常数, $\ell_{0/1}$ 是“0/1损失函数”

$$\ell_{0/1}(z) = \begin{cases} 1, & \text{if } z < 0; \\ 0, & \text{otherwise.} \end{cases} \quad (6.30)$$

显然, 当 C 为无穷大时, 式(6.29)迫使所有样本均满足约束(6.28), 于是式(6.29)等价于(6.6); 当 C 取有限值时, 式(6.29)允许一些样本不满足约束。

然而, $\ell_{0/1}$ 非凸、非连续, 数学性质不太好, 使得式(6.29)不易直接求解。于是, 人们通常用其他一些函数来代替 $\ell_{0/1}$, 称为“替代损失”(surrogate loss)。替代损失函数一般具有较好的数学性质, 如它们通常是凸的连续函数且是 $\ell_{0/1}$ 的上界。图 6.5 给出了三种常用的替代损失函数:

对率损失是对率函数的变形, 对率函数参见 3.3 节。

对率损失函数通常表示为 $\ell_{log}(\cdot)$, 因此式(6.33)把式(3.15)中的 $\ln(\cdot)$ 改写为 $\log(\cdot)$ 。

$$\text{hinge 损失: } \ell_{hinge}(z) = \max(0, 1 - z); \quad (6.31)$$

$$\text{指数损失(exponential loss): } \ell_{exp}(z) = \exp(-z); \quad (6.32)$$

$$\text{对率损失(logistic loss): } \ell_{log}(z) = \log(1 + \exp(-z)). \quad (6.33)$$

若采用 hinge 损失, 则式(6.29)变成

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \max(0, 1 - y_i (\mathbf{w}^T \mathbf{x}_i + b)). \quad (6.34)$$

引入“松弛变量”(slack variables) $\xi_i \geq 0$, 可将式(6.34)重写为

$$\min_{\mathbf{w}, b, \xi_i} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i \quad (6.35)$$

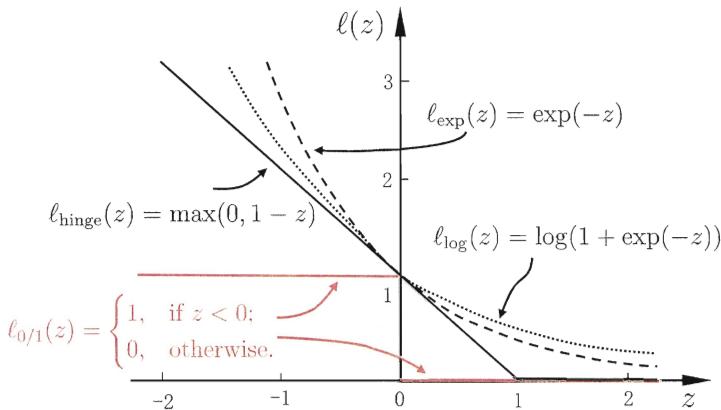


图 6.5 三种常见的替代损失函数: hinge 损失、指数损失、对率损失

$$\text{s.t. } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0, i = 1, 2, \dots, m.$$

这就是常用的“软间隔支持向量机”.

显然, 式(6.35)中每个样本都有一个对应的松弛变量, 用以表征该样本不满足约束(6.28)的程度. 但是, 与式(6.6)相似, 这仍是一个二次规划问题. 于是, 类似式(6.8), 通过拉格朗日乘子法可得到式(6.35)的拉格朗日函数

$$\begin{aligned} L(\mathbf{w}, b, \alpha, \xi, \mu) &= \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i \\ &\quad + \sum_{i=1}^m \alpha_i (1 - \xi_i - y_i (\mathbf{w}^T \mathbf{x}_i + b)) - \sum_{i=1}^m \mu_i \xi_i, \end{aligned} \quad (6.36)$$

其中 $\alpha_i \geq 0, \mu_i \geq 0$ 是拉格朗日乘子.

令 $L(\mathbf{w}, b, \alpha, \xi, \mu)$ 对 \mathbf{w}, b, ξ_i 的偏导为零可得

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i, \quad (6.37)$$

$$0 = \sum_{i=1}^m \alpha_i y_i, \quad (6.38)$$

$$C = \alpha_i + \mu_i. \quad (6.39)$$

将式(6.37)–(6.39)代入式(6.36)即可得到式(6.35)的对偶问题

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, m. \end{aligned} \quad (6.40)$$

将式(6.40)与硬间隔下的对偶问题(6.11)对比可看出, 两者唯一的差别就在于对偶变量的约束不同: 前者是 $0 \leq \alpha_i \leq C$, 后者是 $0 \leq \alpha_i$. 于是, 可采用 6.2 节中同样的算法求解式(6.40); 在引入核函数后能得到与式(6.24)同样的支持向量展式.

类似式(6.13), 对软间隔支持向量机, KKT 条件要求

$$\left\{ \begin{array}{l} \alpha_i \geq 0, \quad \mu_i \geq 0, \\ y_i f(\mathbf{x}_i) - 1 + \xi_i \geq 0, \\ \alpha_i (y_i f(\mathbf{x}_i) - 1 + \xi_i) = 0, \\ \xi_i \geq 0, \quad \mu_i \xi_i = 0. \end{array} \right. \quad (6.41)$$

于是, 对任意训练样本 (\mathbf{x}_i, y_i) , 总有 $\alpha_i = 0$ 或 $y_i f(\mathbf{x}_i) = 1 - \xi_i$. 若 $\alpha_i = 0$, 则该样本不会对 $f(\mathbf{x})$ 有任何影响; 若 $\alpha_i > 0$, 则必有 $y_i f(\mathbf{x}_i) = 1 - \xi_i$, 即该样本是支持向量: 由式(6.39)可知, 若 $\alpha_i < C$, 则 $\mu_i > 0$, 进而有 $\xi_i = 0$, 即该样本恰在最大间隔边界上; 若 $\alpha_i = C$, 则有 $\mu_i = 0$, 此时若 $\xi_i \leq 1$ 则该样本落在最大间隔内部, 若 $\xi_i > 1$ 则该样本被错误分类. 由此可看出, 软间隔支持向量机的最终模型仅与支持向量有关, 即通过采用 hinge 损失函数仍保持了稀疏性.

那么, 能否对式(6.29)使用其他的替代损失函数呢?

可以发现, 如果使用对率损失函数 ℓ_{log} 来替代式(6.29)中的 0/1 损失函数, 则几乎就得到了对率回归模型(3.27). 实际上, 支持向量机与对率回归的优化目标相近, 通常情形下它们的性能也相当. 对率回归的优势主要在于其输出具有自然的概率意义, 即在给出预测标记的同时也给出了概率, 而支持向量机的输出不具有概率意义, 欲得到概率输出需进行特殊处理 [Platt, 2000]; 此外, 对率回归能直接用于多分类任务, 支持向量机为此则需进行推广 [Hsu and Lin, 2002]. 另一方面, 从图 6.5 可看出, hinge 损失有一块“平坦”的零区域, 这使

得支持向量机的解具有稀疏性, 而对率损失是光滑的单调递减函数, 不能导出类似支持向量的概念, 因此对率回归的解依赖于更多的训练样本, 其预测开销更大.

我们还可以把式(6.29)中的 0/1 损失函数换成别的替代损失函数以得到其他学习模型, 这些模型的性质与所用的替代函数直接相关, 但它们具有一个共性: 优化目标中的第一项用来描述划分超平面的“间隔”大小, 另一项 $\sum_{i=1}^m \ell(f(\mathbf{x}_i), y_i)$ 用来表述训练集上的误差, 可写为更一般的形式

$$\min_f \Omega(f) + C \sum_{i=1}^m \ell(f(\mathbf{x}_i), y_i), \quad (6.42)$$

其中 $\Omega(f)$ 称为“结构风险”(structural risk), 用于描述模型 f 的某些性质; 第二项 $\sum_{i=1}^m \ell(f(\mathbf{x}_i), y_i)$ 称为“经验风险”(empirical risk), 用于描述模型与训练数据的契合程度; C 用于对二者进行折中. 从经验风险最小化的角度来看, $\Omega(f)$

正则化可理解为一种“罚函数法”, 即对不希望得到的结果施以惩罚, 从而使得优化过程趋向于希望目标. 从贝叶斯估计的角度来看, 正则化项可认为是提供了模型的先验概率.

参见 11.4 节.

L_p 范数 (norm) 是常用的正则化项, 其中 L_2 范数 $\|\mathbf{w}\|_2$ 倾向于 \mathbf{w} 的分量取值尽量均衡, 即非零分量个数尽量稠密, 而 L_0 范数 $\|\mathbf{w}\|_0$ 和 L_1 范数 $\|\mathbf{w}\|_1$ 则倾向于 \mathbf{w} 的分量尽量稀疏, 即非零分量个数尽量少.

6.5 支持向量回归

现在我们来考虑回归问题. 给定训练样本 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$, $y_i \in \mathbb{R}$, 希望学得一个形如式(6.7)的回归模型, 使得 $f(\mathbf{x})$ 与 y 尽可能接近, \mathbf{w} 和 b 是待确定的模型参数.

对样本 (\mathbf{x}, y) , 传统回归模型通常直接基于模型输出 $f(\mathbf{x})$ 与真实输出 y 之间的差别来计算损失, 当且仅当 $f(\mathbf{x})$ 与 y 完全相同时, 损失才为零. 与此不同, 支持向量回归(Support Vector Regression, 简称 SVR)假设我们能容忍 $f(\mathbf{x})$ 与 y 之间最多有 ϵ 的偏差, 即仅当 $f(\mathbf{x})$ 与 y 之间的差别绝对值大于 ϵ 时才计算损失. 如图 6.6 所示, 这相当于以 $f(\mathbf{x})$ 为中心, 构建了一个宽度为 2ϵ 的间隔带, 若训练样本落入此间隔带, 则认为是被预测正确的.

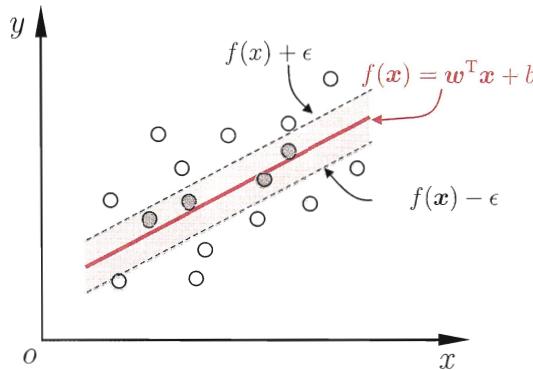


图 6.6 支持向量回归示意图. 红色显示出 ϵ -间隔带, 落入其中的样本不计算损失.

于是, SVR 问题可形式化为

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \ell_\epsilon(f(\mathbf{x}_i) - y_i), \quad (6.43)$$

其中 C 为正则化常数, ℓ_ϵ 是图 6.7 所示的 ϵ -不敏感损失 (ϵ -insensitive loss) 函数

$$\ell_\epsilon(z) = \begin{cases} 0, & \text{if } |z| \leq \epsilon; \\ |z| - \epsilon, & \text{otherwise.} \end{cases} \quad (6.44)$$

间隔带两侧的松弛程度
可有所不同.

引入松弛变量 ξ_i 和 $\hat{\xi}_i$, 可将式(6.43)重写为

$$\min_{\mathbf{w}, b, \xi_i, \hat{\xi}_i} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m (\xi_i + \hat{\xi}_i) \quad (6.45)$$

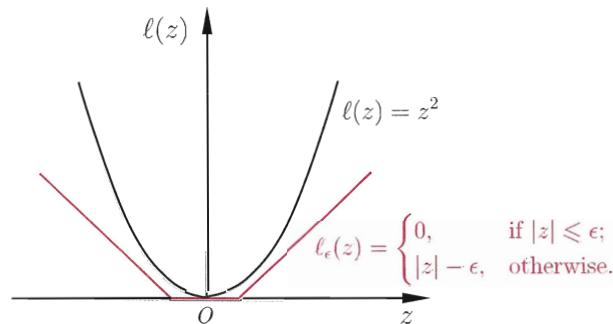


图 6.7 ϵ -不敏感损失函数

$$\begin{aligned} \text{s.t. } & f(\mathbf{x}_i) - y_i \leq \epsilon + \xi_i, \\ & y_i - f(\mathbf{x}_i) \leq \epsilon + \hat{\xi}_i, \\ & \xi_i \geq 0, \hat{\xi}_i \geq 0, \quad i = 1, 2, \dots, m. \end{aligned}$$

类似式(6.36), 通过引入拉格朗日乘子 $\mu_i \geq 0, \hat{\mu}_i \geq 0, \alpha_i \geq 0, \hat{\alpha}_i \geq 0$, 由拉格朗日乘子法可得到式(6.45)的拉格朗日函数

$$\begin{aligned} L(\mathbf{w}, b, \boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}, \boldsymbol{\xi}, \hat{\boldsymbol{\xi}}, \boldsymbol{\mu}, \hat{\boldsymbol{\mu}}) &= \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m (\xi_i + \hat{\xi}_i) - \sum_{i=1}^m \mu_i \xi_i - \sum_{i=1}^m \hat{\mu}_i \hat{\xi}_i \\ &+ \sum_{i=1}^m \alpha_i (f(\mathbf{x}_i) - y_i - \epsilon - \xi_i) + \sum_{i=1}^m \hat{\alpha}_i (y_i - f(\mathbf{x}_i) - \epsilon - \hat{\xi}_i). \end{aligned} \quad (6.46)$$

将式(6.7)代入, 再令 $L(\mathbf{w}, b, \boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}, \boldsymbol{\xi}, \hat{\boldsymbol{\xi}}, \boldsymbol{\mu}, \hat{\boldsymbol{\mu}})$ 对 \mathbf{w}, b, ξ_i 和 $\hat{\xi}_i$ 的偏导为零可得

$$\mathbf{w} = \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) \mathbf{x}_i, \quad (6.47)$$

$$0 = \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i), \quad (6.48)$$

$$C = \alpha_i + \mu_i, \quad (6.49)$$

$$C = \hat{\alpha}_i + \hat{\mu}_i. \quad (6.50)$$

将式(6.47)–(6.50)代入式(6.46), 即可得到 SVR 的对偶问题

$$\begin{aligned} \max_{\boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}} \quad & \sum_{i=1}^m y_i (\hat{\alpha}_i - \alpha_i) - \epsilon (\hat{\alpha}_i + \alpha_i) \\ & - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m (\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j) \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \quad & \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) = 0, \\ & 0 \leq \alpha_i, \hat{\alpha}_i \leq C. \end{aligned} \quad (6.51)$$

上述过程中需满足 KKT 条件, 即要求

$$\begin{cases} \alpha_i(f(\mathbf{x}_i) - y_i - \epsilon - \xi_i) = 0, \\ \hat{\alpha}_i(y_i - f(\mathbf{x}_i) - \epsilon - \hat{\xi}_i) = 0, \\ \alpha_i \hat{\alpha}_i = 0, \quad \xi_i \hat{\xi}_i = 0, \\ (C - \alpha_i)\xi_i = 0, \quad (C - \hat{\alpha}_i)\hat{\xi}_i = 0. \end{cases} \quad (6.52)$$

可以看出, 当且仅当 $f(\mathbf{x}_i) - y_i - \epsilon - \xi_i = 0$ 时 α_i 能取非零值, 当且仅当 $y_i - f(\mathbf{x}_i) - \epsilon - \hat{\xi}_i = 0$ 时 $\hat{\alpha}_i$ 能取非零值. 换言之, 仅当样本 (\mathbf{x}_i, y_i) 不落入 ϵ -间隔带中, 相应的 α_i 和 $\hat{\alpha}_i$ 才能取非零值. 此外, 约束 $f(\mathbf{x}_i) - y_i - \epsilon - \xi_i = 0$ 和 $y_i - f(\mathbf{x}_i) - \epsilon - \hat{\xi}_i = 0$ 不能同时成立, 因此 α_i 和 $\hat{\alpha}_i$ 中至少有一个为零.

将式(6.47)代入(6.7), 则 SVR 的解形如

$$f(\mathbf{x}) = \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) \mathbf{x}_i^T \mathbf{x} + b. \quad (6.53)$$

落在 ϵ -间隔带中的样本都满足 $\alpha_i = 0$ 且 $\hat{\alpha}_i = 0$.

能使式(6.53)中的 $(\hat{\alpha}_i - \alpha_i) \neq 0$ 的样本即为 SVR 的支持向量, 它们必落在 ϵ -间隔带之外. 显然, SVR 的支持向量仅是训练样本的一部分, 即其解仍具有稀疏性.

由 KKT 条件(6.52)可看出, 对每个样本 (\mathbf{x}_i, y_i) 都有 $(C - \alpha_i)\xi_i = 0$ 且 $\alpha_i(f(\mathbf{x}_i) - y_i - \epsilon - \xi_i) = 0$. 于是, 在得到 α_i 后, 若 $0 < \alpha_i < C$, 则必有 $\xi_i = 0$, 进而有

$$b = y_i + \epsilon - \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) \mathbf{x}_i^T \mathbf{x}. \quad (6.54)$$

因此, 在求解式(6.51)得到 α_i 后, 理论上来说, 可任意选取满足 $0 < \alpha_i < C$ 的样本通过式(6.54)求得 b . 实践中常采用一种更鲁棒的办法: 选取多个(或所有)满足条件 $0 < \alpha_i < C$ 的样本求解 b 后取平均值.

若考虑特征映射形式(6.19), 则相应的, 式(6.47)将形如

$$\mathbf{w} = \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) \phi(\mathbf{x}_i). \quad (6.55)$$

将式(6.55)代入(6.19), 则 SVR 可表示为

$$f(\mathbf{x}) = \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) \kappa(\mathbf{x}, \mathbf{x}_i) + b, \quad (6.56)$$

其中 $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ 为核函数.

6.6 核方法

回顾式(6.24)和(6.56)可发现, 给定训练样本 $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$, 若不考虑偏移项 b , 则无论 SVM 还是 SVR, 学得的模型总能表示成核函数 $\kappa(\mathbf{x}, \mathbf{x}_i)$ 的线性组合. 不仅如此, 事实上我们有下面这个称为“表示定理”(representer theorem)的更一般的结论:

证明参阅 [Schölkopf and Smola, 2002], 其中用到了关于实对称矩阵正定性充要条件的 Mercer 定理.

定理 6.2 (表示定理) 令 \mathbb{H} 为核函数 κ 对应的再生核希尔伯特空间, $\|h\|_{\mathbb{H}}$ 表示 \mathbb{H} 空间中关于 h 的范数, 对于任意单调递增函数 $\Omega : [0, \infty] \mapsto \mathbb{R}$ 和任意非负损失函数 $\ell : \mathbb{R}^m \mapsto [0, \infty]$, 优化问题

$$\min_{h \in \mathbb{H}} F(h) = \Omega(\|h\|_{\mathbb{H}}) + \ell(h(\mathbf{x}_1), h(\mathbf{x}_2), \dots, h(\mathbf{x}_m)) \quad (6.57)$$

的解总可写为

$$h^*(\mathbf{x}) = \sum_{i=1}^m \alpha_i \kappa(\mathbf{x}, \mathbf{x}_i). \quad (6.58)$$

表示定理对损失函数没有限制, 对正则化项 Ω 仅要求单调递增, 甚至不要求 Ω 是凸函数, 意味着对于一般的损失函数和正则化项, 优化问题(6.57)的最优解 $h^*(\mathbf{x})$ 都可表示为核函数 $\kappa(\mathbf{x}, \mathbf{x}_i)$ 的线性组合; 这显示出核函数的巨大威力.

线性判别分析见 3.4 节.

人们发展出一系列基于核函数的学习方法, 统称为“核方法”(kernel methods). 最常见的, 是通过“核化”(即引入核函数)来将线性学习器拓展为非线性学习器. 下面我们以线性判别分析为例来演示如何通过核化来对其进行非线性拓展, 从而得到“核线性判别分析”(Kernelized Linear Discriminant Analysis, 简称 KLDA).

我们先假设可通过某种映射 $\phi : \mathcal{X} \mapsto \mathbb{F}$ 将样本映射到一个特征空间 \mathbb{F} , 然后在 \mathbb{F} 中执行线性判别分析, 以求得

$$h(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}). \quad (6.59)$$

类似于式(3.35), KLDA 的学习目标是

$$\max_{\mathbf{w}} J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_b^\phi \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w^\phi \mathbf{w}}, \quad (6.60)$$

其中 \mathbf{S}_b^ϕ 和 \mathbf{S}_w^ϕ 分别为训练样本在特征空间 \mathbb{F} 中的类间散度矩阵和类内散度矩阵. 令 X_i 表示第 $i \in \{0, 1\}$ 类样本的集合, 其样本数为 m_i ; 总样本数 $m = m_0 + m_1$. 第 i 类样本在特征空间 \mathbb{F} 中的均值为

$$\boldsymbol{\mu}_i^\phi = \frac{1}{m_i} \sum_{\mathbf{x} \in X_i} \phi(\mathbf{x}), \quad (6.61)$$

两个散度矩阵分别为

$$\mathbf{S}_b^\phi = (\boldsymbol{\mu}_1^\phi - \boldsymbol{\mu}_0^\phi)(\boldsymbol{\mu}_1^\phi - \boldsymbol{\mu}_0^\phi)^T; \quad (6.62)$$

$$\mathbf{S}_w^\phi = \sum_{i=0}^1 \sum_{\mathbf{x} \in X_i} (\phi(\mathbf{x}) - \boldsymbol{\mu}_i^\phi)(\phi(\mathbf{x}) - \boldsymbol{\mu}_i^\phi)^T. \quad (6.63)$$

通常我们难以知道映射 ϕ 的具体形式, 因此使用核函数 $\kappa(\mathbf{x}, \mathbf{x}_i) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$ 来隐式地表达这个映射和特征空间 \mathbb{F} . 把 $J(\mathbf{w})$ 作为式(6.57)中的损失函数 ℓ , 再令 $\Omega \equiv 0$, 由表示定理, 函数 $h(\mathbf{x})$ 可写为

$$h(\mathbf{x}) = \sum_{i=1}^m \alpha_i \kappa(\mathbf{x}, \mathbf{x}_i), \quad (6.64)$$

于是由式(6.59)可得

$$\mathbf{w} = \sum_{i=1}^m \alpha_i \phi(\mathbf{x}_i). \quad (6.65)$$

令 $\mathbf{K} \in \mathbb{R}^{m \times m}$ 为核函数 κ 所对应的核矩阵, $(\mathbf{K})_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. 令 $\mathbf{1}_i \in \{1, 0\}^{m \times 1}$ 为第 i 类样本的指示向量, 即 $\mathbf{1}_i$ 的第 j 个分量为 1 当且仅当 $\mathbf{x}_j \in X_i$, 否则 $\mathbf{1}_i$ 的第 j 个分量为 0. 再令

$$\hat{\boldsymbol{\mu}}_0 = \frac{1}{m_0} \mathbf{K} \mathbf{1}_0, \quad (6.66)$$

$$\hat{\boldsymbol{\mu}}_1 = \frac{1}{m_1} \mathbf{K} \mathbf{1}_1, \quad (6.67)$$

$$\mathbf{M} = (\hat{\boldsymbol{\mu}}_0 - \hat{\boldsymbol{\mu}}_1)(\hat{\boldsymbol{\mu}}_0 - \hat{\boldsymbol{\mu}}_1)^T, \quad (6.68)$$

$$\mathbf{N} = \mathbf{K}\mathbf{K}^T - \sum_{i=0}^1 m_i \hat{\boldsymbol{\mu}}_i \hat{\boldsymbol{\mu}}_i^T. \quad (6.69)$$

于是, 式(6.60)等价为

$$\max_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T \mathbf{N} \boldsymbol{\alpha}}. \quad (6.70)$$

求解方法参见 3.4 节.

显然, 使用线性判别分析求解方法即可得到 $\boldsymbol{\alpha}$, 进而可由式(6.64)得到投影函数 $h(\mathbf{x})$.

6.7 阅读材料

线性核 SVM 迄今仍是文本分类的首选技术。一个重要原因可能是: 若将每个单词作为文本数据的一个属性, 则该属性空间维数很高, 冗余度很大, 其描述能力足以将不同文档“打散”。关于打散, 参见 12.4 节。

支持向量机于 1995 年正式发表 [Cortes and Vapnik, 1995], 由于在文本分类任务中显示出卓越性能 [Joachims, 1998], 很快成为机器学习的主流技术, 并直接掀起了“统计学习”(statistical learning)在 2000 年前后的高潮。但实际上, 支持向量的概念早在二十世纪六十年代就已出现, 统计学习理论在七十年代就已成型。对核函数的研究更早, Mercer 定理 [Cristianini and Shawe-Taylor, 2000] 可追溯到 1909 年, RKHS 则在四十年代就已被研究, 但在统计学习兴起后, 核技巧才真正成为机器学习的通用基本技术。关于支持向量机和核方法有很多专门书籍和介绍性文章 [Cristianini and Shawe-Taylor, 2000; Burges, 1998; 邓乃扬与田英杰, 2009; Schölkopf et al., 1999; Schölkopf and Smola, 2002], 统计学习理论则可参阅 [Vapnik, 1995, 1998, 1999].

m 是样本个数。

支持向量机的求解通常是借助于凸优化技术 [Boyd and Vandenberghe, 2004]。如何提高效率, 使 SVM 能适用于大规模数据一直是研究重点。对线性核 SVM 已有很多成果, 例如基于割平面法(cutting plane algorithm)的 SVMperf 具有线性复杂度 [Joachims, 2006], 基于随机梯度下降的 Pegasos 速度甚至更快 [Shalev-Shwartz et al., 2011], 而坐标下降法则在稀疏数据上有很高的效率 [Hsieh et al., 2008]。非线性核 SVM 的时间复杂度在理论上不可能低于 $O(m^2)$, 因此研究重点是设计快速近似算法, 如基于采样的 CVM [Tsang et al., 2006]、基于低秩逼近的 Nyström 方法 [Williams and Seeger, 2001]、基于随机傅里叶特征的方法 [Rahimi and Recht, 2007] 等。最近有研究显示, 当核矩阵特征值有很大差别时, Nyström 方法往往优于随机傅里叶特征方法 [Yang et al., 2012]。

支持向量机是针对二分类任务设计的, 对多分类任务要进行专门的推广 [Hsu and Lin, 2002], 对带结构输出的任务也已有相应的算法 [Tschantzidis

et al., 2005]. 支持向量回归的研究始于 [Drucker et al., 1997], [Smola and Schölkopf, 2004] 给出了一个较为全面的介绍.

核函数直接决定了支持向量机与核方法的最终性能, 但遗憾的是, 核函数的选择是一个未决问题. 多核学习(multiple kernel learning) 使用多个核函数并通过学习获得其最优凸组合作为最终的核函数 [Lanckriet et al., 2004; Bach et al., 2004], 这实际上是在借助集成学习机制.

集成学习参见第8章.

一致性亦称“相合性”

替代损失函数在机器学习中被广泛使用. 但是, 通过求解替代损失函数得到的是否仍是原问题的解? 这在理论上称为替代损失的“一致性”(consistency)问题. [Vapnik and Chervonenkis, 1991] 给出了基于替代损失进行经验风险最小化的一致性充要条件, [Zhang, 2004] 证明了几种常见凸替代损失函数的一致性.

SVM 已有很多软件包, 比较著名的有 LIBSVM [Chang and Lin, 2011] 和 LIBLINEAR [Fan et al., 2008] 等.

习题

LIBSVM 见 <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.

西瓜数据集 3.0 α 见 p.89
的表 4.5.

UCI 数据集见
<http://archive.ics.uci.edu/ml/>.

- 6.1** 试证明样本空间中任意点 x 到超平面 (w, b) 的距离为式(6.2).
- 6.2** 试使用 LIBSVM, 在西瓜数据集 3.0 α 上分别用线性核和高斯核训练一个 SVM, 并比较其支持向量的差别.
- 6.3** 选择两个 UCI 数据集, 分别用线性核和高斯核训练一个 SVM, 并与 BP 神经网络和 C4.5 决策树进行实验比较.
- 6.4** 试讨论线性判别分析与线性核支持向量机在何种条件下等价.
- 6.5** 试述高斯核 SVM 与 RBF 神经网络之间的联系.
- 6.6** 试析 SVM 对噪声敏感的原因.
- 6.7** 试给出式(6.52)的完整 KKT 条件.
- 6.8** 以西瓜数据集 3.0 α 的“密度”为输入, “含糖率”为输出, 试使用 LIBSVM 训练一个 SVR.
- 6.9** 试使用核技巧推广对率回归, 产生“核对率回归”.
- 6.10*** 试设计一个能显著减少 SVM 中支持向量的数目而不显著降低泛化性能的方法.

参考文献

- 邓乃扬与田英杰. (2009). 支持向量机: 理论、算法与拓展. 科学出版社, 北京.
- Bach, R. R., G. R. G. Lanckriet, and M. I. Jordan. (2004). “Multiple kernel learning, conic duality, and the SMO algorithm.” In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 6–13, Banff, Canada.
- Boyd, S. and L. Vandenberghe. (2004). *Convex Optimization*. Cambridge University Press, Cambridge, UK.
- Burges, C. J. C. (1998). “A tutorial on support vector machines for pattern recognition.” *Data Mining and Knowledge Discovery*, 2(1):121–167.
- Chang, C.-C. and C.-J. Lin. (2011). “LIBSVM: A library for support vector machines.” *ACM Transactions on Intelligent Systems and Technology*, 2(3): 27.
- Cortes, C. and V. N. Vapnik. (1995). “Support vector networks.” *Machine Learning*, 20(3):273–297.
- Cristianini, N. and J. Shawe-Taylor. (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, UK.
- Drucker, H., C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. (1997). “Support vector regression machines.” In *Advances in Neural Information Processing Systems 9 (NIPS)* (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), 155–161, MIT Press, Cambridge, MA.
- Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. (2008). “LIBLINEAR: A library for large linear classification.” *Journal of Machine Learning Research*, 9:1871–1874.
- Hsieh, C.-J., K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. (2008). “A dual coordinate descent method for large-scale linear SVM.” In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 408–415, Helsinki, Finland.
- Hsu, C.-W. and C.-J. Lin. (2002). “A comparison of methods for multi-class support vector machines.” *IEEE Transactions on Neural Networks*, 13(2): 415–425.

- Joachims, T. (1998). "Text classification with support vector machines: Learning with many relevant features." In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, 137–142, Chemnitz, Germany.
- Joachims, T. (2006). "Training linear SVMs in linear time." In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 217–226, Philadelphia, PA.
- Lanckriet, G. R. G., N. Cristianini, and M. I. Jordan P. Bartlett, L. El Ghaoui. (2004). "Learning the kernel matrix with semidefinite programming." *Journal of Machine Learning Research*, 5:27–72.
- Osuna, E., R. Freund, and F. Girosi. (1997). "An improved training algorithm for support vector machines." In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing (NNSP)*, 276–285, Amelia Island, FL.
- Platt, J. (1998). "Sequential minimal optimization: A fast algorithm for training support vector machines." Technical Report MSR-TR-98-14, Microsoft Research.
- Platt, J. (2000). "Probabilities for (SV) machines." In *Advances in Large Margin Classifiers* (A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, eds.), 61–74, MIT Press, Cambridge, MA.
- Rahimi, A. and B. Recht. (2007). "Random features for large-scale kernel machines." In *Advances in Neural Information Processing Systems 20 (NIPS)* (J.C. Platt, D. Koller, Y. Singer, and S. Roweis, eds.), 1177–1184, MIT Press, Cambridge, MA.
- Schölkopf, B., C. J. C. Burges, and A. J. Smola, eds. (1999). *Advances in Kernel Methods: Support Vector Learning*. MIT Press, Cambridge, MA.
- Schölkopf, B. and A. J. Smola, eds. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, Cambridge, MA.
- Shalev-Shwartz, S., Y. Singer, N. Srebro, and A. Cotter. (2011). "Pegasos: Primal estimated sub-gradient solver for SVM." *Mathematical Programming*, 127(1):3–30.
- Smola, A. J. and B. Schölkopf. (2004). "A tutorial on support vector regression." *Statistics and Computing*, 14(3):199–222.

- Tsang, I. W., J. T. Kwok, and P. Cheung. (2006). "Core vector machines: Fast SVM training on very large data sets." *Journal of Machine Learning Research*, 6:363–392.
- Tsochantaridis, I., T. Joachims, T. Hofmann, and Y. Altun. (2005). "Large margin methods for structured and interdependent output variables." *Journal of Machine Learning Research*, 6:1453–1484.
- Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer, New York, NY.
- Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley, New York, NY.
- Vapnik, V. N. (1999). "An overview of statistical learning theory." *IEEE Transactions on Neural Networks*, 10(5):988–999.
- Vapnik, V. N. and A. J. Chervonenkis. (1991). "The necessary and sufficient conditions for consistency of the method of empirical risk." *Pattern Recognition and Image Analysis*, 1(3):284–305.
- Williams, C. K. and M. Seeger. (2001). "Using the Nyström method to speed up kernel machines." In *Advances in Neural Information Processing Systems 13 (NIPS)* (T. K. Leen, T. G. Dietterich, and V. Tresp, eds.), 682–688, MIT Press, Cambridge, MA.
- Yang, T.-B., Y.-F. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou. (2012), "Nyström method vs random Fourier features: A theoretical and empirical comparison." In *Advances in Neural Information Processing Systems 25 (NIPS)* (P. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), 485–493, MIT Press, Cambridge, MA.
- Zhang, T. (2004). "Statistical behavior and consistency of classification methods based on convex risk minimization (with discussion)." *Annals of Statistics*, 32(5):56–85.

休息一会儿

小故事：统计学习理论之父弗拉基米尔·瓦普尼克

弗拉基米尔·瓦普尼克 (Vladimir N. Vapnik, 1936—) 是杰出的数学家、统计学家、计算机科学家。他出生于苏联，1958 年在乌兹别克国立大学获数学硕士学位，1964 年在莫斯科控制科学学院获统计学博士学位，此后一直在该校工作并担任计算机系主任。1990 年(苏联解体的前一年)他离开苏联来到新泽西州的美国电话电报公司贝尔实验室工作，1995 年发表了最初的 SVM 文章。当时神经网络正当红，因此这篇文章被权威期刊 *Machine Learning* 要求以“支持向量网络”的名义发表。



SVM 的确与神经网络有密切联系：若将隐层神经元数设置为训练样本数，且每个训练样本对应一个神经元中心，则以高斯径向基函数为激活函数的 RBF 网络(参见 5.5.1 节)恰与高斯核 SVM 的预测函数相同。

实际上，瓦普尼克在 1963 年就已提出了支持向量的概念，1968 年他与另一位苏联数学家 A. Chervonenkis 提出了以他们两人的姓氏命名的“VC 维”，1974 年又提出了结构风险最小化原则，使得统计学习理论在二十世纪七十年代就已成型。但这些工作主要是以俄文发表的，直到瓦普尼克随着东欧剧变和苏联解体导致的苏联科学家移民潮来到美国，这方面的研究才在西方学术界引起重视，统计学习理论、支持向量机、核方法在二十世纪末大红大紫。

瓦普尼克 2002 年离开美国电话电报公司加入普林斯顿的 NEC 实验室，2014 年加盟脸书(Facebook)公司人工智能实验室。1995 年之后他还在伦敦大学、哥伦比亚大学等校任教授。据说瓦普尼克在苏联根据一本字典自学了英语及其发音。他有一句名言被广为传诵：“*Nothing is more practical than a good theory.*”

第 7 章 贝叶斯分类器

7.1 贝叶斯决策论

贝叶斯决策论(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。下面我们以多分类任务为例来解释其基本原理。

假设有 N 种可能的类别标记，即 $\mathcal{Y} = \{c_1, c_2, \dots, c_N\}$ ， λ_{ij} 是将一个真实标记为 c_j 的样本误分类为 c_i 所产生的损失。基于后验概率 $P(c_i | \mathbf{x})$ 可获得将样本 \mathbf{x} 分类为 c_i 所产生的期望损失(expected loss)，即在样本 \mathbf{x} 上的“条件风险”(conditional risk)

决策论中将“期望损失”称为“风险”(risk)。

$$R(c_i | \mathbf{x}) = \sum_{j=1}^N \lambda_{ij} P(c_j | \mathbf{x}). \quad (7.1)$$

我们的任务是寻找一个判定准则 $h : \mathcal{X} \mapsto \mathcal{Y}$ 以最小化总体风险

$$R(h) = \mathbb{E}_{\mathbf{x}}[R(h(\mathbf{x}) | \mathbf{x})]. \quad (7.2)$$

显然，对每个样本 \mathbf{x} ，若 h 能最小化条件风险 $R(h(\mathbf{x}) | \mathbf{x})$ ，则总体风险 $R(h)$ 也将被最小化。这就产生了贝叶斯判定准则(Bayes decision rule)：为最小化总体风险，只需在每个样本上选择那个能使条件风险 $R(c | \mathbf{x})$ 最小的类别标记，即

$$h^*(\mathbf{x}) = \arg \min_{c \in \mathcal{Y}} R(c | \mathbf{x}), \quad (7.3)$$

此时， h^* 称为贝叶斯最优分类器(Bayes optimal classifier)，与之对应的总体风险 $R(h^*)$ 称为贝叶斯风险(Bayes risk)。 $1 - R(h^*)$ 反映了分类器所能达到的最好性能，即通过机器学习所能产生的模型精度的理论上限。

错误率对应于 0/1 损失函数，参见第 6 章。

相对来说，若目标是最小化分类错误率，则误判损失 λ_{ij} 可写为

$$\lambda_{ij} = \begin{cases} 0, & \text{if } i = j; \\ 1, & \text{otherwise,} \end{cases} \quad (7.4)$$

此时条件风险

$$R(c | \mathbf{x}) = 1 - P(c | \mathbf{x}), \quad (7.5)$$

于是, 最小化分类错误率的贝叶斯最优分类器为

$$h^*(\mathbf{x}) = \arg \max_{c \in \mathcal{Y}} P(c | \mathbf{x}), \quad (7.6)$$

即对每个样本 \mathbf{x} , 选择能使后验概率 $P(c | \mathbf{x})$ 最大的类别标记.

注意, 这只是从概率框架的角度来理解机器学习; 事实上很多机器学习技术无须准确估计出后验概率就能准确进行分类.

不难看出, 欲使用贝叶斯判定准则来最小化决策风险, 首先要获得后验概率 $P(c | \mathbf{x})$. 然而, 在现实任务中这通常难以直接获得. 从这个角度来看, 机器学习所要实现的是基于有限的训练样本集尽可能准确地估计出后验概率 $P(c | \mathbf{x})$. 大体来说, 主要有两种策略: 给定 \mathbf{x} , 可通过直接建模 $P(c | \mathbf{x})$ 来预测 c , 这样得到的是“判别式模型”(discriminative models); 也可先对联合概率分布 $P(\mathbf{x}, c)$ 建模, 然后再由此获得 $P(c | \mathbf{x})$, 这样得到的是“生成式模型”(generative models). 显然, 前面介绍的决策树、BP 神经网络、支持向量机等, 都可归入判别式模型的范畴. 对生成式模型来说, 必然考虑

$$P(c | \mathbf{x}) = \frac{P(\mathbf{x}, c)}{P(\mathbf{x})}. \quad (7.7)$$

基于贝叶斯定理, $P(c | \mathbf{x})$ 可写为

$$P(c | \mathbf{x}) = \frac{P(c) P(\mathbf{x} | c)}{P(\mathbf{x})}, \quad (7.8)$$

$P(\mathbf{x})$ 对所有类标记均相同.

为便于讨论, 我们假设所有属性均为离散型, 对连续属性, 可将概率质量函数 $P(\cdot)$ 换成概率密度函数 $p(\cdot)$.

其中, $P(c)$ 是类“先验”(prior)概率; $P(\mathbf{x} | c)$ 是样本 \mathbf{x} 相对于类标记 c 的类条件概率(class-conditional probability), 或称为“似然”(likelihood); $P(\mathbf{x})$ 是用于归一化的“证据”(evidence)因子. 对给定样本 \mathbf{x} , 证据因子 $P(\mathbf{x})$ 与类标记无关, 因此估计 $P(c | \mathbf{x})$ 的问题就转化为如何基于训练数据 D 来估计先验 $P(c)$ 和似然 $P(\mathbf{x} | c)$.

类先验概率 $P(c)$ 表达了样本空间中各类样本所占的比例, 根据大数定律, 当训练集包含充足的独立同分布样本时, $P(c)$ 可通过各类样本出现的频率来进行估计.

对类条件概率 $P(\mathbf{x} | c)$ 来说, 由于它涉及关于 \mathbf{x} 所有属性的联合概率, 直

参见 7.3 节.

接根据样本出现的频率来估计将会遇到严重的困难. 例如, 假设样本的 d 个属性都是二值的, 则样本空间将有 2^d 种可能的取值, 在现实应用中, 这个值往往远大于训练样本数 m , 也就是说, 很多样本取值在训练集中根本没有出现, 直接使用频率来估计 $P(\mathbf{x} | c)$ 显然不可行, 因为“未被观测到”与“出现概率为零”通常是不同的.

7.2 极大似然估计

连续分布下为概率密度函数 $p(\mathbf{x} | c)$.

从二十世纪二三十年代开始出现了频率主义学派和贝叶斯学派的争论, 至今仍在继续. 两派在很多重要问题上观点不同, 甚至在对概率的基本解释上就有分歧. 有兴趣的读者可参阅 [Efron, 2005; Samaniego, 2010].

亦称“极大似然法”.

估计类条件概率的一种常用策略是先假定其具有某种确定的概率分布形式, 再基于训练样本对概率分布的参数进行估计. 具体地, 记关于类别 c 的类条件概率为 $P(\mathbf{x} | c)$, 假设 $P(\mathbf{x} | c)$ 具有确定的形式并且被参数向量 $\boldsymbol{\theta}_c$ 唯一确定, 则我们的任务就是利用训练集 D 估计参数 $\boldsymbol{\theta}_c$. 为明确起见, 我们将 $P(\mathbf{x} | c)$ 记为 $P(\mathbf{x} | \boldsymbol{\theta}_c)$.

事实上, 概率模型的训练过程就是参数估计(parameter estimation)过程. 对于参数估计, 统计学界的两个学派分别提供了不同的解决方案: 频率主义学派(Frequentist)认为参数虽然未知, 但却是客观存在的固定值, 因此, 可通过优化似然函数等准则来确定参数值; 贝叶斯学派(Bayesian)则认为参数是未观察到的随机变量, 其本身也可有分布, 因此, 可假定参数服从一个先验分布, 然后基于观测到的数据来计算参数的后验分布. 本节介绍源自频率主义学派的极大似然估计(Maximum Likelihood Estimation, 简称 MLE), 这是根据数据采样来估计概率分布参数的经典方法.

令 D_c 表示训练集 D 中第 c 类样本组成的集合, 假设这些样本是独立同分布的, 则参数 $\boldsymbol{\theta}_c$ 对于数据集 D_c 的似然是

$$P(D_c | \boldsymbol{\theta}_c) = \prod_{\mathbf{x} \in D_c} P(\mathbf{x} | \boldsymbol{\theta}_c). \quad (7.9)$$

对 $\boldsymbol{\theta}_c$ 进行极大似然估计, 就是去寻找能最大化似然 $P(D_c | \boldsymbol{\theta}_c)$ 的参数值 $\hat{\boldsymbol{\theta}}_c$. 直观上看, 极大似然估计是试图在 $\boldsymbol{\theta}_c$ 所有可能的取值中, 找到一个能使数据出现的“可能性”最大的值.

式(7.9)中的连乘操作易造成下溢, 通常使用对数似然(log-likelihood)

$$\begin{aligned} LL(\boldsymbol{\theta}_c) &= \log P(D_c | \boldsymbol{\theta}_c) \\ &= \sum_{\mathbf{x} \in D_c} \log P(\mathbf{x} | \boldsymbol{\theta}_c), \end{aligned} \quad (7.10)$$

此时参数 θ_c 的极大似然估计 $\hat{\theta}_c$ 为

$$\hat{\theta}_c = \arg \max_{\theta_c} LL(\theta_c). \quad (7.11)$$

\mathcal{N} 为正态分布, 参见附录 C.1.7.

例如, 在连续属性情形下, 假设概率密度函数 $p(\mathbf{x} | c) \sim \mathcal{N}(\mu_c, \sigma_c^2)$, 则参数 μ_c 和 σ_c^2 的极大似然估计为

$$\hat{\mu}_c = \frac{1}{|D_c|} \sum_{\mathbf{x} \in D_c} \mathbf{x}, \quad (7.12)$$

$$\hat{\sigma}_c^2 = \frac{1}{|D_c|} \sum_{\mathbf{x} \in D_c} (\mathbf{x} - \hat{\mu}_c)(\mathbf{x} - \hat{\mu}_c)^T. \quad (7.13)$$

也就是说, 通过极大似然法得到的正态分布均值就是样本均值, 方差就是 $(\mathbf{x} - \hat{\mu}_c)(\mathbf{x} - \hat{\mu}_c)^T$ 的均值, 这显然是一个符合直觉的结果. 在离散属性情形下, 也可通过类似的方式估计类条件概率.

需注意的是, 这种参数化的方法虽能使类条件概率估计变得相对简单, 但估计结果的准确性严重依赖于所假设的概率分布形式是否符合潜在的真实数据分布. 在现实应用中, 欲做出能较好地接近潜在真实分布的假设, 往往需在一定程度上利用关于应用任务本身的经验知识, 否则若仅凭“猜测”来假设概率分布形式, 很可能产生误导性的结果.

7.3 朴素贝叶斯分类器

不难发现, 基于贝叶斯公式(7.8)来估计后验概率 $P(c | \mathbf{x})$ 的主要困难在于: 类条件概率 $P(\mathbf{x} | c)$ 是所有属性上的联合概率, 难以从有限的训练样本直接估计而得. 为避开这个障碍, 朴素贝叶斯分类器(naïve Bayes classifier)采用了“属性条件独立性假设”(attribute conditional independence assumption): 对已知类别, 假设所有属性相互独立. 换言之, 假设每个属性独立地对分类结果发生影响.

基于有限训练样本直接估计联合概率, 在计算上将会遭遇组合爆炸问题, 在数据上将会遭遇样本稀疏问题; 属性数越多, 问题越严重.

基于属性条件独立性假设, 式(7.8)可重写为

$$P(c | \mathbf{x}) = \frac{P(c) P(\mathbf{x} | c)}{P(\mathbf{x})} = \frac{P(c)}{P(\mathbf{x})} \prod_{i=1}^d P(x_i | c), \quad (7.14)$$

x_i 实际上是一个“属性-值”对，例如“色泽=青绿”。为便于讨论，在上下文明确时，有时我们用 x_i 表示第 i 个属性对应的变量（如“色泽”），有时直接用其指代 x 在第 i 个属性上的取值（如“青绿”）。

其中 d 为属性数目， x_i 为 x 在第 i 个属性上的取值。

由于对所有类别来说 $P(x)$ 相同，因此基于式(7.6)的贝叶斯判定准则有

$$h_{nb}(x) = \arg \max_{c \in \mathcal{Y}} P(c) \prod_{i=1}^d P(x_i | c), \quad (7.15)$$

这就是朴素贝叶斯分类器的表达式。

显然，朴素贝叶斯分类器的训练过程就是基于训练集 D 来估计类先验概率 $P(c)$ ，并为每个属性估计条件概率 $P(x_i | c)$ 。

令 D_c 表示训练集 D 中第 c 类样本组成的集合，若有充足的独立同分布样本，则可容易地估计出类先验概率

$$P(c) = \frac{|D_c|}{|D|}. \quad (7.16)$$

对离散属性而言，令 D_{c,x_i} 表示 D_c 中在第 i 个属性上取值为 x_i 的样本组成的集合，则条件概率 $P(x_i | c)$ 可估计为

$$P(x_i | c) = \frac{|D_{c,x_i}|}{|D_c|}. \quad (7.17)$$

对连续属性可考虑概率密度函数，假定 $p(x_i | c) \sim \mathcal{N}(\mu_{c,i}, \sigma_{c,i}^2)$ ，其中 $\mu_{c,i}$ 和 $\sigma_{c,i}^2$ 分别是第 c 类样本在第 i 个属性上取值的均值和方差，则有

$$p(x_i | c) = \frac{1}{\sqrt{2\pi}\sigma_{c,i}} \exp\left(-\frac{(x_i - \mu_{c,i})^2}{2\sigma_{c,i}^2}\right). \quad (7.18)$$

下面我们用西瓜数据集 3.0 训练一个朴素贝叶斯分类器，对测试例“测 1”进行分类：

西瓜数据集 3.0 见 p.84
表 4.3.

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
测 1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	？

首先估计类先验概率 $P(c)$ ，显然有

$$P(\text{好瓜} = \text{是}) = \frac{8}{17} \approx 0.471,$$

$$P(\text{好瓜} = \text{否}) = \frac{9}{17} \approx 0.529.$$

注意，当样本数目足够多时才能进行有意义的概率估计。本书仅是以西瓜数据集3.0对估计过程做一个简单的演示。

然后，为每个属性估计条件概率 $P(x_i | c)$:

$$P_{\text{青绿|是}} = P(\text{色泽} = \text{青绿} | \text{好瓜} = \text{是}) = \frac{3}{8} = 0.375,$$

$$P_{\text{青绿|否}} = P(\text{色泽} = \text{青绿} | \text{好瓜} = \text{否}) = \frac{3}{9} \approx 0.333,$$

$$P_{\text{蜷缩|是}} = P(\text{根蒂} = \text{蜷缩} | \text{好瓜} = \text{是}) = \frac{5}{8} = 0.375,$$

$$P_{\text{蜷缩|否}} = P(\text{根蒂} = \text{蜷缩} | \text{好瓜} = \text{否}) = \frac{3}{9} \approx 0.333,$$

$$P_{\text{浊响|是}} = P(\text{敲声} = \text{浊响} | \text{好瓜} = \text{是}) = \frac{6}{8} = 0.750,$$

$$P_{\text{浊响|否}} = P(\text{敲声} = \text{浊响} | \text{好瓜} = \text{否}) = \frac{4}{9} \approx 0.444,$$

$$P_{\text{清晰|是}} = P(\text{纹理} = \text{清晰} | \text{好瓜} = \text{是}) = \frac{7}{8} = 0.875,$$

$$P_{\text{清晰|否}} = P(\text{纹理} = \text{清晰} | \text{好瓜} = \text{否}) = \frac{2}{9} \approx 0.222,$$

$$P_{\text{凹陷|是}} = P(\text{脐部} = \text{凹陷} | \text{好瓜} = \text{是}) = \frac{6}{8} = 0.750,$$

$$P_{\text{凹陷|否}} = P(\text{脐部} = \text{凹陷} | \text{好瓜} = \text{否}) = \frac{2}{9} \approx 0.222,$$

$$P_{\text{硬滑|是}} = P(\text{触感} = \text{硬滑} | \text{好瓜} = \text{是}) = \frac{6}{8} = 0.750,$$

$$P_{\text{硬滑|否}} = P(\text{触感} = \text{硬滑} | \text{好瓜} = \text{否}) = \frac{6}{9} \approx 0.667,$$

$$p_{\text{密度: 0.697|是}} = p(\text{密度} = 0.697 | \text{好瓜} = \text{是})$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.129} \exp\left(-\frac{(0.697 - 0.574)^2}{2 \cdot 0.129^2}\right) \approx 1.959,$$

$$p_{\text{密度: 0.697|否}} = p(\text{密度} = 0.697 | \text{好瓜} = \text{否})$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.195} \exp\left(-\frac{(0.697 - 0.496)^2}{2 \cdot 0.195^2}\right) \approx 1.203,$$

$$p_{\text{含糖: 0.460|是}} = p(\text{含糖率} = 0.460 | \text{好瓜} = \text{是})$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.101} \exp\left(-\frac{(0.460 - 0.279)^2}{2 \cdot 0.101^2}\right) \approx 0.788,$$

$$p_{\text{含糖: 0.460|否}} = p(\text{含糖率} = 0.460 | \text{好瓜} = \text{否})$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.108} \exp\left(-\frac{(0.460 - 0.154)^2}{2 \cdot 0.108^2}\right) \approx 0.066.$$

于是, 有

实践中常通过取对数的方式来将“连乘”转化为“连加”以避免数值下溢.

$$\begin{aligned} P(\text{好瓜} = \text{是}) & \times P_{\text{青绿}}|\text{是} \times P_{\text{蜷缩}}|\text{是} \times P_{\text{浊响}}|\text{是} \times P_{\text{清晰}}|\text{是} \times P_{\text{凹陷}}|\text{是} \\ & \times P_{\text{硬滑}}|\text{是} \times p_{\text{密度: 0.697}}|\text{是} \times p_{\text{含糖: 0.460}}|\text{是} \approx 0.038, \\ P(\text{好瓜} = \text{否}) & \times P_{\text{青绿}}|\text{否} \times P_{\text{蜷缩}}|\text{否} \times P_{\text{浊响}}|\text{否} \times P_{\text{清晰}}|\text{否} \times P_{\text{凹陷}}|\text{否} \\ & \times P_{\text{硬滑}}|\text{否} \times p_{\text{密度: 0.697}}|\text{否} \times p_{\text{含糖: 0.460}}|\text{否} \approx 6.80 \times 10^{-5}. \end{aligned}$$

由于 $0.038 > 6.80 \times 10^{-5}$, 因此, 朴素贝叶斯分类器将测试样本“测 1”判别为“好瓜”.

需注意, 若某个属性值在训练集中没有与某个类同时出现过, 则直接基于式(7.17)进行概率估计, 再根据式(7.15)进行判别将出现问题. 例如, 在使用西瓜数据集 3.0 训练朴素贝叶斯分类器时, 对一个“敲声=清脆”的测试例, 有

$$P_{\text{清脆}}|\text{是} = P(\text{敲声} = \text{清脆} | \text{好瓜} = \text{是}) = \frac{0}{8} = 0,$$

由于式(7.15)的连乘式计算出的概率值为零, 因此, 无论该样本的其他属性是什么, 哪怕在其他属性上明显像好瓜, 分类的结果都将是“好瓜=否”, 这显然不太合理.

为了避免其他属性携带的信息被训练集中未出现的属性值“抹去”, 在估计概率值时通常要进行“平滑”(smoothing), 常用“拉普拉斯修正”(Laplacian correction). 具体来说, 令 N 表示训练集 D 中可能的类别数, N_i 表示第 i 个属性可能的取值数, 则式(7.16)和(7.17)分别修正为

$$\hat{P}(c) = \frac{|D_c| + 1}{|D| + N}, \quad (7.19)$$

$$\hat{P}(x_i | c) = \frac{|D_{c,x_i}| + 1}{|D_c| + N_i}. \quad (7.20)$$

例如, 在本节的例子中, 类先验概率可估计为

$$\hat{P}(\text{好瓜} = \text{是}) = \frac{8 + 1}{17 + 2} \approx 0.474, \quad \hat{P}(\text{好瓜} = \text{否}) = \frac{9 + 1}{17 + 2} \approx 0.526.$$

类似地, $P_{\text{青绿}}|\text{是}$ 和 $P_{\text{青绿}}|\text{否}$ 可估计为

$$\hat{P}_{\text{青绿}}|\text{是} = \hat{P}(\text{色泽} = \text{青绿} | \text{好瓜} = \text{是}) = \frac{3 + 1}{8 + 3} \approx 0.364,$$

$$\hat{P}_{\text{青绿|否}} = \hat{P}(\text{色泽} = \text{青绿} \mid \text{好瓜} = \text{否}) = \frac{3+1}{9+3} \approx 0.333.$$

同时, 上文提到的概率 $P_{\text{清脆|是}}$ 可估计为

$$\hat{P}_{\text{清脆|是}} = \hat{P}(\text{敲声} = \text{清脆} \mid \text{好瓜} = \text{是}) = \frac{0+1}{8+3} \approx 0.091.$$

拉普拉斯修正实质上假设了属性值与类别均匀分布, 这是在朴素贝叶斯学习过程中额外引入的关于数据的先验.

懒惰学习参见 10.1 节.

增量学习参见 5.5.2 节.

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”(lazy learning)方式, 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.

7.4 半朴素贝叶斯分类器

为了降低贝叶斯公式(7.8)中估计后验概率 $P(c \mid \mathbf{x})$ 的困难, 朴素贝叶斯分类器采用了属性条件独立性假设, 但在现实任务中这个假设往往很难成立. 于是, 人们尝试对属性条件独立性假设进行一定程度的放松, 由此产生了一类称为“半朴素贝叶斯分类器”(semi-naïve Bayes classifiers)的学习方法.

半朴素贝叶斯分类器的基本想法是适当考虑一部分属性间的相互依赖信息, 从而既不需进行完全联合概率计算, 又不至于彻底忽略了比较强的属性依赖关系.“独依赖估计”(One-Dependent Estimator, 简称 ODE)是半朴素贝叶斯分类器最常用的一种策略. 顾名思议, 所谓“独依赖”就是假设每个属性在类别之外最多仅依赖于一个其他属性, 即

$$P(c \mid \mathbf{x}) \propto P(c) \prod_{i=1}^d P(x_i \mid c, pa_i), \quad (7.21)$$

其中 pa_i 为属性 x_i 所依赖的属性, 称为 x_i 的父属性. 此时, 对每个属性 x_i , 若其父属性 pa_i 已知, 则可采用类似式(7.20)的办法来估计概率值 $P(x_i \mid c, pa_i)$. 于是, 问题的关键就转化为如何确定每个属性的父属性, 不同的做法产生不同

的独依赖分类器.

最直接的做法是假设所有属性都依赖于同一个属性, 称为“超父”(super-parent), 然后通过交叉验证等模型选择方法来确定超父属性, 由此形成了SPODE(Super-Parent ODE)方法. 例如, 在图7.1(b)中, x_1 是超父属性.

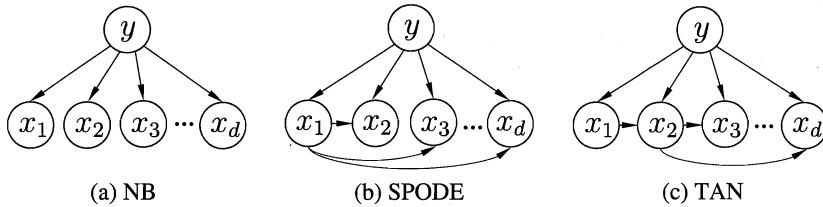


图 7.1 朴素贝叶斯与两种半朴素贝叶斯分类器所考虑的属性依赖关系

TAN(Tree Augmented naïve Bayes) [Friedman et al., 1997] 则是在最大带权生成树(maximum weighted spanning tree)算法 [Chow and Liu, 1968] 的基础上, 通过以下步骤将属性间依赖关系约简为如图7.1(c)所示的树形结构:

(1) 计算任意两个属性之间的条件互信息(conditional mutual information)

$$I(x_i, x_j | y) = \sum_{x_i, x_j; c \in \mathcal{Y}} P(x_i, x_j | c) \log \frac{P(x_i, x_j | c)}{P(x_i | c)P(x_j | c)} ; \quad (7.22)$$

(2) 以属性为结点构建完全图, 任意两个结点之间边的权重设为 $I(x_i, x_j | y)$;

(3) 构建此完全图的最大带权生成树, 挑选根变量, 将边置为有向;

(4) 加入类别结点 y , 增加从 y 到每个属性的有向边.

容易看出, 条件互信息 $I(x_i, x_j | y)$ 刻画了属性 x_i 和 x_j 在已知类别情况下的相关性, 因此, 通过最大生成树算法, TAN 实际上仅保留了强相关属性之间的依赖性.

AODE(Averaged One-Dependent Estimator) [Webb et al., 2005] 是一种基于集成学习机制、更为强大的独依赖分类器. 与 SPODE 通过模型选择确定超父属性不同, AODE 尝试将每个属性作为超父来构建 SPODE, 然后将那些

具有足够训练数据支撑的 SPODE 集成起来作为最终结果, 即

$$P(c \mid \mathbf{x}) \propto \sum_{\substack{i=1 \\ |D_{x_i}| \geq m'}}^d P(c, x_i) \prod_{j=1}^d P(x_j \mid c, x_i), \quad (7.23)$$

m' 默认设为 30 [Webb et al., 2005]. 其中 D_{x_i} 是在第 i 个属性上取值为 x_i 的样本的集合, m' 为阈值常数. 显然, AODE 需估计 $P(c, x_i)$ 和 $P(x_j \mid c, x_i)$. 类似式(7.20), 有

$$\hat{P}(c, x_i) = \frac{|D_{c,x_i}| + 1}{|D| + N_i}, \quad (7.24)$$

$$\hat{P}(x_j \mid c, x_i) = \frac{|D_{c,x_i,x_j}| + 1}{|D_{c,x_i}| + N_j}, \quad (7.25)$$

其中 N_i 是第 i 个属性可能的取值数, D_{c,x_i} 是类别为 c 且在第 i 个属性上取值为 x_i 的样本集合, D_{c,x_i,x_j} 是类别为 c 且在第 i 和第 j 个属性上取值分别为 x_i 和 x_j 的样本集合. 例如, 对西瓜数据集 3.0 有

$$\hat{P}_{\text{是, 浊响}} = \hat{P}(\text{好瓜} = \text{是}, \text{敲声} = \text{浊响}) = \frac{6+1}{17+3} = 0.350,$$

$$\hat{P}_{\text{凹陷|是, 浊响}} = \hat{P}(\text{脐部} = \text{凹陷} \mid \text{好瓜} = \text{是}, \text{敲声} = \text{浊响}) = \frac{3+1}{6+3} = 0.444.$$

不难看出, 与朴素贝叶斯分类器类似, AODE 的训练过程也是“计数”, 即在训练数据集上对符合条件的样本进行计数的过程. 与朴素贝叶斯分类器相似, AODE 无需模型选择, 既能通过预算节省预测时间, 也能采取懒惰学习方式在预测时再进行计数, 并且易于实现增量学习.

“高阶依赖”即对多个属性依赖.

既然将属性条件独立性假设放松为独依赖假设可能获得泛化性能的提升, 那么, 能否通过考虑属性间的高阶依赖来进一步提升泛化性能呢? 也就是说, 将式(7.23)中的属性 pa_i 替换为包含 k 个属性的集合 \mathbf{pa}_i , 从而将 ODE 拓展为 kDE. 需注意的是, 随着 k 的增加, 准确估计概率 $P(x_i \mid y, \mathbf{pa}_i)$ 所需的训练样本数量将以指数级增加. 因此, 若训练数据非常充分, 泛化性能有可能提升; 但在有限样本条件下, 则又陷入估计高阶联合概率的泥沼.

贝叶斯网是一种经典的概率图模型. 概率图模型参见第 14 章.

7.5 贝叶斯网

贝叶斯网(Bayesian network)亦称“信念网”(belief network), 它借助有向无环图(Directed Acyclic Graph, 简称 DAG)来刻画属性之间的依赖关系, 并使

为了简化讨论, 本节假定所有属性均为离散型。对于连续属性, 条件概率表可推广为条件概率密度函数。

这里已将西瓜数据集的连续属性“含糖率”转化为离散属性“甜度”。

用条件概率表(Conditional Probability Table, 简称 CPT)来描述属性的联合概率分布。

具体来说, 一个贝叶斯网 B 由结构 G 和参数 Θ 两部分构成, 即 $B = \langle G, \Theta \rangle$ 。网络结构 G 是一个有向无环图, 其每个结点对应于一个属性, 若两个属性有直接依赖关系, 则它们由一条边连接起来; 参数 Θ 定量描述这种依赖关系, 假设属性 x_i 在 G 中的父结点集为 π_i , 则 Θ 包含了每个属性的条件概率表 $\theta_{x_i|\pi_i} = P_B(x_i | \pi_i)$ 。

作为一个例子, 图 7.2 给出了西瓜问题的一种贝叶斯网结构和属性“根蒂”的条件概率表。从图中网络结构可看出, “色泽”直接依赖于“好瓜”和“甜度”, 而“根蒂”则直接依赖于“甜度”; 进一步从条件概率表能得到“根蒂”对“甜度”量化依赖关系, 如 $P(\text{根蒂} = \text{硬挺} | \text{甜度} = \text{高}) = 0.1$ 等。

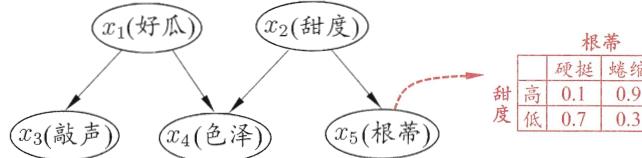


图 7.2 西瓜问题的一种贝叶斯网结构以及属性“根蒂”的条件概率表

7.5.1 结构

贝叶斯网结构有效地表达了属性间的条件独立性。给定父结点集, 贝叶斯网假设每个属性与它的非后裔属性独立, 于是 $B = \langle G, \Theta \rangle$ 将属性 x_1, x_2, \dots, x_d 的联合概率分布定义为

$$P_B(x_1, x_2, \dots, x_d) = \prod_{i=1}^d P_B(x_i | \pi_i) = \prod_{i=1}^d \theta_{x_i|\pi_i}. \quad (7.26)$$

以图 7.2 为例, 联合概率分布定义为

$$P(x_1, x_2, x_3, x_4, x_5) = P(x_1)P(x_2)P(x_3 | x_1)P(x_4 | x_1, x_2)P(x_5 | x_2),$$

这里并未列举出所有的条件独立关系。

显然, x_3 和 x_4 在给定 x_1 的取值时独立, x_4 和 x_5 在给定 x_2 的取值时独立, 分别简记为 $x_3 \perp x_4 | x_1$ 和 $x_4 \perp x_5 | x_2$ 。

图 7.3 显示出贝叶斯网中三个变量之间的典型依赖关系, 其中前两种在式(7.26)中已有所体现。

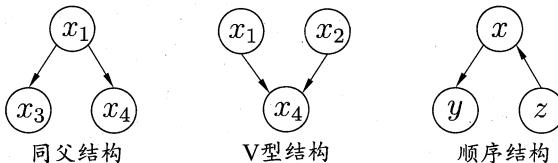


图 7.3 贝叶斯网中三个变量之间的典型依赖关系

在“同父”(common parent)结构中,给定父结点 x_1 的取值,则 x_3 与 x_4 条件独立。在“顺序”结构中,给定 x 的值,则 y 与 z 条件独立。 V 型结构(V -structure)亦称“冲撞”结构,给定子结点 x_4 的取值, x_1 与 x_2 必不独立;奇妙的是,若 x_4 的取值完全未知,则 V 型结构下 x_1 与 x_2 却是相互独立的。我们做一个简单的验证:

$$\begin{aligned} P(x_1, x_2) &= \sum_{x_4} P(x_1, x_2, x_4) \\ &= \sum_{x_4} P(x_4 | x_1, x_2)P(x_1)P(x_2) \\ &= P(x_1)P(x_2). \end{aligned} \quad (7.27)$$

对变量做积分或求和亦称“边际化”(marginalization).

这样的独立性称为“边际独立性”(marginal independence),记为 $x_1 \perp\!\!\!\perp x_2$.

事实上,一个变量取值的确定与否,能对另两个变量间的独立性发生影响,这个现象并非 V 型结构所特有。例如在同父结构中,条件独立性 $x_3 \perp\!\!\!\perp x_4 | x_1$ 成立,但若 x_1 的取值未知,则 x_3 和 x_4 就不独立,即 $x_3 \perp\!\!\!\perp x_4$ 不成立;在顺序结构中, $y \perp\!\!\!\perp z | x$,但 $y \perp\!\!\!\perp z$ 不成立。

D是指“有向”(directed).

同父、顺序和 V 型结构的发现以及有向分离的提出推动了因果发现方面的研究,参阅[Pearl, 1988].

为了分析有向图中变量间的条件独立性,可使用“有向分离”(D-separation)。我们先把有向图转变为一个无向图:

- 找出有向图中的所有 V 型结构,在 V 型结构的两个父结点之间加上一条无向边;
- 将所有有向边改为无向边。

也有译为“端正图”。

“道德化”的蕴义:孩子的父母应建立牢靠的关系,否则是不道德的。

由此产生的无向图称为“道德图”(moral graph),令父结点相连的过程称为“道德化”(moralization)[Cowell et al., 1999]。

基于道德图能直观、迅速地找到变量间的条件独立性。假定道德图中有变量 x , y 和变量集合 $\mathbf{z} = \{z_i\}$,若变量 x 和 y 能在图上被 \mathbf{z} 分开,即从道德图中将

变量集合 \mathbf{z} 去除后, x 和 y 分属两个连通分支, 则称变量 x 和 y 被 \mathbf{z} 有向分离, $x \perp y \mid \mathbf{z}$ 成立. 例如, 图 7.2 所对应的道德图如图 7.4 所示, 从图中能容易地找出所有的条件独立关系: $x_3 \perp x_4 \mid x_1$, $x_4 \perp x_5 \mid x_2$, $x_3 \perp x_2 \mid x_1$, $x_3 \perp x_5 \mid x_1$, $x_3 \perp x_5 \mid x_2$ 等.

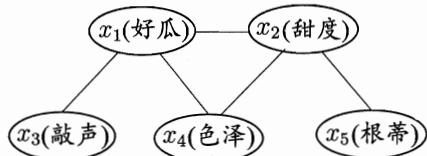


图 7.4 图 7.2 对应的道德图

7.5.2 学习

若网络结构已知, 即属性间的依赖关系已知, 则贝叶斯网的学习过程相对简单, 只需通过对训练样本“计数”, 估计出每个结点的条件概率表即可. 但在现实应用中我们往往并不知晓网络结构, 于是, 贝叶斯网学习的首要任务就是根据训练数据集来找出结构最“恰当”的贝叶斯网.“评分搜索”是求解这一问题的常用办法. 具体来说, 我们先定义一个评分函数(score function), 以此来评估贝叶斯网与训练数据的契合程度, 然后基于这个评分函数来寻找结构最优的贝叶斯网. 显然, 评分函数引入了关于我们希望获得什么样的贝叶斯网的归纳偏好.

归纳偏好参见 1.4 节.

常用评分函数通常基于信息论准则, 此类准则将学习问题看作一个数据压缩任务, 学习的目标是找到一个能以最短编码长度描述训练数据的模型, 此时编码的长度包括了描述模型自身所需的字节长度和使用该模型描述数据所需的字节长度. 对贝叶斯网学习而言, 模型就是一个贝叶斯网, 同时, 每个贝叶斯网描述了一个在训练数据上的概率分布, 自有一套编码机制能使那些经常出现的样本有更短的编码. 于是, 我们应选择那个综合编码长度(包括描述网络和编码数据)最短的贝叶斯网, 这就是“最小描述长度”(Minimal Description Length, 简称 MDL)准则.

这里我们把类别也看作一个属性, 即 \mathbf{x}_i 是一个包括示例和类别的向量.

给定训练集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, 贝叶斯网 $B = \langle G, \Theta \rangle$ 在 D 上的评分函数可写为

$$s(B \mid D) = f(\theta)|B| - LL(B \mid D), \quad (7.28)$$

其中, $|B|$ 是贝叶斯网的参数个数; $f(\theta)$ 表示描述每个参数 θ 所需的字节数; 而

$$LL(B | D) = \sum_{i=1}^m \log P_B(x_i) \quad (7.29)$$

是贝叶斯网 B 的对数似然。显然, 式(7.28)的第一项是计算编码贝叶斯网 B 所需的字节数, 第二项是计算 B 所对应的概率分布 P_B 需多少字节来描述 D 。于是, 学习任务就转化为一个优化任务, 即寻找一个贝叶斯网 B 使评分函数 $s(B | D)$ 最小。

若 $f(\theta) = 1$, 即每个参数用 1 字节描述, 则得到 AIC (Akaike Information Criterion) 评分函数

$$AIC(B | D) = |B| - LL(B | D). \quad (7.30)$$

若 $f(\theta) = \frac{1}{2} \log m$, 即每个参数用 $\frac{1}{2} \log m$ 字节描述, 则得到 BIC (Bayesian Information Criterion) 评分函数

$$BIC(B | D) = \frac{\log m}{2} |B| - LL(B | D). \quad (7.31)$$

显然, 若 $f(\theta) = 0$, 即不计算对网络进行编码的长度, 则评分函数退化为负对数似然, 相应的, 学习任务退化为极大似然估计。

不难发现, 若贝叶斯网 $B = \langle G, \Theta \rangle$ 的网络结构 G 固定, 则评分函数 $s(B | D)$ 的第一项为常数。此时, 最小化 $s(B | D)$ 等价于对参数 Θ 的极大似然估计。由式(7.29)和(7.26)可知, 参数 $\theta_{x_i|\pi_i}$ 能直接在训练数据 D 上通过经验估计获得, 即

$$\theta_{x_i|\pi_i} = \hat{P}_D(x_i | \pi_i), \quad (7.32)$$

即事件在训练数据上出现的频率。

其中 $\hat{P}_D(\cdot)$ 是 D 上的经验分布。因此, 为了最小化评分函数 $s(B | D)$, 只需对网络结构进行搜索, 而候选结构的最优参数可直接在训练集上计算得到。

不幸的是, 从所有可能的网络结构空间搜索最优贝叶斯网结构是一个 NP 难问题, 难以快速求解。有两种常用的策略能在有限时间内求得近似解: 第一种是贪心法, 例如从某个网络结构出发, 每次调整一条边(增加、删除或调整方向), 直到评分函数值不再降低为止; 第二种是通过给网络结构施加约束来削减搜索空间, 例如将网络结构限定为树形结构等。

例如 TAN [Friedman et al., 1997] 将结构限定为树形(半朴素贝叶斯分类器可看作贝叶斯网的特例)。

7.5.3 推断

类别也可看作一个属性变量.

更多关于推断的内容见第 14 章.

变分推断也很常用, 参见 14.5 节.

贝叶斯网训练好之后就能用来回答“查询”(query), 即通过一些属性变量的观测值来推测其他属性变量的取值. 例如在西瓜问题中, 若我们观测到西瓜色泽青绿、敲声浊响、根蒂蜷缩, 想知道它是否成熟、甜度如何. 这样通过已知变量观测值来推测待查询变量的过程称为“推断”(inference), 已知变量观测值称为“证据”(evidence).

最理想的是直接根据贝叶斯网定义的联合概率分布来精确计算后验概率, 不幸的是, 这样的“精确推断”已被证明是 NP 难的 [Cooper, 1990]; 换言之, 当网络结点较多、连接稠密时, 难以进行精确推断, 此时需借助“近似推断”, 通过降低精度要求, 在有限时间内求得近似解. 在现实应用中, 贝叶斯网的近似推断常使用吉布斯采样(Gibbs sampling)来完成, 这是一种随机采样方法, 我们来看看它是如何工作的.

令 $\mathbf{Q} = \{Q_1, Q_2, \dots, Q_n\}$ 表示待查询变量, $\mathbf{E} = \{E_1, E_2, \dots, E_k\}$ 为证据变量, 已知其取值为 $\mathbf{e} = \{e_1, e_2, \dots, e_k\}$. 目标是计算后验概率 $P(\mathbf{Q} = \mathbf{q} | \mathbf{E} = \mathbf{e})$, 其中 $\mathbf{q} = \{q_1, q_2, \dots, q_n\}$ 是待查询变量的一组取值. 以西瓜问题为例, 待查询变量为 $\mathbf{Q} = \{\text{好瓜, 甜度}\}$, 证据变量为 $\mathbf{E} = \{\text{色泽, 敲声, 根蒂}\}$ 且已知其取值为 $\mathbf{e} = \{\text{青绿, 浊响, 蜷缩}\}$, 查询的目标值是 $\mathbf{q} = \{\text{是, 高}\}$, 即这是好瓜且甜度高的概率有多大.

如图 7.5 所示, 吉布斯采样算法先随机产生一个与证据 $\mathbf{E} = \mathbf{e}$ 一致的样本 \mathbf{q}^0 作为初始点, 然后每步从当前样本出发产生下一个样本. 具体来说, 在第 t 次采样中, 算法先假设 $\mathbf{q}^t = \mathbf{q}^{t-1}$, 然后对非证据变量逐个进行采样改变其取值, 采样概率根据贝叶斯网 B 和其他变量的当前取值(即 $\mathbf{Z} = \mathbf{z}$)计算获得. 假定经过 T 次采样得到的与 \mathbf{q} 一致的样本共有 n_q 个, 则可近似估算出后验概率

$$P(\mathbf{Q} = \mathbf{q} | \mathbf{E} = \mathbf{e}) \simeq \frac{n_q}{T}. \quad (7.33)$$

更多关于马尔可夫链和吉布斯采样的内容参见 14.5 节.

实质上, 吉布斯采样是在贝叶斯网所有变量的联合状态空间与证据 $\mathbf{E} = \mathbf{e}$ 一致的子空间中进行“随机漫步”(random walk). 每一步仅依赖于前一步的状态, 这是一个“马尔可夫链”(Markov chain). 在一定条件下, 无论从什么初始状态开始, 马尔可夫链第 t 步的状态分布在 $t \rightarrow \infty$ 时必收敛于一个平稳分布(stationary distribution); 对于吉布斯采样来说, 这个分布恰好是 $P(\mathbf{Q} | \mathbf{E} = \mathbf{e})$. 因此, 在 T 很大时, 吉布斯采样相当于根据 $P(\mathbf{Q} | \mathbf{E} = \mathbf{e})$ 采样, 从而保证了式(7.33)收敛于 $P(\mathbf{Q} = \mathbf{q} | \mathbf{E} = \mathbf{e})$.

除去变量 Q_i 外的其他变量.

输入: 贝叶斯网 $B = \langle G, \Theta \rangle$;
 采样次数 T ;
 证据变量 \mathbf{E} 及其取值 \mathbf{e} ;
 待查询变量 \mathbf{Q} 及其取值 \mathbf{q} .

过程:

- 1: $n_q = 0$
- 2: $\mathbf{q}^0 =$ 对 \mathbf{Q} 随机赋初值
- 3: **for** $t = 1, 2, \dots, T$ **do**
- 4: **for** $Q_i \in \mathbf{Q}$ **do**
- 5: $\mathbf{Z} = \mathbf{E} \cup \mathbf{Q} \setminus \{Q_i\}$;
- 6: $\mathbf{z} = \mathbf{e} \cup \mathbf{q}^{t-1} \setminus \{q_i^{t-1}\}$;
- 7: 根据 B 计算分布 $P_B(Q_i | \mathbf{Z} = \mathbf{z})$;
- 8: $q_i^t =$ 根据 $P_B(Q_i | \mathbf{Z} = \mathbf{z})$ 采样所获 Q_i 取值;
- 9: $\mathbf{q}^t =$ 将 \mathbf{q}^{t-1} 中的 q_i^{t-1} 用 q_i^t 替换
- 10: **end for**
- 11: **if** $\mathbf{q}^t = \mathbf{q}$ **then**
- 12: $n_q = n_q + 1$
- 13: **end if**
- 14: **end for**

输出: $P(\mathbf{Q} = \mathbf{q} | \mathbf{E} = \mathbf{e}) \simeq \frac{n_q}{T}$

图 7.5 吉布斯采样算法

需注意的是, 由于马尔可夫链通常需很长时间才能趋于平稳分布, 因此吉布斯采样算法的收敛速度较慢. 此外, 若贝叶斯网中存在极端概率“0”或“1”, 则不能保证马尔可夫链存在平稳分布, 此时吉布斯采样会给出错误的估计结果.

7.6 EM算法

在前面的讨论中, 我们一直假设训练样本所有属性变量的值都已被观测到, 即训练样本是“完整”的. 但在现实应用中往往遇到“不完整”的训练样本, 例如由于西瓜的根蒂已脱落, 无法看出是“蜷缩”还是“硬挺”, 则训练样本的“根蒂”属性变量值未知. 在这种存在“未观测”变量的情形下, 是否仍能对模型参数进行估计呢?

未观测变量的学名是“隐变量”(latent variable). 令 \mathbf{X} 表示已观测变量集, \mathbf{Z} 表示隐变量集, Θ 表示模型参数. 若欲对 Θ 做极大似然估计, 则应最大化对数似然

$$LL(\Theta | \mathbf{X}, \mathbf{Z}) = \ln P(\mathbf{X}, \mathbf{Z} | \Theta). \quad (7.34)$$

由于“似然”常基于指
数族函数来定义, 因此对
数似然及后续 EM 迭代过
程中一般是使用自然对数
 $\ln(\cdot)$.

然而由于 \mathbf{Z} 是隐变量, 上式无法直接求解. 此时我们可通过对 \mathbf{Z} 计算期望, 来

最大化已观测数据的对数“边际似然”(marginal likelihood)

$$LL(\Theta | \mathbf{X}) = \ln P(\mathbf{X} | \Theta) = \ln \sum_{\mathbf{Z}} P(\mathbf{X}, \mathbf{Z} | \Theta). \quad (7.35)$$

直译为“期望最大化算法”，通常直接称 EM 算法。

这里仅给出 EM 算法的一般描述，具体例子参见 9.4.3 节。

EM (Expectation-Maximization) 算法 [Dempster et al., 1977] 是常用的估计参数隐变量的利器，它是一种迭代式的方法，其基本想法是：若参数 Θ 已知，则可根据训练数据推断出最优隐变量 \mathbf{Z} 的值 (E 步)；反之，若 \mathbf{Z} 的值已知，则可方便地对参数 Θ 做极大似然估计 (M 步)。

于是，以初始值 Θ^0 为起点，对式(7.35)，可迭代执行以下步骤直至收敛：

- 基于 Θ^t 推断隐变量 \mathbf{Z} 的期望，记为 \mathbf{Z}^t ；
- 基于已观测变量 \mathbf{X} 和 \mathbf{Z}^t 对参数 Θ 做极大似然估计，记为 Θ^{t+1} ；

这就是 EM 算法的原型。

进一步，若我们不是取 \mathbf{Z} 的期望，而是基于 Θ^t 计算隐变量 \mathbf{Z} 的概率分布 $P(\mathbf{Z} | \mathbf{X}, \Theta^t)$ ，则 EM 算法的两个步骤是：

- **E 步 (Expectation):** 以当前参数 Θ^t 推断隐变量分布 $P(\mathbf{Z} | \mathbf{X}, \Theta^t)$ ，并计算对数似然 $LL(\Theta | \mathbf{X}, \mathbf{Z})$ 关于 \mathbf{Z} 的期望

$$Q(\Theta | \Theta^t) = \mathbb{E}_{\mathbf{Z} | \mathbf{X}, \Theta^t} LL(\Theta | \mathbf{X}, \mathbf{Z}). \quad (7.36)$$

- **M 步 (Maximization):** 寻找参数最大化期望似然，即

$$\Theta^{t+1} = \arg \max_{\Theta} Q(\Theta | \Theta^t). \quad (7.37)$$

简要来说，EM 算法使用两个步骤交替计算：第一步是期望(E)步，利用当前估计的参数值来计算对数似然的期望值；第二步是最大化(M)步，寻找能使 E 步产生的似然期望最大化的参数值。然后，新得到的参数值重新被用于 E 步，……直至收敛到局部最优解。

事实上，隐变量估计问题也可通过梯度下降等优化算法求解，但由于求和的项数将随着隐变量的数目以指数级上升，会给梯度计算带来麻烦；而 EM 算法则可看作一种非梯度优化方法。

EM 算法的收敛性分析
参见 [Wu, 1983]。

EM 算法可看作坐标下降 (coordinate descent) 法来最大化对数似然下界的过程。坐标下降法参见附录 B.5。

7.7 阅读材料

贝叶斯决策论在机器学习、模式识别等诸多关注数据分析的领域都有极为重要的地位。对贝叶斯定理进行近似求解，为机器学习算法的设计提供了一种有效途径。为避免贝叶斯定理求解时面临的组合爆炸、样本稀疏问题，朴素贝叶斯分类器引入了属性条件独立性假设。这个假设在现实应用中往往很难成立，但有趣的是，朴素贝叶斯分类器在很多情形下都能获得相当好的性能 [Domingos and Pazzani, 1997; Ng and Jordan, 2002]。一种解释是对分类任务来说，只需各类别的条件概率排序正确、无须精准概率值即可导致正确分类结果 [Domingos and Pazzani, 1997]；另一种解释是，若属性间依赖对所有类别影响相同，或依赖关系的影响能相互抵消，则属性条件独立性假设在降低计算开销的同时不会对性能产生负面影响 [Zhang, 2004]。朴素贝叶斯分类器在信息检索领域尤为常用 [Lewis, 1998], [McCallum and Nigam, 1998] 对其在文本分类中的两种常见用法进行了比较。

根据对属性间依赖的涉及程度，贝叶斯分类器形成了一个“谱”：朴素贝叶斯分类器不考虑属性间依赖性，贝叶斯网能表示任意属性间的依赖性，二者分别位于“谱”的两端；介于两者之间的则是一系列半朴素贝叶斯分类器，它们基于各种假设和约束来对属性间的部分依赖性进行建模。一般认为，半朴素贝叶斯分类器的研究始于 [Kononenko, 1991]。ODE 仅考虑依赖一个父属性，由此形成了独依赖分类器如 TAN [Friedman et al., 1997]、AODE [Webb et al., 2005]、LBR (lazy Bayesian Rule) [Zheng and Webb, 2000] 等； k DE 则考虑最多依赖 k 个父属性，由此形成了 k 依赖分类器如 KDB [Sahami, 1996]、NBtree [Kohavi, 1996] 等。

贝叶斯分类器(Bayes Classifier)与一般意义上的“贝叶斯学习”(Bayesian Learning)有显著区别，前者是通过最大后验概率进行单点估计，后者则是进行分布估计。关于贝叶斯学习的内容可参阅 [Bishop, 2006]。

J. Pearl 教授因这方面的
卓越贡献而获得 2011 年
图灵奖，参见第 14 章。

贝叶斯网是经典的概率
图模型，参见第 14 章。

贝叶斯网为不确定学习和推断提供了基本框架，因其强大的表示能力、良好的可解释性而广受关注 [Pearl, 1988]。贝叶斯网学习可分为结构学习和参数学习两部分。参数学习通常较为简单，而结构学习则被证明是 NP 难问题 [Cooper, 1990; Chickering et al., 2004]，人们为此提出了多种评分搜索方法 [Friedman and Goldszmidt, 1996]。贝叶斯网通常被看作生成式模型，但近年来也有不少关于贝叶斯网判别式学习的研究 [Grossman and Domingos, 2004]。关于贝叶斯网的更多介绍可参阅 [Jensen, 1997; Heckerman, 1998]。

EM 算法是最常见的隐变量估计方法，在机器学习中有极为广泛的用途，例

如常被用来学习高斯混合模型(Gaussian mixture model, 简称 GMM) 的参数; 9.4 节将介绍的 k 均值聚类算法就是一个典型的 EM 算法. 更多关于 EM 算法的分析、拓展和应用可参阅 [McLachlan and Krishnan, 2008].

“数据挖掘十大算法”
还包括前几章介绍的
C4.5、CART 决策树、支
持向量机, 以及后几章将
要介绍的 AdaBoost、 k 均
值聚类、 k 近邻算法等.

本章介绍的朴素贝叶斯算法和 EM 算法均曾入选“数据挖掘十大算法”
[Wu et al., 2007].

习题

西瓜数据集 3.0 见 p.84
的表 4.3.

假设同先验; 参见 3.4
节.

西瓜数据集 2.0 见 p.76
的表 4.1.

- 7.1** 试使用极大似然法估算西瓜数据集 3.0 中前 3 个属性的类条件概率.
- 7.2*** 试证明: 条件独立性假设不成立时, 朴素贝叶斯分类器仍有可能产生最优贝叶斯分类器.
- 7.3** 试编程实现拉普拉斯修正的朴素贝叶斯分类器, 并以西瓜数据集 3.0 为训练集, 对 p.151 “测 1” 样本进行判别.
- 7.4** 实践中使用式(7.15)决定分类类别时, 若数据的维数非常高, 则概率连乘 $\prod_{i=1}^d P(x_i | c)$ 的结果通常会非常接近于 0 从而导致下溢. 试述防止下溢的可能方案.
- 7.5** 试证明: 二分类任务中两类数据满足高斯分布且方差相同时, 线性判别分析产生贝叶斯最优分类器.
- 7.6** 试编程实现 AODE 分类器, 并以西瓜数据集 3.0 为训练集, 对 p.151 的“测 1” 样本进行判别.
- 7.7** 给定 d 个二值属性的二分类任务, 假设对于任何先验概率项的估算至少需 30 个样例, 则在朴素贝叶斯分类器式(7.15)中估算先验概率项 $P(c)$ 需 $30 \times 2 = 60$ 个样例. 试估计在 AODE 式(7.23)中估算先验概率项 $P(c, x_i)$ 所需的样例数(分别考虑最好和最坏情形).
- 7.8** 考虑图 7.3, 试证明: 在同父结构中, 若 x_1 的取值未知, 则 $x_3 \perp\!\!\!\perp x_4$ 不成立; 在顺序结构中, $y \perp z | x$, 但 $y \perp\!\!\!\perp z$ 不成立.
- 7.9** 以西瓜数据集 2.0 为训练集, 试基于 BIC 准则构建一个贝叶斯网.
- 7.10** 以西瓜数据集 2.0 中属性“脐部”为隐变量, 试基于 EM 算法构建一个贝叶斯网.

参考文献

- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, New York, NY.
- Chickering, D. M., D. Heckerman, and C. Meek. (2004). "Large-sample learning of Bayesian networks is NP-hard." *Journal of Machine Learning Research*, 5:1287–1330.
- Chow, C. K. and C. N. Liu. (1968). "Approximating discrete probability distributions with dependence trees." *IEEE Transactions on Information Theory*, 14(3):462–467.
- Cooper, G. F. (1990). "The computational complexity of probabilistic inference using Bayesian belief networks." *Artificial Intelligence*, 42(2-3):393–405.
- Cowell, R. G., P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. (1999). *Probabilistic Networks and Expert Systems*. Springer, New York, NY.
- Dempster, A. P., N. M. Laird, and D. B. Rubin. (1977). "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society - Series B*, 39(1):1–38.
- Domingos, P. and M. Pazzani. (1997). "On the optimality of the simple Bayesian classifier under zero-one loss." *Machine Learning*, 29(2-3):103–130.
- Efron, B. (2005). "Bayesians, frequentists, and scientists." *Journal of the American Statistical Association*, 100(469):1–5.
- Friedman, N., D. Geiger, and M. Goldszmidt. (1997). "Bayesian network classifiers." *Machine Learning*, 29(2-3):131–163.
- Friedman, N. and M. Goldszmidt. (1996). "Learning Bayesian networks with local structure." In *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 252–262, Portland, OR.
- Grossman, D. and P. Domingos. (2004). "Learning Bayesian network classifiers by maximizing conditional likelihood." In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 46–53, Banff, Canada.
- Heckerman, D. (1998). "A tutorial on learning with Bayesian networks." In *Learning in Graphical Models* (M. I. Jordan, ed.), 301–354, Kluwer, Dordrecht, The Netherlands.
- Jensen, F. V. (1997). *An Introduction to Bayesian Networks*. Springer, NY.

- Kohavi, R. (1996). "Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid." In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, 202–207, Portland, OR.
- Kononenko, I. (1991). "Semi-naive Bayesian classifier." In *Proceedings of the 6th European Working Session on Learning (EWSL)*, 206–219, Porto, Portugal.
- Lewis, D. D. (1998). "Naive (Bayes) at forty: The independence assumption in information retrieval." In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, 4–15, Chemnitz, Germany.
- McCallum, A. and K. Nigam. (1998). "A comparison of event models for naive Bayes text classification." In *Working Notes of the AAAI'98 Workshop on Learning for Text Categorization*, Madison, WI.
- McLachlan, G. and T. Krishnan. (2008). *The EM Algorithm and Extensions*, 2nd edition. John Wiley & Sons, Hoboken, NJ.
- Ng, A. Y. and M. I. Jordan. (2002). "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes." In *Advances in Neural Information Processing Systems 14 (NIPS)* (T. G. Dietterich, S. Becker, and Z. Ghahramani, eds.), 841–848, MIT Press, Cambridge, MA.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA.
- Sahami, M. (1996). "Learning limited dependence Bayesian classifiers." In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, 335–338, Portland, OR.
- Samaniego, F. J. (2010). *A Comparison of the Bayesian and Frequentist Approaches to Estimation*. Springer, New York, NY.
- Webb, G., J. Boughton, and Z. Wang. (2005). "Not so naive Bayes: Aggregating one-dependence estimators." *Machine Learning*, 58(1):5–24.
- Wu, C. F. Jeff. (1983). "On the convergence properties of the EM algorithm." *Annals of Statistics*, 11(1):95–103.
- Wu, X., V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. (2007). "Top 10 algorithms in data mining." *Knowledge*

- and Information Systems*, 14(1):1–37.
- Zhang, H. (2004). “The optimality of naive Bayes.” In *Proceedings of the 17th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, 562–567, Miami, FL.
- Zheng, Z. and G. I. Webb. (2000). “Lazy learning of Bayesian rules.” *Machine Learning*, 41(1):53–84.

休息一会儿

小故事：贝叶斯之谜

英国皇家学会相当于英国科学院，皇家学会会士相当于科学院院士。

1763年12月23日，托马斯·贝叶斯 (Thomas Bayes, 1701?—1761) 的遗产受赠者 R. Price 牧师在英国皇家学会宣读了贝叶斯的遗作《论机会学说中一个问题的求解》，其中给出了贝叶斯定理，这一天现在被当作贝叶斯定理的诞生日。虽然贝叶斯定理在今天已成为概率统计最经典的内容之一，但贝叶斯本人却笼罩在谜团中。

现有资料表明，贝叶斯是一位神职人员，长期担任英国坦布里奇韦尔斯地方教堂的牧师，他从事数学研究的目的是为了证明上帝的存在。他在 1742 年当选英国皇家学会会士，但没有记录表明他此前发表过任何科学或数学论文。他的提名是由皇家学会的重量级人物签署的，但为什么提名以及他为何能当选，至今仍是个谜。贝叶斯的研究工作和他本人在他生活的时代很少有人关注，贝叶斯定理出现后很快就被遗忘了，后来大数学家拉普拉斯使它重新被科学界所熟悉，但直到二十世纪随着统计学的广泛应用才备受瞩目。贝叶斯的出生年份至今也没有清楚确定，甚至关于如今广泛流传的他的画像是不是贝叶斯本人，也仍存在争议。



第8章 集成学习

8.1 个体与集成

ensemble 读音似“昂桑宝”而非“因桑宝”。

集成学习(ensemble learning)通过构建并结合多个学习器来完成学习任务,有时也被称为多分类器系统(multi-classifier system)、基于委员会的学习(committee-based learning)等.

图 8.1 显示出集成学习的一般结构: 先产生一组“个体学习器”(individual learner),再用某种策略将它们结合起来. 个体学习器通常由一个现有的学习算法从训练数据产生,例如 C4.5 决策树算法、BP 神经网络算法等,此时集成中只包含同种类型的个体学习器,例如“决策树集成”中全是决策树,“神经网络集成”中全是神经网络,这样的集成是“同质”的(homogeneous). 同质集成中的个体学习器亦称“基学习器”(base learner),相应的学习算法称为“基学习算法”(base learning algorithm). 集成也可包含不同类型的个体学习器,例如同时包含决策树和神经网络,这样的集成是“异质”的(heterogenous). 异质集成中的个体学习器由不同的学习算法生成,这时就不再有基学习算法; 相应的, 个体学习器一般不称为基学习器, 常称为“组件学习器”(component learner)或直接称为个体学习器.

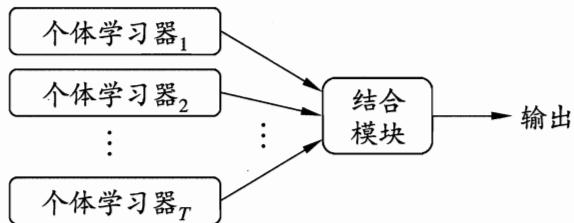


图 8.1 集成学习示意图

弱学习器常指泛化性能略优于随机猜测的学习器; 例如在二分类问题上精度略高于 50% 的分类器.

集成学习通过将多个学习器进行结合,常可获得比单一学习器显著优越的泛化性能. 这对“弱学习器”(weak learner)尤为明显,因此集成学习的很多理论研究都是针对弱学习器进行的,而基学习器有时也被直接称为弱学习器. 但需注意的是,虽然从理论上来说使用弱学习器集成足以获得好的性能,但在实

践中出于种种考虑,例如希望使用较少的个体学习器,或是重用关于常见学习器的一些经验等,人们往往会使用比较强的学习器.

在一般经验中,如果把好坏不等的东西掺到一起,那么通常结果会是比最坏的要好一些,比最好的要坏一些.集成学习把多个学习器结合起来,如何能获得比最好的单一学习器更好的性能呢?

考虑一个简单的例子:在二分类任务中,假定三个分类器在三个测试样本上的表现如图 8.2 所示,其中 \checkmark 表示分类正确, \times 表示分类错误,集成学习的结果通过投票法(voting)产生,即“少数服从多数”.在图 8.2(a) 中,每个分类器都只有 66.6% 的精度,但集成学习却达到了 100%;在图 8.2(b) 中,三个分类器没有差别,集成之后性能没有提高;在图 8.2(c) 中,每个分类器的精度都只有 33.3%,集成学习的结果变得更糟.这个简单的例子显示出:要获得好的集成,个体学习器应“好而不同”,即个体学习器要有一定的“准确性”,即学习器不能太坏,并且要有“多样性”(diversity),即学习器间具有差异.

个体学习器至少不差于弱学习器.

	测试例1	测试例2	测试例3		测试例1	测试例2	测试例3		测试例1	测试例2	测试例3
h_1	\checkmark	\checkmark	\times	h_1	\checkmark	\checkmark	\times	h_1	\checkmark	\times	\times
h_2	\times	\checkmark	\checkmark	h_2	\checkmark	\checkmark	\times	h_2	\times	\checkmark	\times
h_3	\checkmark	\times	\checkmark	h_3	\checkmark	\checkmark	\times	h_3	\times	\times	\checkmark
集成	\checkmark	\checkmark	\checkmark	集成	\checkmark	\checkmark	\times	集成	\times	\times	\times

(a) 集成提升性能
(b) 集成不起作用
(c) 集成起副作用

图 8.2 集成个体应“好而不同” (h_i 表示第 i 个分类器)

我们来做个简单的分析. 考虑二分类问题 $y \in \{-1, +1\}$ 和真实函数 f , 假定基分类器的错误率为 ϵ , 即对每个基分类器 h_i 有

$$P(h_i(\mathbf{x}) \neq f(\mathbf{x})) = \epsilon. \quad (8.1)$$

为简化讨论,假设 T 为奇数.

假设集成通过简单投票法结合 T 个基分类器,若有超过半数的基分类器正确,则集成分类就正确:

$$H(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^T h_i(\mathbf{x}) \right). \quad (8.2)$$

参见习题 8.1.

假设基分类器的错误率相互独立,则由 Hoeffding 不等式可知,集成的错误率为

$$\begin{aligned} P(H(\mathbf{x}) \neq f(\mathbf{x})) &= \sum_{k=0}^{\lfloor T/2 \rfloor} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k} \\ &\leq \exp\left(-\frac{1}{2}T(1-2\epsilon)^2\right). \end{aligned} \quad (8.3)$$

上式显示出, 随着集成中个体分类器数目 T 的增大, 集成的错误率将指数级下降, 最终趋向于零.

然而我们必须注意到, 上面的分析有一个关键假设: 基学习器的误差相互独立. 在现实任务中, 个体学习器是为解决同一个问题训练出来的, 它们显然不可能相互独立! 事实上, 个体学习器的“准确性”和“多样性”本身就存在冲突. 一般的, 准确性很高之后, 要增加多样性就需牺牲准确性. 事实上, 如何产生并结合“好而不同”的个体学习器, 恰是集成学习研究的核心.

根据个体学习器的生成方式, 目前的集成学习方法大致可分为两大类, 即个体学习器间存在强依赖关系、必须串行生成的序列化方法, 以及个体学习器间不存在强依赖关系、可同时生成的并行化方法; 前者的代表是 Boosting, 后者的代表是 Bagging 和“随机森林”(Random Forest).

8.2 Boosting

Boosting 是一族可将弱学习器提升为强学习器的算法. 这族算法的工作机制类似: 先从初始训练集训练出一个基学习器, 再根据基学习器的表现对训练样本分布进行调整, 使得先前基学习器做错的训练样本在后续受到更多关注, 然后基于调整后的样本分布来训练下一个基学习器; 如此重复进行, 直至基学习器数目达到事先指定的值 T , 最终将这 T 个基学习器进行加权结合.

Boosting 族算法最著名的代表是 AdaBoost [Freund and Schapire, 1997], 其描述如图 8.3 所示, 其中 $y_i \in \{-1, +1\}$, f 是真实函数.

AdaBoost 算法有多种推导方式, 比较容易理解的是基于“加性模型”(additive model), 即基学习器的线性组合

$$H(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \quad (8.4)$$

来最小化指数损失函数(exponential loss function) [Friedman et al., 2000]

$$\ell_{\text{exp}}(H \mid \mathcal{D}) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e^{-f(\mathbf{x})H(\mathbf{x})}]. \quad (8.5)$$

初始样本权值分布. 基于分布 D_t 从数据集 D 中训练出分类器 h_t . 估计 h_t 的误差. 确定分类器 h_t 的权重. 更新样本分布, 其中 Z_t 是规范化因子, 以确保 D_{t+1} 是一个分布.	输入: 训练集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$; 基学习算法 \mathcal{L} ; 训练轮数 T . 过程: 1: $\mathcal{D}_1(\mathbf{x}) = 1/m$. 2: for $t = 1, 2, \dots, T$ do 3: $h_t = \mathcal{L}(D, \mathcal{D}_t)$; 4: $\epsilon_t = P_{\mathbf{x} \sim \mathcal{D}_t}(h_t(\mathbf{x}) \neq f(\mathbf{x}))$; 5: if $\epsilon_t > 0.5$ then break 6: $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$; 7: $\mathcal{D}_{t+1}(\mathbf{x}) = \frac{\mathcal{D}_t(\mathbf{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\mathbf{x}) = f(\mathbf{x}) \\ \exp(\alpha_t), & \text{if } h_t(\mathbf{x}) \neq f(\mathbf{x}) \end{cases} \\ = \frac{\mathcal{D}_t(\mathbf{x}) \exp(-\alpha_t f(\mathbf{x}) h_t(\mathbf{x}))}{Z_t}$ 8: end for 输出: $H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$
--	--

图 8.3 AdaBoost 算法

若 $H(\mathbf{x})$ 能令指数损失函数最小化, 则考虑式(8.5)对 $H(\mathbf{x})$ 的偏导

$$\frac{\partial \ell_{\exp}(H | \mathcal{D})}{\partial H(\mathbf{x})} = -e^{-H(\mathbf{x})} P(f(\mathbf{x}) = 1 | \mathbf{x}) + e^{H(\mathbf{x})} P(f(\mathbf{x}) = -1 | \mathbf{x}), \quad (8.6)$$

令式(8.6)为零可解得

$$H(\mathbf{x}) = \frac{1}{2} \ln \frac{P(f(\mathbf{x}) = 1 | \mathbf{x})}{P(f(\mathbf{x}) = -1 | \mathbf{x})}, \quad (8.7)$$

因此, 有

$$\begin{aligned} \text{sign}(H(\mathbf{x})) &= \text{sign} \left(\frac{1}{2} \ln \frac{P(f(\mathbf{x}) = 1 | \mathbf{x})}{P(f(\mathbf{x}) = -1 | \mathbf{x})} \right) \\ &= \begin{cases} 1, & P(f(\mathbf{x}) = 1 | \mathbf{x}) > P(f(\mathbf{x}) = -1 | \mathbf{x}) \\ -1, & P(f(\mathbf{x}) = 1 | \mathbf{x}) < P(f(\mathbf{x}) = -1 | \mathbf{x}) \end{cases} \\ &= \arg \max_{y \in \{-1, 1\}} P(f(\mathbf{x}) = y | \mathbf{x}), \end{aligned} \quad (8.8)$$

这意味着 $\text{sign}(H(\mathbf{x}))$ 达到了贝叶斯最优错误率. 换言之, 若指数损失函数最小化, 则分类错误率也将最小化; 这说明指数损失函数是分类任务原本 0/1 损失函数的一致的(consistent)替代损失函数. 由于这个替代函数有更好的数学性

质, 例如它是连续可微函数, 因此我们用它替代 0/1 损失函数作为优化目标.

在 AdaBoost 算法中, 第一个基分类器 h_1 是通过直接将基学习算法用于初始数据分布而得; 此后迭代地生成 h_t 和 α_t , 当基分类器 h_t 基于分布 \mathcal{D}_t 产生后, 该基分类器的权重 α_t 应使得 $\alpha_t h_t$ 最小化指数损失函数

$$\begin{aligned}\ell_{\exp}(\alpha_t h_t | \mathcal{D}_t) &= \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_t} [e^{-f(\mathbf{x})\alpha_t h_t(\mathbf{x})}] \\ &= \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_t} [e^{-\alpha_t} \mathbb{I}(f(\mathbf{x}) = h_t(\mathbf{x})) + e^{\alpha_t} \mathbb{I}(f(\mathbf{x}) \neq h_t(\mathbf{x}))] \\ &= e^{-\alpha_t} P_{\mathbf{x} \sim \mathcal{D}_t}(f(\mathbf{x}) = h_t(\mathbf{x})) + e^{\alpha_t} P_{\mathbf{x} \sim \mathcal{D}_t}(f(\mathbf{x}) \neq h_t(\mathbf{x})) \\ &= e^{-\alpha_t}(1 - \epsilon_t) + e^{\alpha_t}\epsilon_t,\end{aligned}\quad (8.9)$$

其中 $\epsilon_t = P_{\mathbf{x} \sim \mathcal{D}_t}(h_t(\mathbf{x}) \neq f(\mathbf{x}))$. 考虑指数损失函数的导数

$$\frac{\partial \ell_{\exp}(\alpha_t h_t | \mathcal{D}_t)}{\partial \alpha_t} = -e^{-\alpha_t}(1 - \epsilon_t) + e^{\alpha_t}\epsilon_t,\quad (8.10)$$

令式(8.10)为零可解得

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right),\quad (8.11)$$

这恰是图 8.3 中算法第 6 行的分类器权重更新公式.

AdaBoost 算法在获得 H_{t-1} 之后样本分布将进行调整, 使下一轮的基学习器 h_t 能纠正 H_{t-1} 的一些错误. 理想的 h_t 能纠正 H_{t-1} 的全部错误, 即最小化

$$\begin{aligned}\ell_{\exp}(H_{t-1} + h_t | \mathcal{D}) &= \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} [e^{-f(\mathbf{x})(H_{t-1}(\mathbf{x}) + h_t(\mathbf{x}))}] \\ &= \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} [e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})} e^{-f(\mathbf{x})h_t(\mathbf{x})}].\end{aligned}\quad (8.12)$$

注意到 $f^2(\mathbf{x}) = h_t^2(\mathbf{x}) = 1$, 式(8.12)可使用 $e^{-f(\mathbf{x})h_t(\mathbf{x})}$ 的泰勒展式近似为

$$\begin{aligned}\ell_{\exp}(H_{t-1} + h_t | \mathcal{D}) &\simeq \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})} \left(1 - f(\mathbf{x})h_t(\mathbf{x}) + \frac{f^2(\mathbf{x})h_t^2(\mathbf{x})}{2} \right) \right] \\ &= \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})} \left(1 - f(\mathbf{x})h_t(\mathbf{x}) + \frac{1}{2} \right) \right].\end{aligned}\quad (8.13)$$

于是, 理想的基学习器

$$h_t(\mathbf{x}) = \arg \min_h \ell_{\exp}(H_{t-1} + h | \mathcal{D})$$

$$\begin{aligned}
&= \arg \min_h \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})} \left(1 - f(\mathbf{x})h(\mathbf{x}) + \frac{1}{2} \right) \right] \\
&= \arg \max_h \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})} f(\mathbf{x})h(\mathbf{x}) \right] \\
&= \arg \max_h \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[\frac{e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}}{\mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}]} f(\mathbf{x})h(\mathbf{x}) \right], \tag{8.14}
\end{aligned}$$

注意到 $\mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}]$ 是一个常数。令 \mathcal{D}_t 表示一个分布

$$\mathcal{D}_t(\mathbf{x}) = \frac{\mathcal{D}(\mathbf{x})e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}}{\mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}]}, \tag{8.15}$$

则根据数学期望的定义，这等价于令

$$\begin{aligned}
h_t(\mathbf{x}) &= \arg \max_h \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[\frac{e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}}{\mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}]} f(\mathbf{x})h(\mathbf{x}) \right] \\
&= \arg \max_h \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_t}[f(\mathbf{x})h(\mathbf{x})]. \tag{8.16}
\end{aligned}$$

由 $f(\mathbf{x}), h(\mathbf{x}) \in \{-1, +1\}$ ，有

$$f(\mathbf{x})h(\mathbf{x}) = 1 - 2 \mathbb{I}(f(\mathbf{x}) \neq h(\mathbf{x})), \tag{8.17}$$

则理想的基学习器

$$h_t(\mathbf{x}) = \arg \min_h \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_t} [\mathbb{I}(f(\mathbf{x}) \neq h(\mathbf{x}))]. \tag{8.18}$$

由此可见，理想的 h_t 将在分布 \mathcal{D}_t 下最小化分类误差。因此，弱分类器将基于分布 \mathcal{D}_t 来训练，且针对 \mathcal{D}_t 的分类误差应小于 0.5。这在一定程度上类似“残差逼近”的思想。考虑到 \mathcal{D}_t 和 \mathcal{D}_{t+1} 的关系，有

$$\begin{aligned}
\mathcal{D}_{t+1}(\mathbf{x}) &= \frac{\mathcal{D}(\mathbf{x})e^{-f(\mathbf{x})H_t(\mathbf{x})}}{\mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e^{-f(\mathbf{x})H_t(\mathbf{x})}]} \\
&= \frac{\mathcal{D}(\mathbf{x})e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}e^{-f(\mathbf{x})\alpha_t h_t(\mathbf{x})}}{\mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e^{-f(\mathbf{x})H_t(\mathbf{x})}]} \\
&= \mathcal{D}_t(\mathbf{x}) \cdot e^{-f(\mathbf{x})\alpha_t h_t(\mathbf{x})} \frac{\mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e^{-f(\mathbf{x})H_{t-1}(\mathbf{x})}]}{\mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[e^{-f(\mathbf{x})H_t(\mathbf{x})}]}, \tag{8.19}
\end{aligned}$$

这恰是图 8.3 中算法第 7 行的样本分布更新公式.

于是, 由式(8.11)和(8.19)可见, 我们从基于加性模型迭代式优化指数损失函数的角度推导出了图 8.3 的 AdaBoost 算法.

Boosting 算法要求基学习器能对特定的数据分布进行学习, 这可通过“重赋权法”(re-weighting)实施, 即在训练过程的每一轮中, 根据样本分布为每个训练样本重新赋予一个权重. 对无法接受带权样本的基学习算法, 则可通过“重采样法”(re-sampling)来处理, 即在每一轮学习中, 根据样本分布对训练集重新进行采样, 再用重采样而得的样本集对基学习器进行训练. 一般而言, 这两种做法没有显著的优劣差别. 需注意的是, Boosting 算法在训练的每一轮都要检查当前生成的基学习器是否满足基本条件(例如图 8.3 的第 5 行, 检查当前基分类器是否是比随机猜测好), 一旦条件不满足, 则当前基学习器即被抛弃, 且学习过程停止. 在此种情形下, 初始设置的学习轮数 T 也许还远未达到, 可能导致最终集成中只包含很少的基学习器而性能不佳. 若采用“重采样法”, 则可获得“重启动”机会以避免训练过程过早停止 [Kohavi and Wolpert, 1996], 即在抛弃不满足条件的当前基学习器之后, 可根据当前分布重新对训练样本进行采样, 再基于新的采样结果重新训练出基学习器, 从而使得学习过程可以持续到预设的 T 轮完成.

偏差/方差参见 2.5 节.

决策树桩即单层决策树,
参见 4.3 节.

集成的规模指集成中包
含的个体学习器数目.

从偏差-方差分解的角度看, Boosting 主要关注降低偏差, 因此 Boosting 能基于泛化性能相当弱的学习器构建出很强的集成. 我们以决策树桩为基学习器, 在表 4.5 的西瓜数据集 3.0 α 上运行 AdaBoost 算法, 不同规模(size)的集成及其基学习器所对应的分类边界如图 8.4 所示.

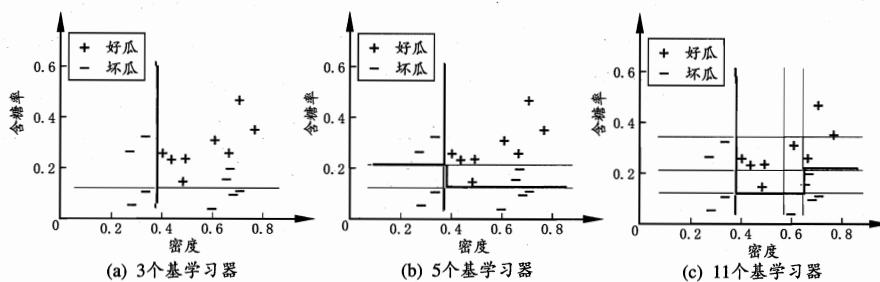


图 8.4 西瓜数据集 3.0 α 上 AdaBoost 集成规模为 3、5、11 时, 集成(红色)与基学习器(黑色)的分类边界.

8.3 Bagging与随机森林

由 8.1 节可知, 欲得到泛化性能强的集成, 集成中的个体学习器应尽可能相互独立; 虽然“独立”在现实任务中无法做到, 但可以设法使基学习器尽可能具有较大的差异。给定一个训练数据集, 一种可能的做法是对训练样本进行采样, 产生出若干个不同的子集, 再从每个数据子集中训练出一个基学习器。这样, 由于训练数据不同, 我们获得的基学习器可望具有比较大的差异。然而, 为获得好的集成, 我们同时还希望个体学习器不能太差。如果采样出的每个子集都完全不同, 则每个基学习器只用到了一小部分训练数据, 甚至不足以进行有效学习, 这显然无法确保产生出比较好的基学习器。为解决这个问题, 我们可考虑使用相互有交叠的采样子集。

8.3.1 Bagging

Bagging 这个名字是由 Bootstrap AGGREGATING 缩写而来。

Bagging [Breiman, 1996a] 是并行式集成学习方法最著名的代表。从名字即可看出, 它直接基于我们在 2.2.3 节介绍过的自助采样法 (bootstrap sampling)。给定包含 m 个样本的数据集, 我们先随机取出一个样本放入采样集中, 再把该样本放回初始数据集, 使得下次采样时该样本仍有可能被选中, 这样, 经过 m 次随机采样操作, 我们得到含 m 个样本的采样集, 初始训练集中有的样本在采样集里多次出现, 有的则从未出现。由式(2.1)可知, 初始训练集中约有 63.2% 的样本出现在采样集中。

照这样, 我们可采样出 T 个含 m 个训练样本的采样集, 然后基于每个采样集训练出一个基学习器, 再将这些基学习器进行结合。这就是 Bagging 的基本流程。在对预测输出进行结合时, Bagging 通常对分类任务使用简单投票法, 对回归任务使用简单平均法。若分类预测时出现两个类收到同样票数的情形, 则最简单的做法是随机选择一个, 也可进一步考察学习器投票的置信度来确定最终胜者。Bagging 的算法描述如图 8.5 所示。

即每个基学习器使用相同权重的投票、平均。

\mathcal{D}_{bs} 是自助采样产生的样本分布。

输入: 训练集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$;
基学习算法 \mathcal{L} ;
训练轮数 T .

过程:

- 1: **for** $t = 1, 2, \dots, T$ **do**
- 2: $h_t = \mathcal{L}(D, \mathcal{D}_{bs})$
- 3: **end for**

输出: $H(\mathbf{x}) = \arg \max_{y \in \mathcal{Y}} \sum_{t=1}^T \mathbb{I}(h_t(\mathbf{x}) = y)$

图 8.5 Bagging 算法

为处理多分类或回归任务, AdaBoost 需进行修改; 目前已有适用的变体算法 [Zhou, 2012].

包外估计参见 2.2.3 节.

假定基学习器的计算复杂度为 $O(m)$, 则 Bagging 的复杂度大致为 $T(O(m) + O(s))$, 考虑到采样与投票/平均过程的复杂度 $O(s)$ 很小, 而 T 通常是一个不太大的常数, 因此, 训练一个 Bagging 集成与直接使用基学习算法训练一个学习器的复杂度同阶, 这说明 Bagging 是一个很高效的集成学习算法. 另外, 与标准 AdaBoost 只适用于二分类任务不同, Bagging 能不经修改地用于多分类、回归等任务.

值得一提的是, 自助采样过程还给 Bagging 带来了另一个优点: 由于每个基学习器只使用了初始训练集中约 63.2% 的样本, 剩下约 36.8% 的样本可用作验证集来对泛化性能进行“包外估计”(out-of-bag estimate) [Breiman, 1996a; Wolpert and Macready, 1999]. 为此需记录每个基学习器所使用的训练样本. 不妨令 D_t 表示 h_t 实际使用的训练样本集, 令 $H^{oob}(\mathbf{x})$ 表示对样本 \mathbf{x} 的包外预测, 即仅考虑那些未使用 \mathbf{x} 训练的基学习器在 \mathbf{x} 上的预测, 有

$$H^{oob}(\mathbf{x}) = \arg \max_{y \in \mathcal{Y}} \sum_{t=1}^T \mathbb{I}(h_t(\mathbf{x}) = y) \cdot \mathbb{I}(\mathbf{x} \notin D_t), \quad (8.20)$$

则 Bagging 泛化误差的包外估计为

$$\epsilon^{oob} = \frac{1}{|D|} \sum_{(\mathbf{x}, y) \in D} \mathbb{I}(H^{oob}(\mathbf{x}) \neq y). \quad (8.21)$$

事实上, 包外样本还有许多其他用途. 例如当基学习器是决策树时, 可使用包外样本来辅助剪枝, 或用于估计决策树中各结点的后验概率以辅助对零训练样本结点的处理; 当基学习器是神经网络时, 可使用包外样本来辅助早期停止以减小过拟合风险.

偏差/方差参见 2.5 节.

关于样本扰动, 参见 8.5.3 节.

从偏差-方差分解的角度看, Bagging 主要关注降低方差, 因此它在不剪枝决策树、神经网络等易受样本扰动的学习器上效用更为明显. 我们以基于信息增益划分的决策树为基学习器, 在表 4.5 的西瓜数据集 3.0 α 上运行 Bagging 算法, 不同规模的集成及其基学习器所对应的分类边界如图 8.6 所示.

8.3.2 随机森林

随机森林(Random Forest, 简称 RF) [Breiman, 2001a] 是 Bagging 的一个扩展变体. RF 在以决策树为基学习器构建 Bagging 集成的基础上, 进一步在决策树的训练过程中引入了随机属性选择. 具体来说, 传统决策树在选择划分属性时是在当前结点的属性集合(假定有 d 个属性)中选择一个最优属性; 而在 RF 中, 对基决策树的每个结点, 先从该结点的属性集合中随机选择一个包含 k

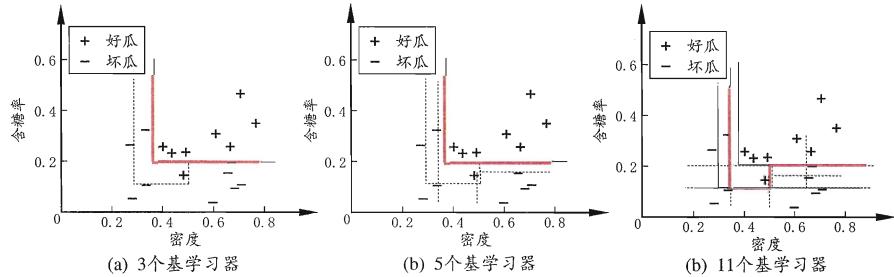


图 8.6 西瓜数据集 3.0 α 上 Bagging 集成规模为 3、5、11 时, 集成(红色)与基学习器(黑色)的分类边界.

个属性的子集, 然后再从这个子集中选择一个最优属性用于划分. 这里的参数 k 控制了随机性的引入程度: 若令 $k = d$, 则基决策树的构建与传统决策树相同; 若令 $k = 1$, 则是随机选择一个属性用于划分; 一般情况下, 推荐值 $k = \log_2 d$ [Breiman, 2001a].

随机森林简单、容易实现、计算开销小, 令人惊奇的是, 它在很多现实任务中展现出强大的性能, 被誉为“代表集成学习技术水平的方法”. 可以看出, 随机森林对 Bagging 只做了小改动, 但是与 Bagging 中基学习器的“多样性”仅通过样本扰动(通过对初始训练集采样)而来不同, 随机森林中基学习器的多样性不仅来自样本扰动, 还来自属性扰动, 这就使得最终集成的泛化性能可通过个体学习器之间差异度的增加而进一步提升.

关于样本扰动、属性扰动等, 参见 8.5.3 节.

随机森林的收敛性与 Bagging 相似. 如图 8.7 所示, 随机森林的起始性能往往相对较差, 特别是在集成中只包含一个基学习器时. 这很容易理解, 因为通过引入属性扰动, 随机森林中个体学习器的性能往往有所降低. 然而, 随着个体

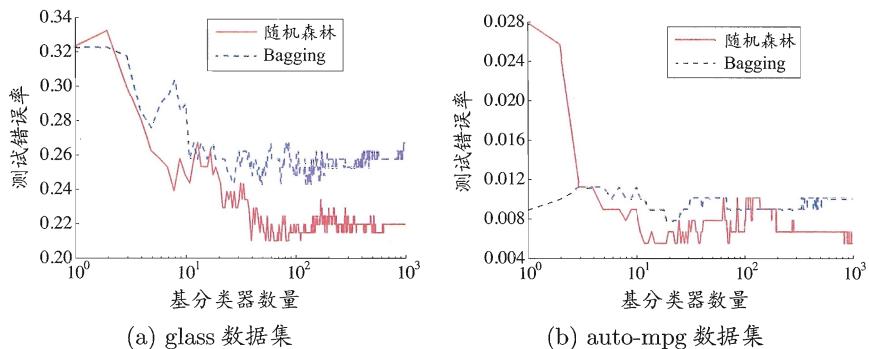


图 8.7 在两个 UCI 数据上, 集成规模对随机森林与 Bagging 的影响

学习器数目的增加, 随机森林通常会收敛到更低的泛化误差. 值得一提的是, 随机森林的训练效率常优于 Bagging, 因为在个体决策树的构建过程中, Bagging 使用的是“确定型”决策树, 在选择划分属性时要对结点的所有属性进行考察, 而随机森林使用的“随机型”决策树则只需考察一个属性子集.

8.4 结合策略

学习器结合可能会从三个方面带来好处 [Dietterich, 2000]: 首先, 从统计的方面来看, 由于学习任务的假设空间往往很大, 可能有多个假设在训练集上达到同等性能, 此时若使用单学习器可能因误选而导致泛化性能不佳, 结合多个学习器则会减小这一风险; 第二, 从计算的方面来看, 学习算法往往会陷入局部极小, 有的局部极小点所对应的泛化性能可能很糟糕, 而通过多次运行之后进行结合, 可降低陷入糟糕局部极小点的风险; 第三, 从表示的方面来看, 某些学习任务的真实假设可能不在当前学习算法所考虑的假设空间中, 此时若使用单学习器则肯定无效, 而通过结合多个学习器, 由于相应的假设空间有所扩大, 有可能学得更好的近似. 图 8.8 给出了一个直观示意图.

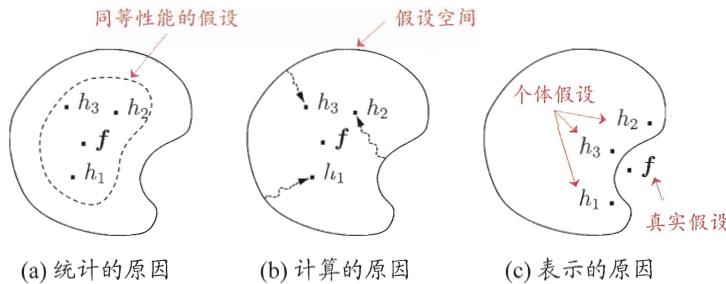


图 8.8 学习器结合可能从三个方面带来好处 [Dietterich, 2000]

假定集成包含 T 个基学习器 $\{h_1, h_2, \dots, h_T\}$, 其中 h_i 在示例 \mathbf{x} 上的输出为 $h_i(\mathbf{x})$. 本节介绍几种对 h_i 进行结合的常见策略.

8.4.1 平均法

对数值型输出 $h_i(\mathbf{x}) \in \mathbb{R}$, 最常见的结合策略是使用平均法 (averaging).

- 简单平均法(simple averaging)

$$H(\mathbf{x}) = \frac{1}{T} \sum_{i=1}^T h_i(\mathbf{x}). \quad (8.22)$$

- 加权平均法(weighted averaging)

$$H(\mathbf{x}) = \sum_{i=1}^T w_i h_i(\mathbf{x}). \quad (8.23)$$

Breiman [1996b] 在研究 Stacking 回归时发现, 必须使用非负权重才能确保集成性能优于单一最佳个体学习器, 因此在集成学习中一般对学习器的权重施以非负约束.

例如估计出个体学习器的误差, 然后令权重大小与误差大小成反比.

其中 w_i 是个体学习器 h_i 的权重, 通常要求 $w_i \geq 0$, $\sum_{i=1}^T w_i = 1$.

显然, 简单平均法是加权平均法令 $w_i = 1/T$ 的特例. 加权平均法在二十世纪五十年代已被广泛使用 [Markowitz, 1952], [Perrone and Cooper, 1993] 正式将其用于集成学习. 它在集成学习中具有特别的意义, 集成学习中的各种结合方法都可视为其特例或变体. 事实上, 加权平均法可认为是集成学习研究的基本出发点, 对给定的基学习器, 不同的集成学习方法可视为通过不同的方式来确定加权平均法中的基学习器权重.

加权平均法的权重一般是从训练数据中学习而得, 现实任务中的训练样本通常不充分或存在噪声, 这将使得学出的权重不完全可靠. 尤其是对规模比较大的集成来说, 要学习的权重比较多, 较容易导致过拟合. 因此, 实验和应用均显示出, 加权平均法未必一定优于简单平均法 [Xu et al., 1992; Ho et al., 1994; Kittler et al., 1998]. 一般而言, 在个体学习器性能相差较大时宜使用加权平均法, 而在个体学习器性能相近时宜使用简单平均法.

8.4.2 投票法

对分类任务来说, 学习器 h_i 将从类别标记集合 $\{c_1, c_2, \dots, c_N\}$ 中预测出一个标记, 最常见的结合策略是使用投票法(voting). 为便于讨论, 我们将 h_i 在样本 \mathbf{x} 上的预测输出表示为一个 N 维向量 $(h_i^1(\mathbf{x}); h_i^2(\mathbf{x}); \dots; h_i^N(\mathbf{x}))$, 其中 $h_i^j(\mathbf{x})$ 是 h_i 在类别标记 c_j 上的输出.

- 绝对多数投票法(majority voting)

$$H(\mathbf{x}) = \begin{cases} c_j, & \text{if } \sum_{i=1}^T h_i^j(\mathbf{x}) > 0.5 \sum_{k=1}^N \sum_{i=1}^T h_i^k(\mathbf{x}); \\ \text{reject}, & \text{otherwise.} \end{cases} \quad (8.24)$$

即若某标记得票过半数, 则预测为该标记; 否则拒绝预测.

- 相对多数投票法(plurality voting)

$$H(\mathbf{x}) = c_{\arg \max_j \sum_{i=1}^T h_i^j(\mathbf{x})}. \quad (8.25)$$

即预测为得票最多的标记, 若同时有多个标记获最高票, 则从中随机选取一个.

- 加权投票法(weighted voting)

$$H(\mathbf{x}) = c \arg \max_j \sum_{i=1}^T w_i h_i^j(\mathbf{x}). \quad (8.26)$$

与加权平均法类似, w_i 是 h_i 的权重, 通常 $w_i \geq 0$, $\sum_{i=1}^T w_i = 1$.

标准的绝对多数投票法(8.24)提供了“拒绝预测”选项, 这在可靠性要求较高的学习任务中是一个很好的机制. 但若学习任务要求必须提供预测结果, 则绝对多数投票法将退化为相对多数投票法. 因此, 在不允许拒绝预测的任务中, 绝对多数、相对多数投票法统称为“多数投票法”.

式(8.24)~(8.26)没有限制个体学习器输出值的类型. 在现实任务中, 不同类型个体学习器可能产生不同类型的 $h_i^j(\mathbf{x})$ 值, 常见的有:

- 类标记: $h_i^j(\mathbf{x}) \in \{0, 1\}$, 若 h_i 将样本 \mathbf{x} 预测为类别 c_j 则取值为 1, 否则为 0. 使用类标记的投票亦称“硬投票”(hard voting).
- 类概率: $h_i^j(\mathbf{x}) \in [0, 1]$, 相当于对后验概率 $P(c_j | \mathbf{x})$ 的一个估计. 使用类概率的投票亦称“软投票”(soft voting).

不同类型的 $h_i^j(\mathbf{x})$ 值不能混用. 对一些能在预测出类别标记的同时产生分类置信度的学习器, 其分类置信度可转化为类概率使用. 若此类值未进行规范化, 例如支持向量机的分类间隔值, 则必须使用一些技术如 Platt 缩放(Platt scaling) [Platt, 2000]、等分回归(isotonic regression) [Zadrozny and Elkan, 2001] 等进行“校准”(calibration)后才能作为类概率使用. 有趣的是, 虽然分类器估计出的类概率值一般都不太准确, 但基于类概率进行结合却往往比直接基于类标记进行结合性能更好. 需注意的是, 若基学习器的类型不同, 则其类概率值不能直接进行比较; 在此种情形下, 通常可将类概率输出转化为类标记输出(例如将类概率输出最大的 $h_i^j(\mathbf{x})$ 设为 1, 其他设为 0)然后再投票.

例如异质集成中不同类型的个体学习器.

Stacking 本身是一种著名的集成学习方法, 且有不少集成学习算法可视为其变体或特例. 它也可看作一种特殊的结合策略, 因此本书在此介绍.

8.4.3 学习法

当训练数据很多时, 一种更为强大的结合策略是使用“学习法”, 即通过另一个学习器来进行结合. Stacking [Wolpert, 1992; Breiman, 1996b] 是学习法的典型代表. 这里我们把个体学习器称为初级学习器, 用于结合的学习器称为次级学习器或元学习器(meta-learner).

初级学习器也可是同质的。

Stacking 先从初始数据集训练出初级学习器, 然后“生成”一个新数据集用于训练次级学习器。在这个新数据集中, 初级学习器的输出被当作样例输入特征, 而初始样本的标记仍被当作样例标记。Stacking 的算法描述如图 8.9 所示, 这里我们假定初级学习器使用不同学习算法产生, 即初级集成是异质的。

使用初级学习算法 \mathcal{L}_t
产生初级学习器 h_t 。

生成次级训练集。

在 D' 上用次级学习算
法 \mathcal{L} 产生次级学习器 h' 。

输入: 训练集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$;
初级学习算法 $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_T$;
次级学习算法 \mathcal{L} 。

过程:

```

1: for  $t = 1, 2, \dots, T$  do
2:    $h_t = \mathcal{L}_t(D)$ ;
3: end for
4:  $D' = \emptyset$ ;
5: for  $i = 1, 2, \dots, m$  do
6:   for  $t = 1, 2, \dots, T$  do
7:      $z_{it} = h_t(x_i)$ ;
8:   end for
9:    $D' = D' \cup ((z_{i1}, z_{i2}, \dots, z_{iT}), y_i)$ ;
10: end for
11:  $h' = \mathcal{L}(D')$ ;
输出:  $H(\mathbf{x}) = h'(h_1(\mathbf{x}), h_2(\mathbf{x}), \dots, h_T(\mathbf{x}))$ 
```

图 8.9 Stacking 算法

在训练阶段, 次级训练集是利用初级学习器产生的, 若直接用初级学习器的训练集来产生次级训练集, 则过拟合风险会比较大; 因此, 一般是通过使用交叉验证或留一法这样的方式, 用训练初级学习器未使用的样本来产生次级学习器的训练样本。以 k 折交叉验证为例, 初始训练集 D 被随机划分为 k 个大小相似的集合 D_1, D_2, \dots, D_k 。令 D_j 和 $\bar{D}_j = D \setminus D_j$ 分别表示第 j 折的测试集和训练集。给定 T 个初级学习算法, 初级学习器 $h_t^{(j)}$ 通过在 \bar{D}_j 上使用第 t 个学习算法而得。对 D_j 中每个样本 x_i , 令 $z_{it} = h_t^{(j)}(x_i)$, 则由 x_i 所产生的次级训练样例的示例部分为 $z_i = (z_{i1}; z_{i2}; \dots; z_{iT})$, 标记部分为 y_i 。于是, 在整个交叉验证过程结束后, 从这 T 个初级学习器产生的次级训练集是 $D' = \{(z_i, y_i)\}_{i=1}^m$, 然后 D' 将用于训练次级学习器。

MLR 是基于线性回归的分类器, 它对每个类分别进行线性回归, 属于该类的训练样例所对应的输出被置为 1, 其他类置为 0; 测试示例将被分给输出值最大的类。

WEKA 中的 StackingC 算法就是这样实现的。

次级学习器的输入属性表示和次级学习算法对 Stacking 集成的泛化性能有很大影响。有研究表明, 将初级学习器的输出类概率作为次级学习器的输入属性, 用多响应线性回归(Multi-response Linear Regression, 简称 MLR) 作为次级学习算法效果较好 [Ting and Witten, 1999], 在 MLR 中使用不同的属性集更佳 [Seewald, 2002]。

贝叶斯模型平均(Bayes Model Averaging, 简称 BMA)基于后验概率来为不同模型赋予权重, 可视为加权平均法的一种特殊实现. [Clarke, 2003] 对 Stacking 和 BMA 进行了比较. 理论上来说, 若数据生成模型恰在当前考虑的模型中, 且数据噪声很少, 则 BMA 不差于 Stacking; 然而, 在现实应用中无法确保数据生成模型一定在当前考虑的模型中, 甚至可能难以用当前考虑的模型来进行近似, 因此, Stacking 通常优于 BMA, 因为其鲁棒性比 BMA 更好, 而且 BMA 对模型近似误差非常敏感.

8.5 多样性

8.5.1 误差-分歧分解

8.1 节提到, 欲构建泛化能力强的集成, 个体学习器应“好而不同”. 现在我们来做一个简单的理论分析.

假定我们用个体学习器 h_1, h_2, \dots, h_T 通过加权平均法(8.23)结合产生的集成来完成回归学习任务 $f : \mathbb{R}^d \mapsto \mathbb{R}$. 对示例 \mathbf{x} , 定义学习器 h_i 的“分歧”(ambiguity)为

$$A(h_i | \mathbf{x}) = (h_i(\mathbf{x}) - H(\mathbf{x}))^2, \quad (8.27)$$

则集成的“分歧”是

$$\begin{aligned} \bar{A}(h | \mathbf{x}) &= \sum_{i=1}^T w_i A(h_i | \mathbf{x}) \\ &= \sum_{i=1}^T w_i (h_i(\mathbf{x}) - H(\mathbf{x}))^2. \end{aligned} \quad (8.28)$$

显然, 这里的“分歧”项表征了个体学习器在样本 \mathbf{x} 上的不一致性, 即在一定程度上反映了个体学习器的多样性. 个体学习器 h_i 和集成 H 的平方误差分别为

$$E(h_i | \mathbf{x}) = (f(\mathbf{x}) - h_i(\mathbf{x}))^2, \quad (8.29)$$

$$E(H | \mathbf{x}) = (f(\mathbf{x}) - H(\mathbf{x}))^2. \quad (8.30)$$

令 $\bar{E}(h | \mathbf{x}) = \sum_{i=1}^T w_i \cdot E(h_i | \mathbf{x})$ 表示个体学习器误差的加权均值, 有

$$\begin{aligned} \bar{A}(h | \mathbf{x}) &= \sum_{i=1}^T w_i E(h_i | \mathbf{x}) - E(H | \mathbf{x}) \\ &= \bar{E}(h | \mathbf{x}) - E(H | \mathbf{x}). \end{aligned} \quad (8.31)$$

式(8.31)对所有样本 \mathbf{x} 均成立, 令 $p(\mathbf{x})$ 表示样本的概率密度, 则在全样本上有

$$\sum_{i=1}^T w_i \int A(h_i | \mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \sum_{i=1}^T w_i \int E(h_i | \mathbf{x}) p(\mathbf{x}) d\mathbf{x} - \int E(H | \mathbf{x}) p(\mathbf{x}) d\mathbf{x}. \quad (8.32)$$

这里我们用 E_i 和 A_i 简化表示 $E(h_i)$ 和 $A(h_i)$.

类似的, 个体学习器 h_i 在全样本上的泛化误差和分歧项分别为

$$E_i = \int E(h_i | \mathbf{x}) p(\mathbf{x}) d\mathbf{x}, \quad (8.33)$$

$$A_i = \int A(h_i | \mathbf{x}) p(\mathbf{x}) d\mathbf{x}. \quad (8.34)$$

集成的泛化误差为

这里我们用 E 简化表示 $E(H)$.

$$E = \int E(H | \mathbf{x}) p(\mathbf{x}) d\mathbf{x}. \quad (8.35)$$

将式(8.33)~(8.35)代入式(8.32), 再令 $\bar{E} = \sum_{i=1}^T w_i E_i$ 表示个体学习器泛化误差的加权均值, $\bar{A} = \sum_{i=1}^T w_i A_i$ 表示个体学习器的加权分歧值, 有

$$E = \bar{E} - \bar{A}. \quad (8.36)$$

式(8.36)这个漂亮的式子明确提示出: 个体学习器准确性越高、多样性越大, 则集成越好. 上面这个分析首先由 [Krogh and Vedelsby, 1995] 给出, 称为“误差-分歧分解”(error-ambiguity decomposition).

至此, 读者可能很高兴: 我们直接把 $\bar{E} - \bar{A}$ 作为优化目标来求解, 不就能得到最优的集成了? 遗憾的是, 在现实任务中很难直接对 $\bar{E} - \bar{A}$ 进行优化, 不仅由于它们是定义在整个样本空间上, 还由于 \bar{A} 不是一个可直接操作的多样性度量, 它仅在集成构造好之后才能进行估计. 此外需注意的是, 上面的推导过程只适用于回归学习, 难以直接推广到分类学习任务上去.

8.5.2 多样性度量

亦称“差异性度量”.

顾名思义, 多样性度量(diversity measure)是用于度量集成中个体分类器的多样性, 即估算个体学习器的多样化程度. 典型做法是考虑个体分类器的两两相似/不相似性.

给定数据集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$, 对二分类任务, $y_i \in \{-1, +1\}$, 分类器 h_i 与 h_j 的预测结果列联表(contingency table)为
参见 2.3.2 节混淆矩阵.

	$h_i = +1$	$h_i = -1$
$h_j = +1$	a	c
$h_j = -1$	b	d

其中, a 表示 h_i 与 h_j 均预测为正类的样本数目; b, c, d 含义由此类推;
 $a + b + c + d = m$. 基于这个列联表, 下面给出一些常见的多样性度量.

- 不合度量(disagreement measure)

$$dis_{ij} = \frac{b + c}{m}. \quad (8.37)$$

dis_{ij} 的值域为 $[0, 1]$. 值越大则多样性越大.

- 相关系数(correlation coefficient)

$$\rho_{ij} = \frac{ad - bc}{\sqrt{(a+b)(a+c)(c+d)(b+d)}}. \quad (8.38)$$

ρ_{ij} 的值域为 $[-1, 1]$. 若 h_i 与 h_j 无关, 则值为 0; 若 h_i 与 h_j 正相关则值为正, 否则为负.

- Q -统计量(Q -statistic)

$$Q_{ij} = \frac{ad - bc}{ad + bc}. \quad (8.39)$$

Q_{ij} 与相关系数 ρ_{ij} 的符号相同, 且 $|Q_{ij}| \leq |\rho_{ij}|$.

- κ -统计量(κ -statistic)

$$\kappa = \frac{p_1 - p_2}{1 - p_2}. \quad (8.40)$$

其中, p_1 是两个分类器取得一致的概率; p_2 是两个分类器偶然达成一致的概率, 它们可由数据集 D 估算:

$$p_1 = \frac{a + d}{m}, \quad (8.41)$$

$$p_2 = \frac{(a+b)(a+c) + (c+d)(b+d)}{m^2}. \quad (8.42)$$

若分类器 h_i 与 h_j 在 D 上完全一致, 则 $\kappa = 1$; 若它们仅是偶然达成一致,

则 $\kappa = 0$. κ 通常为非负值, 仅在 h_i 与 h_j 达成一致的概率甚至低于偶然性的情况下取负值.

以上介绍的都是“成对型”(pairwise)多样性度量, 它们可以容易地通过 2 维图绘制出来. 例如著名的“ κ -误差图”, 就是将每一对分类器作为图上的一个点, 横坐标是这对分类器的 κ 值, 纵坐标是它们的平均误差, 图 8.10 给出了一个例子. 显然, 数据点云的位置越高, 则个体分类器准确性越低; 点云的位置越靠右, 则个体学习器的多样性越小.

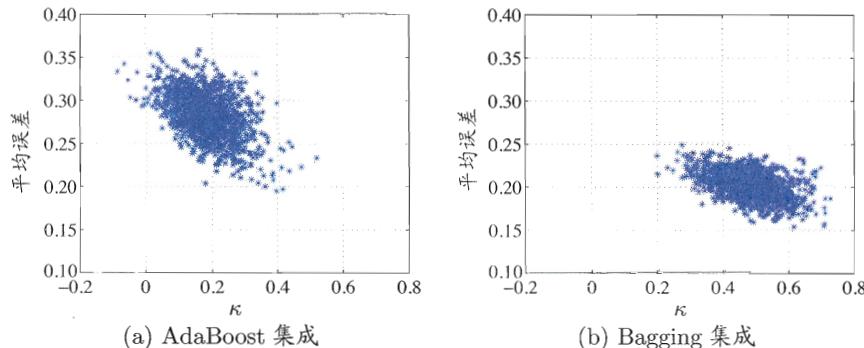


图 8.10 在 UCI 数据集 *tic-tac-toe* 上的 κ -误差图. 每个集成含 50 棵 C4.5 决策树

8.5.3 多样性增强

在集成学习中需有效地生成多样性大的个体学习器. 与简单地直接用初始数据训练出个体学习器相比, 如何增强多样性呢? 一般思路是在学习过程中引入随机性, 常见做法主要是对数据样本、输入属性、输出表示、算法参数进行扰动.

- 数据样本扰动

给定初始数据集, 可从中产生出不同的数据子集, 再利用不同的数据子集训练出不同的个体学习器. 数据样本扰动通常是基于采样法, 例如在 Bagging 中使用自助采样, 在 AdaBoost 中使用序列采样. 此类做法简单高效, 使用最广. 对很多常见的基学习器, 例如决策树、神经网络等, 训练样本稍加变化就会导致学习器有显著变动, 数据样本扰动法对这样的“不稳定基学习器”很有效; 然而, 有一些基学习器对数据样本的扰动不敏感, 例如线性学习器、支持向量机、朴素贝叶斯、 k 近邻学习器等, 这样的基学习器称为稳定基学习器(stable base learner), 对此类基学习器进行集成往往需使用输入属性扰动等其他机制.

- 输入属性扰动

子空间一般指从初始的高维属性空间投影产生的低维属性空间, 描述低维空间的属性是通过初始属性投影变换而得, 未必是初始属性. 参见第 10 章.

训练样本通常由一组属性描述, 不同的“子空间”(subspace, 即属性子集)提供了观察数据的不同视角. 显然, 从不同子空间训练出的个体学习器必然有所不同. 著名的随机子空间(random subspace)算法 [Ho, 1998] 就依赖于输入属性扰动, 该算法从初始属性集中抽取出若干个属性子集, 再基于每个属性子集训练一个基学习器, 算法描述如图 8.11 所示. 对包含大量冗余属性的数据, 在子空间中训练个体学习器不仅能产生多样性大的个体, 还会因属性数的减少而大幅节省时间开销, 同时, 由于冗余属性多, 减少一些属性后训练出的个体学习器也不至于太差. 若数据只包含少量属性, 或者冗余属性很少, 则不宜使用输入属性扰动法.

d' 小于初始属性数 d .

\mathcal{F}_t 包含 d' 个随机选取的属性, D_t 仅保留 \mathcal{F}_t 中的属性.

输入: 训练集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$;

基学习算法 \mathcal{L} ;

基学习器数 T ;

子空间属性数 d' .

过程:

```

1: for  $t = 1, 2, \dots, T$  do
2:    $\mathcal{F}_t = \text{RS}(D, d')$ 
3:    $D_t = \text{Map}_{\mathcal{F}_t}(D)$ 
4:    $h_t = \mathcal{L}(D_t)$ 
5: end for
```

输出: $H(\mathbf{x}) = \arg \max_{y \in \mathcal{Y}} \sum_{t=1}^T \mathbb{I}(h_t(\text{Map}_{\mathcal{F}_t}(\mathbf{x})) = y)$

图 8.11 随机子空间算法

- 输出表示扰动

ECOC 参见 3.5 节.

此类做法的基本思路是对输出表示进行操纵以增强多样性. 可对训练样本的类标记稍作变动, 如“翻转法”(Flipping Output) [Breiman, 2000] 随机改变一些训练样本的标记; 也可对输出表示进行转化, 如“输出调制法”(Output Smearing) [Breiman, 2000] 将分类输出转化为回归输出后构建个体学习器; 还可将原任务拆解为多个可同时求解的子任务, 如 ECOC 法 [Dietterich and Bakiri, 1995] 利用纠错输出码将多分类任务拆解为一系列二分类任务来训练基学习器.

- 算法参数扰动

基学习算法一般都有参数需进行设置, 例如神经网络的隐层神经元数、初始连接权值等, 通过随机设置不同的参数, 往往可产生差别较大的个体学习器.

例如“负相关法”(Negative Correlation) [Liu and Yao, 1999] 显式地通过正则化项来强制个体神经网络使用不同的参数。对参数较少的算法，可通过将其学习过程中某些环节用其他类似方式代替，从而达到扰动的目的，例如可将决策树使用的属性选择机制替换成其他的属性选择机制。值得指出的是，使用单一学习器时通常需使用交叉验证等方法来确定参数值，这事实上已使用了不同参数训练出多个学习器，只不过最终仅选择其中一个学习器进行使用，而集成学习则相当于把这些学习器都利用起来；由此也可看出，集成学习技术的实际计算开销并不比使用单一学习器大很多。

不同的多样性增强机制可同时使用，例如 8.3.2 节介绍的随机森林中同时使用了数据样本扰动和输入属性扰动，有些方法甚至同时使用了更多机制 [Zhou, 2012]。

8.6 阅读材料

集成学习方面的主要推荐读物是 [Zhou, 2012]，本章提及的所有内容在该书中都有更深入详细的介绍。[Kuncheva, 2004; Rokach, 2010b] 可供参考。[Schapire and Freund, 2012] 则是专门关于 Boosting 的著作。

Boosting 源于 [Schapire, 1990] 对 [Kearns and Valiant, 1989] 提出的“弱学习是否等价于强学习”这个重要理论问题的构造性证明。最初的 Boosting 算法仅有理论意义，经数年努力后 [Freund and Schapire, 1997] 提出 AdaBoost，并因此获得理论计算机科学方面的重要奖项——哥德尔奖。不同集成学习方法的工作机理和理论性质往往有显著不同，例如从偏差-方差分解的角度看，Boosting 主要关注降低偏差，而 Bagging 主要关注降低方差。MultiBoosting [Webb, 2000] 等方法尝试将二者的优点加以结合。关于 Boosting 和 Bagging 已有很多理论研究结果，可参阅 [Zhou, 2012] 第 2~3 章。

8.2 节给出的 AdaBoost 推导源于“统计视角”(statistical view) [Friedman et al., 2000]，此派理论认为 AdaBoost 实质上是基于加性模型(additive model)以类似牛顿迭代法来优化指数损失函数。受此启发，通过将迭代优化过程替换为其他优化方法，产生了 GradientBoosting [Friedman, 2001]、LPBoost [Demiriz et al., 2008] 等变体算法。然而，这派理论产生的推论与 AdaBoost 实际行为有相当大的差别 [Mease and Wyner, 2008]，尤其是它不能解释 AdaBoost 为什么没有过拟合这个重要现象，因此不少人认为，统计视角本身虽很有意义，但其阐释的是一个与 AdaBoost 相似的学习过程而非 AdaBoost 本身。“间隔理论”(margin theory) [Schapire et al., 1998] 能直观地解释这个重要现象，

这个现象的严格表述是“为什么 AdaBoost 在训练误差达到零之后继续训练仍能提高泛化性能”；若一直训练下去，过拟合最终仍会出现。

但过去 15 年中一直存有争论, 直到最近的研究结果使它最终得以确立, 并对新型学习方法的设计给出了启示; 相关内容可参阅 [Zhou, 2014].

本章仅介绍了最基本的几种结合方法, 常见的还有基于 D-S 证据理论的方法、动态分类器选择、混合专家(mixture of experts) 等. 本章仅介绍了成对型多样性度量. [Kuncheva and Whitaker, 2003; Tang et al., 2006] 显示出, 现有多样性度量都存在显著缺陷. 如何理解多样性, 被认为是集成学习中的圣杯问题. 关于结合方法和多样性方面的内容, 可参阅 [Zhou, 2012] 第 4~5 章.

在集成产生之后再试图通过去除一些个体学习器来获得较小的集成, 称为集成修剪(ensemble pruning). 这有助于减小模型的存储开销和预测时间开销. 早期研究主要针对序列化集成进行, 减小集成规模后常导致泛化性能下降 [Rokach, 2010a]; [Zhou et al., 2002] 揭示出对并行化集成进行修剪能在减小规模的同时提升泛化性能, 并催生了基于优化的集成修剪技术. 这方面的内容可参阅 [Zhou, 2012] 第 6 章.

关于聚类、半监督学习、代价敏感学习等任务中集成学习的内容, 可参阅 [Zhou, 2012] 第 7~8 章. 事实上, 集成学习已被广泛用于几乎所有的学习任务. 著名数据挖掘竞赛 KDDCup 历年的冠军几乎都使用了集成学习.

由于集成包含多个学习器, 即便个体学习器有较好的可解释性, 集成仍是黑箱模型. 已有一些工作试图改善集成的可解释性, 例如将集成转化为单模型、从集成中抽取符号规则等, 这方面的研究衍生出了能产生性能超越集成的单学习器的“二次学习”(twice-learning)技术, 例如 NeC4.5 算法 [Zhou and Jiang, 2004]. 可视化技术也对改善可解释性有一定帮助. 可参阅 [Zhou, 2012] 第 8 章.

对并行化集成的修剪亦称“选择性集成”(selective ensemble), 但现在一般将选择性集成用作集成修剪的同义语, 亦称“集成选择”(ensemble selection).

习题

- 8.1** 假设抛硬币正面朝上的概率为 p , 反面朝上的概率为 $1 - p$. 令 $H(n)$ 代表抛 n 次硬币所得正面朝上的次数, 则最多 k 次正面朝上的概率为

$$P(H(n) \leq k) = \sum_{i=0}^k \binom{n}{i} p^i (1-p)^{n-i}. \quad (8.43)$$

对 $\delta > 0$, $k = (p - \delta)n$, 有 Hoeffding 不等式

$$P(H(n) \leq (p - \delta)n) \leq e^{-2\delta^2 n}. \quad (8.44)$$

试推导出式(8.3).

西瓜数据集 3.0 α 见 p.89
表 4.5.

- 8.2** 对于 0/1 损失函数来说, 指数损失函数并非仅有的一致替代函数. 考虑式(8.5), 试证明: 任意损失函数 $\ell(-f(\mathbf{x})H(\mathbf{x}))$, 若对于 $H(\mathbf{x})$ 在区间 $[-\infty, \delta]$ ($\delta > 0$) 上单调递减, 则 ℓ 是 0/1 损失函数的一致替代函数.
- 8.3** 从网上下载或自己编程实现 AdaBoost, 以不剪枝决策树为基学习器, 在西瓜数据集 3.0 α 上训练一个 AdaBoost 集成, 并与图 8.4 进行比较.
- 8.4** GradientBoosting [Friedman, 2001] 是一种常用的 Boosting 算法, 试析其与 AdaBoost 的异同.
- 8.5** 试编程实现 Bagging, 以决策树桩为基学习器, 在西瓜数据集 3.0 α 上训练一个 Bagging 集成, 并与图 8.6 进行比较.
- 8.6** 试析 Bagging 通常为何难以提升朴素贝叶斯分类器的性能.
- 8.7** 试析随机森林为何比决策树 Bagging 集成的训练速度更快.
- 8.8** MultiBoosting 算法 [Webb, 2000] 将 AdaBoost 作为 Bagging 的基学习器, Iterative Bagging 算法 [Breiman, 2001b] 则是将 Bagging 作为 AdaBoost 的基学习器. 试比较二者的优缺点.
- 8.9*** 试设计一种可视的多样性度量, 对习题 8.3 和习题 8.5 中得到的集成进行评估, 并与 κ -误差图比较.
- 8.10*** 试设计一种能提升 k 近邻分类器性能的集成学习算法.

参考文献

- Breiman, L. (1996a). "Bagging predictors." *Machine Learning*, 24(2):123–140.
- Breiman, L. (1996b). "Stacked regressions." *Machine Learning*, 24(1):49–64.
- Breiman, L. (2000). "Randomizing outputs to increase prediction accuracy." *Machine Learning*, 40(3):113–120.
- Breiman, L. (2001a). "Random forests." *Machine Learning*, 45(1):5–32.
- Breiman, L. (2001b). "Using iterated bagging to debias regressions." *Machine Learning*, 45(3):261–277.
- Clarke, B. (2003). "Comparing Bayes model averaging and stacking when model approximation error cannot be ignored." *Journal of Machine Learning Research*, 4:683–712.
- Demiriz, A., K. P. Bennett, and J. Shawe-Taylor. (2008). "Linear programming Boosting via column generation." *Machine Learning*, 46(1-3):225–254.
- Dietterich, T. G. (2000). "Ensemble methods in machine learning." In *Proceedings of the 1st International Workshop on Multiple Classifier Systems (MCS)*, 1–15, Cagliari, Italy.
- Dietterich, T. G. and G. Bakiri. (1995). "Solving multiclass learning problems via error-correcting output codes." *Journal of Artificial Intelligence Research*, 2:263–286.
- Freund, Y. and R. E. Schapire. (1997). "A decision-theoretic generalization of on-line learning and an application to boosting." *Journal of Computer and System Sciences*, 55(1):119–139.
- Friedman, J., T. Hastie, and R. Tibshirani. (2000). "Additive logistic regression: A statistical view of boosting (with discussions)." *Annals of Statistics*, 28(2):337–407.
- Friedman, J. H. (2001). "Greedy function approximation: A gradient Boosting machine." *Annals of Statistics*, 29(5):1189–1232.
- Ho, T. K. (1998). "The random subspace method for constructing decision forests." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844.
- Ho, T. K., J. J. Hull, and S. N. Srihari. (1994). "Decision combination in multi-

- ple classifier systems.” *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16(1):66–75.
- Kearns, M. and L. G. Valiant. (1989). “Cryptographic limitations on learning Boolean formulae and finite automata.” In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, 433–444, Seattle, WA.
- Kittler, J., M. Hatef, R. Duin, and J. Matas. (1998). “On combining classifiers.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3): 226–239.
- Kohavi, R. and D. H. Wolpert. (1996). “Bias plus variance decomposition for zero-one loss functions.” In *Proceedings of the 13th International Conference on Machine Learning (ICML)*, 275–283, Bari, Italy.
- Krogh, A. and J. Vedelsby. (1995). “Neural network ensembles, cross validation, and active learning.” In *Advances in Neural Information Processing Systems 7 (NIPS)* (G. Tesauro, D. S. Touretzky, and T. K. Leen, eds.), 231–238, MIT Press, Cambridge, MA.
- Kuncheva, L. I. (2004). *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, Hoboken, NJ.
- Kuncheva, L. I. and C. J. Whitaker. (2003). “Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy.” *Machine Learning*, 51(2):181–207.
- Liu, Y. and X. Yao. (1999). “Ensemble learning via negative correlation.” *Neural Networks*, 12(10):1399–1404.
- Markowitz, H. (1952). “Portfolio selection.” *Journal of Finance*, 7(1):77–91.
- Mease, D. and A. Wyner. (2008). “Evidence contrary to the statistical view of boosting (with discussions).” *Journal of Machine Learning Research*, 9: 131–201.
- Perrone, M. P. and L. N. Cooper. (1993). “When networks disagree: Ensemble method for neural networks.” In *Artificial Neural Networks for Speech and Vision* (R. J. Mammone, ed.), 126–142, Chapman & Hall, New York, NY.
- Platt, J. C. (2000). “Probabilities for SV machines.” In *Advances in Large Margin Classifiers* (A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, eds.), 61–74, MIT Press, Cambridge, MA.

- Rokach, L. (2010a). "Ensemble-based classifiers." *Artificial Intelligence Review*, 33(1):1–39.
- Rokach, L. (2010b). *Pattern Classification Using Ensemble Methods*. World Scientific, Singapore.
- Schapire, R. E. (1990). "The strength of weak learnability." *Machine Learning*, 5(2):197–227.
- Schapire, R. E. and Y. Freund. (2012). *Boosting: Foundations and Algorithms*. MIT Press, Cambridge, MA.
- Schapire, R. E., Y. Freund, P. Bartlett, and W. S. Lee. (1998). "Boosting the margin: A new explanation for the effectiveness of voting methods." *Annals of Statistics*, 26(5):1651–1686.
- Seewald, A. K. (2002). "How to make Stacking better and faster while also taking care of an unknown weakness." In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, 554–561, Sydney, Australia.
- Tang, E. K., P. N. Suganthan, and X. Yao. (2006). "An analysis of diversity measures." *Machine Learning*, 65(1):247–271.
- Ting, K. M. and I. H. Witten. (1999). "Issues in stacked generalization." *Journal of Artificial Intelligence Research*, 10:271–289.
- Webb, G. I. (2000). "MultiBoosting: A technique for combining boosting and wagging." *Machine Learning*, 40(2):159–196.
- Wolpert, D. H. (1992). "Stacked generalization." *Neural Networks*, 5(2):241–260.
- Wolpert, D. H. and W. G. Macready. (1999). "An efficient method to estimate Bagging's generalization error." *Machine Learning*, 35(1):41–55.
- Xu, L., A. Krzyzak, and C. Y. Suen. (1992). "Methods of combining multiple classifiers and their applications to handwriting recognition." *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418–435.
- Zadrozny, B. and C. Elkan. (2001). "Obtaining calibrated probability estimates from decision trees and naïve Bayesian classifiers." In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, 609–616, Williamstown, MA.
- Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*.

- Chapman & Hall/CRC, Boca Raton, FL.
- Zhou, Z.-H. (2014). "Large margin distribution learning." In *Proceedings of the 6th IAPR International Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR)*, 1–11, Montreal, Canada.
- Zhou, Z.-H. and Y. Jiang. (2004). "NeC4.5: Neural ensemble based C4.5." *IEEE Transactions on Knowledge and Data Engineering*, 16(6):770–773.
- Zhou, Z.-H., J. Wu, and W. Tang. (2002). "Ensembling neural networks: Many could be better than all." *Artificial Intelligence*, 137(1-2):239–263.

休息一会儿

小故事：老当益壮的李奥·布瑞曼

李奥·布瑞曼 (Leo Breiman, 1928–2005) 是二十世纪伟大的统计学家。他在二十世纪末公开宣称，统计学界把统计搞成了抽象数学，这偏离了初衷，统计学本该是关于预测、解释和处理数据的学问。他自称与机器学习走得更近，因为这一行是在处理有挑战的数据问题。事实上，布瑞曼是一位卓越的机器学习学家，他不仅是 CART 决策树的作者，还对集成学习有三大贡献：Bagging、随机森林以及关于 Boosting 的理论探讨。有趣的是，这些都是在他 1993 年从加州大学伯克利分校统计系退休后完成的。



布瑞曼早年在加州理工学院获物理学士学位，然后打算到哥伦比亚大学念哲学，但哲学系主任告诉他，自己最优秀的两个博士生没找到工作，于是布瑞曼改学数学，先后在哥伦比亚大学和加州大学伯克利分校获得数学硕士、博士学位。他先是研究概率论，但在加州大学洛杉矶分校(UCLA)做了 7 年教授后他厌倦了概率论，于是主动辞职。为了向概率论告别，辞职后他把自己关在家里半年写了本关于概率论的书，然后他到工业界做了 13 年咨询，再回到加州大学伯克利分校统计系做教授。布瑞曼的经历极为丰富，他曾在 UCLA 学术假期间主动到联合国教科文组织工作，被安排到非洲利比里亚统计失学儿童数。他是一位业余雕塑家，甚至还与人合伙在墨西哥开过制冰厂。他自认为一生最重要的研究成果——随机森林，是 70 多岁时做出来的。

第9章 聚类

9.1 聚类任务

常见的无监督学习任务还有密度估计 (density estimation)、异常检测 (anomaly detection) 等。

对聚类算法而言，样本簇亦称“类”。

聚类任务中也可使用有标记训练样本，如 9.4.2 与 13.6 节，但样本的类标记与聚类产生的簇有所不同。

在“无监督学习”(unsupervised learning)中，训练样本的标记信息是未知的，目标是通过对无标记训练样本的学习来揭示数据的内在性质及规律，为进一步的数据分析提供基础。此类学习任务中研究最多、应用最广的是“聚类”(clustering)。

聚类试图将数据集中的样本划分为若干个通常是不相交的子集，每个子集称为一个“簇”(cluster)。通过这样的划分，每个簇可能对应于一些潜在的概念(类别)，如“浅色瓜”“深色瓜”，“有籽瓜”“无籽瓜”，甚至“本地瓜”“外地瓜”等；需说明的是，这些概念对聚类算法而言事先是未知的，聚类过程仅能自动形成簇结构，簇所对应的概念语义需由使用者来把握和命名。

形式化地说，假定样本集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ 包含 m 个无标记样本，每个样本 $\mathbf{x}_i = (x_{i1}; x_{i2}; \dots; x_{in})$ 是一个 n 维特征向量，则聚类算法将样本集 D 划分为 k 个不相交的簇 $\{C_l \mid l = 1, 2, \dots, k\}$ ，其中 $C_l \cap_{l' \neq l} C_{l'} = \emptyset$ 且 $D = \bigcup_{l=1}^k C_l$ 。相应地，我们用 $\lambda_j \in \{1, 2, \dots, k\}$ 表示样本 \mathbf{x}_j 的“簇标记”(cluster label)，即 $\mathbf{x}_j \in C_{\lambda_j}$ 。于是，聚类的结果可用包含 m 个元素的簇标记向量 $\boldsymbol{\lambda} = (\lambda_1; \lambda_2; \dots; \lambda_m)$ 表示。

聚类既能作为一个单独过程，用于找寻数据内在的分布结构，也可作为分类等其他学习任务的前驱过程。例如，在一些商业应用中需对新用户的类型进行判别，但定义“用户类型”对商家来说却可能不太容易，此时往往可先对用户数据进行聚类，根据聚类结果将每个簇定义为一个类，然后再基于这些类训练分类模型，用于判别新用户的类型。

基于不同的学习策略，人们设计出多种类型的聚类算法。本章后半部分将对不同类型的代表性算法进行介绍，但在此之前，我们先讨论聚类算法涉及的两个基本问题——性能度量和距离计算。

9.2 性能度量

聚类性能度量亦称聚类“有效性指标”(validity index)。与监督学习中的

监督学习中的性能度量
参见 2.3 节.

性能度量作用相似, 对聚类结果, 我们需通过某种性能度量来评估其好坏; 另一方面, 若明确了最终将要使用的性能度量, 则可直接将其作为聚类过程的优化目标, 从而更好地得到符合要求的聚类结果.

聚类是将样本集 D 划分为若干互不相交的子集, 即样本簇. 那么, 什么样的聚类结果比较好呢? 直观上看, 我们希望“物以类聚”, 即同一簇的样本尽可能彼此相似, 不同簇的样本尽可能不同. 换言之, 聚类结果的“簇内相似度”(intra-cluster similarity)高且“簇间相似度”(inter-cluster similarity)低.

例如将领域专家给出的
划分结果作为参考模型.

聚类性能度量大致有两类. 一类是将聚类结果与某个“参考模型”(reference model)进行比较, 称为“外部指标”(external index); 另一类是直接考察聚类结果而不利用任何参考模型, 称为“内部指标”(internal index).

通常 $k \neq s$.

对数据集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, 假定通过聚类给出的簇划分为 $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$, 参考模型给出的簇划分为 $\mathcal{C}^* = \{C_1^*, C_2^*, \dots, C_s^*\}$. 相应地, 令 λ 与 λ^* 分别表示与 \mathcal{C} 和 \mathcal{C}^* 对应的簇标记向量. 我们将样本两两配对考虑, 定义

$$a = |SS|, \quad SS = \{(\mathbf{x}_i, \mathbf{x}_j) \mid \lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}, \quad (9.1)$$

$$b = |SD|, \quad SD = \{(\mathbf{x}_i, \mathbf{x}_j) \mid \lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}, \quad (9.2)$$

$$c = |DS|, \quad DS = \{(\mathbf{x}_i, \mathbf{x}_j) \mid \lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}, \quad (9.3)$$

$$d = |DD|, \quad DD = \{(\mathbf{x}_i, \mathbf{x}_j) \mid \lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}, \quad (9.4)$$

其中集合 SS 包含了在 \mathcal{C} 中隶属于相同簇且在 \mathcal{C}^* 中也隶属于相同簇的样本对, 集合 SD 包含了在 \mathcal{C} 中隶属于相同簇但在 \mathcal{C}^* 中隶属于不同簇的样本对, ……由于每个样本对 $(\mathbf{x}_i, \mathbf{x}_j)$ ($i < j$) 仅能出现在一个集合中, 因此有 $a + b + c + d = m(m - 1)/2$ 成立.

基于式(9.1)~(9.4)可导出下面这些常用的聚类性能度量外部指标:

- Jaccard 系数(Jaccard Coefficient, 简称 JC)

$$JC = \frac{a}{a + b + c}. \quad (9.5)$$

- FM 指数(Fowlkes and Mallows Index, 简称 FMI)

$$FMI = \sqrt{\frac{a}{a+b} \cdot \frac{a}{a+c}}. \quad (9.6)$$

- Rand 指数(Rand Index, 简称 RI)

$$\text{RI} = \frac{2(a+d)}{m(m-1)}. \quad (9.7)$$

显然, 上述性能度量的结果值均在 $[0, 1]$ 区间, 值越大越好.

考虑聚类结果的簇划分 $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$, 定义

$$\text{avg}(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \leq i < j \leq |C|} \text{dist}(\mathbf{x}_i, \mathbf{x}_j), \quad (9.8)$$

$$\text{diam}(C) = \max_{1 \leq i < j \leq |C|} \text{dist}(\mathbf{x}_i, \mathbf{x}_j), \quad (9.9)$$

$$d_{\min}(C_i, C_j) = \min_{\mathbf{x}_i \in C_i, \mathbf{x}_j \in C_j} \text{dist}(\mathbf{x}_i, \mathbf{x}_j), \quad (9.10)$$

$$d_{\text{cen}}(C_i, C_j) = \text{dist}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j), \quad (9.11)$$

距离越大则样本的相似度越低; 距离计算见下节.

其中, $\text{dist}(\cdot, \cdot)$ 用于计算两个样本之间的距离; $\boldsymbol{\mu}$ 代表簇 C 的中心点 $\boldsymbol{\mu} = \frac{1}{|C|} \sum_{1 \leq i \leq |C|} \mathbf{x}_i$. 显然, $\text{avg}(C)$ 对应于簇 C 内样本间的平均距离, $\text{diam}(C)$ 对应于簇 C 内样本间的最远距离, $d_{\min}(C_i, C_j)$ 对应于簇 C_i 与簇 C_j 最近样本间的距离, $d_{\text{cen}}(C_i, C_j)$ 对应于簇 C_i 与簇 C_j 中心点间的距离.

基于式(9.8)~(9.11)可导出下面这些常用的聚类性能度量内部指标:

- DB 指数(Davies-Bouldin Index, 简称 DBI)

$$\text{DBI} = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left(\frac{\text{avg}(C_i) + \text{avg}(C_j)}{d_{\text{cen}}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)} \right). \quad (9.12)$$

- Dunn 指数(Dunn Index, 简称 DI)

$$\text{DI} = \min_{1 \leq i \leq k} \left\{ \min_{j \neq i} \left(\frac{d_{\min}(C_i, C_j)}{\max_{1 \leq l \leq k} \text{diam}(C_l)} \right) \right\}. \quad (9.13)$$

显然, DBI 的值越小越好, 而 DI 则相反, 值越大越好.

9.3 距离计算

对函数 $\text{dist}(\cdot, \cdot)$, 若它是一个“距离度量”(distance measure), 则需满足一些基本性质:

$$\text{非负性: } \text{dist}(\mathbf{x}_i, \mathbf{x}_j) \geq 0; \quad (9.14)$$

$$\text{同一性: } \text{dist}(\mathbf{x}_i, \mathbf{x}_j) = 0 \text{ 当且仅当 } \mathbf{x}_i = \mathbf{x}_j; \quad (9.15)$$

$$\text{对称性: } \text{dist}(\mathbf{x}_i, \mathbf{x}_j) = \text{dist}(\mathbf{x}_j, \mathbf{x}_i); \quad (9.16)$$

直递性常被直接称为“三角不等式”.

$$\text{直递性: } \text{dist}(\mathbf{x}_i, \mathbf{x}_j) \leq \text{dist}(\mathbf{x}_i, \mathbf{x}_k) + \text{dist}(\mathbf{x}_k, \mathbf{x}_j). \quad (9.17)$$

给定样本 $\mathbf{x}_i = (x_{i1}; x_{i2}; \dots; x_{in})$ 与 $\mathbf{x}_j = (x_{j1}; x_{j2}; \dots; x_{jn})$, 最常用的是“闵可夫斯基距离”(Minkowski distance)

式(9.18)即为 $\mathbf{x}_i - \mathbf{x}_j$ 的 L_p 范数 $\|\mathbf{x}_i - \mathbf{x}_j\|_p$.

$$\text{dist}_{mk}(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{u=1}^n |x_{iu} - x_{ju}|^p \right)^{\frac{1}{p}}. \quad (9.18)$$

对 $p \geq 1$, 式(9.18)显然满足式(9.14)~(9.17)的距离度量基本性质.

$p \mapsto \infty$ 时则得到切比雪夫距离.

$p = 2$ 时, 闵可夫斯基距离即欧氏距离(Euclidean distance)

$$\text{dist}_{ed}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 = \sqrt{\sum_{u=1}^n |x_{iu} - x_{ju}|^2}. \quad (9.19)$$

亦称“街区距离”(city block distance).

$p = 1$ 时, 闵可夫斯基距离即曼哈顿距离(Manhattan distance)

$$\text{dist}_{man}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_1 = \sum_{u=1}^n |x_{iu} - x_{ju}|. \quad (9.20)$$

连续属性亦称“数值属性”(numerical attribute), “离散属性”亦称“列名属性”(nominal attribute).

我们常将属性划分为“连续属性”(continuous attribute)和“离散属性”(categorical attribute), 前者在定义域上有无穷多个可能的取值, 后者在定义域上是有限个取值. 然而, 在讨论距离计算时, 属性上是否定义了“序”关系更为重要. 例如定义域为 {1, 2, 3} 的离散属性与连续属性的性质更接近一些, 能直接在属性值上计算距离: “1”与“2”比较接近、与“3”比较远, 这样的属性称为“有序属性”(ordinal attribute); 而定义域为{飞机, 火车, 轮船}这样的离散属性则不能直接在属性值上计算距离, 称为“无序属性”(non-ordinal attribute). 显然, 闵可夫斯基距离可用于有序属性.

样本类别已知时 k 通常设置为类别数.

对无序属性可采用 VDM (Value Difference Metric) [Stanfill and Waltz, 1986]. 令 $m_{u,a}$ 表示在属性 u 上取值为 a 的样本数, $m_{u,a,i}$ 表示在第 i 个样本簇中在属性 u 上取值为 a 的样本数, k 为样本簇数, 则属性 u 上两个离散值 a 与 b 之间的 VDM 距离为

$$\text{VDM}_p(a, b) = \sum_{i=1}^k \left| \frac{m_{u,a,i}}{m_{u,a}} - \frac{m_{u,b,i}}{m_{u,b}} \right|^p. \quad (9.21)$$

于是, 将闵可夫斯基距离和 VDM 结合即可处理混合属性. 假定有 n_c 个有序属性、 $n - n_c$ 个无序属性, 不失一般性, 令有序属性排列在无序属性之前, 则

$$\text{MinkovDM}_p(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{u=1}^{n_c} |x_{iu} - x_{ju}|^p + \sum_{u=n_c+1}^n \text{VDM}_p(x_{iu}, x_{ju}) \right)^{\frac{1}{p}}. \quad (9.22)$$

当样本空间中不同属性的重要性不同时, 可使用 “加权距离” (weighted distance). 以加权闵可夫斯基距离为例:

$$\text{dist}_{\text{wmk}}(\mathbf{x}_i, \mathbf{x}_j) = (w_1 \cdot |x_{i1} - x_{j1}|^p + \dots + w_n \cdot |x_{in} - x_{jn}|^p)^{\frac{1}{p}}, \quad (9.23)$$

其中权重 $w_i \geq 0$ ($i = 1, 2, \dots, n$) 表征不同属性的重要性, 通常 $\sum_{i=1}^n w_i = 1$.

需注意的是, 通常我们是基于某种形式的距离来定义 “相似度度量” (similarity measure), 距离越大, 相似度越小. 然而, 用于相似度度量的距离未必一定要满足距离度量的所有基本性质, 尤其是直递性(9.17). 例如在某些任务中我们可能希望有这样的相似度度量: “人” “马” 分别与 “人马” 相似, 但 “人” 与 “马” 很不相似; 要达到这个目的, 可以令 “人” “马” 与 “人马” 之间的距离都比较小, 但 “人” 与 “马” 之间的距离很大, 如图 9.1 所示, 此时该距离不再满足直递性; 这样的距离称为 “非度量距离” (non-metric distance). 此外, 本节介绍的距离计算式都是事先定义好的, 但在不少现实任务中, 有必要基于数据样本来确定合适距离计算式, 这可通过 “距离度量学习” (distance metric learning) 来实现.

参见 10.6 节.

这个例子中, 从数学上看, 令 $d_3 = 3$ 即可满足直递性; 但从语义上看, d_3 应远大于 d_1 与 d_2 .

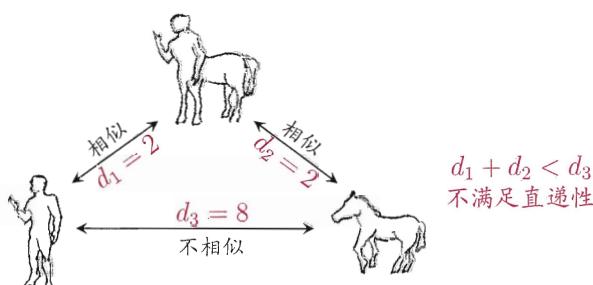


图 9.1 非度量距离的一个例子

9.4 原型聚类

“原型”是指样本空间中具有代表性的点。

原型聚类亦称“基于原型的聚类”(prototype-based clustering),此类算法假设聚类结构能通过一组原型刻画,在现实聚类任务中极为常用。通常情形下,算法先对原型进行初始化,然后对原型进行迭代更新求解。采用不同的原型表示、不同的求解方式,将产生不同的算法。下面介绍几种著名的原型聚类算法。

9.4.1 k 均值算法

给定样本集 $D = \{x_1, x_2, \dots, x_m\}$, “ k 均值”(k -means)算法针对聚类所得簇划分 $C = \{C_1, C_2, \dots, C_k\}$ 最小化平方误差

$$E = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|_2^2, \quad (9.24)$$

其中 $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$ 是簇 C_i 的均值向量。直观来看,式(9.24)在一定程度上刻画了簇内样本围绕簇均值向量的紧密程度, E 值越小则簇内样本相似度越高。

最小化式(9.24)不容易,找到它的最优解需考察样本集 D 所有可能的簇划分,这是一个 NP 难问题[Aloise et al., 2009]。因此, k 均值算法采用了贪心策略,通过迭代优化来近似求解式(9.24)。算法流程如图 9.2 所示,其中第 1 行对均值向量进行初始化,在第 4~8 行与第 9~16 行依次对当前簇划分及均值向量迭代更新,若迭代更新后聚类结果保持不变,则在第 18 行将当前簇划分结果返回。

下面以表 9.1 的西瓜数据集 4.0 为例来演示 k 均值算法的学习过程。为方便叙述,我们将编号为 i 的样本称为 x_i ,这是一个包含“密度”与“含糖率”两个属性值的二维向量。

p.89 的西瓜数据集 3.0 α
是西瓜数据集 4.0 的子集。

表 9.1 西瓜数据集 4.0

编号	密度	含糖率	编号	密度	含糖率	编号	密度	含糖率
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

样本 9~21 的类别是“好瓜=否”,其他样本的类别是“好瓜=是”。由于本节使用无标记样本,因此类别标记信息未在表中给出。

输入: 样本集 $D = \{x_1, x_2, \dots, x_m\}$;
聚类簇数 k .

过程:

- 1: 从 D 中随机选择 k 个样本作为初始均值向量 $\{\mu_1, \mu_2, \dots, \mu_k\}$
- 2: **repeat**
- 3: 令 $C_i = \emptyset (1 \leq i \leq k)$
- 4: **for** $j = 1, 2, \dots, m$ **do**
- 5: 计算样本 x_j 与各均值向量 $\mu_i (1 \leq i \leq k)$ 的距离: $d_{ji} = \|x_j - \mu_i\|_2$;
- 6: 根据距离最近的均值向量确定 x_j 的簇标记: $\lambda_j = \arg \min_{i \in \{1, 2, \dots, k\}} d_{ji}$;
- 7: 将样本 x_j 划入相应的簇: $C_{\lambda_j} = C_{\lambda_j} \cup \{x_j\}$;
- 8: **end for**
- 9: **for** $i = 1, 2, \dots, k$ **do**
- 10: 计算新均值向量: $\mu'_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$;
- 11: **if** $\mu'_i \neq \mu_i$ **then**
- 12: 将当前均值向量 μ_i 更新为 μ'_i
- 13: **else**
- 14: 保持当前均值向量不变
- 15: **end if**
- 16: **end for**
- 17: **until** 当前均值向量均未更新

输出: 簇划分 $C = \{C_1, C_2, \dots, C_k\}$

为了避免运行时间过长,
通常设置一个最大运行轮
数或最小调整幅度阈值,
若达到最大轮数或调整幅
度小于阈值, 则停止运行.

图 9.2 k 均值算法

假定聚类簇数 $k = 3$, 算法开始时随机选取三个样本 x_6, x_{12}, x_{27} 作为初始均值向量, 即

$$\mu_1 = (0.403; 0.237), \mu_2 = (0.343; 0.099), \mu_3 = (0.532; 0.472).$$

考察样本 $x_1 = (0.697; 0.460)$, 它与当前均值向量 μ_1, μ_2, μ_3 的距离分别为 0.369, 0.506, 0.166, 因此 x_1 将被划入簇 C_3 中. 类似的, 对数据集中的所有样本考察一遍后, 可得当前簇划分为

$$C_1 = \{x_5, x_6, x_7, x_8, x_9, x_{10}, x_{13}, x_{14}, x_{15}, x_{17}, x_{18}, x_{19}, x_{20}, x_{23}\};$$

$$C_2 = \{x_{11}, x_{12}, x_{16}\};$$

$$C_3 = \{x_1, x_2, x_3, x_4, x_{21}, x_{22}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}\}.$$

于是, 可从 C_1, C_2, C_3 分别求出新的均值向量

$$\mu'_1 = (0.473; 0.214), \mu'_2 = (0.394; 0.066), \mu'_3 = (0.623; 0.388).$$

更新当前均值向量后, 不断重复上述过程, 如图 9.3 所示, 第五轮迭代产生的结果与第四轮迭代相同, 于是算法停止, 得到最终的簇划分.

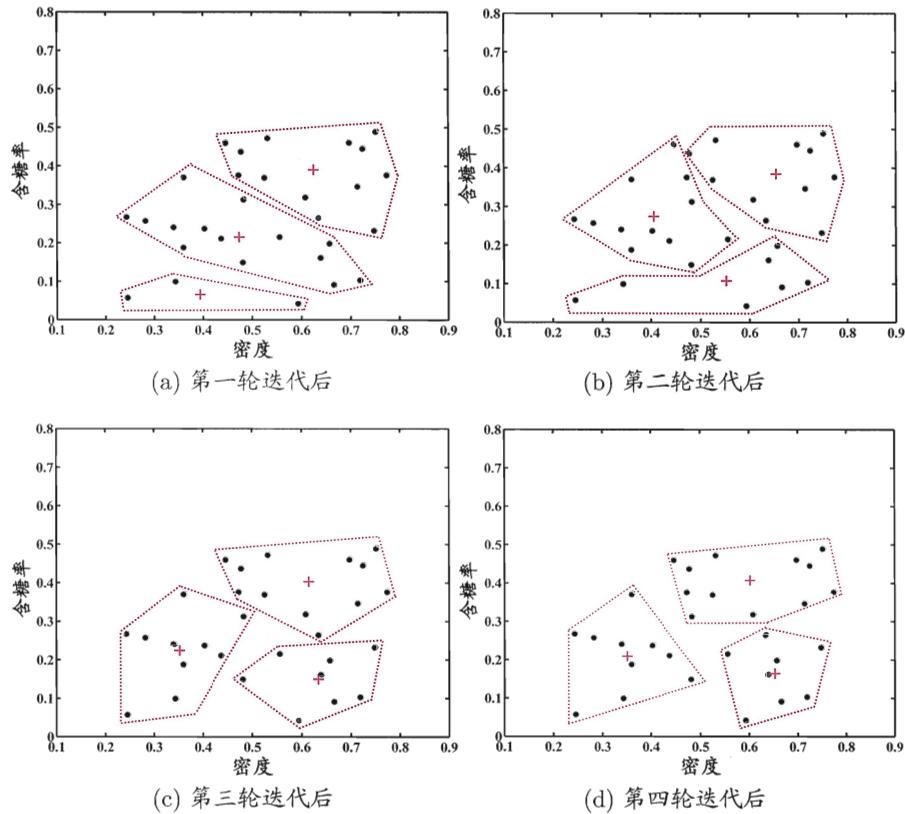


图 9.3 西瓜数据集 4.0 上 k 均值算法 ($k = 3$) 在各轮迭代后的结果. 样本点与均值向量分别用“●”与“+”表示, 红色虚线显示出簇划分.

9.4.2 学习向量量化

与 k 均值算法类似, “学习向量量化” (Learning Vector Quantization, 简称 LVQ)也是试图找到一组原型向量来刻画聚类结构, 但与一般聚类算法不同的是, LVQ 假设数据样本带有类别标记, 学习过程利用样本的这些监督信息来辅助聚类.

给定样本集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$, 每个样本 \mathbf{x}_j 是由 n 个属性描述的特征向量 $(x_{j1}; x_{j2}; \dots; x_{jn})$, $y_j \in \mathcal{Y}$ 是样本 \mathbf{x}_j 的类别标记. LVQ 的目标是学得一组 n 维原型向量 $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_q\}$, 每个原型向量代表一个聚类簇, 簇标记 $t_i \in \mathcal{Y}$.

LVQ 算法描述如图 9.4 所示. 算法第 1 行先对原型向量进行初始化, 例如对第 q 个簇可从类别标记为 t_q 的样本中随机选取一个作为原型向量. 算法第

可看作通过聚类来形成类别“子类”结构, 每个子类对应一个聚类簇.

x_j 与 p_{i^*} 的类别相同.
 x_j 与 p_{i^*} 的类别不同.
 如达到最大迭代轮数.

输入: 样本集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$;
 原型向量个数 q , 各原型向量预设的类别标记 $\{t_1, t_2, \dots, t_q\}$;
 学习率 $\eta \in (0, 1)$.

过程:

- 1: 初始化一组原型向量 $\{p_1, p_2, \dots, p_q\}$
- 2: **repeat**
- 3: 从样本集 D 随机选取样本 (x_j, y_j) ;
- 4: 计算样本 x_j 与 p_i ($1 \leq i \leq q$) 的距离: $d_{ji} = \|x_j - p_i\|_2$;
- 5: 找出与 x_j 距离最近的原型向量 p_{i^*} , $i^* = \arg \min_{i \in \{1, 2, \dots, q\}} d_{ji}$;
- 6: **if** $y_j = t_{i^*}$ **then**
- 7: $p' = p_{i^*} + \eta \cdot (x_j - p_{i^*})$
- 8: **else**
- 9: $p' = p_{i^*} - \eta \cdot (x_j - p_{i^*})$
- 10: **end if**
- 11: 将原型向量 p_{i^*} 更新为 p'
- 12: **until** 满足停止条件

输出: 原型向量 $\{p_1, p_2, \dots, p_q\}$

图 9.4 学习向量量化算法

第 5 行是竞争学习的“胜者为王”策略. SOM 是基于无标记样本的聚类算法, 而 LVQ 可看作 SOM 基于监督信息的扩展. 关于竞争学习与 SOM, 参见 5.5.2 和 5.5.3 节.

2~12 行对原型向量进行迭代优化. 在每一轮迭代中, 算法随机选取一个有标记训练样本, 找出与其距离最近的原型向量, 并根据两者的类别标记是否一致来对原型向量进行相应的更新. 在第 12 行中, 若算法的停止条件已满足(例如已达到最大迭代轮数, 或原型向量更新很小甚至不再更新), 则将当前原型向量作为最终结果返回.

显然, LVQ 的关键是第 6~10 行, 即如何更新原型向量. 直观上看, 对样本 x_j , 若最近的原型向量 p_{i^*} 与 x_j 的类别标记相同, 则令 p_{i^*} 向 x_j 的方向靠拢, 如第 7 行所示, 此时新原型向量为

$$p' = p_{i^*} + \eta \cdot (x_j - p_{i^*}), \quad (9.25)$$

p' 与 x_j 之间的距离为

$$\begin{aligned} \|p' - x_j\|_2 &= \|(p_{i^*} + \eta \cdot (x_j - p_{i^*})) - x_j\|_2 \\ &= (1 - \eta) \cdot \|p_{i^*} - x_j\|_2. \end{aligned} \quad (9.26)$$

令学习率 $\eta \in (0, 1)$, 则原型向量 p_{i^*} 在更新为 p' 之后将更接近 x_j .

类似的, 若 p_{i^*} 与 x_j 的类别标记不同, 则更新后的原型向量与 x_j 之间的距离将增大为 $(1 + \eta) \cdot \|p_{i^*} - x_j\|_2$, 从而更远离 x_j .

在学得一组原型向量 $\{p_1, p_2, \dots, p_q\}$ 后, 即可实现对样本空间 \mathcal{X} 的簇划

若将 R_i 中样本全用原型向量 \mathbf{p}_i 表示, 则可实现数据的“有损压缩”(lossy compression), 这称为“向量量化”(vector quantization); LVQ 由此而得名.

即希望为“好瓜=是”
找到3个簇, “好瓜=否”
找到2个簇.

分. 对任意样本 \mathbf{x} , 它将被划入与其距离最近的原型向量所代表的簇中; 换言之, 每个原型向量 \mathbf{p}_i 定义了与之相关的一个区域 R_i , 该区域中每个样本与 \mathbf{p}_i 的距离不大于它与其他原型向量 $\mathbf{p}_{i'} (i' \neq i)$ 的距离, 即

$$R_i = \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{p}_i\|_2 \leq \|\mathbf{x} - \mathbf{p}_{i'}\|_2, i' \neq i\}. \quad (9.27)$$

由此形成了对样本空间 \mathcal{X} 的簇划分 $\{R_1, R_2, \dots, R_q\}$, 该划分通常称为“Voronoi剖分”(Voronoi tessellation).

下面我们以表 9.1 的西瓜数据集 4.0 为例来演示 LVQ 的学习过程. 令 9-21 号样本的类别标记为 c_2 , 其他样本的类别标记为 c_1 . 假定 $q = 5$, 即学习目标是找到 5 个原型向量 $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5$, 并假定其对应的类别标记分别为 c_1, c_2, c_2, c_1, c_1 .

算法开始时, 根据样本的类别标记和簇的预设类别标记对原型向量进行随机初始化, 假定初始化为样本 $\mathbf{x}_5, \mathbf{x}_{12}, \mathbf{x}_{18}, \mathbf{x}_{23}, \mathbf{x}_{29}$. 在第一轮迭代中, 假定随机选取的样本为 \mathbf{x}_1 , 该样本与当前原型向量 $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5$ 的距离分别为 0.283, 0.506, 0.434, 0.260, 0.032. 由于 \mathbf{p}_5 与 \mathbf{x}_1 距离最近且两者具有相同的类别标记 c_2 , 假定学习率 $\eta = 0.1$, 则 LVQ 更新 \mathbf{p}_5 得到新原型向量

$$\begin{aligned} \mathbf{p}' &= \mathbf{p}_5 + \eta \cdot (\mathbf{x}_1 - \mathbf{p}_5) \\ &= (0.725; 0.445) + 0.1 \cdot ((0.697; 0.460) - (0.725; 0.445)) \\ &= (0.722; 0.442). \end{aligned}$$

将 \mathbf{p}_5 更新为 \mathbf{p}' 后, 不断重复上述过程, 不同轮数之后的聚类结果如图 9.5 所示.

9.4.3 高斯混合聚类

与 k 均值、LVQ 用原型向量来刻画聚类结构不同, 高斯混合(Mixture-of-Gaussian)聚类采用概率模型来表达聚类原型.

我们先简单回顾一下(多元)高斯分布的定义. 对 n 维样本空间 \mathcal{X} 中的随机向量 \mathbf{x} , 若 \mathbf{x} 服从高斯分布, 其概率密度函数为

Σ : 对称正定矩阵;
 $|\Sigma|$: Σ 的行列式;
 Σ^{-1} : Σ 的逆矩阵.

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})}, \quad (9.28)$$

其中 $\boldsymbol{\mu}$ 是 n 维均值向量, Σ 是 $n \times n$ 的协方差矩阵. 由式(9.28)可看出, 高斯分布完全由均值向量 $\boldsymbol{\mu}$ 和协方差矩阵 Σ 这两个参数确定. 为了明确显示高斯分

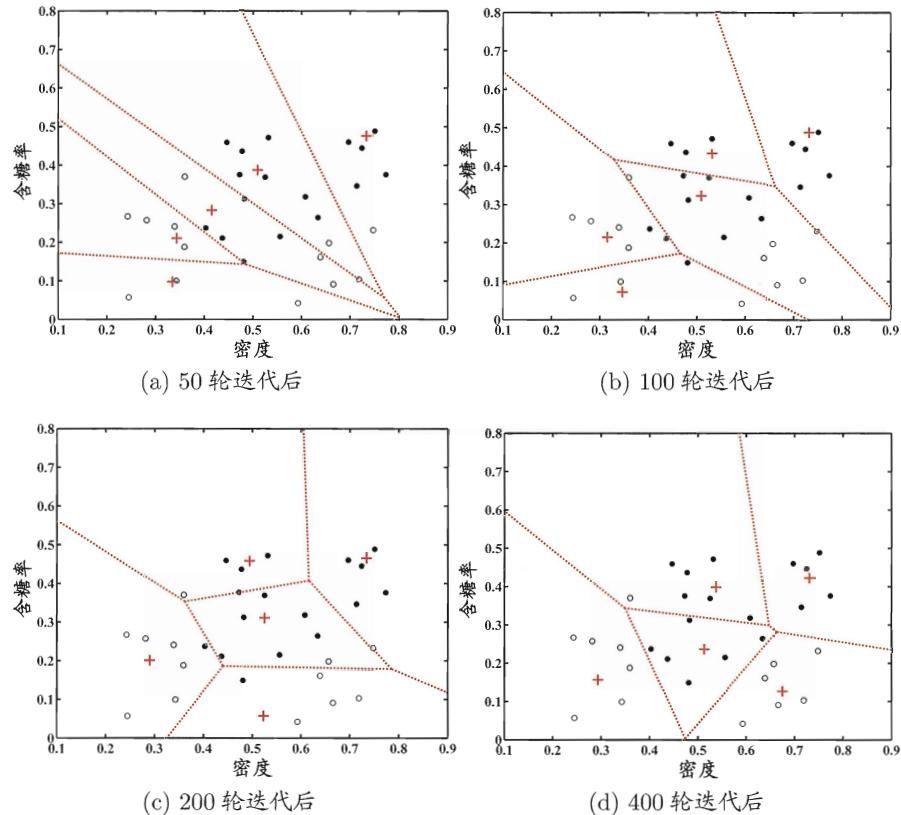


图 9.5 西瓜数据集 4.0 上 LVQ 算法($q = 5$)在不同轮数迭代后的聚类结果。 c_1, c_2 类样本点与原型向量分别用“●”、“○”与“+”表示，红色虚线显示出聚类形成的 Voronoi 割分。

布与相应参数的依赖关系, 将概率密度函数记为 $p(x | \mu, \Sigma)$.

我们可定义高斯混合分布

$p_M(\cdot)$ 也是概率密度函数, $\int p_M(x)dx = 1$.

$$p_{\mathcal{M}}(\mathbf{x}) = \sum_{i=1}^k \alpha_i \cdot p(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) , \quad (9.29)$$

该分布共由 k 个混合成分组成, 每个混合成分对应一个高斯分布. 其中 μ_i 与 Σ_i 是第 i 个高斯混合成分的参数, 而 $\alpha_i > 0$ 为相应的“混合系数”(mixture coefficient), $\sum_{i=1}^k \alpha_i = 1$.

假设样本的生成过程由高斯混合分布给出：首先，根据 $\alpha_1, \alpha_2, \dots, \alpha_k$ 定义的先验分布选择高斯混合成分，其中 α_i 为选择第 i 个混合成分的概率；然后，根据被选择的混合成分的概率密度函数进行采样，从而生成相应的样本。

若训练集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ 由上述过程生成, 令随机变量 $z_j \in \{1, 2, \dots, k\}$ 表示生成样本 \mathbf{x}_j 的高斯混合成分, 其取值未知。显然, z_j 的先验概率 $P(z_j = i)$ 对应于 α_i ($i = 1, 2, \dots, k$)。根据贝叶斯定理, z_j 的后验分布对应于

$$\begin{aligned} p_{\mathcal{M}}(z_j = i | \mathbf{x}_j) &= \frac{P(z_j = i) \cdot p_{\mathcal{M}}(\mathbf{x}_j | z_j = i)}{p_{\mathcal{M}}(\mathbf{x}_j)} \\ &= \frac{\alpha_i \cdot p(\mathbf{x}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{l=1}^k \alpha_l \cdot p(\mathbf{x}_j | \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)} . \end{aligned} \quad (9.30)$$

换言之, $p_{\mathcal{M}}(z_j = i | \mathbf{x}_j)$ 给出了样本 \mathbf{x}_j 由第 i 个高斯混合成分生成的后验概率。为方便叙述, 将其简记为 γ_{ji} ($i = 1, 2, \dots, k$)。

当高斯混合分布(9.29)已知时, 高斯混合聚类将把样本集 D 划分为 k 个簇 $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$, 每个样本 \mathbf{x}_j 的簇标记 λ_j 如下确定:

$$\lambda_j = \arg \max_{i \in \{1, 2, \dots, k\}} \gamma_{ji} . \quad (9.31)$$

因此, 从原型聚类的角度来看, 高斯混合聚类是采用概率模型(高斯分布)对原型进行刻画, 簇划分则由原型对应后验概率确定。

那么, 对于式(9.29), 模型参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leq i \leq k\}$ 如何求解呢? 显然, 极大似然估计参见 7.2 节。给定样本集 D , 可采用极大似然估计, 即最大化(对数)似然

$$\begin{aligned} LL(D) &= \ln \left(\prod_{j=1}^m p_{\mathcal{M}}(\mathbf{x}_j) \right) \\ &= \sum_{j=1}^m \ln \left(\sum_{i=1}^k \alpha_i \cdot p(\mathbf{x}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \right) , \end{aligned} \quad (9.32)$$

EM 算法参见 7.6 节。

常采用 EM 算法进行迭代优化求解。下面我们做一个简单的推导。

若参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leq i \leq k\}$ 能使式(9.32)最大化, 则由 $\frac{\partial LL(D)}{\partial \boldsymbol{\mu}_i} = 0$ 有

$$\sum_{j=1}^m \frac{\alpha_i \cdot p(\mathbf{x}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{l=1}^k \alpha_l \cdot p(\mathbf{x}_j | \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)} (\mathbf{x}_j - \boldsymbol{\mu}_i) = 0 , \quad (9.33)$$

由式(9.30)以及 $\gamma_{ji} = p_{\mathcal{M}}(z_j = i | \mathbf{x}_j)$, 有

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^m \gamma_{ji} \mathbf{x}_j}{\sum_{j=1}^m \gamma_{ji}}, \quad (9.34)$$

即各混合成分的均值可通过样本加权平均来估计, 样本权重是每个样本属于该成分的后验概率. 类似的, 由 $\frac{\partial LL(D)}{\partial \Sigma_i} = 0$ 可得

$$\boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^m \gamma_{ji} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^m \gamma_{ji}}. \quad (9.35)$$

对于混合系数 α_i , 除了要最大化 $LL(D)$, 还需满足 $\alpha_i \geq 0$, $\sum_{i=1}^k \alpha_i = 1$. 考虑 $LL(D)$ 的拉格朗日形式

$$LL(D) + \lambda \left(\sum_{i=1}^k \alpha_i - 1 \right), \quad (9.36)$$

其中 λ 为拉格朗日乘子. 由式(9.36)对 α_i 的导数为 0, 有

$$\sum_{j=1}^m \frac{p(\mathbf{x}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{l=1}^k \alpha_l \cdot p(\mathbf{x}_j | \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)} + \lambda = 0, \quad (9.37)$$

两边同乘以 α_i , 对所有样本求和可知 $\lambda = -m$, 有

$$\alpha_i = \frac{1}{m} \sum_{j=1}^m \gamma_{ji}, \quad (9.38)$$

即每个高斯成分的混合系数由样本属于该成分的平均后验概率确定.

由上述推导即可获得高斯混合模型的 EM 算法: 在每步迭代中, 先根据当前参数来计算每个样本属于每个高斯成分的后验概率 γ_{ji} (E步), 再根据式(9.34)、(9.35)和(9.38)更新模型参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leq i \leq k\}$ (M步).

高斯混合聚类算法描述如图 9.6 所示. 算法第 1 行对高斯混合分布的模型参数进行初始化. 然后, 在第 2–12 行基于 EM 算法对模型参数进行迭代更新. 若 EM 算法的停止条件满足(例如已达到最大迭代轮数, 或似然函数 $LL(D)$ 增

输入: 样本集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$;
高斯混合成分个数 k .

过程:

- 1: 初始化高斯混合分布的模型参数 $\{(\alpha_i, \mu_i, \Sigma_i) \mid 1 \leq i \leq k\}$
- 2: repeat
- 3: for $j = 1, 2, \dots, m$ do
- 4: 根据式(9.30)计算 \mathbf{x}_j 由各混合成分生成的后验概率, 即
 $\gamma_{ji} = p_M(z_j = i \mid \mathbf{x}_j) (1 \leq i \leq k)$
- 5: end for
- 6: for $i = 1, 2, \dots, k$ do
- 7: 计算新均值向量: $\mu'_i = \frac{\sum_{j=1}^m \gamma_{ji} \mathbf{x}_j}{\sum_{j=1}^m \gamma_{ji}}$;
- 8: 计算新协方差矩阵: $\Sigma'_i = \frac{\sum_{j=1}^m \gamma_{ji} (\mathbf{x}_j - \mu'_i)(\mathbf{x}_j - \mu'_i)^T}{\sum_{j=1}^m \gamma_{ji}}$;
- 9: 计算新混合系数: $\alpha'_i = \frac{\sum_{j=1}^m \gamma_{ji}}{m}$;
- 10: end for
- 11: 将模型参数 $\{(\alpha_i, \mu_i, \Sigma_i) \mid 1 \leq i \leq k\}$ 更新为 $\{(\alpha'_i, \mu'_i, \Sigma'_i) \mid 1 \leq i \leq k\}$
- 12: until 满足停止条件
- 13: $C_i = \emptyset (1 \leq i \leq k)$
- 14: for $j = 1, 2, \dots, m$ do
- 15: 根据式(9.31)确定 \mathbf{x}_j 的簇标记 λ_j ;
- 16: 将 \mathbf{x}_j 划入相应的簇: $C_{\lambda_j} = C_{\lambda_j} \cup \{\mathbf{x}_j\}$
- 17: end for

输出: 簇划分 $C = \{C_1, C_2, \dots, C_k\}$

图 9.6 高斯混合聚类算法

长很少甚至不再增长), 则在第 14–17 行根据高斯混合分布确定簇划分, 在第 18 行返回最终结果.

以表 9.1 的西瓜数据集 4.0 为例, 令高斯混合成分的个数 $k = 3$. 算法开始时, 假定将高斯混合分布的模型参数初始化为: $\alpha_1 = \alpha_2 = \alpha_3 = \frac{1}{3}$; $\mu_1 = \mathbf{x}_6$, $\mu_2 = \mathbf{x}_{22}$, $\mu_3 = \mathbf{x}_{27}$; $\Sigma_1 = \Sigma_2 = \Sigma_3 = \begin{pmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{pmatrix}$.

在第一轮迭代中, 先计算样本由各混合成分生成的后验概率. 以 \mathbf{x}_1 为例, 由式(9.30)算出后验概率 $\gamma_{11} = 0.219$, $\gamma_{12} = 0.404$, $\gamma_{13} = 0.377$. 所有样本的后验概率算完后, 得到如下新的模型参数:

$$\alpha'_1 = 0.361, \alpha'_2 = 0.323, \alpha'_3 = 0.316$$

$$\mu'_1 = (0.491; 0.251), \mu'_2 = (0.571; 0.281), \mu'_3 = (0.534; 0.295)$$

$$\Sigma'_1 = \begin{pmatrix} 0.025 & 0.004 \\ 0.004 & 0.016 \end{pmatrix}, \Sigma'_2 = \begin{pmatrix} 0.023 & 0.004 \\ 0.004 & 0.017 \end{pmatrix}, \Sigma'_3 = \begin{pmatrix} 0.024 & 0.005 \\ 0.005 & 0.016 \end{pmatrix}$$

模型参数更新后, 不断重复上述过程, 不同轮数之后的聚类结果如图 9.7 所示.

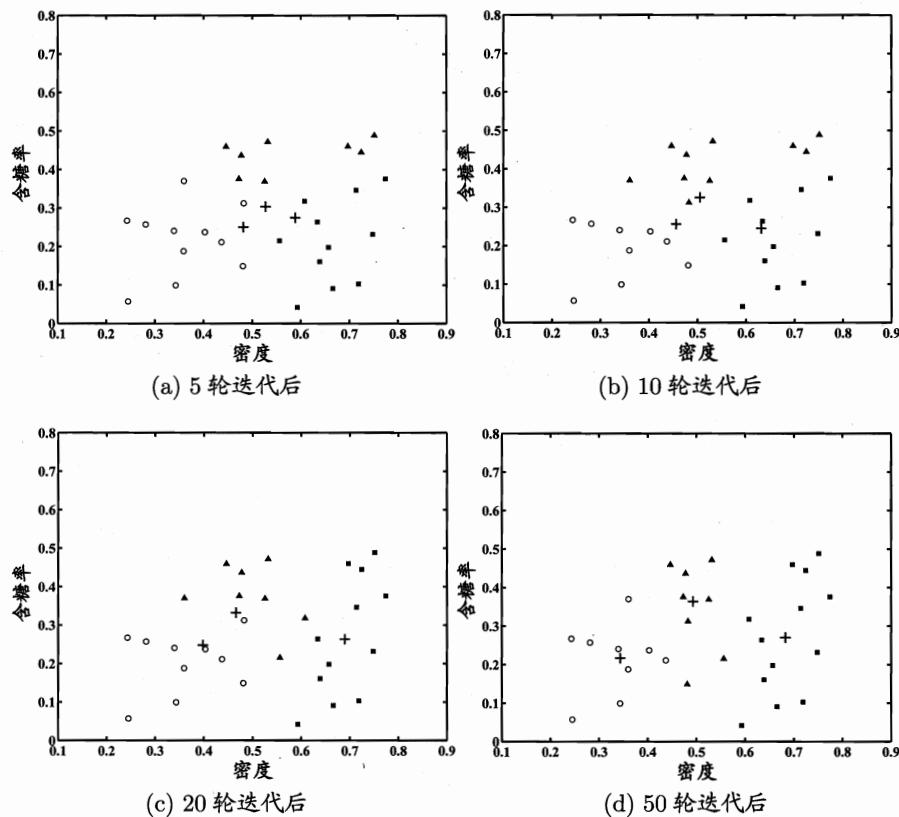


图 9.7 高斯混合聚类($k = 3$)在不同轮数迭代后的聚类结果. 其中样本簇 C_1 , C_2 与 C_3 中的样本点分别用“○”, “■”与“▲”表示, 各高斯混合成分的均值向量用“+”表示.

9.5 密度聚类

密度聚类亦称“基于密度的聚类”(density-based clustering), 此类算法假设聚类结构能通过样本分布的紧密程度确定. 通常情形下, 密度聚类算法从样本密度的角度来考察样本之间的可连接性, 并基于可连接样本不断扩展聚类簇以获得最终的聚类结果.

全称“Density-Based Spatial Clustering of Applications with Noise”.

DBSCAN 是一种著名的密度聚类算法, 它基于一组“邻域”(neighborhood)参数 $(\epsilon, MinPts)$ 来刻画样本分布的紧密程度. 给定数据集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, 定义下面这几个概念:

在本章后续内容中, 距离函数 $\text{dist}(\cdot, \cdot)$ 在默认情况下设为欧氏距离.

- ϵ -邻域: 对 $\mathbf{x}_j \in D$, 其 ϵ -邻域包含样本集 D 中与 \mathbf{x}_j 的距离不大于 ϵ 的样本, 即 $N_\epsilon(\mathbf{x}_j) = \{\mathbf{x}_i \in D \mid \text{dist}(\mathbf{x}_i, \mathbf{x}_j) \leq \epsilon\}$;

- 核心对象(core object): 若 x_j 的 ϵ -邻域至少包含 $MinPts$ 个样本, 即 $|N_\epsilon(x_j)| \geq MinPts$, 则 x_j 是一个核心对象;
- 密度直达(directly density-reachable): 若 x_j 位于 x_i 的 ϵ -邻域中, 且 x_i 是核心对象, 则称 x_j 由 x_i 密度直达;
- 密度可达(density-reachable): 对 x_i 与 x_j , 若存在样本序列 p_1, p_2, \dots, p_n , 其中 $p_1 = x_i$, $p_n = x_j$ 且 p_{i+1} 由 p_i 密度直达, 则称 x_j 由 x_i 密度可达;
- 密度相连(density-connected): 对 x_i 与 x_j , 若存在 x_k 使得 x_i 与 x_j 均由 x_k 密度可达, 则称 x_i 与 x_j 密度相连.

图 9.8 给出了上述概念的直观显示.

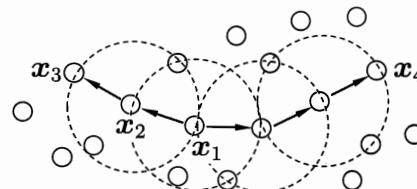


图 9.8 DBSCAN 定义的基本概念($MinPts = 3$): 虚线显示出 ϵ -邻域, x_1 是核心对象, x_2 由 x_1 密度直达, x_3 由 x_1 密度可达, x_3 与 x_4 密度相连.

D 中不属于任何簇的样本被认为是噪声(noise)或异常(anomaly)样本.

基于这些概念, DBSCAN 将“簇”定义为: 由密度可达关系导出的最大的密度相连样本集合. 形式化地说, 给定邻域参数 $(\epsilon, MinPts)$, 簇 $C \subseteq D$ 是满足以下性质的非空样本子集:

$$\text{连接性(connectivity): } x_i \in C, x_j \in C \Rightarrow x_i \text{ 与 } x_j \text{ 密度相连} \quad (9.39)$$

$$\text{最大性(maximality): } x_i \in C, x_j \text{ 由 } x_i \text{ 密度可达} \Rightarrow x_j \in C \quad (9.40)$$

那么, 如何从数据集 D 中找出满足以上性质的聚类簇呢? 实际上, 若 x 为核心对象, 由 x 密度可达的所有样本组成的集合记为 $X = \{x' \in D \mid x'$ 由 x 密度可达}, 则不难证明 X 即为满足连接性与最大性的簇.

于是, DBSCAN 算法先任选数据集中的一个核心对象为“种子”(seed), 再由此出发确定相应的聚类簇, 算法描述如图 9.9 所示. 在第 1~7 行中, 算法先根据给定的邻域参数 $(\epsilon, MinPts)$ 找出所有核心对象; 然后在第 10~24 行中, 以任一核心对象为出发点, 找出由其密度可达的样本生成聚类簇, 直到所有核心对象均被访问过为止.

输入: 样本集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$;
邻域参数 $(\epsilon, MinPts)$.

过程:

- 1: 初始化核心对象集合: $\Omega = \emptyset$
- 2: **for** $j = 1, 2, \dots, m$ **do**
- 3: 确定样本 \mathbf{x}_j 的 ϵ -邻域 $N_\epsilon(\mathbf{x}_j)$;
- 4: **if** $|N_\epsilon(\mathbf{x}_j)| \geq MinPts$ **then**
- 5: 将样本 \mathbf{x}_j 加入核心对象集合: $\Omega = \Omega \cup \{\mathbf{x}_j\}$
- 6: **end if**
- 7: **end for**
- 8: 初始化聚类簇数: $k = 0$
- 9: 初始化未访问样本集合: $\Gamma = D$
- 10: **while** $\Omega \neq \emptyset$ **do**
- 11: 记录当前未访问样本集合: $\Gamma_{old} = \Gamma$;
- 12: 随机选取一个核心对象 $\mathbf{o} \in \Omega$, 初始化队列 $Q = <\mathbf{o}>$;
- 13: $\Gamma = \Gamma \setminus \{\mathbf{o}\}$;
- 14: **while** $Q \neq \emptyset$ **do**
- 15: 取出队列 Q 中的首个样本 \mathbf{q} ;
- 16: **if** $|N_\epsilon(\mathbf{q})| \geq MinPts$ **then**
- 17: 令 $\Delta = N_\epsilon(\mathbf{q}) \cap \Gamma$;
- 18: 将 Δ 中的样本加入队列 Q ;
- 19: $\Gamma = \Gamma \setminus \Delta$;
- 20: **end if**
- 21: **end while**
- 22: $k = k + 1$, 生成聚类簇 $C_k = \Gamma_{old} \setminus \Gamma$;
- 23: $\Omega = \Omega \setminus C_k$
- 24: **end while**

输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$

图 9.9 DBSCAN 算法

以表 9.1 的西瓜数据集 4.0 为例, 假定邻域参数 $(\epsilon, MinPts)$ 设置为 $\epsilon = 0.11$, $MinPts = 5$. DBSCAN 算法先找出各样的 ϵ -邻域并确定核心对象集合: $\Omega = \{\mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_8, \mathbf{x}_9, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{24}, \mathbf{x}_{25}, \mathbf{x}_{28}, \mathbf{x}_{29}\}$. 然后, 从 Ω 中随机选取一个核心对象作为种子, 找出由它密度可达的所有样本, 这就构成了第一个聚类簇. 不失一般性, 假定核心对象 \mathbf{x}_8 被选中作为种子, 则 DBSCAN 生成的第一个聚类簇为

$$C_1 = \{\mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_{10}, \mathbf{x}_{12}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{20}, \mathbf{x}_{23}\}.$$

然后, DBSCAN 将 C_1 中包含的核心对象从 Ω 中去除: $\Omega = \Omega \setminus C_1 = \{\mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_9, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{24}, \mathbf{x}_{25}, \mathbf{x}_{28}, \mathbf{x}_{29}\}$. 再从更新后的集合 Ω 中随机选取一个核心对象作为种子来生成下一个聚类簇. 上述过程不断重复, 直至 Ω 为空. 图 9.10 显示出 DBSCAN 先后生成聚类簇的情况. C_1 之后生成的聚类簇为

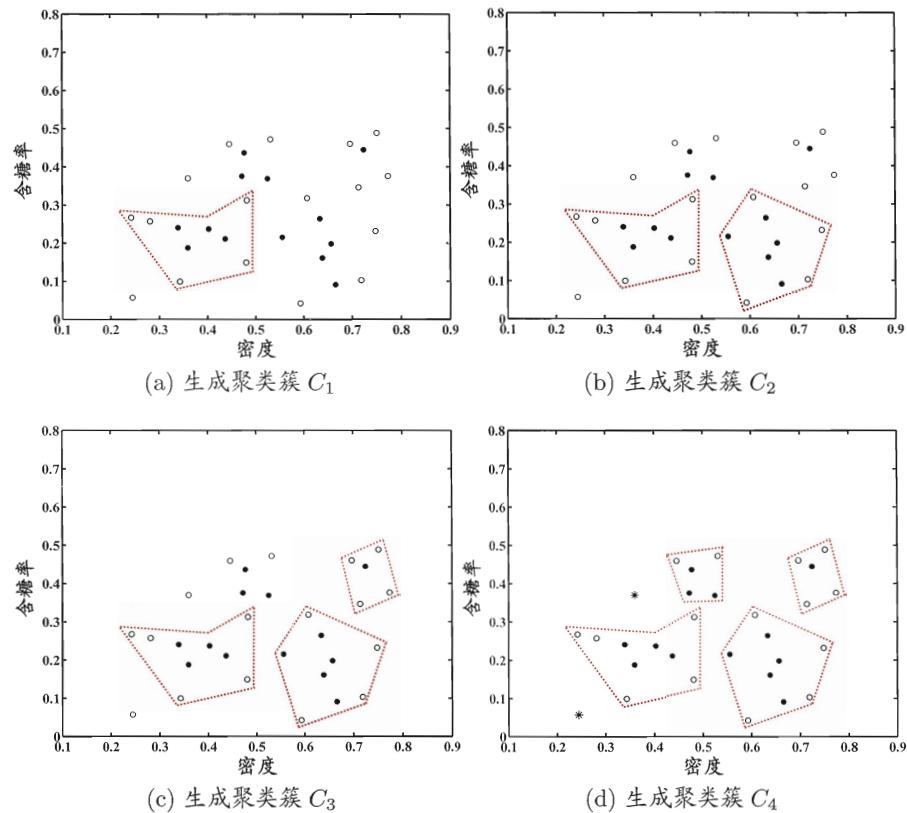


图 9.10 DBSCAN 算法($\epsilon = 0.11$, $MinPts = 5$)生成聚类簇的先后情况. 核心对象、非核心对象、噪声样本分别用“●”“○”“*”表示, 红色虚线显示出簇划分.

$$C_2 = \{x_3, x_4, x_5, x_9, x_{13}, x_{14}, x_{16}, x_{17}, x_{21}\} ;$$

$$C_3 = \{x_1, x_2, x_{22}, x_{26}, x_{29}\} ;$$

$$C_4 = \{x_{24}, x_{25}, x_{27}, x_{28}, x_{30}\} .$$

9.6 层次聚类

层次聚类(hierarchical clustering)试图在不同层次对数据集进行划分, 从而形成树形的聚类结构. 数据集的划分可采用“自底向上”的聚合策略, 也可采用“自顶向下”的分拆策略.

AGNES 是 AGglomerative NESting 的简写.

AGNES 是一种采用自底向上聚合策略的层次聚类算法. 它先将数据集中的每个样本看作一个初始聚类簇, 然后在算法运行的每一步中找出距离最近的

集合间的距离计算常采用豪斯多夫距离 (Hausdorff distance), 参见习题 9.2.

两个聚类簇进行合并, 该过程不断重复, 直至达到预设的聚类簇个数. 这里的关键是如何计算聚类簇之间的距离. 实际上, 每个簇是一个样本集合, 因此, 只需采用关于集合的某种距离即可. 例如, 给定聚类簇 C_i 与 C_j , 可通过下面的式子来计算距离:

$$\text{最小距离: } d_{\min}(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{z} \in C_j} \text{dist}(\mathbf{x}, \mathbf{z}), \quad (9.41)$$

$$\text{最大距离: } d_{\max}(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{z} \in C_j} \text{dist}(\mathbf{x}, \mathbf{z}), \quad (9.42)$$

$$\text{平均距离: } d_{\text{avg}}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{z} \in C_j} \text{dist}(\mathbf{x}, \mathbf{z}). \quad (9.43)$$

显然, 最小距离由两个簇的最近样本决定, 最大距离由两个簇的最远样本决定, 而平均距离则由两个簇的所有样本共同决定. 当聚类簇距离由 d_{\min} 、 d_{\max} 或

通常使用 d_{\min} , d_{\max}
或 d_{avg} .

初始化单样本聚类簇.

初始化聚类簇距离矩阵.

$i^* < j^*$.

输入: 样本集 $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$;
聚类簇距离度量函数 d ;
聚类簇数 k .

过程:

```

1: for  $j = 1, 2, \dots, m$  do
2:    $C_j = \{\mathbf{x}_j\}$ 
3: end for
4: for  $i = 1, 2, \dots, m$  do
5:   for  $j = 1, 2, \dots, m$  do
6:      $M(i, j) = d(C_i, C_j);$ 
7:      $M(j, i) = M(i, j)$ 
8:   end for
9: end for
10: 设置当前聚类簇个数:  $q = m$ 
11: while  $q > k$  do
12:   找出距离最近的两个聚类簇  $C_{i^*}$  和  $C_{j^*}$ ;
13:   合并  $C_{i^*}$  和  $C_{j^*}$ :  $C_{i^*} = C_{i^*} \cup C_{j^*};$ 
14:   for  $j = j^* + 1, j^* + 2, \dots, q$  do
15:     将聚类簇  $C_j$  重编号为  $C_{j-1}$ 
16:   end for
17:   删除距离矩阵  $M$  的第  $j^*$  行与第  $j^*$  列;
18:   for  $j = 1, 2, \dots, q - 1$  do
19:      $M(i^*, j) = d(C_{i^*}, C_j);$ 
20:      $M(j, i^*) = M(i^*, j)$ 
21:   end for
22:    $q = q - 1$ 
23: end while
```

输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$

图 9.11 AGNES 算法

d_{avg} 计算时, AGNES 算法被相应地称为“单链接”(single-linkage)、“全链接”(complete-linkage)或“均链接”(average-linkage)算法.

AGNES 算法描述如图 9.11 所示. 在第 1–9 行, 算法先对仅含一个样本的初始聚类簇和相应的距离矩阵进行初始化; 然后在第 11–23 行, AGNES 不断合并距离最近的聚类簇, 并对合并得到的聚类簇的距离矩阵进行更新; 上述过程不断重复, 直至达到预设的聚类簇数.

西瓜数据集 4.0 见 p.202
的表 9.1.

以西瓜数据集 4.0 为例, 令 AGNES 算法一直执行到所有样本出现在同一个簇中, 即 $k = 1$, 则可得到图 9.12 所示的“树状图”(dendrogram), 其中每层连接一组聚类簇.

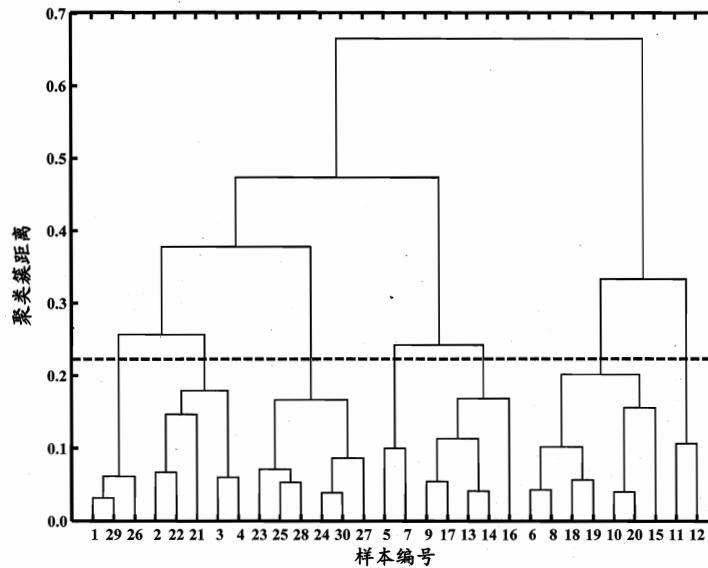


图 9.12 西瓜数据集 4.0 上 AGNES 算法生成的树状图(采用 d_{max}). 横轴对应于样本编号, 纵轴对应于聚类簇距离.

在树状图的特定层次上进行分割, 则可得到相应的簇划分结果. 例如, 以图 9.12 中所示虚线分割树状图, 将得到包含 7 个聚类簇的结果:

$$C_1 = \{x_1, x_{26}, x_{29}\}; C_2 = \{x_2, x_3, x_4, x_{21}, x_{22}\};$$

$$C_3 = \{x_{23}, x_{24}, x_{25}, x_{27}, x_{28}, x_{30}\}; C_4 = \{x_5, x_7\};$$

$$C_5 = \{x_9, x_{13}, x_{14}, x_{16}, x_{17}\}; C_6 = \{x_6, x_8, x_{10}, x_{15}, x_{18}, x_{19}, x_{20}\};$$

$$C_7 = \{x_{11}, x_{12}\}.$$

将分割层逐步提升，则可得到聚类簇逐渐减少的聚类结果。例如图 9.13 显示出了从图 9.12 中产生 7 至 4 个聚类簇的划分结果。

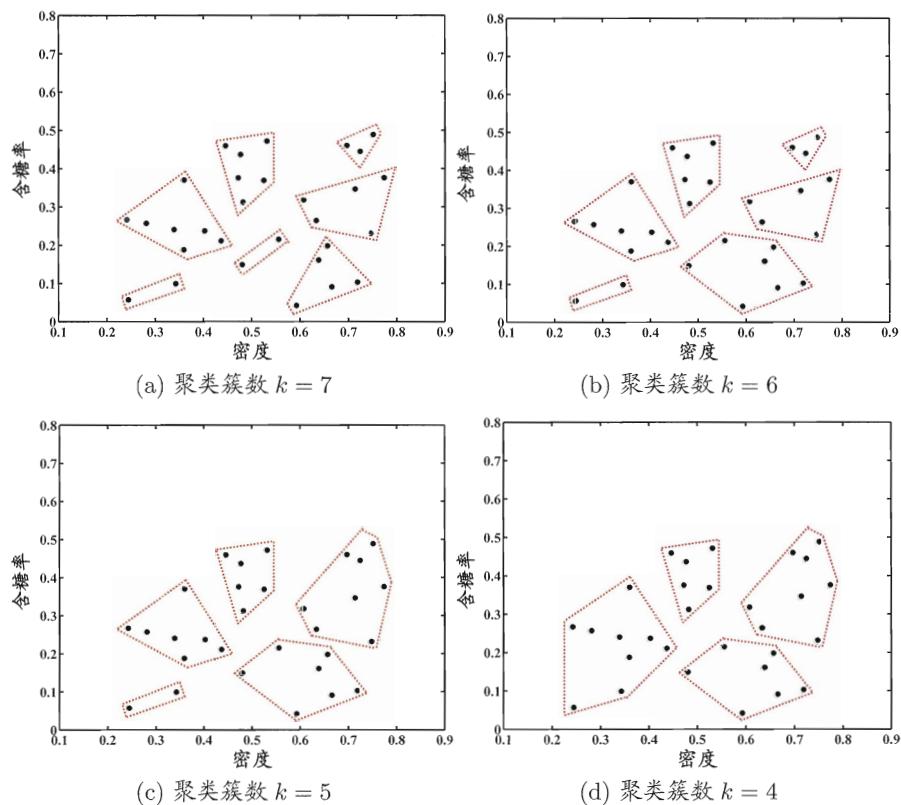


图 9.13 西瓜数据集 4.0 上 AGNES 算法(采用 d_{\max})在不同聚类簇数($k = 7, 6, 5, 4$)时的簇划分结果。样本点用“●”表示，红色虚线显示出簇划分。

9.7 阅读材料

例如同一堆水果，既能按大小，也能按颜色，甚至能按产地聚类。

聚类也许是机器学习中“新算法”出现最多、最快的领域。一个重要原因是聚类不存在客观标准；给定数据集，总能从某个角度找到以往算法未覆盖的某种标准从而设计出新算法 [Estivill-Castro, 2002]。相对于机器学习其他分支来说，聚类的知识还不够系统化，因此著名教科书 [Mitchell, 1997] 中甚至没有关于聚类的章节。但聚类技术本身在现实任务中非常重要，因此本章勉强采用了“列举式”的叙述方式，相较于其他各章给出了更多的算法描述。关于聚类

更多的内容, 可参阅这方面的专门书籍和综述文章如 [Jain and Dubes, 1988; Jain et al., 1999; Xu and Wunsch II, 2005; Jain, 2009] 等.

聚类性能度量除 9.2 节的内容外, 常见的还有 F 值、互信息 (mutual information)、平均廓宽 (average silhouette width) [Rousseeuw, 1987] 等, 可参阅 [Jain and Dubes, 1988; Halkidi et al., 2001; Maulik and Bandyopadhyay, 2002].

距离计算是很多学习任务的核心技术. 闵可夫斯基距离提供了距离计算的一般形式. 除闵可夫斯基距离之外, 内积距离、余弦距离等也很常用, 可参阅 [Deza and Deza, 2009]. MinkovDM 在 [Zhou and Yu, 2005] 中正式给出. 模式识别、图像检索等涉及复杂语义的应用中常会涉及非度量距离 [Jacobs et al., 2000; Tan et al., 2009]. 距离度量学习可直接嵌入到聚类学习过程中 [Xing et al., 2003].

距离度量学习参见 10.6 节.

k 均值算法可看作高斯混合聚类在混合成分方差相等、且每个样本仅指派给一个混合成分时的特例. 该算法在历史上曾被不同领域的学者多次重新发明, 如 Steinhaus 在 1956 年、Lloyd 在 1957 年、McQueen 在 1967 年等 [Jain and Dubes, 1988; Jain, 2009]. k 均值算法有大量变体, 如 k -medoids 算法 [Kaufman and Rousseeuw, 1987] 强制原型向量必为训练样本, k -modes 算法 [Huang, 1998] 可处理离散属性, Fuzzy C -means (简称 FCM) [Bezdek, 1981] 则是“软聚类”(soft clustering) 算法, 允许每个样本以不同程度同时属于多个原型. 需注意的是, k 均值类算法仅在凸形簇结构上效果较好. 最近研究表明, 若采用某种 Bregman 距离, 则可显著增强此类算法对更多类型簇结构的适用性 [Banerjee et al., 2005]. 引入核技巧则可得到核 k 均值 (kernel k -means) 算法 [Schölkopf et al., 1998], 这与谱聚类 (spectral clustering) [von Luxburg, 2007] 有密切联系 [Dhillon et al., 2004], 后者可看作在拉普拉斯特征映射 (Laplacian Eigenmap) 降维后执行 k 均值聚类. 聚类簇数 k 通常需由用户提供, 有一些启发式用于自动确定 k [Pelleg and Moore, 2000; Tibshirani et al., 2001], 但常用的仍是基于不同 k 值多次运行后选取最佳结果.

LVQ 算法在每轮迭代中仅更新与当前样本距离最近的原型向量. 同时更新多个原型向量能显著提高收敛速度, 相应的改进算法有 LVQ2、LVQ3 等 [Kohonen, 2001]. [McLachlan and Peel, 2000] 详细介绍了高斯混合聚类, 算法中 EM 迭代优化的推导过程可参阅 [Bilmes, 1998; Jain and Dubes, 1988].

采用不同方式表征样本分布的紧密程度, 可设计出不同的密度聚类算法, 除 DBSCAN [Ester et al., 1996] 外, 较常用的还有 OPTICS [Ankerst et al.,

凸形簇结构即形似“椭球”的簇结构.

Bregman 距离亦称 Bregman divergence, 是一类不满足对称性和直递性的距离.

降维参见第 10 章.

1999]、DENCLUE [Hinneburg and Keim, 1998] 等。AGNES [Kaufman and Rousseeuw, 1990] 采用了自底向上的聚合策略来产生层次聚类结构, 与之相反, DIANA [Kaufman and Rousseeuw, 1990] 则是采用自顶向下的分拆策略。AGNES 和 DIANA 都不能对已合并或已分拆的聚类簇进行回溯调整, 常用的层次聚类算法如 BIRCH [Zhang et al., 1996]、ROCK [Guha et al., 1999] 等对此进行了改进。

聚类集成 (clustering ensemble) 通过对多个聚类学习器进行集成, 能有效降低聚类假设与真实聚类结构不符、聚类过程中的随机性等因素带来的不利影响, 可参阅 [Zhou, 2012] 第 7 章。

亦称 outlier detection.

异常检测 (anomaly detection) [Hodge and Austin, 2004; Chandola et al., 2009] 常借助聚类或距离计算进行, 如将远离所有簇中心的样本作为异常点, 或将密度极低处的样本作为异常点。最近有研究提出基于“隔离性” (isolation) 可快速检测出异常点 [Liu et al., 2012]。

习题

- 9.1** 试证明: $p \geq 1$ 时, 闵可夫斯基距离满足距离度量的四条基本性质; $0 < p < 1$ 时, 闵可夫斯基距离不满足直递性, 但满足非负性、同一性、对称性; p 趋向无穷大时, 闵可夫斯基距离等于对应分量的最大绝对距离, 即

$$\lim_{p \rightarrow +\infty} \left(\sum_{u=1}^n |x_{iu} - x_{ju}|^p \right)^{\frac{1}{p}} = \max_u |x_{iu} - x_{ju}|.$$

- 9.2** 同一样本空间中的集合 X 与 Z 之间的距离可通过“豪斯多夫距离”(Hausdorff distance)计算:

$$\text{dist}_H(X, Z) = \max (\text{dist}_h(X, Z), \text{dist}_h(Z, X)) , \quad (9.44)$$

其中

$$\text{dist}_h(X, Z) = \max_{x \in X} \min_{z \in Z} \|x - z\|_2 . \quad (9.45)$$

试证明: 豪斯多夫距离满足距离度量的四条基本性质.

- 9.3** 试析 k 均值算法能否找到最小化式(9.24)的最优解.
- 9.4** 试编程实现 k 均值算法, 设置三组不同的 k 值、三组不同初始中心点, 在西瓜数据集 4.0 上进行实验比较, 并讨论什么样的初始中心有利于取得好结果.
- 9.5** 基于 DBSCAN 的概念定义, 若 x 为核对象, 由 x 密度可达的所有样本构成的集合为 X . 试证明: X 满足连接性(9.39)与最大性(9.40).
- 9.6** 试析 AGNES 算法使用最小距离和最大距离的区别.
- 9.7** 聚类结果中若每个簇都有一个凸包(包含簇样本的凸多面体), 且这些凸包不相交, 则称为凸聚类. 试析本章介绍的哪些聚类算法只能产生凸聚类, 哪些能产生非凸聚类.
- 9.8** 试设计一个聚类性能度量指标, 并与 9.2 节中的指标比较.
- 9.9*** 试设计一个能用于混合属性的非度量距离.
- 9.10*** 试设计一个能自动确定聚类数的改进 k 均值算法, 编程实现并在西瓜数据集 4.0 上运行.

西瓜数据集 4.0 见 p.202
表 9.1.

即凸形簇结构.

参考文献

- Aloise, D., A. Deshpande, P. Hansen, and P. Popat. (2009). "NP-hardness of Euclidean sum-of-squares clustering." *Machine Learning*, 75(2):245–248.
- Ankerst, M., M. Breunig, H.-P. Kriegel, and J. Sander. (1999). "OPTICS: Ordering points to identify the clustering structure." In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 49–60, Philadelphia, PA.
- Banerjee, A., S. Merugu, I. Dhillon, and J. Ghosh. (2005). "Clustering with Bregman divergences." *Journal of Machine Learning Research*, 6:1705–1749.
- Bezdek, J. C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, NY.
- Bilmes, J. A. (1998). "A gentle tutorial of the EM algorithm and its applications to parameter estimation for Gaussian mixture and hidden Markov models." Technical Report TR-97-021, Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA.
- Chandola, V., A. Banerjee, and V. Kumar. (2009). "Anomaly detection: A survey." *ACM Computing Surveys*, 41(3):Article 15.
- Deza, M. and E. Deza. (2009). *Encyclopedia of Distances*. Springer, Berlin.
- Dhillon, I. S., Y. Guan, and B. Kulis. (2004). "Kernel k -means: Spectral clustering and normalized cuts." In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 551–556, Seattle, WA.
- Ester, M., H. P. Kriegel, J. Sander, and X. Xu. (1996). "A density-based algorithm for discovering clusters in large spatial databases." In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, 226–231, Portland, OR.
- Estivill-Castro, V. (2002). "Why so many clustering algorithms - a position paper." *SIGKDD Explorations*, 1(4):65–75.
- Guha, S., R. Rastogi, and K. Shim. (1999). "ROCK: A robust clustering algorithm for categorical attributes." In *Proceedings of the 15th International Conference on Data Engineering (ICDE)*, 512–521, Sydney, Australia.
- Halkidi, M., Y. Batistakis, and M. Vazirgiannis. (2001). "On clustering valida-

- tion techniques.” *Journal of Intelligent Information Systems*, 27(2-3):107–145.
- Hinneburg, A. and D. A. Keim. (1998). “An efficient approach to clustering in large multimedia databases with noise.” In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD)*, 58–65, New York, NY.
- Hodge, V. J. and J. Austin. (2004). “A survey of outlier detection methodologies.” *Artificial Intelligence Review*, 22(2):85–126.
- Huang, Z. (1998). “Extensions to the k -means algorithm for clustering large data sets with categorical values.” *Data Mining and Knowledge Discovery*, 2(3):283–304.
- Jacobs, D. W., D. Weinshall, and Y. Gdalyahu. (2000). “Classification with non-metric distances: Image retrieval and class representation.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(22):583–600.
- Jain, A. K. (2009). “Data clustering: 50 years beyond k -means.” *Pattern Recognition Letters*, 31(8):651–666.
- Jain, A. K. and R. C. Dubes. (1988). *Algorithms for Clustering Data*. Prentice Hall, Upper Saddle River, NJ.
- Jain, A. K., M. N. Murty, and P. J. Flynn. (1999). “Data clustering: A review.” *ACM Computing Surveys*, 3(31):264–323.
- Kaufman, L. and P. J. Rousseeuw. (1987). “Clustering by means of medoids.” In *Statistical Data Analysis Based on the L_1 -Norm and Related Methods* (Y. Dodge, ed.), 405–416, Elsevier, Amsterdam, The Netherlands.
- Kaufman, L. and P. J. Rousseeuw. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, New York, NY.
- Kohonen, T. (2001). *Self-Organizing Maps*, 3rd edition. Springer, Berlin.
- Liu, F. T., K. M. Ting, and Z.-H. Zhou. (2012). “Isolation-based anomaly detection.” *ACM Transactions on Knowledge Discovery from Data*, 6(1):Article 3.
- Maulik, U. and S. Bandyopadhyay. (2002). “Performance evaluation of some clustering algorithms and validity indices.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654.

- McLachlan, G. and D. Peel. (2000). *Finite Mixture Models*. John Wiley & Sons, New York, NY.
- Mitchell, T. (1997). *Machine Learning*. McGraw Hill, New York, NY.
- Pelleg, D. and A. Moore. (2000). "X-means: Extending k -means with efficient estimation of the number of clusters." In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, 727–734, Stanford, CA.
- Rousseeuw, P. J. (1987). "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis." *Journal of Computational and Applied Mathematics*, 20:53–65.
- Schölkopf, B., A. Smola, and K.-R. Müller. (1998). "Nonlinear component analysis as a kernel eigenvalue problem." *Neural Computation*, 10(5):1299–1319.
- Stanfill, C. and D. Waltz. (1986). "Toward memory-based reasoning." *Communications of the ACM*, 29(12):1213–1228.
- Tan, X., S. Chen, Z.-H. Zhou, and J. Liu. (2009). "Face recognition under occlusions and variant expressions with partial similarity." *IEEE Transactions on Information Forensics and Security*, 2(4):217–230.
- Tibshirani, R., G. Walther, and T. Hastie. (2001). "Estimating the number of clusters in a data set via the gap statistic." *Journal of the Royal Statistical Society - Series B*, 63(2):411–423.
- von Luxburg, U. (2007). "A tutorial on spectral clustering." *Statistics and Computing*, 17(4):395–416.
- Xing, E. P., A. Y. Ng, M. I. Jordan, and S. Russell. (2003). "Distance metric learning, with application to clustering with side-information." In *Advances in Neural Information Processing Systems 15 (NIPS)* (S. Becker, S. Thrun, and K. Obermayer, eds.), 505–512, MIT Press, Cambridge, MA.
- Xu, R. and D. Wunsch II. (2005). "Survey of clustering algorithms." *IEEE Transactions on Neural Networks*, 3(16):645–678.
- Zhang, T., R. Ramakrishnan, and M. Livny. (1996). "BIRCH: An efficient data clustering method for very large databases." In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 103–114, Montreal, Canada.
- Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*. Chap-

- man & Hall/CRC, Boca Raton, FL.
- Zhou, Z.-H. and Y. Yu. (2005). “Ensembling local learners through multimodal perturbation.” *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 35(4):725–735.

休息一会儿

小故事：曼哈顿距离与赫尔曼·闵可夫斯基

曼哈顿距离(Manhattan distance)亦称“出租车几何”(Taxicab geometry), 是德国大数学家赫尔曼·闵可夫斯基(Hermann Minkowski, 1864—1909)所创的词汇, 其得名是由于该距离标明了几何度量空间中两点在标准坐标系上的绝对轴距总和, 这恰是规划为方形区块的城市里两点之间的最短行程, 例如从曼哈顿的第五大道与33街交点前往第三大道与23街交点, 需走过 $(5-3)+(33-23)=12$ 个街区。



今立陶宛的考纳斯(Kaunas).

哥尼斯堡是著名的“七桥问题”发源地, 今俄罗斯加里宁格勒。

四维时空亦称“闵可夫斯基时空”或“闵可夫斯基空间”。

闵可夫斯基出生于俄国亚力克索塔斯(Alexotas)的一个犹太人家庭, 由于当时俄国政府迫害犹太人, 他八岁时随全家移居普鲁士哥尼斯堡, 与后来成为大数学家的希尔伯特一河之隔。闵可夫斯基从小就是著名神童, 他熟读莎士比亚、席勒和歌德的作品, 几乎能全文背诵《浮士德》; 八岁进入预科学校, 仅用五年半就完成了八年的学业; 十七岁时建立了 n 元二次型的完整理论体系, 解决了法国科学院公开悬赏的数学难题。1908年9月他在科隆的一次学术会议上做了《空间与时间》的著名演讲, 提出了四维时空理论, 为广义相对论的建立开辟了道路。不幸的是, 三个月后他死于急性阑尾炎。

1896年闵可夫斯基在苏黎世大学任教期间, 是爱因斯坦的数学老师。诺贝尔物理学奖得主玻恩曾说, 在闵可夫斯基的数学工作中找到了“相对论的整个武器库”。闵可夫斯基去世后, 其生前好友希尔伯特整理了他的遗作, 于1911年出版了《闵可夫斯基全集》。闵可夫斯基的哥哥奥斯卡是“胰岛素之父”, 侄子鲁道夫是美国著名天文学家。