第5章 数据挖掘

随着信息化的发展，企业积累了越来越多的数据，人们对数据的应用需求也日益强烈。许多对决策起重要作用的知识往往隐含在海量的数据中，为了充分利用这些数据资产，需要利用一定的方法把这些知识挖掘出来。数据挖掘（DM）就是从数据中获取知识的手段，例如，零售商可以利用数据挖掘分析顾客的购物行为和偏好，预测顾客消费的趋势。

数据挖掘又称为数据库中的知识发现（knowledge discovery in database，KDD），它是一个利用机器学习和统计学等多学科理论分析大量的数据，进行归纳性推理，从事务数据库、文本数据库、空间数据库、多媒体数据库、数据仓库以及其他数据文件中提取正确的、新颖的、有效的以及人们感兴趣的知识的高级处理过程。数据挖掘的任务是从大量的数据中发现对决策有用的知识，发现数据特性以及数据之间的关系，这些知识表现为概念、规则、模式和规律等多种形式。

5.1 数据挖掘的基础

企业经常需要从大量运营数据中获取信息和知识以辅助决策，但现有的管理信息系统难以满足这样的需求。常见的查询、统计和报表都是对指定的数据进行简单的统计处理，而不能对这些数据所蕴含的模式进行有效的分析。此外，数据挖掘与信息检索也不同，信息检索是针对数据的特征来寻找信息。例如，使用 Google 等搜索引擎寻找含有某关键词的网页。如何从大量数据中提取出隐藏的知识，就成为数据挖掘发展的动力。本节首先介绍了数据挖掘的概念，然后讨论数据挖掘的发展、分类、步骤以及其他的一些相关主题。

5.1.1 数据挖掘的概念

与第 4 章提到的 OLAP 不同，数据挖掘不是验证某个假设的正确性，而是在数据中寻找未知模式，本质上是一个归纳学习的过程。数据挖掘是一门涉及面很广的交叉学科，融合了模式识别、数据库、统计学、机器学习、粗糙集、模糊数学和神经网络等多个领域的理论。数据挖掘有一些替代词，如数据库中的知识发现、知识提炼、模式识别、数据考古、数据捕捞和信息获取等。由于“数据挖掘”能表现“挖掘”的本质，因此在学术界和企业界被广泛应用。

截至目前，数据挖掘还没有一个公认的精确定义，在不同的文献或应用领域也有不同的说法。例如，有学者认为数据挖掘是一个从大型数据库中提取以前未知的、可理解的、可用

的知识，并把这些知识用于关键的商业决策过程。也有学者把数据挖掘定义为在知识发现过程中，辨识存在于数据中的未知关系和模式的一些方法。或者认为数据挖掘是为那些未知的信息模式而分析大型数据集的一个决策支持过程。数据挖掘的过程比较复杂，其结果的评价也不是一件轻松的事情：数据挖掘是否完成了预定的目标？数据挖掘是否能给企业带来价值？投资回报率如何？在实际应用中，数据挖掘的结果最终还要看挖掘出的知识转化为行动的效果。

概括而言，数据挖掘是从大量的、不完全的、有噪声的、模糊的、随机的数据中提取正确的、有用的、未知的、综合的以及用户感兴趣的知识并用于决策支持的过程。其中“正确”意味着提取的信息、知识应该是正确的，保证在挖掘结果中正确信息的比例。数据挖掘的结果往往很多，“有用”意味着挖掘出的模式能够指导实践。要让用户接受一个挖掘出的业务模型，仅靠正确的结果是不够的，还需要考虑模型的可用性和可解释性，即模型有什么业务价值。数据挖掘毕竟不是为了建立一个完美的数学模型，而是要切实解决实际业务中出现的问题。“未知”强调挖掘的模式具有预测功能，不仅是对过去业务的总结，也可以预测业务的未来发展。“综合”说明数据挖掘的过程应当运用多种方法，从多个角度得出结论，挖掘结果不应该是片面的。此外，数据挖掘的结果是用户感兴趣的。同一组数据用不同的数据挖掘方法也可能得到不同的模式。在数据挖掘产生的大量模式中，通常只有一小部分是用户感兴趣的，这就需要通过设定兴趣度度量评价过滤掉用户不感兴趣的模式。每一种兴趣度度量都可以由用户设定阈值，低于阈值的规则被认为是不感兴趣的。兴趣度度量包括客观兴趣度度量和主观兴趣度度量，前者使用从数据推导出来的统计量来确定模式是否有趣，而后者需要领域专家的先验知识，可能需要领域专家解释和检验被发现的模式。下面简要介绍这些兴趣度度量，其中模式的简洁性、确定性、实用性和提升度属于客观兴趣度度量，而新颖性是主观兴趣度度量。

(1) 简洁性：模式兴趣度的一个重要因素是简洁，符合最小描述长度(minimum description length，MDL)的要求，便于理解和应用。模式简洁的客观度量可以看作模式结构的函数，用模式的二进位位数、属性数或模式中出现的操作符数进行度量。一个规则的条件越复杂，它就越难解释，用户对它的兴趣度可能就比较低。

(2) 确定性：每个发现的模式都有一个表示其有效性或值得信赖的确定性度量，如分类规则的置信度、关联规则的置信度等。

(3) 实用性：挖掘的模式或规则能带来一定的经济效益，如关联规则的支持度必须大于一定的阈值才可能有商业价值。对于分类或预测型任务，模型的实用性可以通过测试集的预测错误率来判断。而对于连续变量的估计，可以考虑估算值和实际值之间的差别。

(4) 提升度：比较模型的好坏还可以用提升度(lift)的概念。以顾客响应分析为例，假设从潜在的顾客群中抽取一定数量的样本进行市场推广，发现有 30％的响应者，而利用分类模型挑选同样数量的潜在顾客进行推广，有 65％的响应者，那么此分类模型的提升度 lift＝65/30＝2.17。

(5) 新颖性：新颖的模式是指那些提供新知识的模式，能够解释意料不到的信息，经常使用户感到意外。一个例外的规则可以认为是新颖的，它不同于根据统计模型和用户的信念所期望的模式。

5.1.2 数据挖掘的发展

数据挖掘是一门不断发展的学科,尽管作为一门独立的学科只有数十年的时间,但数据挖掘的起源可追溯到早期的模式识别、机器学习等人工智能技术以及统计学的抽样、估计和假设检验等。这些技术虽然没有被冠以数据挖掘之名,但至今仍然是数据挖掘的技术基础。随着数据库技术的发展,尤其是近年来计算机的性价比按摩尔定律增长,数据库技术被应用于越来越多的领域。企业存储的数据量越来越大,数据越来越复杂,高级数据库、并行处理和分布式技术也先后应用于数据挖掘领域。Oracle、Microsoft 和 IBM 等主流的数据库厂商聚焦商务智能,已在其产品中增加了数据仓库、在线分析处理和数据挖掘等功能。

在电子商务时代,各行业业务流程的自动化和各类信息系统不断深入的应用在企业内产生了大量的数据,这些数据最初不是为了分析的目的而收集的,而是在企业的日常运营中产生的。根据有关调查,每 12 个月左右,企业的数据量就会翻一番,而 93%～95%的数据进入数据库后并没有得到有效利用。换句话说,海量的、未被充分利用的数据并没有成为企业的财富,反而因占用企业的资源而成了负担。因此企业面临着两个问题:一方面,全球化竞争的加剧要求企业比任何时候都需要更快、更好地做出决策;另一方面,许多企业在面对逐年增长的业务数据时,不知道真正有价值的模式在哪里,难以发现数据中存在的关系以及根据现有的数据预测未来的发展趋势。数据挖掘正是在这个背景下应运而生的。

数据挖掘是一类深层次的数据分析方法,能够揭示隐藏的、未知的业务规律,以达到增加收入、降低成本的目的,使企业处于更有利的竞争位置。表 5.1 所示为数据挖掘的大致演变过程。

表 5.1 数据挖掘的大致演变过程

时 间	挖 掘 对 象	解决的问题
20 世纪 60 年代	文件系统	过去 5 年中公司总收入是多少,利润是多少
20 世纪 80 年代早期	关系数据模型 关系数据库管理系统	某分公司在去年 3 月的销售额是多少
20 世纪 80 年代晚期	各种高级数据库系统(扩展的关系数据库、面向对象的数据库等)和面向应用的数据库系统(时序数据库、多媒体数据库等)	购买产品 A 的顾客过一段时间是否会购买产品 B
20 世纪 90 年代	数据仓库、多媒体数据库和网络数据库	某分公司去年各个月份的销售额是多少
2000 年至今	流数据管理和挖掘 Web 挖掘 XML 数据库和分布异构数据分析 非结构化复杂数据挖掘 大数据分析 文本分析、情感分析和基于流数据的分析等	顾客智能、电子推荐、流程智能化管理等 异常分析、日志分析

数据挖掘软件的进展体现了数据挖掘技术的发展,其发展大致经历了以下阶段。

(1) 第一代数据挖掘软件是独立的,可以支持少数几种数据挖掘算法,典型的代表是 Salford System 公司的 CART 系统,其缺点是在数据量较大或者数据变化频繁时效率不高。

（2）第二代数据挖掘软件和数据库系统进行了集成，能够处理大规模的数据，但缺少对业务的预测能力。

（3）第三代数据挖掘软件有显著的进步，不仅增加了预测功能，而且还能在分布式系统中运行，挖掘网络环境下的数据，但不能支持移动应用，此问题在第四代数据挖掘软件中得到了解决。

（4）第四代数据挖掘软件支持移动计算和各种嵌入式系统，扩展了数据挖掘的应用领域。

新一代的数据挖掘方法面对的大数据环境更加复杂，不仅数据量猛增，而且非结构化程度增加，数据呈现分布和异构的特点，这些问题都对数据挖掘提出了挑战。

5.1.3　数据挖掘的过程

数据挖掘的过程由以下步骤组成：定义业务问题，抽取与预处理数据，选择挖掘方法分析，解释挖掘结果，探查新模式以及运用发现的知识，各步骤所占的工作量如图 5.1 所示。整个过程需要数据库管理员、业务分析师、数据挖掘专家（数据科学家、数据分析师、数据工程师等）、数据质量分析人员、系统开发人员等共同合作才能顺利完成。其中业务人员提出业务需求，协助熟悉数据挖掘算法和相关数据挖掘软件的数据分析员把业务问题转化为数据挖掘问题，并评价数据挖掘结果，最终把数据挖掘模型转化为企业的行动，创造价值。数据挖掘是一个非平凡的过程，一些步骤很难自动完成，如果后续步骤的结果不令人满意可能会回溯，这个过程需要循环多次才能达到目标。

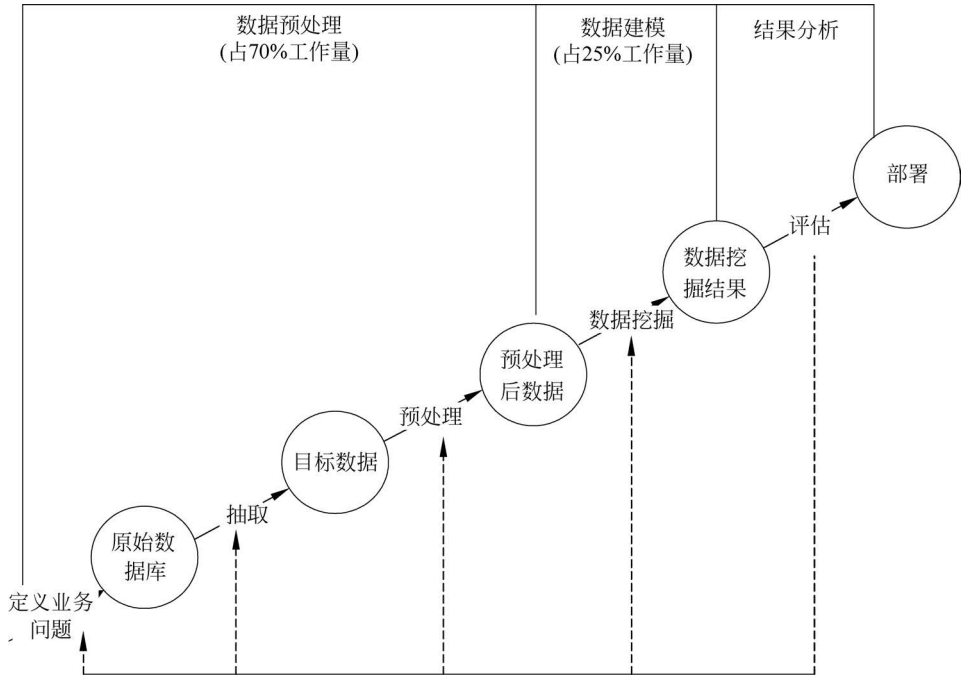


图 5.1　数据挖掘过程

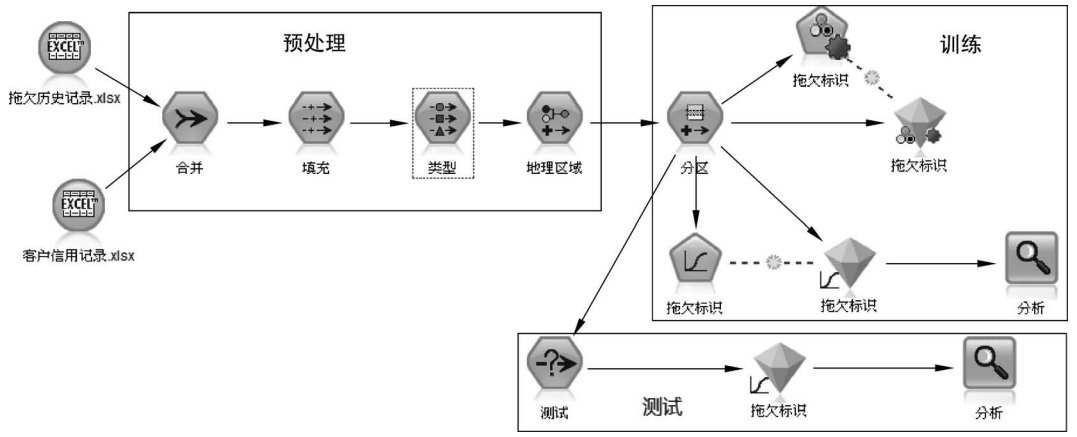


图 5.1 （续）

1. 定义业务问题

数据挖掘不是简单地把数据输入算法就可以解决问题，业务决策大多数情况下是比较复杂的。因此无论是处理大数据，还是常规数据，在做数据挖掘时都需要熟悉业务，与业务专家紧密协同，准确把握业务分析问题。在此基础上设计或选择合适的算法，并对挖掘结果进行严密的验证。这不是一个简单的过程。

数据挖掘第一步不是分析数据，而是理解业务需求，清晰定义业务问题，从而避免迷失在大量数据中。评价一个数据挖掘项目的成败，主要看挖掘的结果是否解决了业务问题。对于同一个数据集，不同的业务问题会需要不同的分析过程。这里所说的业务问题并不限于纯商业领域的问题，而是使用数据挖掘技术能够解决的问题。在定义业务问题时，只有了解相关领域的背景知识，才能确定挖掘什么内容。例如，在市场营销领域，用户感兴趣的可能是顾客的购买行为和购买方式，而在天文学领域，相关的知识是天体运动的规律以及某些天体在同一个地方同时出现的概率等。在此阶段与业务人员的充分交流是有必要的。

在定义业务问题时，首先，需要考虑是否有充足的与业务问题有关的数据。识别数据挖掘分析的数据是否包含需要的模式是很重要的，这甚至决定了一个数据挖掘项目的成败。其次，需要仔细考虑如何应用已发现的知识，思考如何把数据挖掘的结果应用到业务中有助于洞察业务存在的实际问题。例如，在分析顾客的购买模式时，数据挖掘的最终目的是通过了解顾客的购买模式，确定哪些潜在顾客会对公司的新产品感兴趣，从而针对这些目标顾客制定出相应的市场策略，以实现利润最大化。

2. 数据抽取与探测

高质量的数据可简化数据挖掘的过程，这需要从数据源头控制。数据挖掘在确定业务问题后就要抽取相关的数据，这些数据一般用简单文件、文本或数据库表的数据结构表示，不同的数据需要用到不同的工具和语言。分析什么数据？需要多少数据？如何进行各种数据的平衡？又需要什么转换才能进行有效的挖掘？解决这些问题是比较耗时的。数据挖掘往往需要使用大量的数据，但只有包含业务模式的数据才是真正需要的。例如对某公司的

顾客购买模式进行分析，很明显需要顾客购物记录和人口统计等方面的数据，这些数据分布在电子商务交易网站或者连锁店的数据库中，很多情况下数据的质量难以保证，因此充分的数据探测(exploration)是必要的。

数据抽取后不能马上进行数据建模。在数据挖掘之前，通过绘制各种图表，对数据的分布、变化趋势和相关关系等数据特征进行描述性的统计分析，理解数据的分布与统计信息，有助于全面了解数据特点并建立合适的数据挖掘模型。

3. 数据预处理

数据预处理有助于为数据挖掘提供高质量、易于处理的数据。良好的数据源是数据挖掘成功的重要保证，但现实的数据源中存在不完整的、异常(outlier)的和不一致的数据。在数据挖掘中，由于某种原因，例如，在该输入时操作人员没有输入，或者由于硬件的故障，或者有些数据被删除或修改等，造成了某些数据的空值(missing value)，而这些数据可能包含了重要的信息。此外，有些变量也可能是相关的。这些有问题的数据在数据挖掘早期必须被有效地处理，否则就会影响数据挖掘的质量。因此在数据挖掘前，需要通过可视化、统计学理论等手段对数据进行评价和预处理，有关数据预处理的内容将在 5.3 节详细讨论。事实证明，只要从数据源开始时控制数据质量，不断纠正各种数据质量问题，就可以推动数据质量的不断改善。

4. 数据建模

选择一个合适的数据挖掘算法还是多种挖掘算法的组合主要由定义的业务问题决定，其中参数的选择是一个比较棘手的问题(需要理解数据挖掘算法及其参数的作用)。一旦业务问题明确后，就可以从分类、聚类、关联、预测或者序列分析等方法中选择相应的数据挖掘方法。这些方法可以分为发现型(discovery)数据挖掘和预测型分析(predictive analysis)，前者不需要一些业务相关的先验知识(prior knowledge)，包括聚类、关联和序列分析，后者包括分类、回归分析等。然后在此基础上确定合适的挖掘算法。按照学习方式的不同，数据挖掘算法分为监督学习和无监督学习。在监督学习中，输入数据(训练样本)有明确的类别或标识，在训练过程中不断调整预测模型，使得预测结果与数据的实际结果尽量接近。监督学习算法包括常见的分类算法、回归分析方法等。在无监督学习中，输入数据没有类别或标识，通过训练得到数据中的内在规律。常见的无监督学习有关联挖掘、聚类等。

每种数据挖掘算法都有适用的范围(处理的业务问题类型、数据量、数据类型等)和局限性，需要的数据预处理方法也是不同的，数据建模算法也不是越复杂越好。通常每一类问题可通过多种算法解决，每个算法可能生成不同的结果，具体选择哪种(些)算法没有固定的思路，有时需要综合多种方法才能挖掘出较满意的结果。此外，数据挖掘的结果只是辅助决策，最终的决策还要结合决策人员的业务经验。

面对日益复杂的应用场景，使用单一的数据挖掘算法可能难以满足应用的需求。混合数据挖掘(hybrid data mining)综合运用多种数据挖掘算法，以解决更复杂的问题。例如，在银行客户分析时，可以先使用聚类算法，对客户进行细分，掌握各类客户的群体描述。在此基础上，再使用决策树算法对各类客户的特征进行识别，便于对新客户的类别进行预测，辅助企业的精准营销、产品推荐、客户价值分析以及风险评估等业务决策。

5. 评估数据挖掘结果

为了判断模型的有效性和可靠性,需要评估数据挖掘结果。评估模型的好坏可用准确率、召回率、均方根误差、速度、鲁棒性、可解释性等指标。数据挖掘算法会输出许多模式,但并不是所有的模式都是用户感兴趣的,因此需要对这些模式进行评估,这个阶段与业务人员的充分沟通是必要的。可视化的工具把数据挖掘结果以一种直观的形式呈现,有助于解释数据挖掘的结果。

6. 部署

数据挖掘的价值体现在把挖掘结果应用到商务决策,更好地辅助管理人员和业务人员的决策,产生经济效益。这里需要注意,挖掘得到的模式应该回到数据产生的业务背景。此外,这些模式有一定的时效性,需要补充新的数据增量挖掘、更新。

下面以电商客户评论的情感分析为例,说明数据挖掘的过程。随着电子商务的发展,竞争逐渐激化,如何改善客户的体验尤其重要。通过对电商平台客户评论文本的分析,挖掘客户对商品的情感倾向以及客户对这些商品满意或不满意的原因就变得尤其重要。

在确定客户情感分析的主题后,就可以利用数据抓取软件(例如八爪鱼软件 http://www.bazhuayu.com/download)或编程,从相应的商品页面抓取客户的评论数据。这是业务理解和数据准备阶段。然后进行数据理解以及预处理。这个步骤主要是利用分词软件,通过字符串匹配、句法分析、关联分析或者基于机器学习的方法,提取重要的关键词,并删除一些停用词、语气词、连词、介词等无用的数据。这里关键词的重要性可以用文献检索的 TF-IDF 等方法计算。为了提高分析的效果,可以使用 LDA 等方法提取评论文本中隐含的主题,而不是简单地根据词频多少提取关键词,从而可以进行评论的语义分析。在此基础上,可以对用户的情感进行统计分析,找出客户对商品持有的各种情感的分布,并利用标签云等可视化的方法,展示反映客户正面和负面情感的主题。上述分析不仅为改善客户的体验提供了有用的信息,也为网商改善经营,制造商完善产品设计都提供了指导。

5.1.4 数据挖掘原语与语言

数据挖掘原语用于定义数据挖掘任务。例如,一种原语是用来说明待挖掘的数据来源。通常,用户感兴趣的只是数据库的一个子集。一般情况下发现的许多模式与用户的兴趣无关。此外,说明挖掘什么类型的知识是非常重要的,可以使用元模式或概念分层来实现。知识类型包括概念分层、关联、分类、预测和聚类等。概念分层是一种有用的背景知识,它使原始数据可以在较高的、一般化的抽象层次上进行处理。数据的泛化或上卷可以通过用较高层概念替换较低层的概念实现,以方便用户在较高的抽象层观察数据。泛化的另一个用处是压缩数据,与未压缩的数据相比,效率更高。

数据挖掘面临着一些问题,例如,目前的数据挖掘系统基于不同的技术和方法,仅提供孤立的知识发现功能,很难嵌入大型应用。数据挖掘引擎与数据库系统是松散耦合的,没有提供独立于应用的操作原语等。在这种背景下,人们设计了数据挖掘语言。数据挖掘语言由多种数据挖掘原语组成,完成一项数据挖掘任务,常见的数据挖掘语言有数据挖掘查询语

言、数据挖掘建模语言和通用数据挖掘语言等。

1. 数据挖掘查询语言

在数据挖掘查询语言(DMQL)中,DBMiner 系统采用的 DMQL 具有一定的代表性。通过 DMQL 原语,可以在多个抽象层上挖掘多种类型知识。DMQL 中的原语主要包括以下内容。

1) 任务相关的数据原语

任务相关的数据原语说明挖掘涉及的相关数据所在的数据库或数据仓库。通常,挖掘的对象不是整个数据库或数据仓库,而是与具体业务问题相关的数据集。任务相关数据原语包括使用的数据库或数据仓库、数据过滤条件、相关属性以及数据分组标准等。

2) 知识类型原语

知识类型原语说明数据挖掘的知识类型,在 DMQL 中把挖掘的知识分为特征化、区分/比较、关联规则、分类模型和聚类等,其中特征化用于描述所挖掘的数据所具有的特性,区分把给定目标类对象与其他一个或多个对比类对象进行比较,关联规则用于表示数据集中不同项目之间的关联,分类模型用于从样本集的属性中找出分类特征,而聚类则是对样本集中数据的分组,从而确定每个样本的类别。

3) 背景知识原语

背景知识是关于挖掘领域的知识。这些知识对于指导知识发现过程和评估发现模式都是非常有用的。在 DMQL 中有一种简单但功能强大的背景知识原语,即前面提到的概念分层。通过使用概念分层,用户可以在多个抽象层次上对数据进行挖掘。概念分层结构通常采用树的形式表示,但也可以表示成偏序或格。利用概念分层,用户可以有效地对数据集进行上钻和下钻,采用不同的抽象层视图观察数据,挖掘隐藏的数据之间的联系。

4) 兴趣度度量原语

通过使用任务相关数据,挖掘知识类型和背景知识原语,可以缩减所要处理的数据集规模,从而减少数据挖掘产生的模式数量。但数据挖掘过程中仍然会产生大量的模式,其中大多数可能是用户不感兴趣的,需要进一步缩减这些模式的数量。通过对用户兴趣度设定阈值,可以排除用户不感兴趣的模式。

5) 知识的可视化和表示原语

一个有效的数据挖掘系统能够使用多种容易理解的方式表示数据挖掘产生的知识,如规则、表、交叉表、报表、图形、决策树(decision tree)和立方体等。采用多种可视化的方式表示挖掘结果有利于用户理解挖掘的模式,方便用户与系统交互并指导挖掘过程。下面以银行信用卡用户信用等级的分类挖掘为例,给出 DMQL 的一个应用示例。

```
use database Bank_db
use hierarchy age_hierarchy for C.age
mine classification as classifyingCustomerCreditRating analyze
    Card.credit_info
in relevance to C.age, C.income, C.occupation
from Customer C, Credit_Card Card, Credit_Charge_Log Log
where C.id = Card.user_id and Card.id = Log.card_id and Log.charge > 50
```

```
with noise threshold = 0.05
display as table
```

2. 数据挖掘建模语言

数据挖掘建模语言是对数据挖掘模型进行描述的语言。这种语言为数据挖掘系统提供了模型定义和描述等方面的标准,而且数据挖掘系统之间可以共享模型。数据挖掘建模语言还可以在其他应用系统中嵌入数据挖掘模型,增强这些应用系统的分析能力。

预言模型标识语言(predictive model markup language,PMML)是由 Data Mining Group 于 1998 年开发的数据挖掘标准*。它是一种基于 XML 的语言,用于描述数据挖掘以及数据建模前需要的数据清洗和变换等操作。PMML 包括定义属性的数据字典、挖掘模式、数据变换(标准化、离散和值聚集等)字典和模型参数描述等。这种语言为不同的应用程序共享模型提供了一种快速、简单的方式。PMML 使用标准的 XML 解析器解释数据挖掘模型,有助于应用程序判断模型输入和输出的数据类型、模型的格式并且按照标准的数据挖掘术语解释模型。此外,使用 PMML 在不同的应用程序之间共享预言模型简单方便,例如,可以把数据挖掘模型通过 PMML 导入操作型 CRM,以便对目标顾客进行预测或者交叉推荐。

IBM 是使用 PMML 的业界领先者。IBM 在数据库产品 DB2 中使用基于 PMML 的 intelligent mining scoring(IMS)作为 DB2 通用数据库的一个组件。IMS 服务使企业可以依据既定的标准对顾客进行归类。IBM 也推出了基于 PMML 规范的 AnswerTree 和 SmartScore,其中 AnswerTree 用于建立决策树模型,而 SmartScore 可以对 SPSS 建立的(多元)回归模型进行评分。

下面给出 PMML 的一个简单应用实例。对某数据集训练一棵决策树,分类属性是“是否有车(car)”,条件属性包括性别(sex)、收入是否大于 8000(income)、是否结婚(married)和是否有小孩(haveChild)。训练后得到一个典型规则:If sex = male, income = yes and haveChild = no then car = yes。描述该规则的 PMML 文件如下。

```
<?xml version = "1.0"?>
< PMML version = "1.1" >
< Header copyright = "DB2 Magazine"
         description = "Predict car."/>
< DataDictionary numberOfFields = "5">
< DataField name = "sex" optype = "categorical"/>
      < Value value = "male"/>< Value value = "female"/>
</DataField >
< DataField name = "income" optype = "categorical"/>
      < Value value = "yes"/>< Value value = "no"/>
</DataField >
< DataField name = "married" optype = "categorical"/>
      < Value value = "yes"/>< Value value = "no"/>
</DataField >
< DataField name = "haveChild" optype = "categorical"/>
      < Value value = "yes"/>< Value value = "no"/>
```

* http://www.dmg.org。

```
</DataField>
<DataField name = " car" optype = "categorical"/>
     <Value value = " yes"/><Value value = " no"/>
</DataField>
</DataDictionary>
<TreeModel modelName = "haveCar">
<MiningSchema>
<MiningField name = " sex"/>
<MiningField name = " income"/>
<MiningField name = " haveChild"/>
<MiningField name = "car" usageType = "predicted"/>
</MiningSchema>
</TreeModel>
</PMML>
```

5.1.5 基于组件的数据挖掘

组件(component)的概念已经广泛地应用于各类软件中。由于组件具有灵活、可复用和安全等特点,因此数据挖掘系统也逐渐开始采用基于组件的架构。一种基于组件的数据挖掘系统框架如图 5.2 所示,此系统通过统一的数据访问接口从多个异构数据源收集数据,经过预处理后利用数据挖掘算法产生知识模型,这些模型都存放在模式库中。此外,这些模型还可作为其他数据挖掘算法的输入。

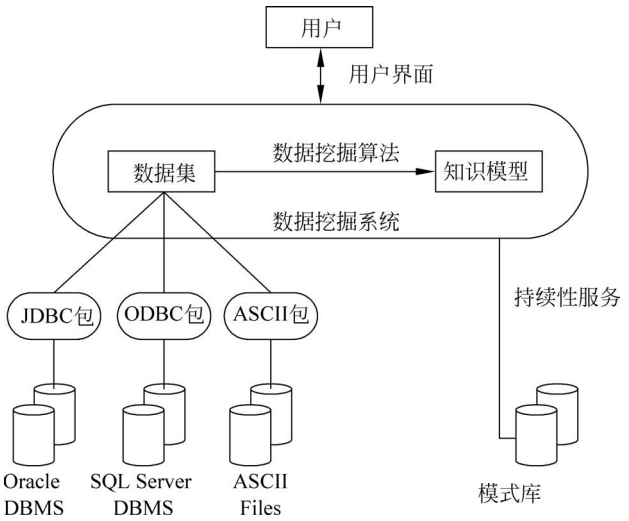


图 5.2 基于组件的数据挖掘系统框架

相对于传统的数据挖掘系统构建方式,基于组件的构建方式更加灵活方便。在相似的数据挖掘情境下,只要对已有的数据挖掘系统调整某些组件就可以适应新的需求,使用组件方便了数据挖掘系统的开发,可灵活适应市场环境的变化。目前,为了提高系统的易用性,很多数据挖掘系统都采用了组件方式组合完成整个数据挖掘过程。数据源的选择、数据的探测、数据的预处理、数据建模和评估等步骤都可由不同组件完成。目前主流的数据挖掘工具 SAS、IBM SPSS Modeler、SAP KXEN 等数据挖掘软件以及开源工具 Weka 等都是基于

组件,且符合数据挖掘标准 CRISP-DM(cross industry standard process for data mining,跨行业数据挖掘标准流程)。例如,SPSS Modeler 就是一个典型的基于组件的数据挖掘系统,如图 5.3 所示。使用组件的数据挖掘系统优势在于,如果数据挖掘的条件和对象发生变化,只需替换相关的组件或重新设置组件的参数即可。

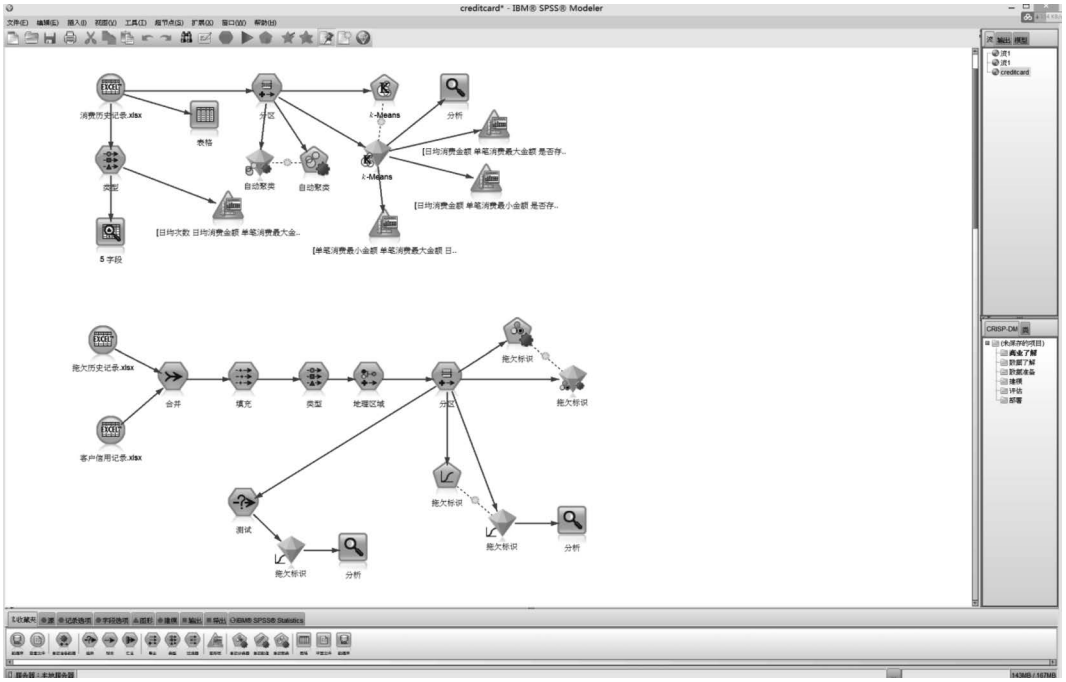


图 5.3 SPSS Modeler 数据挖掘系统

5.1.6 数据可视化

在信息爆炸的时代,如何将海量数据转换为有价值的信息成为一个急需解决的问题,对数据进行可视化是解决该问题的一个有效途径,因为人类对于图形的接受能力更强,更容易发现其中隐藏的模式。

数据可视化是指通过图表的形式,对处理后的数据进行分析,揭示其中蕴含的业务问题、规律,把数据转化为有用的信息,辅助决策。可视化分析贯穿了数据分析流程的全过程;促进对用户需求的把握,直观地发现数据中的异常点,了解数据的概率分布情况,发现数据之间的关联,选择和评估合适的数据模型,展示数据分析的结果。可视化也提供了多维分析功能,用户可从多视角、多层次地分析数据,并且能对变量的影响因素进行深入的分析。例如在卷积神经网络中使用可视化技术让研究人员更加直观地理解各层神经网络运行的结果。

1. 数据可视化技术的发展

数据可视化技术历史悠久。计算机的出现使得处理数据的能力突破了原先的瓶颈,研究人员开始使用计算机来代替人工绘制图表。由于计算机在运算速度和精度方面拥有巨大优势,在这一时期开始出现了树状图、聚类图等较为复杂的可视化图表,数据可视化的应用

领域相较之前也有较大拓展。

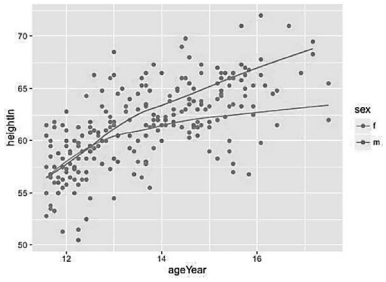
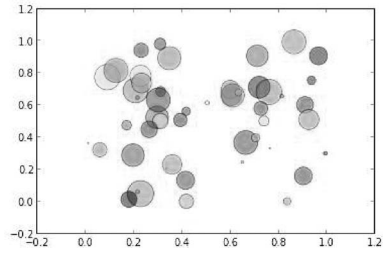
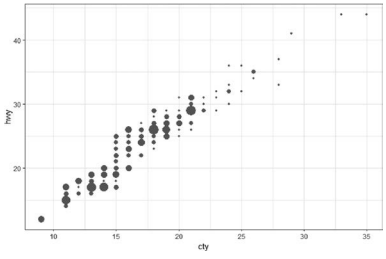
早期数据可视化多以静态图表为主，绘制这些图表需要分析人员事先汇总数据，建立数据模型。当数据或分析需求发生变更时，需要重新建模并修改图表，且图表所能显示的数据维度也仅限于一维和二维。随着可视化需求的不断扩大，20 世纪 80 年代开始出现动态交互图表，允许操作人员与图形进行实时交互，数据维度也由二维扩展至高维。

进入 21 世纪，大数据的处理提上日程，在完成数据处理前数据或需求就可能已经发生变化。目前可视化的发展趋势正由小数据处理发展为大数据挖掘，可视化的工具也由少数专家掌握转变为面向大众群体，且应用范围涉及多个领域，注重实时处理功能。

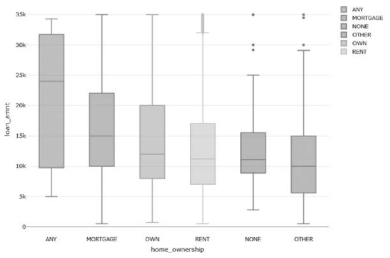
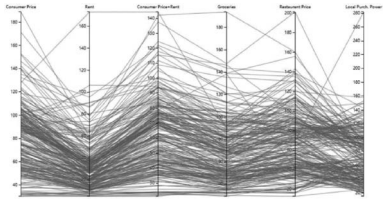
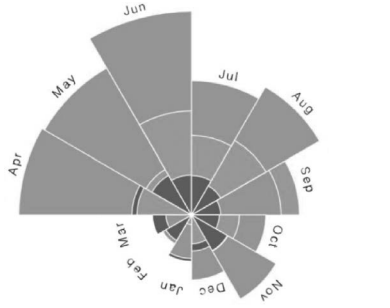
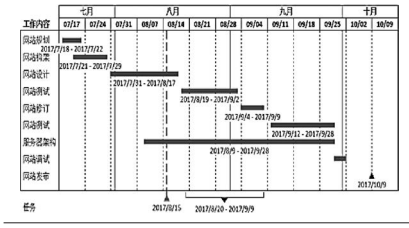
2. 常用的数据可视化图表

传统数据可视化图表包括柱状图、饼图、箱图、折线图、地图、气泡图、树状图等几十种图形，并且还有新的图表不断出现。几种目前常用的数据可视化图表如表 5.2 所示。

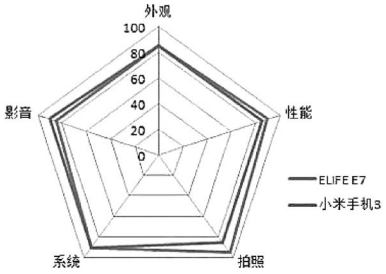
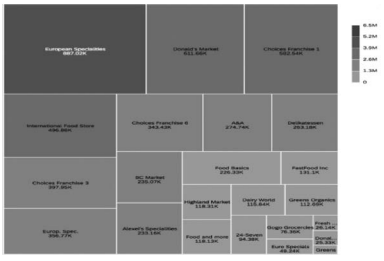
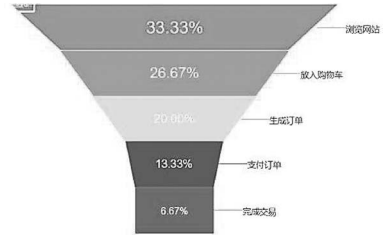
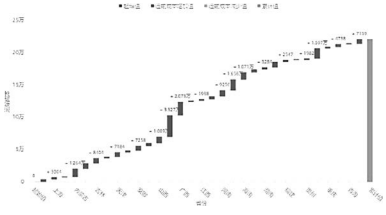
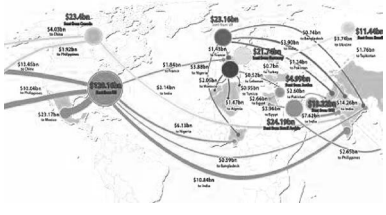
表 5.2　常用的数据可视化图表

图　　形	图形的作用与应用场合
散点图 	散点图用于将两组变量数据投影在平面直角坐标系上，通过观察坐标点的分布情况，判断两组变量之间的关系，例如线性关系、指数关系等
气泡图 	气泡图是在散点图的基础上增加了一个变量，表示气泡的大小
计数图 	计数图是在散点图的基础上通过点的大小来表示数据，用于解决散点图中数据点的重叠问题，通过点的大小反映该点处数据取值的个数

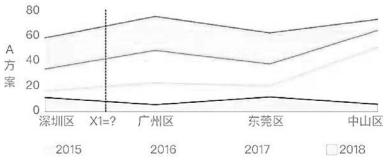
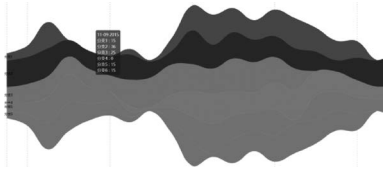
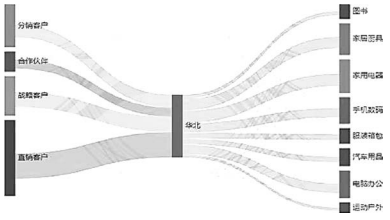
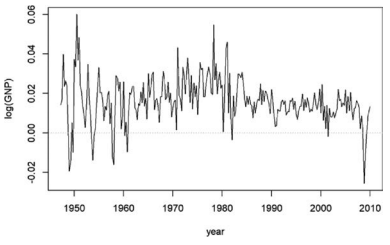
续表

图　　形	图形的作用与应用场合
箱图 	箱图是使用上四分位数、下四分位数、四分位距、上边界和下边界等表示数据的总体分布情况，用于识别数据中的异常值，也可判断数据的偏态和尾重
平行坐标图 	平行坐标图可以呈现多个变量之间的关系，常用于表示三维以上的高维数据可视化。平行坐标轴(竖线)用于表示多个数据的相关性
热力图 	热力图使用不同的颜色区分不同区域数据密度或数值大小。热力图常用于分析某一地区的人口分布、某一景区最热门的景点，或是分析某网页上用户最常点击或浏览的区域。热力图也可用于表示气象气温或降雨情况等
玫瑰图 	玫瑰图适用于展示类目较多的数据。玫瑰图使用不同的颜色表示不同类别的数据，可直观地对比多组数据。或是在原玫瑰图的每一扇区上进行细化。例如，将某一地区各方位风向出现的次数进行统计，绘制成玫瑰图
甘特图 	甘特图是指通过条形图来表示项目或进程与时间之间的关系，用于反映项目活动之间的先后顺序和持续时间

续表

图　　形	图形的作用与应用场合
雷达图 	雷达图用于多维度对比，主要应用于分析企业经营状况等，通过添加表示业内平均水平或最低水平的辅助线，可直观了解企业各指标在业内所处的位置。也可比较多组数据在各指标方面的强弱程度
树状图 	树状图将数据按某一分类标准分为几个矩形，代表不同的分支。通过矩形块的层级可以识别数据类别整体与部分的包含关系；通过矩形块的大小可以识别各数据的数值高低或其重要程度
漏斗图 	漏斗图由若干流程环节组成，各环节使用梯形分别表示当前环节的输入和输出。漏斗图常用于表示业务流程，可体现业务各环节的完成情况及各环节转换率，可帮助管理者了解关键环节的转换情况或是发现转换率不足的环节
瀑布图 	瀑布图采用了绝对值与相对值结合的方式，常用于表示数值之间增减关系的变化。例如可表示公司财务报表中的收入或支出的变化情况
流向图 	流向图结合了地图和流程图，直观展示了事物之间的联系以及运动趋势，常用于表示网络流向、航运路线或研究运输问题等。例如春运期间人口流动情况、贸易中货物交换的情况等

续表

图　　形	图形的作用与应用场合
面积图 	面积图用于反映变量随时间的变化，面积图不仅强调峰和谷，而且还强调高低点的持续时间。层叠面积图则是在上一个数据集所作面积图基础上作新的面积图
河流图 	河流图用于表示某段时间内发生的不同事件及其变化情况。通过不同颜色的条带表示某段时间内发生的不同事件；使用条带的宽度表示数据值的大小，表示事件在一段时间内的变化情况
桑基图 	桑基图由边、流量和支点组成，分别用于表示数据的流动情况、流动数据量的大小和分类。例如在能源领域，桑基图可以更好理解能源转换情况；在金融领域，桑基图可以表示各笔资金的详细流向，其总量不变的特性也适用于校核资金的数额
时间序列图 	时间序列图可用于直观显示某一变量随时间的变化趋势，通过对变化趋势的分析，可以对未来数据的变化走向进行预测。广泛应用于财务数据分析等场合

3. 数据可视化的典型工具

传统的数据可视化工具包括 Excel 等，适合制作一些基础的可视化图表，如柱状图、折线图、饼图、散点图等，但其功能有一定的局限性，也缺乏交互能力。

新兴的数据可视化工具中，具有代表性的是 Echarts、D3. js、Tableau 等。Echarts 基于 HTML5，有良好的动画渲染效果，提供可交互、个性化的可视化图表。Echarts 提供多种图表组合，可以无缝连接地图。通过交互组件对数据进行多维数据筛取、细节展示等操作。D3. js 提供了各种易用的函数以实现各种功能，可借助 HTML、SVG、CSS 等实现各种数据的可视化。Tableau 是一款专业的商业化数据可视化工具，操作简单，能够调用多种工业标准的数据库，数据导入便捷，支持在线分析处理、即时查询、数据动态更新等功能。Tableau

可集成显示多个相关的工作表，同时提供完整的分析功能，支持数据下钻等，可对数据进行深度探索。

数据可视化处理也可以使用一些编程语言，例如 Python 和 R 等。Python 可处理大批量数据，能够胜任繁重的分析工作，并提供了大量的可视化库，例如 Matplotlib、Seaborn、PIL 等，创建复杂的数据图形。R 适合做统计分析，也拥有众多的工具包，可以创建各种数据图形，从而顺利实现数据分析。

4. 数据可视化的应用

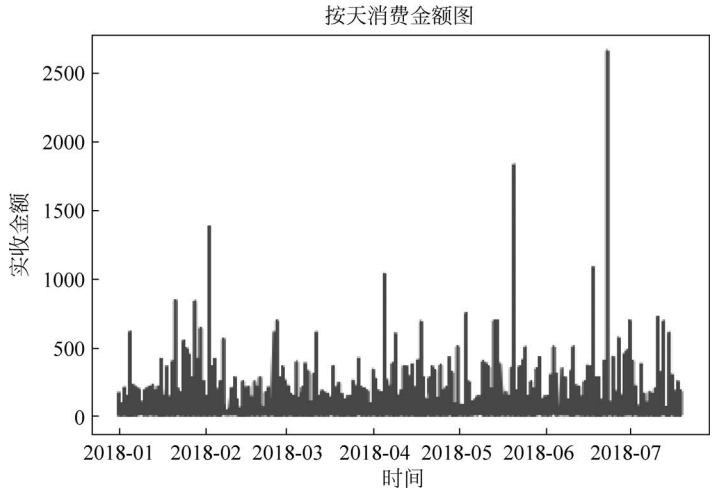
在商业领域，数据可视化技术被应用于商品销售分析、渠道分析等，例如在零售行业，通过采集各门店数据，可根据消费者的习惯性轨迹、货架上各类商品停留时间等，以热力图的形式表示客流情况，确定热门商品或是商场内顾客最常经过的区域，或是进行商业选址等应用。

药品销售可视化 Python 代码如下：

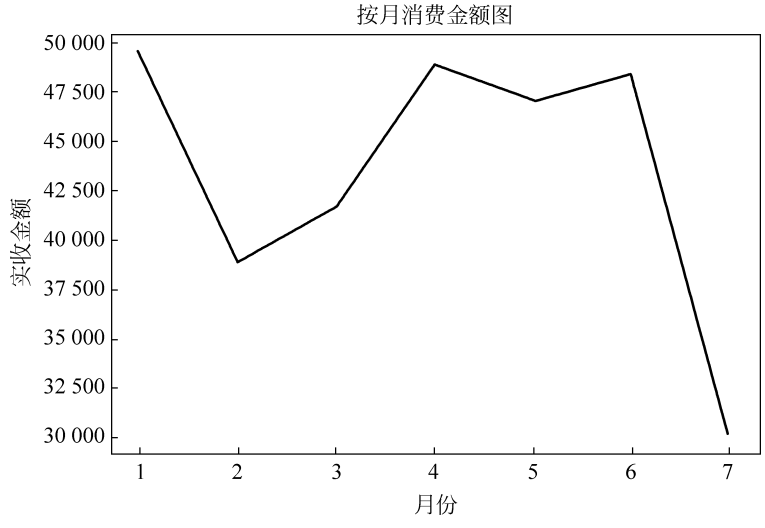
```
import pandas as pd
import matplotlib.pyplot as plt
from pylab import mpl
# 读取数据(最好使用 object 类型读取)
data = pd.read_excel("朝阳医院 2018 年销售数据.xlsx", dtype = "object")
# 修改为 DataFrame 格式
dataDF = pd.DataFrame(data)
# 使用 rename 函数,把"购药时间" 改为 "销售时间"
dataDF.rename(columns = {"购药时间": "销售时间"}, inplace = True)
# 使用 dropna 函数删除缺失值
dataDF = dataDF.dropna()
# 将字符串转为浮点型数据
dataDF["销售数量"] = dataDF["销售数量"].astype("f8")
dataDF["应收金额"] = dataDF["应收金额"].astype("f8")
dataDF["实收金额"] = dataDF["实收金额"].astype("f8")

def splitsaletime(timeColser):
  timelist = []
  for t in timeColser:
      # [0]表示选取的分片,这里表示切割完后选取第一个分片
    timelist.append(t.split(" ")[0])
     # 将列表转行为一维数据 Series 类型
    timeser = pd.Series(timelist)
  return timeser
# 获取"销售时间"数据
t = dataDF.loc[:, "销售时间"]
# 调用函数去除星期,获取日期
timeser = splitsaletime(t)
# 修改"销售时间"日期
dataDF.loc[:, "销售时间"] = timeser
# 字符串转日期
dataDF.loc[:, "销售时间"] = pd.to_datetime(dataDF.loc[:, "销售时间"], errors = 'coerce')
#删除为空的行
```

```
dataDF = dataDF.dropna()
# 按销售时间进行升序排序
dataDF = dataDF.sort_values(by = '销售时间', ascending = True)
# 重置索引(index)
dataDF = dataDF.reset_index(drop = True)
# 将"销售数量"列中小于 0 的数排除
pop = dataDF.loc[:, "销售数量"] > 0
dataDF = dataDF.loc[pop, :]
# 删除重复数据
kpi1_Df = dataDF.drop_duplicates(subset = ['销售时间', '社保卡号'])
kpi1_Df = dataDF.drop_duplicates(subset = ['销售时间', '社保卡号'])
# 统计行数
totall = kpi1_Df.shape[0]
print('总消费次数:', totall)
# 按销售时间升序排序
kpi1_Df = kpi1_Df.sort_values(by = '销售时间', ascending = True)
# 重命名行名(index)
kpi1_Df = kpi1_Df.reset_index(drop = True)
# 获取时间范围
startTime = kpi1_Df.loc[0, '销售时间']
endTime = kpi1_Df.loc[totall - 1, '销售时间']
# 计算天数
daysI = (endTime - startTime).days
# 月份数:运算符"//"表示取整除,返回商的整数部分
monthsI = daysI // 30
print('月份数:', monthsI)
# 计算月均消费次数
kpi1_I = totall // monthsI
print('业务指标 1:月均消费次数 = ', kpi1_I)
# 总消费金额
totalMoneyF = dataDF.loc[:, '实收金额'].sum()
# 月均消费金额
monthMoneyF = totalMoneyF / monthsI
print('业务指标 2:月均消费金额 = ', monthMoneyF)
# 客单价 = 总消费金额 / 总消费次数
pct = totalMoneyF / totall
print('业务指标 3:客单价 = ', pct)
mpl.rcParams['font.sans - serif'] = ['SimHei'] # SimHei 是黑体的意思
# 在操作之前先复制一份数据,防止影响清洗后的数据
groupDf = dataDF
# 重命名行(index)为销售时间所在列的值
groupDf.index = groupDf['销售时间']
groupDf.head()
# 画图
plt.plot(groupDf['实收金额'])
plt.title('按天消费金额图')
plt.xlabel('时间')
plt.ylabel('实收金额')
plt.show()
```

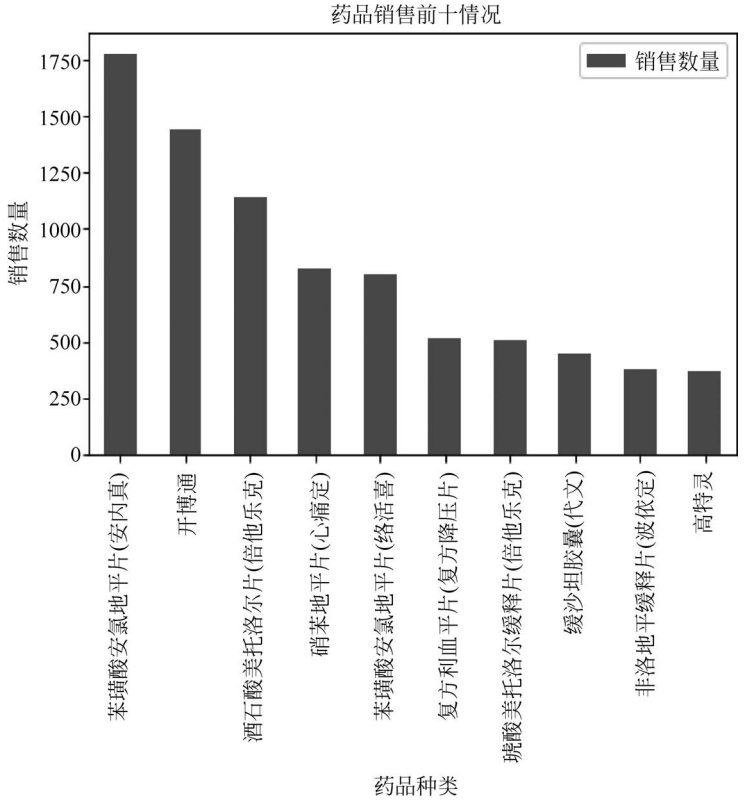



```
# 将销售时间聚合按月分组
gb = groupDf.groupby(groupDf.index.month)
# 应计算每个月的消费总额
monthDf = gb.sum()
# 描绘按月消费金额图
plt.plot(monthDf['实收金额'])
plt.title('按月消费金额图')
plt.xlabel('月份')
plt.ylabel('实收金额')
plt.show()
```




```
# 聚合统计各种药品的销售数量
medicine = groupDf[['商品名称','销售数量']]
bk = medicine.groupby('商品名称')[['销售数量']]
re_medicine = bk.sum()
# 对药品销售数量按降序排序
re_medicine = re_medicine.sort_values(by = '销售数量',ascending = False)
re_medicine.head()
```

```
# 销售数量最多的十种药品
top_medicine = re_medicine.iloc[:10,:]
# 用条形图展示销售数量前十的药品
top_medicine.plot(kind = 'bar')
plt.title('药品销售前十情况')
plt.xlabel('药品种类')
plt.ylabel('销售数量')
plt.legend(loc = 0)
plt.show()
```



5.1.7 数据挖掘的隐私保护

数据挖掘是从数据中挖掘出隐藏的、有价值的知识,在实际应用中,大量的数据中可能包含用户的个人隐私信息,如银行信用卡数据、用户注册数据、消费记录和手机通话记录等,对这些数据进行挖掘可能会侵犯用户的隐私。调查显示,绝大多数网站的用户都不愿提供真实信息,担心网站会滥用这些信息或侵害个人隐私。对很多公司而言,挖掘什么数据在数据收集时也是不确定的,用户很难知道公司如何利用包含个人隐私的数据。因此,如何确保数据挖掘过程中不泄露或尽量少地泄露隐私信息,已成为数据挖掘的一个重要研究方向。W3C 提出的 P3P(platform for privacy preferences,个人隐私安全平台)标准允许网民控制个人资料在网络上的开放程度。事实上,为了保护用户的隐私,有学者早在 2000 年就提出了隐私保护数据挖掘的新算法。

常用的隐私保护方法主要包括数据预处理法、基于关联规则的方法和基于分类的方法

等多种。数据预处理法是人们使用比较早的方法，其主要思想是在数据预处理阶段删除数据中最敏感的某些字段，如姓名和证件号码等，或者在数据集中随机添加、修改和转换某些字段的数据，这些数据能起到干扰作用，从而避免隐私泄露。上述方法比较简单，但也可能影响挖掘结果。基于关联规则的方法首先在数据集中挖掘关联规则，然后通过预先设定的学习方法或人工方法区分敏感规则和非敏感规则，根据敏感规则可以删除其中的部分敏感项或者给予较低的权重。基于分类的方法是建立一个没有隐私泄露的分类规则，用于区分包含隐私的信息和不包含隐私的信息。对于每条数据，该方法尝试都使用一些字段代替敏感字段，再进一步计算这种替换对于数据集本身的影响，从而找出一种尽量少地泄露隐私，又不破坏数据集完整性的方案。上述几种方法都可以起到保护隐私的作用，但后面两种方法的可扩展性更强、实际效果更好，是比较实用的算法。

此外，还有其他的隐私保护方法。例如，根据不同的安全级别对数据进行分类和限制，仅允许用户访问授权的安全级别或使用加密技术对数据编码。还有一种方法，称为匿名法，它通过泛化数据标识符来防止隐私数据泄露。

隐私保护和隐私攻击竞相发展，这对各种隐私保护方法提出了挑战。近期出现了差分隐私保护的方法，这种方法通过向查询和数据分析结果中掺杂适量的噪声，达到隐私保护的目的。

当然，保护用户的隐私仅仅依靠技术手段是不够的，制定相关的法律法规也是很有必要的。

【例 5.1】 医疗数据挖掘隐私保护

医学领域的大量数据处理往往需要由专业机构完成，所以在数据挖掘的过程中可能会出现病人隐私泄露的问题。下面介绍一个基于数据预处理法的医学隐私保护实例，某医院的病人原始病历如表 5.3 所示。

表 5.3 病人原始病历

编号	姓名	性别	年龄/岁	是否发热	是否呼吸困难	淋巴细胞数 /(10^9/L)
1	张三	男	40	是	是	1.2
2	李四	男	25	否	是	0.6
3	王五	女	29	是	是	0.8
…	…	…	…	…	…	…

表 5.3 中的病人姓名对于数据挖掘是不重要的，在数据挖掘前可以删除。对于其他字段，该医院采用了数据转换的方法进行隐私保护。定义一个转换函数 F，该函数采用 Hash 算法把任意长度的字符串转换为 10 位数字。表 5.3 中的数据经过转换后得到表 5.4 所示的病历信息。

表 5.4 转换后的病历信息

0086504692	0212459792	0071164880	0153471795	0248929060	0016528691
0000000012	0267625744	0000000168	0150185280	0150185280	0002500126
0000000016	0267625744	0000000108	0325654575	0150185280	0002500102
0000000020	0032745200	0000000124	0150185280	0150185280	0002500112
…	…	…	…	…	…

经过转换后的数据发送给专业机构进行数据挖掘是比较安全的,且不会影响数据挖掘的质量,挖掘结果可通过转换函数 F 的逆过程进行解码得到,例如,0071164880＞0000000168 and 0153471795＝0150185280→ 0016528691＞0002500126,支持度为 80%。经过解码后可以得出一条有意义的关联规则:有 80%的年龄大于 40 岁且发热的病人中淋巴细胞数大于 1.2×10^9/L。

5.2 数据挖掘的典型应用领域

数据挖掘的应用范围相当广泛。数据挖掘不仅在一些传统行业中得到了应用,而且在电子商务等新兴的科技领域中也引起了人们的注意。在过去的十几年中,大型商业数据库(特别是数据仓库)的使用和人们需要了解数据之间的内在规律的需求迅速增长,导致数据挖掘广泛地应用于多样化的商业领域。下面简单介绍数据挖掘在一些典型行业的应用。

1. 银行

通过数据挖掘,一方面可以对顾客的信用卡使用模式进行分类,检测信用卡欺诈行为,并按顾客等级和类型建立信贷发放模型,避免顾客出现信贷危机,减少信贷损失;另一方面,根据信用卡的使用模式,可以识别为银行带来较高利润的顾客,进行收益分析。

2. 证券

数据挖掘在金融业的应用还包括从股票交易的历史数据中得到股票交易的规则或规律,或者探测金融政策与金融业行情的相互影响等。

3. 保险

在保险业领域,可以通过历史数据预测何种顾客将会购买什么样的保险,从而推出具有针对性的保险产品,根据顾客的消费特征制订营销计划;可以分析如何对不同行业、不同年龄段、不同层次的顾客确定保险金数额;数据挖掘也可以进行险种关联分析,分析购买了某种保险的顾客是否会同时购买另一种保险,进而促进保险公司的业务。此外,利用数据挖掘还可以分析承保新险种和新顾客的风险,发现高风险市场区域,减少赔偿。

4. 零售

在零售业中,数据挖掘的主要应用之一是分析顾客的购买行为和习惯。例如,“某地区的男性顾客在购买尿布的同时购买啤酒”“顾客一般在购买了睡袋和背包后,过了一定的时间也会购买野营帐篷”等,这些模式促使零售企业改进营销手段。数据挖掘也可以分析企业销售商品的构成,例如,把商品按照利润的多少分成多个类别,然后分析属于同一类别商品的共同特征。这些知识有助于决策商品的市场定位、商品的定价等。数据挖掘工具还可以用于预测商品销售量、分析商品价格和选择零售点等,例如,聚类可用于顾客细分,把顾客分成不同的群组进行有针对性的营销。

5. 电信

数据挖掘在电信行业主要用于分析顾客的消费记录,确定高收益的产品和顾客分布。

通过分析历史记录、竞争和交流渠道数据，对个人呼叫行为特点进行全面分析，设计面向特定顾客群的服务和营销策略，并预测顾客将来的产品需求和服务需求。

6. 科学研究

数据挖掘在科学研究领域也受到了重视。例如，在气象学中，可以对不同的海流情况进行聚类，根据以往的经验判断海流对未来气象的影响。在生物信息领域，数据挖掘还用于基因分析。在疾病治疗中，数据挖掘可以从健康组织、病变组织中分离出基因序列，结合疾病和药物的情况发现一些疾病的致病机理和治疗措施，例如，运用聚类算法分析遗传数据和基因数据，从而发现具有类似功能或特征的遗传基因组。

5.3 数据预处理

数据的提取和预处理是数据挖掘过程中最耗时、最费力的工作。数据提取包括理解业务问题、搜集并分析数据源、确定数据的相关性以及使用工具提取数据等。数据挖掘的数据来源包括内部数据源和外部数据源，数据有不同的格式。数据挖掘与其他的分析工具一样，通常要求数据存放在表格或者文件中。如果数据分布在多个数据源中，那么数据整合是有必要的。如果数据存储在关系数据库中，那么创建一个新表或者新的视图是可行的，在构建这些表或者视图时，可能会执行复杂的聚合计算把数据整合成数据挖掘方法所需要的形式。经验表明，数据预处理除了需要一些统计学、数据分布等知识外，对数据的数理特征的理解也是有必要的。

数据预处理是数据挖掘过程的基础工作，一般占整个数据挖掘过程 70％的工作量。数据预处理技术用于数据清洗、数据集成、数据变换和数据归约等。数据清洗是指删除噪声和不一致的数据。数据集成是把多个数据源的数据合并存储，如数据仓库。数据变换通过规范化的方法改善数据挖掘算法的精度和有效性。数据归约通过删除冗余属性，使用聚集或聚类方法压缩数据。有效地使用这些数据预处理技术能够在不同程度上改善数据挖掘的质量，但也可能损失一些对数据挖掘有用的信息。

1. 数据清洗

在现实世界里，很多情况下数据中都存在不一致、不完整和噪声数据。通常，从低质量的数据中很难挖掘出有价值的知识，因此这些数据往往不能直接用于数据挖掘。这就需要通过数据清洗来修补空缺的值，识别出数据中的孤立点，去除噪声，消除数据中的不一致。为了有效地清洗数据，可以利用清洗工具维护、控制数据源的质量，避免无用的、过期的、残缺不全的和重复的数据进入系统。下面介绍几种常用的数据清洗方法。

1）聚类

通过聚类可以检测孤立点，图 5.4 中落在聚类集合外的点被视为孤立点。

此外，消除噪声的方法还有人工检测和回归分析两种。人工检测是指由专业人员识别孤立点。通过人与计算机的结合，相比单纯手动检查整个数据库，人工检测可以提高效率。回归分析是通过回归函数平滑数据，也可以利用一个变量预测另一个变量。当涉及多个变量时，可以使用多元回归。使用回归分析可以找出合适的数学模型，帮助消除噪声。

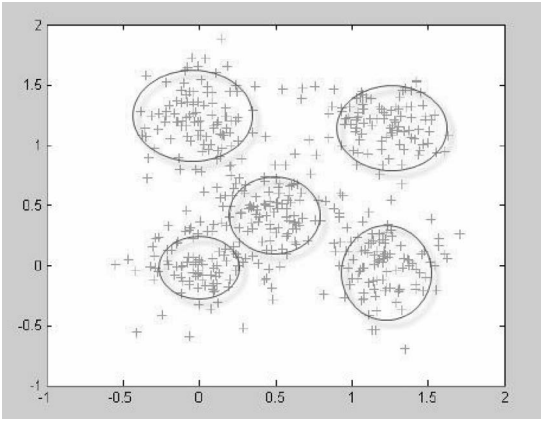


图 5.4 聚类分析检测孤立点

2) 空值处理

在数据预处理中,空值也是常见的,有些记录的属性值可能存在空缺。有时可以用人工方法填写遗漏的空值,但这种方法费时费力,并不常被采用。也可以忽略某些类标号缺失的记录,但这种方法可能会遗漏某些重要信息,当空值的百分比很高时,数据挖掘的性能可能就比较差。常见的处理空值的方法包括以下几种。

(1) 忽略包含空值的记录或属性。这种方法简单,但当空值比例较高或含空值的属性比较重要时,数据挖掘性能可能较差。

(2) 使用一个常量填充遗漏值,把遗漏的属性值用一个常数替换。但空值都用同一个值替换,也会影响挖掘结果的准确性。

(3) 使用数值型属性的平均值或与给定记录属同一类(近邻)的所有样本的平均值填充空值也是一种有效的方法,但这种方法在挖掘一些特殊规则时并不适用。

(4) 把空值属性看作决策属性,使用已知属性的值预测未知属性,即用数据挖掘(分类)的方法预测空值。

(5) 使用最可能的值填充空值,可以用统计分析、贝叶斯方法、相关分析或决策树等方法确定空值最可能的取值(数据挖掘也是数据清洗的一种工具)。这种方法的结果最接近原值,但相对比较复杂。例如,借助相关性分析数据相关程度、数据分布规律等特征,由所获得的信息对空值进行修补。这里两个属性 a、b 之间的皮尔逊相关系数定义为

$$r_{ab}=\frac{\sum_{i=1}^{n}(a_i-\bar{a})(b_i-\bar{b})}{(n-1)\sigma_a\sigma_b}$$

其中:n 是数据集的样本个数;$\bar{a}$ 和 $\bar{b}$ 分别是属性 a 和 b 的平均值;σ_a 和 σ_b 分别是 a 和 b 的标准差。

$$\sigma_a=\sqrt{\frac{\sum_{i=1}^{n}(a_i-\bar{a})^2}{n-1}}$$

$$\sigma_b=\sqrt{\frac{\sum_{i=1}^{n}(b_i-\bar{b})^2}{n-1}}$$

r_{ab} 的绝对值越大，表明属性 a、b 相关性越强：$r_{ab}>0$ 表示 a、b 正相关，即 a 的值随 b 增加而增加；$r_{ab}<0$ 表示 a、b 负相关，即 a 的值随 b 增加而减少；$r_{ab}=0$ 表示 a、b 是独立的且不存在相关关系。去掉空值 x 所在的列或行，对剩下的数据分别进行列或行相关分析，找到空值所在的列或行与其他列或行的相关系数。然后再考虑含空值 x 的列或行的相关系数，求解关于 x 的方程，可以得到空值的可能修补值。

3）冗余和重复

冗余和重复也是在数据集成过程中非常值得注意的问题。数据属性命名或者维命名的不一致都可能导致数据集中的冗余。冗余可以通过上述相关分析来检测。重复是指相同的数据在数据库中存储了多次，这种重复的数据会使数据挖掘的结果产生倾斜，所以需要进行检测。

2. 数据集成

数据集成把来自多个数据库或者平面文件等不同数据源的数据整合成一致的数据存储。数据集成时，需要考虑实体识别问题。例如，在一个数据库中用学号(student_No)作为学生的标识，而在另一个数据库中学号可能被命名为(S_ID)。通常使用元数据来避免数据集成中出现的错误。

在实际应用时，来自不同数据源的数据对于同一实体的描述也可能各不相同。这可能是由编码、单位或者比例的不同造成的。例如，相同商品价格在不同国家以不同的货币单位记录，相同大小和质量的商品在不同的数据库中用不同的度量单位表示。这些问题对数据集成来说都是挑战。为了提高数据挖掘的精度和减少数据挖掘使用的时间，对多个数据源的数据进行集成，减少数据集中的冗余和不一致是十分有必要的。

3. 数据变换

数据变换把数据转化成适于挖掘的形式。通过对某些属性按比例进行缩放，使属性取值落在较小的区间，例如，数值型属性可以规范化到[0,1]区间，这种变换对聚类、神经网络等算法都是必要的。连续属性离散化也是决策树等分类分析常用的预处理。

属性规范化会减少挖掘过程所用的时间，而且规范化可以有效地避免较大取值的属性对数据挖掘的过度影响。数据变换的常见方法如下。

(1) 平滑。平滑可以有效地去掉噪声，常用的方法有分箱(binning)、聚类和回归分析。这里简要介绍一下分箱，聚类和回归分析将在本章后面介绍。分箱是通过分析邻近的值平滑存储数据的值，可处理连续型和分类型变量，得到更少的变量取值种类，以便于分析。数据被分布到箱中，分箱的方法是进行局部的平滑，也可以作为一种离散化技术使用。在图 5.5 中，学生的数学成绩被划分并存入等深的深度为 3 的箱中，然后采用下面的方法之一进行平滑。

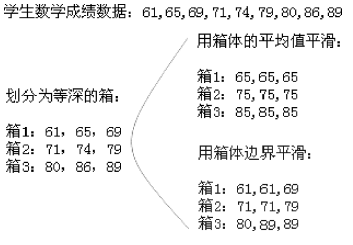


图 5.5　分箱操作

① 按箱平均值平滑：箱中每一个值都按箱中的平均值替换，例如，箱 1 中的值 61、65、69 的平均值是 65，该箱中的每一个值都被箱中的平均值 65 替换。

② 按箱中值平滑：箱中的每一个值，都按箱中的中值替换。

③ 按箱边界平滑：箱中的最大和最小值被视为箱边界。箱中的每一个值都被最近的

边界替换。

(2) 聚集。对数据进行汇总,例如,对某些产品的每周销售额感兴趣,而现有的数据是这些产品每天的销售量,此时就需要把数据汇总。聚集产生较小的数据集,使得分析的数据更稳定,但也应注意可能会丢失有趣的细节。

(3) 数据泛化。把任务相关的数据集从较低的概念层抽象到较高的概念层,例如,在分析顾客属性的年龄分布时,可以把顾客划分为年轻人、中年人和老年人等。

(4) 标准化(standardization)或规范化(normalization)。如果描述样本或记录的变量单位不统一,数值差别比较大,那么就需要通过把数据归一化、指数化或标准化,把不同的属性进行比例缩放,使它们的值落在大致相同的范围内。这在聚类分析、神经网络等数据挖掘算法的数据预处理中经常用到。

① 假定 $\min_a$ 和 $\max_a$ 分别为属性 a 的最小值和最大值,可通过变换 $a'=\dfrac{a-\min_a}{\max_a-\min_a}$ 把 a 转换为区间[0,1]的值 a',或者把 a 转化为区间[lower,upper]之间的值:$a'=\dfrac{a-\min_a}{\max_a-\min_a}(\text{upper}-\text{lower})+\text{lower}$。

② 通过变换 $\dfrac{a-\bar{a}}{\sigma_a}$ 把 a 的值 z-score 标准化,即转为平均值为 0,标准差为 1 的正态分布变量,其中 $\bar{a}$ 是 a 的平均值,而 σ_a 是 a 的标准差。

③ 把属性值除以该属性所有取值的均值,称为变量指数化。

4. 数据归约

数据挖掘时一般需要对数据集进行归约处理。对归约的数据集进行数据挖掘与原数据应该有相同或差不多的效果,但效率更高。常见的数据归约技术包括以下几种。

1) 数据立方体聚集

数据立方体聚集的基础是概念的分层,用于处理数据立方体中的数据,例如,收集的数据是某公司过去几年中每个季度的数据,而感兴趣的数据是年销售数据,可以通过对数据聚集汇总得到年总销售额。数据立方体聚集为在线分析处理的上钻等操作提供了可以快速访问的汇总数据。

2) 维归约

维归约可以剔除相关性较弱或者冗余的属性,例如,有些属性可能是由其他属性导出的。在实际应用中,数据挖掘只关心部分相关的属性,例如,进行购物篮分析,顾客的生日和电话号码等并不需要考虑,多余的数据会影响数据挖掘的效率,但遗漏了相关的属性或者选择了错误的属性都会对挖掘结果产生影响。维归约就是从决策分析相关的属性集中选择重要的属性(特征)子集,这需要启发式的算法解决,常用的方法有决策树、粗糙集(rough set)和遗传算法(genetic algorithm)等。其中决策树通过 ID3 等算法确定,出现在树根与树叶之间的属性形成属性子集,这部分的内容将在 5.5 节中详细讨论。粗糙集也是一种分析不精确、不确定性知识的数学工具,其最大特点是无须提供问题所需处理的数据集合之外的任何先验信息,利用定义在数据集上的等价关系对数据集进行划分,用不同的属性及其组合把数据划分成不同的基本类,在这些类的基础上进一步求得最小约简数据集。

目前的机器学习算法所涉及的数据集变得日益多样和复杂，常涉及数百甚至数千维。高维数据能够提供更丰富的信息，但同时也为分析带来了挑战。首先，过高的维度会使一些机器学习算法变得困难，因复杂度上升，训练时间增加，导致效率低下，容易引起维数灾难。其次，高维度会导致样本在高维空间中过于稀疏，导致训练的结果更容易产生过拟合现象，其质量反而会远低于低维度数据的预测。维数过多也难免会造成各维度间存在一定的相关性，导致冗余信息的产生。因此在处理维度较大的数据时，降维是非常必要的。

降维是指采用某种映射方法，将高维空间中的数据映射到低维空间，该映射可以是显式的或隐式的，也可以是线性的或非线性的。在减少特征数量的同时，使用降维避免丢失太多信息并保持模型性能。降维后的数据更易于分析。

降维方法分为线性降维和非线性降维，其中线性方法对于数据维数相对较低，且对于全局线性结构的数据具有很好的降维效果，包括主成分分析(principal component analysis，PCA)和线性判别分析(linear discriminant analysis，LDA)。而非线性方法则有多维标度分析(multidimensional scaling，MDS)等。

(1) 主成分分析。

主成分分析是一种常见的无监督降维方法。它通过正交变换，将线性相关变量表示的观测数据转换为少数几个由线性无关变量表示的数据，从而发现数据中的基本结构。原有数据集在经过正交变换后，形成新的特征集合，然后从中选择比较重要的一部分子特征，称为主成分，这样得到的低维数据集保留了原始数据的特征，从而实现降维。

主成分分析对原坐标系中的数据进行正交变换，将数据在新坐标系表示：首先选择方差最大的方向作为新坐标系的第一坐标轴，然后选择与此坐标轴正交，且方差次之的方向，作为第二坐标轴，以此类推。直至选取 k 个坐标轴，即可将原数据集投影到新的 k 维空间。

主成分分析是一种无标签的数据降维方法，可以利用主成分近似表示原始数据，从而发现数据的基本结构，但在降维后无法保留原始维度。主成分分析的主要计算步骤如下。

① 去中心化。对原样本矩阵 $\boldsymbol{X}$ 的各维度计算其平均值，假设 $\boldsymbol{x}=(x_1,x_2,\cdots,x_m)^{\mathrm{T}}$ 为 m 维随机变量，其均值向量为 $\boldsymbol{\mu}=E(\boldsymbol{x})=(\mu_1,\mu_2,\cdots,\mu_m)^{\mathrm{T}}$，而后每个元素减去所在列的平均值得到 $\boldsymbol{x}-\boldsymbol{\mu}$；

② 求出协方差矩阵 $\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}=E[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}]$，并求得协方差矩阵的特征值及其对应的特征向量；

③ 取最大的 k 个特征值对应的特征向量$(w_1,w_2,\cdots,w_k)$组成投影矩阵 $\boldsymbol{W}$，$\boldsymbol{Z}=\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}$。

(2) 线性判别分析。

主成分分析通常用于无标签数据的降维，而线性判别分析则是对于已知类别的数据进行判别，从而对于新样本进行分类。线性判别分析将训练样本投影到低维空间，使得同类别的投影点尽可能接近，而不同类别的投影点尽可能相互远离，以最大化类间差异，最小化类内差异，使降维后的数据更容易被区分。

考虑两类样本的情况，对于两个类 C_1 和 C_2，找到向量$\boldsymbol{w}$ 定义的方向，使得当数据投影到$\boldsymbol{w}$ 上时，来自两个类的样本尽可能分开，可以使用费希尔线性判别式：

$$J(\boldsymbol{w})=\frac{(m_1-m_2)^2}{s_1^2+s_2^2}$$

其中：m_1 和 m_2 表示投影后 C_1 和 C_2 类样本的均值；s_1 和 s_2 表示投影后的散布。为了

使不同类能被很好地分开,因此希望 $|m_1-m_2|$ 尽可能大,而 $s_1^2+s_2^2$ 尽可能小,即最大化 $J(\boldsymbol{w})$。对于两类以上的情况,则希望能够找到投影矩阵 $\boldsymbol{W}$,使得 $J(W)$最大化。

$$J(W)=\frac{|\boldsymbol{W}^{\mathrm{T}}\boldsymbol{S}_B\boldsymbol{W}|}{|\boldsymbol{W}^{\mathrm{T}}\boldsymbol{S}_W\boldsymbol{W}|}$$

其中:$\boldsymbol{S}_B$ 为类间散布矩阵;$\boldsymbol{S}_W$ 为类内散布矩阵。

(3) 多维标度分析。

多维标度分析是一种通过直观空间图表示数据的感知和偏好的传统降维方法,将高维数据转化为低维数据后进行定位、分析以及归类,并通过可视化手段,直观了解降维后数据的分布情况。多维标度分析可分为计量多维标度分析和非计量多维标度分析,计量数据即按间隔尺度或比例尺度所测定的数据,非计量数据则是按顺序尺度确定的数据。多维标度分析以数据点间的某种距离关系,如欧氏距离、相似系数或亲疏程度等,利用成对样本间的相似性,构建合适的低维空间,将数据在低维空间中给出标度或者位置,从而再现各数据点间的相对关系,并使得在低维空间中的样本距离与高维空间中的样本的相似程度尽可能保持一致,将降维引起的变形降至最小。例如使用多维标度分析,可通过已知各城市间的距离数据,将之转化至二维平面上,以反映其真实的地理位置,同时也可以直观地观察各城市间的相对距离关系。

主题模型是文本挖掘领域重要的分析内容,能够挖掘文本背后隐含的信息,其中概率潜在语义分析(probabilistic latent semantic analysis,PLSA)和线性判别分析是很有影响力的识别大规模文档集或语料库(corpus)中潜藏主题的统计方法。线性判别分析作为常用的非监督文档主题生成方法,用来识别大规模文档集或语料库中潜藏的主题信息。一篇文档通常包含多个主题,每个主题在文档中的重要性也不一样。线性判别分析可以估计给定文档集合的主题分布和每个主题上的词语概率分布,把一篇文档视为一个词频向量,通常文档到主题服从 Dirichlet 分布,主题到词服从多项式分布。线性判别分析实际上将词项空间描述的文档变换到主题空间,由于主题的数量可能少于关键词的数量,因此线性判别分析也是获取文档特征的降维方法。

3) 数据压缩

数据压缩应用数据编码或变换,得到源数据的归约或者压缩表示。数据压缩根据压缩后信息是否丢失可以分有损压缩(数据压缩后信息有丢失)和无损压缩。例如,小波变换(wavelet transformation)是目前比较有效的有损压缩方法。作为一种信号处理技术,小波变换可以在保留数据主要特征的同时过滤数据中的噪声,从而提高了数据处理的效率。

4) 数值归约

数值归约通过选择替代的、较小的数据表示形式减少数据量,它分为有参数和无参数两种方法。有参数的方法使用一个数学模型拟合数据,而无参数的方法包括直方图、聚类和样本抽样等。其中,直方图是一种常用的数值归约形式,它考察数值数据不同分段(等宽或等深)的频率以及数据的大致分布规律。通常,被挖掘的数据有几十万、几百万个记录,对所有数据进行处理,代价很大也没有必要,因此需要对数据进行抽样。有效的抽样要求样本有代表性,使用样本的效果与使用整个数据集的效果差不多。不同的抽样方法对挖掘结果的影响也不同。

5) 离散化和概念分层

原始数据可以用区间或者高层的概念替换。离散化是连续值属性归约的常用方法,它

可以减少属性的取值个数，决策树等算法是必不可少的步骤，划分区间是连续属性离散的常用方法（具体内容请参考 5.5 节）。此外，连续属性离散化也用于概念分层，允许对多个抽象层上的数据进行挖掘。除决策树外，多层关联分析也可以在不同的概念分层上挖掘关联规则。

5.4 聚类分析

在自然科学和社会科学中，存在着大量的聚类（clustering）问题。通俗地说，类是指相似对象的集合。聚类分析是数据挖掘中的一种重要方法，在银行、零售和保险等领域都有着广泛的应用。聚类分析既可以作为一个独立的方法透视数据分布，也可以作为其他分析方法的预处理步骤。

5.4.1 聚类的概念

聚类是把对象或样本的集合分组成为多个簇（类）的过程，使同一个组中的对象具有较高的相似度，而不同类的对象差别较大。相异度是根据描述对象的属性值进行计算的，距离经常采用相异度度量方式。在许多应用场合，可以把一个簇中的对象作为一个整体对待。与分类、回归分析等不同，聚类的每个样本都没有类标号，因此一般是无监督（unsupervised）方法。

在数据挖掘领域，聚类分析已经被广泛应用，其应用领域包括模式识别、图像处理和市场研究（市场细分、客户群细分）等。通过聚类，人们能够识别密集的和稀疏的区域，进而发现全局的分布模式。

目前已出现多种聚类方法：基于划分的方法、基于层次的方法、基于密度的方法、基于网格的方法、基于模型的方法以及模糊聚类等。聚类方法的选择取决于数据的类型、聚类目的和应用场合。一个好的聚类方法可以产生高质量的聚类结果，这些类有高的类内相似性和低的类间相似性。一般地，聚类分析需要有良好的可伸缩性，能够处理不同类型的属性，发现任意形状的类。此外，聚类分析应该有效地处理噪声数据、异常数据和高维数据，产生满足用户指定约束的聚类结果，并且聚类结果是可解释、可理解和可用的。

5.4.2 聚类分析的统计量

聚类分析可表示为给定 n 个待聚类的对象（也称为样本）组成的集合 $S=\{t_1,t_2,\cdots,t_n\}$ 和整数值 k，聚类问题就是定义一个映射 $f:S\rightarrow\{1,\cdots,k\}$，其中第 i 个对象 t_i 被映射到第 j 个簇中。第 j 个簇 K_j 由所有被映射到该簇中的对象组成，即 $K_j=\{t_i\mid f(t_i)=j,1\leqslant i\leqslant n,1\leqslant j\leqslant k,t_i\in S\}$。

通过引进一些表示样本间相似程度的度量标准把性质相似的对象归为一类，这些度量标准称为聚类统计量。最常用的聚类统计量可分为距离和相似系数等，这些统计量处理数值型数据比较有效。

1. 距离

距离的定义有多种，对于连续值数据，可以采用欧几里得距离、曼哈顿距离、闵可夫斯基

距离和切比雪夫距离等几何距离。其中最常用的是欧几里得距离和曼哈顿距离。

假定每个样本包含有 p 项指标,如果有 n 个对象的观测数据,则

$$\boldsymbol{X}_1=\begin{bmatrix}x_{11}\\x_{21}\\\vdots\\x_{p1}\end{bmatrix},\quad \boldsymbol{X}_2=\begin{bmatrix}x_{12}\\x_{22}\\\vdots\\x_{p2}\end{bmatrix},\quad \cdots,\quad \boldsymbol{X}_n=\begin{bmatrix}x_{1n}\\x_{2n}\\\vdots\\x_{pn}\end{bmatrix}$$

每个样本可看作 p 维空间的一个点,并把 p 维空间距离相近的对象划为一类,把二维平面中两个点的距离推广到 p 维空间中,p 维空间中两个点 $\boldsymbol{X}_i$ 与 $\boldsymbol{X}_j$ 之间的欧几里得距离 d_{ij} 表示为

$$d_{ij}=\sqrt{\sum_{k=1}^{p}(x_{ki}-x_{kj})^2}$$

其中:x_{ki} 是第 i 个对象 $\boldsymbol{X}_i$ 的第 k 个维(属性)的值;x_{kj} 是第 j 个对象 $\boldsymbol{X}_j$ 的第 k 个维的值,其中 $i,j=1,2,\cdots,n$;$k=1,2,\cdots,p$。与欧几里得距离不同,曼哈顿距离可减少某一维产生大的差异而支配总的距离。

$$d_{ij}=\sum_{k=1}^{p}|x_{ki}-x_{kj}|$$

如 $\boldsymbol{X}_1=(2,1)$ 和 $\boldsymbol{X}_2=(5,3)$ 表示二维空间的两个对象,则它们的欧几里得距离是 3.61,曼哈顿距离为 5,如图 5.6 所示。

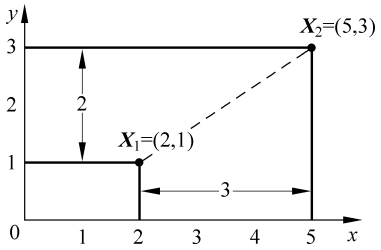


图 5.6 二维空间中的欧几里得距离和曼哈顿距离

如果对象包含其他数据类型的属性聚类,如可分类变量、二元变量、标称变量(nominal variables)、序数型变量和文本等,需要设计相应的距离公式,对象之间总的距离可由不同类型属性的距离加权和求得。

对于可分类变量,常见的对象之间距离可用 Jaccard 系数或 Dice 系数计算,其中 Jaccard 系数是两个对象共有的可分类属性的个数与两个对象属性个数和(属性集并集的元素个数)的比值。而 Dice 系数是两个对象共有可分类属性的个数与两个对象属性个数的平均值的比值。

二元变量只有两种状态 1 或 0,例如某病人的属性发烧与否。根据这两种状态的权重是否相同,二元变量分为对称的二元变量和非对称的二元变量,前者表示二元变量的两个状态优先权相同,例如性别有男女两种取值。而非对称的二元变量的两个状态出现的概率不同、重要程度不同(通常比较重要的状态用 1 表示),因此在距离计算上有别于对称的二元变量。例如病人的属性咳嗽比较重要,用状态 1 表示(不咳嗽用 0 表示)。为便于计算两个对象 $\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$ 的距离,构造下面的可能性矩阵。

$\boldsymbol{X}_1\backslash\boldsymbol{X}_2$	1	0
1	a	b
0	c	d

其中:a 表示对象 $\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$ 值都为 1 的属性的个数;b 是在对象 $\boldsymbol{X}_1$ 中值为 1,在对象 $\boldsymbol{X}_2$ 中值为 0 的属性个数;c 是在对象 $\boldsymbol{X}_1$ 中值为 0,在对象 $\boldsymbol{X}_2$ 中值为 1 的属性个数;d 是在对象

$\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$ 中值都为 0 的属性个数。如果对象 $\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$ 的属性都为对称的二元变量，它们之间的距离可用下面的简单匹配系数计算。

$$d(\boldsymbol{X}_1,\boldsymbol{X}_2)=\frac{b+c}{a+b+c+d}$$

对于属性为非对称的二元变量的两个对象 $\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$，采用 Jaccard 系数度量两对象之间的距离。

$$d(\boldsymbol{X}_1,\boldsymbol{X}_2)=\frac{b+c}{a+b+c}$$

【例 5.2】 比较表 5.5 中三个包含非对称的二元变量对象的距离

表 5.5 病人的数据

对象	a_1	a_2	a_3	a_4	a_5	a_6
Jack	1	0	1	0	0	0
Mary	1	0	1	0	1	0
Tim	1	1	0	0	0	0

表 5.5 中的 $a_1 \sim a_6$ 是对象 Jack、Mary 和 Tim 的属性，都为非对称的二元变量。它们之间的距离用 Jaccard 系数计算分别为 d(Jack,Mary)=(0+1)/(2+0+1)=1/3；d(Jack,Tim)=2/3；T(Mary,Tim)=3/4。可见 Jack 和 Mary 的距离最小，而 Mary 和 Tim 的距离最大。

此外，由于簇之间的距离有多种解释，对于给定的两个簇，有如下常用计算方法。

(1) 单一链接(single link)。簇之间的距离由不同簇中两个最接近的样本(成员)的距离确定。

(2) 完全链接(complete link)。簇之间的距离由不同簇中两个最远样本之间的距离确定。

(3) 质心(centroid)。计算代表簇的质心，质心距离是指两个簇质心之间的距离。

2. 相似系数

对于连续型数据，常用的相似系数 C_{ij} 有夹角余弦和相关系数等。而包含分类型变量的对象之间的相似性可用匹配(取值是否相等)变量与总变量数的比率表示。

1) 夹角余弦

如果把 n 个具有 p 项指标的观察数据看成 p 维空间中的 n 个向量，则有

$$\boldsymbol{X}_1=\begin{bmatrix}x_{11}\\x_{21}\\\vdots\\x_{p1}\end{bmatrix},\quad \boldsymbol{X}_2=\begin{bmatrix}x_{12}\\x_{22}\\\vdots\\x_{p2}\end{bmatrix},\quad \cdots,\quad \boldsymbol{X}_n=\begin{bmatrix}x_{1n}\\x_{2n}\\\vdots\\x_{pn}\end{bmatrix}$$

此时，任意两个向量 $\boldsymbol{X}_i$ 和 $\boldsymbol{X}_j$ 之间的夹角 θ_{ij} 余弦表示它们之间的亲疏程度：

$$C_{ij}=\cos\theta_{ij}=\frac{\boldsymbol{X}_i\cdot\boldsymbol{X}_j}{|\boldsymbol{X}_i||\boldsymbol{X}_j|}$$

其中：$\boldsymbol{X}_i\cdot\boldsymbol{X}_j$ 表示向量 $\boldsymbol{X}_i$ 和 $\boldsymbol{X}_j$ 的内积；$|\boldsymbol{X}_i|$和$|\boldsymbol{X}_j|$分别表示向量 $\boldsymbol{X}_i$ 和 $\boldsymbol{X}_j$ 的长度。例

如,有两个变量 $\boldsymbol{X}_1=(0,0,1,1)$和 $\boldsymbol{X}_2=(1,0,1,1)$,则 $\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$ 之间的相似度为 $C_{12}=(0+0+1+1)/(\sqrt{2}\times\sqrt{3})=0.816$。在文本聚类中,文档用此向量表示,其中每个分量的值是关键词在文档出现的次数。利用夹角余弦可以计算文档的相似程度,从而能对文档进行聚类。

2) 相关系数

$\boldsymbol{X}_i$ 与 $\boldsymbol{X}_j$ 之间的相关系数 C_{ij} 为

$$C_{ij}=\frac{\sum_{k=1}^{p}(x_{ki}-\bar{x}_i)(x_{kj}-\bar{x}_j)}{\sqrt{\sum_{k=1}^{p}(x_{ki}-\bar{x}_i)^2\sum_{k=1}^{p}(x_{kj}-\bar{x}_j)^2}}$$

其中:$\bar{x}_i$、$\bar{x}_j$ 分别表示第 i、第 j 个分量的平均值。相关系数越大,表示 $\boldsymbol{X}_i$ 与 $\boldsymbol{X}_j$ 越相关。

在聚类时,描述样本用到了不同的属性,这些数据单位不同,数量级也不同,因此为了获得更好的聚类效果,在聚类前需要对这些属性进行标准化处理。

通常,并不是每个样本的属性对聚类过程都有贡献,多余的属性不仅会增加计算量,而且会影响聚类的结果。因此在聚类前也需要进行归约,常用的方法是单因素方差分析或均值描述,判断聚类结果中各类样本属性在这些指标上的差异是否显著,没有显著差异的属性会被剔除。图 5.7 是利用 SPSS Statistics 对大众点评网的餐馆进行单因素方差分析的结果,可见餐馆点评条数对餐馆聚类没多大影响,在聚类时可以不考虑。

ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
口味	Between Groups	21.520	2	10.760	125.172	.000
	Within Groups	5.330	62	.086		
	Total	26.850	64			
环境	Between Groups	33.471	2	16.736	106.459	.000
	Within Groups	9.746	62	.157		
	Total	43.218	64			
服务	Between Groups	30.026	2	15.013	126.944	.000
	Within Groups	7.332	62	.118		
	Total	37.358	64			
点评条数	Between Groups	57926345.88	2	28963172.94	1.805	.173
	Within Groups	9.947E8	62	16043612.70		
	Total	1.053E9	64			

图 5.7 属性选择的方差分析

5.4.3 常用聚类算法

最基本的聚类算法是 k-means 算法。k-means 算法比较简单,对凸形分布数据的聚类效率比较高,但这种聚类算法不能有效处理非数值型数据。然而实际应用中的非数值数据经常出现,因此有些学者对 k-means 算法进行了扩展,如 k-modes 算法和 k-prototypes 算法,以处理包含分类型属性和混合型属性的数据。k-modes 算法用一种简单的相异度测量处理可分类型数据,聚类的过程和 k-means 算法是相似的。而 k-prototypes 算法结合了 k-means 算法和 k-modes 算法的相异度测量方法处理数值型和分类型的混合数据聚类。这两种扩展的算法对大规模的数据集聚类也比较有效。下面简要介绍这几种算法。

1. k-means 算法

k-means 算法是常见的基于划分的聚类方法,其中相异度基于对象与类中心(簇中心)

的距离计算，与簇中心距离最近的对象可以划为一个簇。此算法目标是使每个对象与簇中心距离的平方和最小。

$$\mathrm{SSE}=\sum_{i=1}^{k}\sum_{t\in K_i}d(\boldsymbol{t},\boldsymbol{m}_i)^2$$

其中：d 表示对象 $\boldsymbol{t}$ 与簇中心 $\boldsymbol{m}_i$ 的欧几里得距离。由 $\frac{\partial \mathrm{SSE}}{\partial \boldsymbol{m}_i}=0$ 可以证明这里的簇中心为质心，即一组点的均值。

k-means 算法过程比较简单。首先，用户指定聚类的类别数 k，随机选择 k 个对象作为 k 个初始聚类中心。对剩余的每个对象，分别计算与初始聚类中心的距离，根据距离划到不同的簇。然后重新计算每个簇的平均值，求出新的聚类中心，再重新聚类。这个过程不断重复，直到收敛（相邻两次计算的聚类中心相同）为止或迭代次数小于设定的值。k-means 算法以 k 为参数，把 n 个对象分为 k 个簇，簇内对象具有较高相似度，簇间的对象相似度较低。k-means 算法的时间复杂度是 $O(knt)$，其中 n 是所有对象数目，t 是迭代次数。通常 k 远远小于 n，且 t 也远远小于 n，k-means 算法经常以局部最优结束，效率比较高，其缺点是对数值型的且簇呈球状分布的数据比较有效，对离群点和噪声数据非常敏感。而且这种算法不能处理非凸形（非球形）的簇和不同大小的簇。此外，初始聚类中心的选择对聚类结果影响比较大，随机选择的初始聚类中心可能会导致出现不同的迭代次数和聚类结果，通常可以用不同的初始值多次运行再确定合适的聚类结果。注意这些初始的聚类中心应相互远离。k-means 算法的大致过程如下。

（1）给定 k，从 n 个对象中任意选择 k 个对象作为初始聚类中心。

（2）重复如下过程：

- 计算每个对象与聚类中心的距离，把它们划到不同的簇；
- 重新计算每个簇的聚类中心。

直到聚类中心不再发生变化。

图 5.8 是 k-means 算法聚类过程示意图（$k=2$，△号表示质心）。

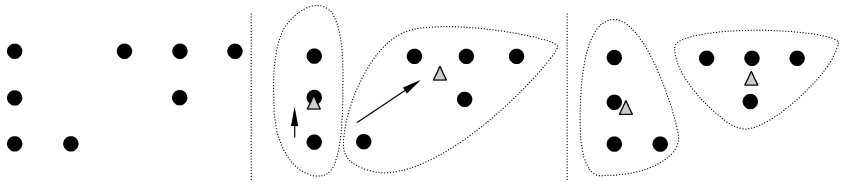


图 5.8　k-means 算法聚类过程示意图

k-means 聚类算法比较适合处理凸形分布的连续数值型数据的聚类。

k-means 聚类算法对异常点也比较敏感，因此少数异常点会极大影响聚类中心的计算，因此在预处理前去异常样本是必要的。聚类后一些样本较少的组可以合并到其他样本较多的邻近组。

每个样本维也可根据需要赋予不同的权重。k 值的选择也需要视应用而定，而不是简单地看聚类效果，例如使 SSE 最小。通常可以使用不同的 k 值进行聚类，然后利用合适的准则选择合适的 k 值。例如可以不断增加聚类数，直到满足一定的停止条件。

聚类经常与其他数据挖掘方法结合使用,例如,聚类可以作为决策树、神经网络等方法的基础,聚类的结果也可以结合数据可视化技术分析各簇的特点。此外,下面讨论的离群点对聚类的影响很大,在聚类前可以识别离群点并删除,这里离群点的检测作为预处理的一部分。

【例 5.3】 k-means 算法在安全检测中的应用

入侵检测用于识别非授权使用计算机系统的用户(如黑客)和虽有合法授权但滥用其权限的用户(如内部攻击),通过从计算机网络或计算机系统的关键点收集信息并进行分析,从中发现网络中是否有违反安全策略的行为和被攻击的迹象。现有的入侵检测系统大都采用专家系统或基于统计的方法,需要使用者有较多的经验。而数据挖掘方法可从大量数据中提取人们感兴趣的、事先未知的知识,而不过于依赖经验。应用 k-means 聚类算法,分析入侵检测模型数据库,自动地分析原有数据,从中挖掘出潜在的模式,预测用户的行为。更重要的是聚类分析能优化原有的模型。

入侵检测在很大程度上依赖于收集数据的可靠性和正确性,选择哪些数据表现用户行为是首要问题。黑客们经常在网络日志中留下踪迹,通常日志文件记录了各种行为类型,每种类型又包含不同的信息,例如,记录"用户活动"类型的日志,就包含登录、用户 ID 改变、用户对文件的访问、授权和认证信息等内容,可以充分利用这些日志数据检测入侵。因此,选择的特征应充分反映用户行为特征,数据提取难度不可太大,还要考虑学习过程的时间、用户行为的时效性等。表 5.6 以目标端口与当前连接相同次数和目标主机不同连接所占百分比作为特征,列出了 20 条网络访问记录。聚类前初始数据如图 5.9 所示,其中坐标轴横轴 x_1 为目标端口与当前连接相同的连接次数,纵轴 x_2 为目标主机不同连接所占百分比。

表 5.6 网络访问记录

序号	目标端口与当前连接相同的连接次数 x_1	目标主机不同连接所占百分比 x_2	聚类结果
1	5	0.6	正常
2	4	0.5	正常
3	25	0	攻击
4	9	0	异常
5	13	0.3	异常
6	10	0	异常
7	2	0	正常
8	2	0	正常
9	3	0.33	正常
10	5	0.55	正常
11	6	0.5	正常
12	10	0.15	异常
13	9	0	异常
14	5	0.45	正常
15	4	0.65	正常
16	4	0	正常
17	5	0.1	正常
18	6	0.2	正常
19	13	0.2	异常
20	11	0	异常

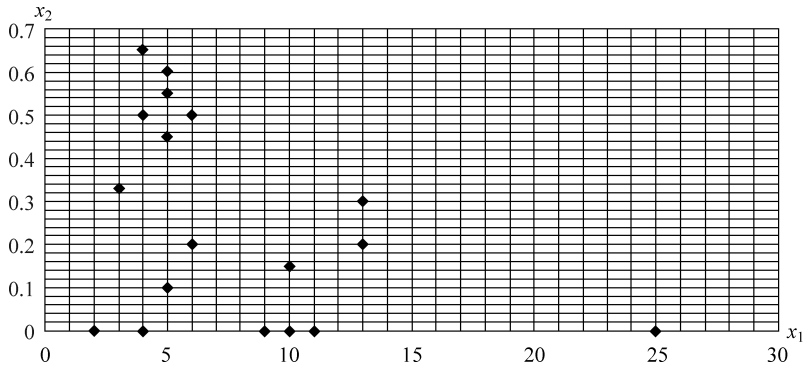


图 5.9 聚类前初始数据

程序经过 8 次迭代识别出攻击、异常和安全三种类型的样本。把该算法用于不同大小的数据集。实验表明，迭代次数总是一个小于数据集的整数，并与之保持近似线性关系。应用 k-means 聚类后，样本 3 是唯一具有攻击倾向的，而样本 4、5、6、12、13、19、20 是具有异常行为的样本，需要进一步观察。剩下的样本 1、2、7、8、9、10、11、14、15、16、17、18 是安全的。k-means 算法按特征参数的性质，把该网络行为数据集归为三类。图 5.10 是聚类的初步结果。然后对其中的异常行为进一步分析，再次应用 k-means 算法进行识别，经过 4 次迭代，聚成两类，分别由样本 4、6、12、13、20 和样本 5、19 组成。

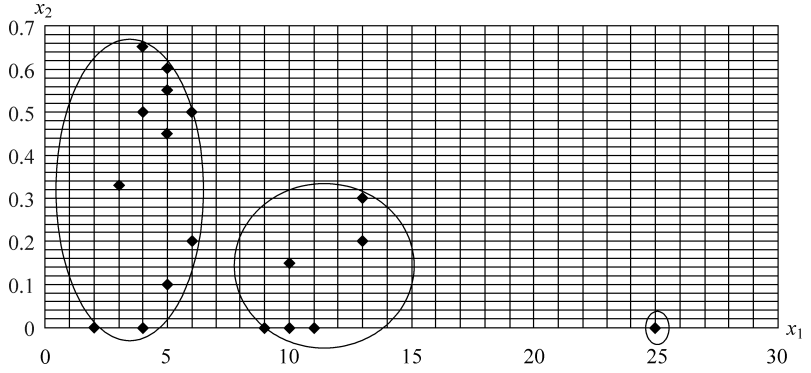


图 5.10 聚类的初步结果

图 5.11 所示为继续应用聚类算法识别异常行为的结果。对样本数据进行合理性分析，可以得出样本 4、6、12、13、20 的用户行为不具备攻击特性，可提高其安全等级。而由网络访问数据得知样本 5、19 出现 SYN 错误达 60%，同一主机连接中出现 SYN 错误超过 90%时，应予以重点监控。在这里，SYN 错误是由 SYN 攻击引起的。SYN 攻击属于 DOS 攻击的一种，它利用 TCP 协议缺陷，通过发送大量的半连接请求，耗费服务器的 CPU 和内存资源。SYN 攻击除了影响服务主机外，还会危害路由器、防火墙等网络设备。

k-means 用于预测银行客户聚类的 Python 代码如下：

```
import pandas as pd
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
```

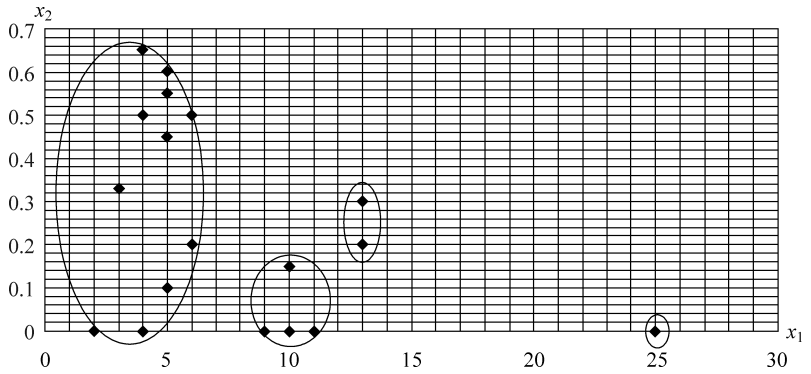


图 5.11 应用聚类算法识别异常行为的结果

```
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv("select - data.csv")
data = []
for i in range(0, len(df["EstimatedSalary"])):
    mid = []
    mid.append(df["Geography"][i])
    mid.append(df["Gender"][i])
    mid.append(df["EB"][i])
    mid.append(df["Age"][i])
    mid.append(df["EstimatedSalary"][i])
    mid.append(df["NumOfProducts"][i])
    mid.append(df["CreditScore"][i])
    mid.append(df["Tenure"][i])
    mid.append(df["HasCrCard"][i])
    data.append(mid)
data = np.array(data)
# 确定合适的 k 值
distortions = []
K = range(1, 15)
for k in K:
    kmeanModel = KMeans(n_clusters = k).fit(data)
    kmeanModel.fit(data)
    distortions.append(sum(np.min(cdist(data, kmeanModel.cluster_centers_, 'euclidean'),
axis = 1)) / data.shape[0])
# Plot the elbow
plt.plot(K, distortions, 'bx - ')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
kmeans = KMeans(10).fit(data)
kmeans.fit(data)
y_means = kmeans.predict(data)
print(y_means)
```

2. k-modes 算法

k-modes 算法改变了 k-means 算法的相异度测量方法，这种算法用一个简单的匹配相异度测量对数据进行聚类处理。k-modes 算法把 k-means 算法扩展到可分类数据，定义了新的度量可分类数据相异度的距离公式，并给出了相应的更新聚类中心的方式，能够迅速处理可分类型数据。

k-modes 算法根据可分类属性值出现的频率更新聚类中心，聚类中出现频率最高的属性值被选为聚类中心，即 modes(类模式)。但这种基于频率的 modes 更新方式也有一些问题，例如，出现频率同样高的两个属性值时就很难决定选择哪个属性值作为 modes。此外，如果作为 modes 的属性值不占绝对多数，那么用其表示聚类中心可能不太准确。

k-modes 算法改变了 k-means 算法的相异度测量方法，用一个简单的相异度测量对数据进行聚类。假设 X、Y 是数据集中的两个对象，它们用 m 维属性描述，则这两个对象之间的相异度为

$$d(X,Y)=\sum_{j=1}^{m}\delta(x_j,y_j)$$

当 $x_j=y_j$ 时，$\delta(x_j,y_j)=0$；当 $x_j\neq y_j$ 时，$\delta(x_j,y_j)=1$。

k-modes 算法不断更新 modes，使得所有对象与其最近 modes 的相异度总和最小：首先，计算每一簇在某一属性值的对象所占百分数；然后，取每个簇中频率最大的一个属性值作为类模式 Q；最后，分别对每个属性进行上述计算，得到类模式 Q，即初始聚类中心。k-modes 算法与 k-means 算法的步骤类似。

(1) 预先定义好 k 类，确定各个类的初始类模式 Q。

(2) 根据类模式 Q 把每个对象赋给最近邻的类，然后更新类模式 Q。

(3) 不断重复步骤(2)，直到不再发生变化为止。

3. k-prototypes 算法

在实际应用中，数据可能是数值型的，也可能是可分类型的。k-prototypes 算法综合了 k-means 和 k-modes 算法，采用新的距离度量方法，能够快速处理混合类型数据集的聚类问题。

k-prototypes 算法的聚类中心由数值型数据的聚类中心和可分类数据的聚类中心两部分组成，其中，数值型属性的聚类中心和 k-means 算法类似，都可通过计算数值型属性的平均值得到。而可分类型属性的中心采用类似 k-modes 算法聚类中心的更新方式，通过计算可分类属性值出现的频率确定。

在本节的最后简要介绍使用 SPSS Modeler 对客户数据进行聚类。

mailshot.txt 文件记录了某个销售商的客户基本数据，包括 300 条记录、12 个字段，如图 5.12 所示。

对客户进行聚类后，销售商可以根据每类别客户的特征设置促销活动。聚类时选择客户的 age、sex、region、income、married、children、car、save_act、current_act、mortgage 和 mailshot_YN 作为聚类的输入，聚类算法选择 k-means。聚类结果如图 5.13 所示。

表(12个字段，300条记录) #2

文件 编辑 生成

表 注解

	id	age	sex	region	income	married	children	car	save_act	current_act	mortgage	mailshot_YN
1	ID12101	48	FEMALE	INNER_CITY	17546....	NO	1	NO	NO	NO	NO	YES
2	ID12102	40	MALE	TOWN	30085....	YES	3	Y...	NO	YES	YES	NO
3	ID12103	51	FEMALE	INNER_CITY	16575....	YES	0	Y...	YES	YES	NO	NO
4	ID12104	23	FEMALE	TOWN	20375....	YES	3	NO	NO	YES	NO	NO
5	ID12105	57	FEMALE	RURAL	50576....	YES	0	NO	YES	NO	NO	NO
6	ID12106	57	FEMALE	TOWN	37869....	YES	2	NO	YES	YES	NO	YES
7	ID12107	22	MALE	RURAL	8877.0...	NO	0	NO	NO	YES	NO	YES
8	ID12108	58	MALE	TOWN	24946....	YES	0	Y...	YES	YES	NO	NO
9	ID12109	37	FEMALE	SUBURBAN	25304....	YES	2	Y...	NO	NO	NO	NO
10	ID12110	54	MALE	TOWN	24212....	YES	2	Y...	YES	YES	NO	NO
11	ID12111	66	FEMALE	TOWN	59803....	YES	0	NO	YES	YES	NO	NO
12	ID12112	52	FEMALE	INNER_CITY	26658....	NO	0	Y...	YES	YES	YES	NO
13	ID12113	44	FEMALE	TOWN	15735....	YES	1	NO	YES	YES	YES	YES
14	ID12114	66	FEMALE	TOWN	55204....	YES	1	Y...	YES	YES	YES	YES
15	ID12115	36	MALE	RURAL	19474....	YES	0	NO	YES	YES	YES	NO
16	ID12116	38	FEMALE	INNER_CITY	22342....	YES	0	Y...	YES	YES	YES	NO
17	ID12117	37	FEMALE	TOWN	17729....	YES	2	NO	NO	NO	YES	NO
18	ID12118	46	FEMALE	SUBURBAN	41016....	YES	0	NO	YES	NO	YES	NO
19	ID12119	62	FEMALE	INNER_CITY	26909....	YES	0	NO	YES	NO	NO	YES
20	ID12120	31	MALE	TOWN	22522....	YES	0	Y...	YES	YES	NO	NO
21	ID12121	61	MALE	INNER_CITY	57880....	YES	2	NO	YES	NO	NO	YES
22	ID12122	50	MALE	TOWN	16497....	YES	2	NO	YES	YES	NO	NO
23	ID12123	54	MALE	INNER_CITY	38446....	YES	0	NO	YES	YES	NO	NO
24	ID12124	27	FEMALE	TOWN	15538....	NO	0	Y...	YES	YES	YES	NO
25	ID12125	22	MALE	INNER_CITY	12640....	NO	2	Y...	YES	YES	NO	NO

YES

确定

图 5.12 某销售商的客户基本数据

聚类

输入（预测变量）重要性

1.0 0.8 0.6 0.4 0.2 0.0

聚类	聚类-5	聚类-4	聚类-1	聚类-2	聚类-3
标签					
说明					
大小	24.3% (73)	21.7% (65)	19.7% (59)	17.3% (52)	17.0% (51)
输入	married YES (100.0%)	married YES (100.0%)	married NO (100.0%)	married YES (88.5%)	married NO (66.7%)
	mailshot_YN YES (100.0%)	mailshot_YN NO (100.0%)	mailshot_YN YES (81.4%)	mailshot_YN NO (82.7%)	mailshot_YN NO (90.2%)
	mortgage NO (67.1%)	mortgage NO (100.0%)	mortgage NO (94.9%)	mortgage YES (53.8%)	mortgage YES (82.4%)
	car NO (61.6%)	car NO (69.2%)	car YES (61.0%)	car YES (100.0%)	car NO (86.3%)
	sex MALE (56.2%)	sex FEMALE (69.2%)	sex MALE (52.5%)	sex MALE (84.6%)	sex FEMALE (62.7%)
	income 31,660	income 22,794	income 29,468	income 25,674	income 25,660

图 5.13 k-means 聚类结果

对于可分类数据的聚类，还可以使用 CLOPE 算法。这种算法在市场交易数据和网络服务器日志等高维的大型数据聚类中具有速度快、节省内存、对样本数据出现顺序不敏感以及较好的可拓展性等特点。CLOPE 算法采用新的全局聚类评估函数，不需要耗费大量时间计算每对样本数据的距离，可通过选择较大的聚类分组直方图来控制样本数据分组的紧密性。有兴趣的读者可以查阅相关资料。

在聚类过程中，确定聚类数目是一个很重要的问题。由于事先不容易确定最佳的聚类数，因此需要利用一些启发式方法，例如，使用基于熵的拟合优度测度 Aakaike 信息准则(Aakaike information criterion)、基于最大似然估计的模型选择标准贝叶斯信息准则(Bayesian information criterion，BIC)或者用经验公式$(n/2)^{1/2}$ 等方法。

5.4.4 其他聚类方法

除了以上基于划分的聚类算法外，还有基于层次的聚类算法以及神经网络。层次聚类(hierarchical clustering)方法把数据组织成若干簇，并形成一个相应的树状图进行聚类。它可以分为两类：自底向上的聚合聚类和自顶向下的分裂聚类。聚合层次聚类采用自底向上的策略，首先把每个对象单独作为一类，然后根据一定的规则，例如，把簇间距离最小的相似簇合并成越来越大的簇，直到所有样本凝聚成一个大的簇，针对给定应用选择最好结果的聚类层次。与聚合型方法相反，分裂聚类采用自顶向下的方法，先把所有的对象都看成一个簇，然后不断分解直至满足一定的条件。可以看出，层次聚类的一个重要问题是如何评价两个簇的相似性。大多数层次聚类方法都属于聚合型方法，它们对噪声、异常数据比较敏感。层次聚类常用的方法有 BIRCH 和 CURE 等。

两步(two step)聚类算法也是常用的聚类方法，在 SPSS Modeler 数据挖掘工具中就有两步聚类建模组件。顾名思义，两步聚类算法由两个阶段组成：第一步是预聚类，把具有较少样本的子聚类视为离群值，生成若干子聚类；第二步利用分层聚类方法对上述子聚类进行合并，形成大的聚类。与 k-means 算法不同，两步聚类可以确定最佳聚类数：首先基于贝叶斯信息准则选择聚类数的上限，然后从聚类数更少的所有聚类模型中找出聚类间最小距离的差异，最终选择距离最大差异的聚类模型。需要注意的是聚类结果与训练数据的顺序有关。

基于划分的聚类和基于层次的聚类还有其他实现方法，如基于密度的聚类、基于网格的聚类、基于模型的聚类以及模糊聚类等，每种方法都有各自的优缺点，适用范围也有限。选择哪种聚类方法，需要综合考虑实际的应用需求、簇的类型与特征、数据的特性、数据质量、数据集的规模(样本个数、样本属性个数)等因素。

1. 基于密度的聚类

簇是对象的稠密区域。基于密度的聚类方法与 k-means 算法使用簇的中心不同，它使用密度的概念。这种聚类方法根据样本点周围的密度不断增长聚类，克服了基于距离的算法只能发现凸形分布数据聚类的缺点。基于密度的聚类方法首先计算一个区域中点的密度，如果大于某个阈值，就把它添加到相近的聚类中，主要算法包括 DBSCAN 算法和 OPTICS 算法(DBSCAN 的扩展算法)等。

DBSCAN 算法是一种常见的基于密度的聚类方法，大致过程如下。首先，把所有的样

本标记为核心点、边界点或噪声点：其中一个样本是核心点，满足在该样本的邻域(由距离函数和用户指定的参数 R 确定)内的样本的个数大于给定的阈值 min；边界点是位于某核心样本邻域的非核心样本；噪声点指既不是核心样本又不是边界样本的样本。然后，对每个样本做如下处理：删除噪声点，而足够靠近的核心点(它们的距离小于 R)聚集在同一簇中，与核心点足够靠近(它们的距离小于 R)的边界点也聚集在与核心点相同的簇中。DBSCAN 算法可以有效地发现数据库中任意形状的簇，自动确定聚类的簇个数，但也存在一定的局限性，例如，参数 R 和 min 仍然需要用户依靠经验来设置。

DBSCAN 用于聚类的 Python 代码如下：

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import DBSCAN
X1, y1 = datasets.make_circles(n_samples = 5000, factor = .6,noise = .05)
X2,y2 = datasets.make_blobs(n_samples = 1000, n_features = 2, centers = [[1.2,1.2]], cluster_
std = [[.1]],random_state = 9)
X = np.concatenate((X1, X2))
#eps 和 min_samples 需要进行调参
y_pred = DBSCAN(eps = 0.1, min_samples = 15).fit_predict(X)
#分类结果
plt.scatter(X[:, 0], X[:, 1], c = y_pred)
plt.show()
```

2. 基于网格的聚类

基于网格的聚类方法大多数是基于密度的。这种方法的基本思想是首先把样本的属性值域分成许多区间(如通过离散化处理)，这样，数据空间被划分为许多网格单元。然后计算落入每个单元的对象数目。在删除密度(单位体积的样本数)小于阈值的单元后，由邻接的高密度单元组成簇。基于网格的聚类方法处理速度比较快，但密度阈值较难确定，对高维数据的聚类效果也不理想。

在一些应用中，样本集在少数属性的子空间存在有趣的簇。CLIQUE 算法就是一种发现子空间簇的有效方法，这种算法也是基于网格的。CLIQUE 算法的基础是如果样本集在 k 维属性空间是一个满足密度阈值的簇，那么此样本集在任何小于 k 维的属性子空间中都是满足密度阈值的簇。利用这个性质，可以由高密度的低维单元逐渐生成高维的候选高密度单元，最后再把邻接的高密度单元组成簇。

3. 基于统计模型的聚类

基于统计模型的聚类假定样本集是由某种统计过程产生的，因此找出最佳拟合数据的统计模型及其参数就可以描述数据。特殊地，每个簇都对应一个统计分布，这种统计模型是混合模型，它可以发现不同大小和椭球形状的簇。期望最大化等是常见的使用最大似然估计(maximum likelihood estimation)混合模型参数的算法，它迭代改进模型参数。

4. 模糊聚类

模糊(fuzzy)聚类并不是把每个样本硬性地划分到一个簇中，尤其是对靠近两个簇边界

的对象，而是把簇看成模糊集，样本对于每个簇都有不同的隶属度值。实际应用时，一般把样本指派给具有最高隶属度的簇。模糊 c-means 是典型的模糊聚类方法。

5. EM 算法

EM(expectation maximization)算法，也称为最大期望算法，是数据挖掘常用的十大算法之一，常用于计算机视觉等领域的数据聚类，可以从非完整数据集中对参数进行最大似然估计。

EM 算法的大致步骤如下：首先，进行初始化，在缺少先验知识的情况下，通常从均匀分布开始，也可以选择随机概率作为起点。接着，在 E 步骤中计算期望，用最可能的值填补数据中的缺陷，并计算其最大似然估计值。然后，在 M 步骤中找出 E 步骤中得到的最大似然估计值的极大值，并计算参数的值。数据集在给定变量估计值后得到了扩充。可以简化为只考虑最优估计，但更精确的方法是根据概率的不同对所有可能的估计进行加权。M 步骤上找到的参数估计值被用于下一次 E 步骤的计算中，上述过程不断交替，直至收敛。

神经网络也有一些算法可用于聚类，例如 Kohonen 神经网络，又称为自组织映射网络(SOM)等。Kohonen 神经网络是一种前馈型无监督学习网络，能够根据样本的特征自动聚类。如图 5.14 所示，Kohonen 神经网络的拓扑结构由输入层和输出层组成，其中输入层的每个节点与输出层的所有节点相连，每个连接对应一个权值(组成权向量)，输出层的每个神经元与同层临近的若干神经元相连。输入层的主要功能是计算样本输入向量与权向量之间的距离，输出层的功能则是计算这两个向量之间的距离，确定样本与输出神经元的匹配程度，距离最小的输出层神经元获胜。

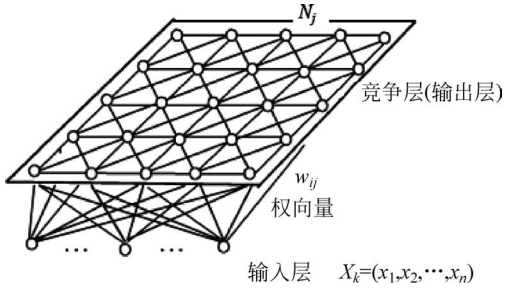


图 5.14 Kohonen 神经网络结构

在 Kohonen 神经网络的运行过程中，匹配竞争胜出的神经元及其邻近的神经元与相应输入层神经元之间的权向量朝着样本输入(特征)向量方向更新，如此经过多次迭代，这些权向量就可以对样本进行自动聚类，完成自组织学习(映射)过程。

Kohonen 神经网络算法的大致过程如下。

(1) 网络初始化。对输入层到输出层所有神经元的连接权值 w_{ij} 赋予随机的小数。

(2) 对网络的输入样本 $X_k=(x_1^k,x_2^k,\cdots,x_n^k)$，计算 X_k 与所有输出神经网络节点连接向量的距离：

$$d_j=\sum_{i=1}^{n}(x_i^k-w_{ij})^2,\quad i\in\{1,2,\cdots,n\},j\in\{1,2,\cdots,m\}$$

(3) 找出上述距离最小的输出节点 j^*：$d_{j^*}=\min\limits_{j\in\{1,2,\cdots,m\}}\{d_j\}$。

(4) 调整输出节点 j^* 连接的权值以及领域 NE_{j^*} 内输出节点的连接权值：$\Delta w_{ij}=$

$\eta(x_i^k - w_{ij})$，$j \in NE_{j^*}$，其中 η 是学习因子，随着学习的进行利用它逐渐减少权值调整的幅度。

(5) 对其他的样本，重复上述过程。

【例 5.4】 层次型聚类在网络社区发现中的应用

社会化网络作为一种复杂网络，具有显著的社区结构：相似节点聚集在同一社区，社区内部节点之间连接稠密，社区之间连接相对稀疏。目前在社会化网络分析领域，社区发现已成为一个重要的热点问题，这里以 Zachary 的 karate club 数据集为例，分析层次聚类算法在社会化网络社区发现中的应用。

Zachary 耗时两年研究了 karate club 中的 34 名成员，发现经过俱乐部管理者和教员的分析，这 34 个成员可以分成两个社区。若俱乐部成员以网络节点表示，节点之间的关系以无权重的连接边表示。karate club 网络关系如图 5.15 所示(图中数字表示成员编号)。

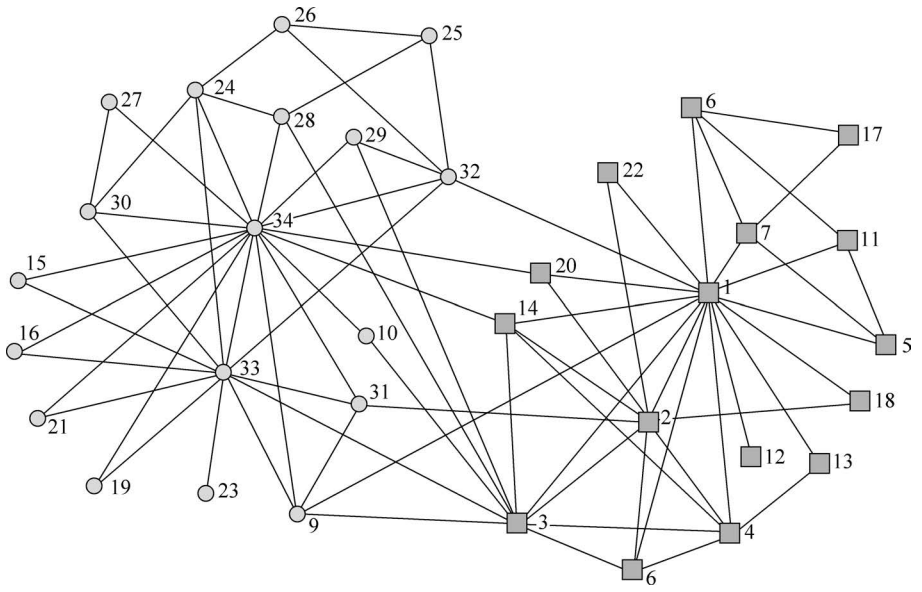


图 5.15 karate club 网络关系图

图 5.15 中○节点表示支持教员的成员，■节点表示支持管理者的成员。利用改进的层次聚类算法解决了这两个网络社区发现的问题，聚类结果如图 5.16 所示。不难发现，聚类得到的层次树状结构与 Zachary 的观察结果一致。

5.4.5 离群点检测

离群点也称为异常，是指数据集中显著不同的对象，表现为某个或者某些属性是异常的。这些对象不是随机的偏差引起的，而是来源于不同的类，让人怀疑它们产生于不同的机制。离群点与数据测量、数据收集误差导致的噪声不同，它们通常是有趣的。相应地，离群点检测的目的在于找出隐含在海量数据中相对稀疏而孤立的异常数据模式。在数据挖掘的早期，对数据集进行预处理时，通常把离群点当作噪声处理，以减少它们对数据挖掘质量的影响。然而离群点检测有时比正常数据的挖掘更有价值，离群点可能意味着十分有用的模

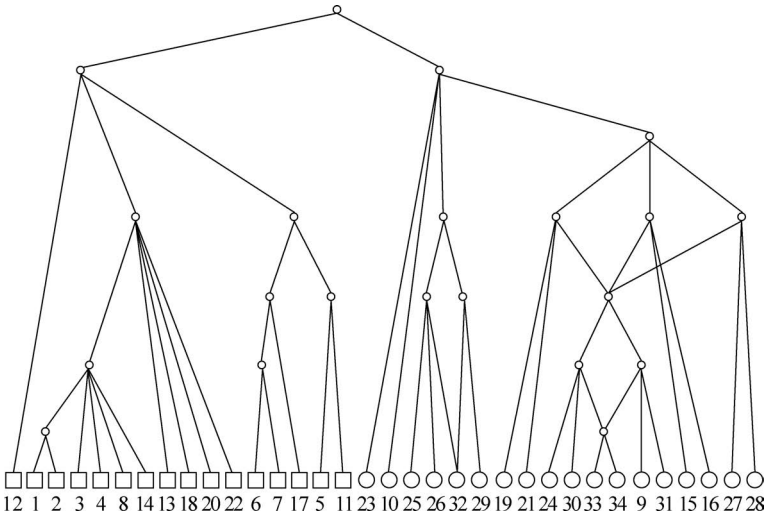


图 5.16　网络社区层次聚类结果

式。在一些应用领域中，例如，网络入侵检测、顾客流失分析、银行的信用卡欺诈、移动话费拖欠以及医学中特殊病情的征兆分析等，离群点检测具有一定的商业价值。在医保解决方案中可以对医疗保险交易中的异常进行检测，找出可疑的模式，防止保险交易中的诈骗以及医疗过程中的异常收费。

常见的离群点检测算法包括以下几种。

1. 基于统计的离群点检测

统计方法是较早的离群点检测方法。这种方法为数据集构建一个概率统计模型，如常见的正态分布、泊松分布、二项式分布或者非标准分布，其中的参数由数据求得。然后根据对象拟合该模型的情况评价它是否异常。一般而言，基于对小概率事件实现对象异常的鉴别，通过不一致检验把那些严重偏离分布曲线的对象视为离群点。例如，对于一元正态分布 $N(0,1)$，落在离分布中心 ± 3 标准差以外的样本概率很小，可以视为离群点。

2. 基于距离的离群点检测

基于距离的离群点检测，也称为基于近邻的离群点检测，其基本概念是把对象视为多维空间的点，离群点是那些数据集中与大多数对象之间的距离大于某个阈值的点，即远离大部分对象的点。基于距离的离群点检测方法与基于统计的离群点检测方法相比，不需要用户拥有任何领域的专业知识。但这种方法的计算复杂度比较高，不适合处理大型数据集。

3. 基于密度的离群点检测

基于密度的离群点检测方法的主要思想是低密度区域中的对象是离群点。这种离群点检测方法需要计算对象近邻的距离，一个对象近邻的密度则可以定义为该对象某邻域内点的个数或者对象 k 个近邻的平均距离的倒数。从某种意义上可以说，基于密度的方法是基

于距离的方法的特例。这种离群点检测方法能够发现基于距离的离群点检测方法所不能识别的一类异常,即局部异常。如果离群点位于不同密度的区域,那么就需要对密度定义进行调整。基于密度的离群点检测方法的复杂度也比较高。

此外,还可以使用分类和聚类检测离群点。把正常对象和异常对象看作不同的类,如果有充分的训练数据集,就可以挖掘异常对象的分类模型。对于不属于任何簇的对象,也可以应用聚类检测异常。

5.5 分类分析

分类也是数据挖掘的主要方法。如图 5.17 所示,分类要解决的问题是利用训练样本集获得分类函数或分类模型(分类器)。分类模型能很好地拟合训练样本集中属性集与类别之间的关系,也可以预测一个新样本属于哪一类。分类和回归都属于预测建模:分类用于预测可分类属性或变量,而回归用于预测连续的属性取值(有些书籍认为对分类型属性进行预测,对连续型属性进行估计,这里不加严格区分)。

下面简要介绍贝叶斯分类器、决策树、支持向量机和神经网络等分类算法,这些算法都是常用的有监督分类方法。

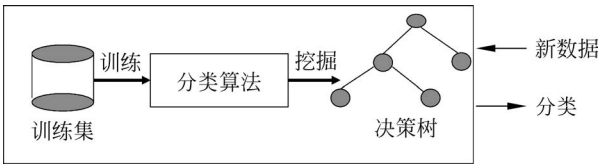


图 5.17 分类

5.5.1 贝叶斯分类器

在实际应用中,样本的属性集与类别的关系一般是不确定的,但可能存在一种概率关系。贝叶斯分类器是一种基于统计概率的分类器,通过比较样本属于不同类别的概率大小对其进行分类。这里介绍一种样本的属性集与类别的概率关系建模方法——贝叶斯(Bayes)定理及其常用的实现方法:朴素贝叶斯分类器。贝叶斯定理是由英国数学家 Tomas Bayes 提出的,它是一种把先验知识与样本中得到的新信息相结合的统计方法,在分类中得到了比较广泛的应用。

1. 贝叶斯定理

假设 X 和 Y 在分类中可以分别表示样本的属性集和类别。$p(X,Y)$表示它们的联合概率,$p(X|Y)$和 $p(Y|X)$表示条件概率,其中 $p(Y|X)$是后验概率,而 $p(Y)$称为 Y 的先验概率。X 和 Y 的联合概率和条件概率满足下列关系:

$$p(X,Y)=p(Y\mid X)p(X)=p(X\mid Y)p(Y)$$

变换后可得

$$p(Y \mid X)=\frac{p(X \mid Y)p(Y)}{p(X)}$$

上面公式称为贝叶斯定理，它提供了从先验概率 $p(Y)$ 计算后验概率 $p(Y|X)$ 的方法。在分类时，给定测试样本的属性集 X。利用训练样本数据可以计算不同类别 Y 值的后验概率，后验概率 $p(Y|X)$ 最大的类别 Y 可以作为样本的分类。

2. 朴素贝叶斯分类器

在应用贝叶斯定理时，$p(X|Y)$ 的计算比较麻烦。但对于属性集 $X=\{X_1,X_2,\cdots,X_n\}$，如果 $X_1,X_2,\cdots,X_n$ 之间相互独立，即 $p(X \mid Y)=\prod_{i=1}^{n} p(X_i \mid Y)$，这个问题就可以由朴素贝叶斯分类器来解决：

$$p(Y \mid X)=\frac{p(Y)\prod_{i=1}^{n} p(X_i \mid Y)}{p(X)}$$

其中：$p(X)$ 是常数，先验概率 $p(Y)$ 可以通过训练集中每类样本所占的比例进行估计。给定 $Y=y$，如果要估计测试样本 X 的分类，由朴素贝叶斯分类器得到 y 类的后验概率为

$$p(Y=y \mid X)=\frac{p(Y=y)\prod_{i=1}^{n} p(X_i \mid Y=y)}{p(X)}$$

只要找出使 $p(Y=y)\prod_{i=1}^{n} p(X_i \mid Y=y)$ 最大的类别 y 即可。

$p(X_i|Y)$ 的计算方法根据属性性质的不同而有所不同。

(1) 对于分类属性 X_i，可以用类 Y 中属性值等于 X_i 的样本比例来进行估计。

(2) 对于连续型属性 X_i，通常先把 X_i 离散化，然后计算属于类 Y 的训练样本落在 X_i 对应离散区间的比例估计 $p(X_i|Y)$。离散化的方法将在 5.5.2 节讨论。也可以假设 $p(X_i|Y)$ 的概率分布，如符合正态分布，然后用训练样本估计其中的参数。

朴素贝叶斯分类器简单高效，经常被用于入侵检测和文本分类等领域。这种分类模型能较好地处理训练样本的噪声和无关属性，减少对数据的过度拟合。朴素贝叶斯分类器要求严格的条件独立性假设，但属性之间一般都存在着一定的相关性。因此对于实际应用中某些属性有一定相关性的分类问题，效果往往并不理想。这个问题可以用拓展的朴素贝叶斯模型——贝叶斯网络(Bayesian network)，也称为贝叶斯信任网络来解决。

贝叶斯网络于 1988 年由 Judea Pearl 提出，已在智能系统、决策支持、故障诊断等领域得到关注。贝叶斯网络是一种描述变量之间依赖关系的图形化概率模型，基于多个变量的联合概率进行不确定性推理，常用于分类。例如，对于集装箱运输公司而言，当客户订舱后，在集装箱出运前，多种因素会导致客户中止，订舱的最终状态也就带有一定的不确定性。可以利用集装箱订舱管理系统积累的大量历史数据，通过贝叶斯网络计算订舱被中止的概率，以便预测订舱的风险。

贝叶斯网络由一个有向无环图和一个条件概率表组成，其中一个节点(属性)需要与其父节点的所有非后代节点是相互独立的。在有向无环图中，每一个节点都对应一个条件概率表，表中的数据表示在父节点出现的条件下子节点的条件概率。如图 5.18 所示，使用

SPSS Modeler 中的贝叶斯网络模型分析银行贷款客户的风险。银行贷款客户的风险受到此前是否违约(predicted defaulted)、债务收入比(debt to income ratio)、工作年限(years with current employer)、居住年限(years at current address)等因素影响,其中箭头方向表示节点之间的概率依赖关系。从图 5.18 中可以看出,这几个影响因素的重要性递减。

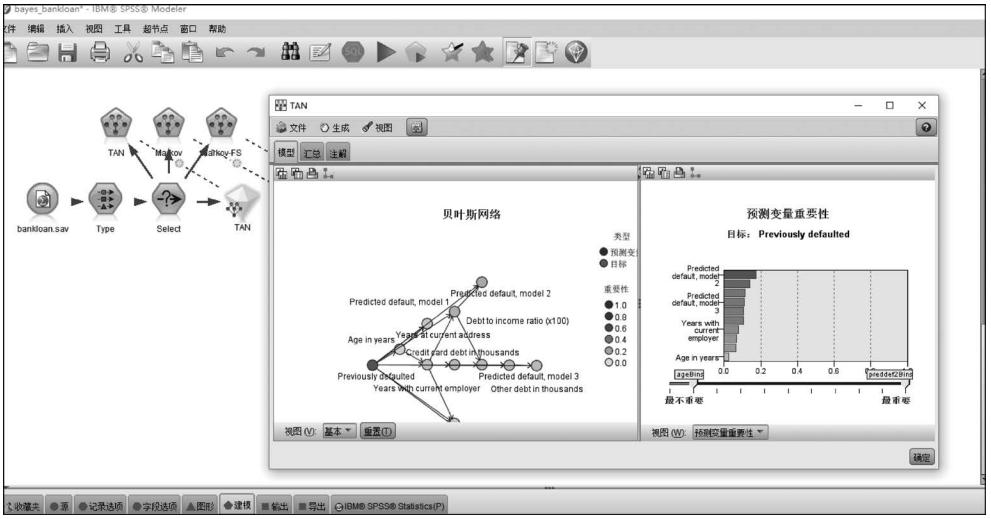


图 5.18 银行贷款客户的风险预测

与决策树、神经网络等分类算法相比,贝叶斯网络可以围绕分类变量,帮助分析多个变量之间的依赖关系,包括预测变量(条件属性)之间的关系,而不限于描述条件属性(变量)与分类属性之间的关系,但这种方法可能运算量比较大。贝叶斯网络的具体介绍读者可查阅相关资料。

【例 5.5】 贝叶斯分类器在供电电容生产中的应用

供电电容是计算机主板生产商必备的工业组件,质量好的供电电容可以提高主板的供电效率,所以供电电容的质量也就直接决定了主板的使用寿命。假设某段时期内某计算机主板制造商所用的供电电容是由三家电容生产商提供的。对制造商在这段时期内的业务数据进行抽样,得到表 5.7 所示数据。

表 5.7 电容次品率及所占份额

电容生产商标识	次品率/%	提供电容的份额/%
1	2	15
2	1	80
3	3	5

三家电容工厂的供电电容在计算机主板生产商的仓库中是均匀混合的,并无明显的区别标志。现在计算机主板生产商想通过对数据进行分析,解决下面两个问题。

(1) 随机地从仓库中取一只供电电容是次品的概率。

(2) 从仓库中随机地取一只供电电容,若已知取到的是一只次品,那么此次品来自哪家工厂的可能性最大。

假设 X 表示“取到的是一只次品”，$Y=i(i=1,2,3)$表示“取到的产品是由第 i 家工厂提供的”，则问题转化为求解 $p(X)$与 $p(Y=i|X)$。由表 5.7 得到后验概率为

$$p(X\mid Y=1)=2\%,\quad p(X\mid Y=2)=1\%,\quad p(X\mid Y=3)=3\%$$

先验概率为

$$p(Y=1)=15\%,\quad p(Y=2)=80\%,\quad p(Y=3)=5\%$$

由全概率公式计算得

$$\begin{aligned}p(X)&=p(X\mid Y=1)p(Y=1)+p(X\mid Y=2)p(Y=2)+p(X\mid Y=3)p(Y=3)\\&=0.02\times0.15+0.01\times0.8+0.03\times0.05=0.0125\end{aligned}$$

然后求解 $p(Y=i|X)$，根据贝叶斯定理可得

$$p(Y=i\mid X)=\frac{p(X\mid Y=i)p(Y=i)}{p(X)}$$

由上式可以计算次品出自生产厂商 1 的概率为

$$p(Y=1\mid X)=\frac{p(X\mid Y=1)p(Y=1)}{p(X)}=\frac{0.02\times0.15}{0.0125}=0.24$$

类似地可以计算次品出自其他两个厂商的概率为

$$p(Y=2\mid X)=0.64,\quad p(Y=3\mid X)=0.12$$

可见，从仓库中随机地取一只电容，如果是一只次品，那么此次品来自工厂 2 的可能性最大。

【例 5.6】 贝叶斯分类器在数字图书自动标引中的应用

随着图书馆的信息化发展，越来越多的图书、科技文献电子化。如何对这些电子资源进行快速地检索，就需要利用“中图法分类号”进行标引。采用人工分类的方式进行标引，效率比较低，难以满足实际应用的要求。机器学习技术的发展，为机器自动标引提供了有效的自动化工具。

数字资源的标题、摘要、关键词、刊载期刊或者会议名称等元数据中蕴藏了类别的重要信息，人工在标引时通常也是通过这些重要的关键词确定资源的类别的。因此，可以利用贝叶斯分类器确定数字资源的 1～2 级分类目录，通过从元数据中的关键词推理得到数字资源的类别。其中的分类属性是数字资源中对分类影响的词，而类是数字资源在“中图法分类号”的 1～2 级目录的位置。

为了计算贝叶斯分类器的先验概率，需要选择一定量的图书馆馆藏的各类数字资源作为训练语料。然后利用分词算法进行分词，删除停用词等对资源分类无用的词，并利用信息检索的方法(例如 TF-IDF)或卡方值的方法提取对分类有帮助的词，这些词有比较高的出现频率。由于不同大类的数字文献样本不平衡，因此就需要对类别较多的样本进行欠采样。

经过以上的预处理后就可以使用贝叶斯分类进行训练和测试。为了提高贝叶斯分类的性能指标，包括正确率和召回率等，需要对训练过程进行优化。例如，将训练集中预测错误的样本的权重提高以进行多次训练。为了弥补一些类的样本不足的问题，还可以采用样本增强的方法，例如将样本的一些词随机删除，作为新的样本加入训练数据。此外，还可以将检验分错的样本加入训练集，从而提升贝叶斯分类器的泛化能力。

5.5.2 决策树

决策树是由决策节点、分支和叶子节点组成的。其中每个内部节点都表示在一个属性上的测试,每个分支代表一个测试输出,而每个叶节点代表类,树的最顶层节点是根节点。沿决策树从上到下遍历的过程中,在每个节点都会遇到一个测试,每个节点上的测试输出导致不同的分支,最后到达一个叶子节点,这个过程就是利用决策树进行分类的过程。决策树可转化为一些分类规则,具有较好的解释性,从树根到树叶的每条路径都对应一个规则,例如,图 5.19 最左边的路径对应的规则如下:如果年龄<30 岁而且家庭经济情况一般,那么这种顾客不购买跑步机。因为决策树等算法属于归纳学习,所以从训练集得到的决策树并不能完全拟合所有训练样本,最终得到的分类规则置信度一般小于 1。这里的置信度表示决策树的分类正确率。

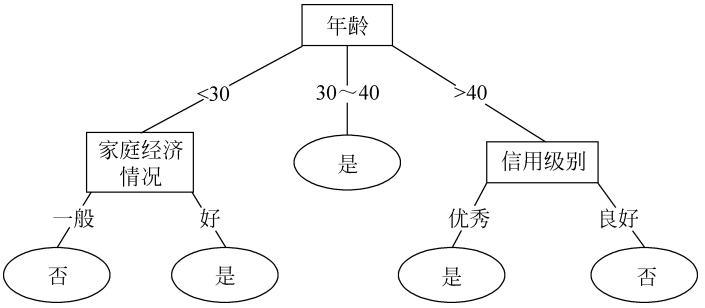


图 5.19 跑步机购买决策树

为了便于分析,决策树算法的输入一般整理成决策表(decision table)的形式。决策表的每行是一个样本(实例),每个样本用若干属性(变量)描述。这些属性分为条件属性和决策属性,其中条件属性用于描述实例,重要的条件属性可能成为决策树的分支属性,而决策属性标明每个样本的类别(可通过聚类确定)。决策树可用于与决策属性相关的重要属性分析。

1. 决策树的属性选择

给定一个决策表,可以构造很多决策树。搜索最优的决策树一般是不现实的,可以采用启发式的方式来构造次优决策树。属性选择依赖分支准则,一般不需要领域知识。属性选取是决策树算法中重要的步骤,一般需要最大程度地增加样本集的纯度,而且不要产生样本数量太少的分支。常见的属性选择标准包括 ID3 算法使用的信息增益(information gain)、Gini 指数(gini index)和 χ^2 检验等。

1) 信息增益

信息增益是决策树常用的分支准则:在树的每个节点上选择具有最高信息增益的属性作为当前节点的分支属性。这种分支的方法只关心目前分支的优化,因此属于贪婪的搜索方法。设 S 是 n 个样本的集合。假定分类属性具有 m 个不同值,定义 m 个不同类 $C_i(i=1,2,\cdots,m)$,s_i 是类 C_i 中的样本数。对给定的样本分类的期望信息:

$$I(s_1,s_2,\cdots,s_m)=-\sum_{i=1}^{m}p_i\log_2(p_i)$$

其中：p_i 表示样本属于类别 i 的概率，可用 s_i/s 估计，因此 S 中的样本要有一定的数量和代表性。设属性 A 具有 v 个不同值$\{a_1,a_2,\cdots,a_v\}$，可以用属性 A 把 S 划分为 v 个子集$\{S_1,S_2,\cdots,S_v\}$，其中 S_j 包含 S 中在属性 A 上取值 a_j 的样本。如果 A 选作测试属性，那么 A 的 v 个不同值对应各个分支。设 s_{ij} 是样本子集 S_j 中类 C_i 的样本数。由 A 划分样本子集的熵确定为

$$E(A)=\sum_{j=1}^{v}\frac{s_{1j}+s_{2j}+\cdots+s_{mj}}{n}I(s_{1j},s_{2j},\cdots,s_{mj})$$

其中：$(s_{1j}+s_{2j}+\cdots+s_{mj})/n$ 表示子集中的样本个数除以 S 中的样本总数，即第 j 个子集的权。如果熵值越小，样本子集划分的纯度就越高。给定样本子集 S_j 的期望信息：

$$I(s_{1j},s_{2j},\cdots,s_{mj})=-\sum_{i=1}^{m}p_{ij}\log_2(p_{ij})$$

其中：$p_{ij}=s_{ij}/|S_j|$，是 S_j 中的样本属于 C_i 的概率，$|S_j|$ 表示集合 S_j 中的样本数量。

A 作为分支属性的信息增益：

$$\text{gain}(A)=I(s_1,s_2,\cdots,s_m)-E(A)$$

计算每个属性的信息增益，具有最高信息增益的属性选作给定集合 S 的分支属性。创建一个节点，对该属性的每个值创建分支。

决策树的生成过程可以看成将空间用超平面进行划分，每次用某个属性分割时，都把当前的空间分为该属性取值的种数。

ID3 用于鸢尾花的分类 Python 代码如下：

```
from sklearn import datasets                                    # 导入方法类
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
iris = datasets.load_iris()                                     # 加载鸢尾花 iris 数据集
iris_feature = iris.data                                        # 属性数据
iris_target = iris.target                                       # 分类数据
#把样本集分割成训练集和测试集(20%比例)
feature_train, feature_test, target_train, target_test = train_test_split(iris_feature,
iris_target, test_size = 0.2,
random_state = 42)
dt_model = DecisionTreeClassifier()                             # ID3 决策树
dt_model.fit(feature_train, target_train)                       # 使用训练集训练模型
predict = dt_model.predict(feature_test)                        # 使用模型对测试集进行预测
print("正确率:",accuracy_score(predict, target_test))          # 测试样本的正确率
```

2) Gini 指数

Gini 指数是一种不纯度函数(impurity function)，已用于分类回归树(classification and regression trees，CART)等分类算法，适合可分类型和数值型数据的分类。不纯度函数是用来度量数据集中的数据关于类的“纯度”或同质性的，度量每个属性的二元划分：对于可分类型属性，取产生最小 Gini 指数的子集作为该属性的分支子集；对于连续型属性，选择产生最小 Gini 指数的分割点作为该属性的分支点。如果数据均匀地分布于各个类中，则数

据集的不纯度就大。反之,数据集的不纯度就小。当根据属性的不同取值拆分数据集时,会导致数据集不纯度的减小。Gini 指数首先计算各个属性的纯度增量,然后选取纯度增量最大的属性拆分数据集。

3) χ^2 检验

1900 年,Karl Pearson 提出了使用 χ^2(卡方)作为度量统计学显著性的方法。1975 年,John A. Hartigan 把 χ^2 用于 CHAID(chi-squared automatic interaction detector,卡方自动交互检测)算法,作为决策树的分支标准。较高的 χ^2 值表示用某属性拆分决策树时,可以把样本集拆分为有显著分布差异的分组。

χ^2 检验可以用于分析两个变量之间是否存在关系,因此常用于特征提取或特征降维。χ^2 检验在分类资料统计推断的应用中,χ^2 值表示观察值与理论值之间的偏离程度,偏离程度越大,说明两个变量之间的相关性越大:

$$\chi^2=\sum_{i=1}^{k}\frac{(Y_i-e_i)^2}{e_i}$$

其中:Y_i 和 e_i 分别表示变量 Y(对应后面分类问题的分类属性)在第 i 种情况下(i 对应下面分类问题的类别数)的实际值和理论值(样本数)。例如分析客户性别对某类产品的营销响应是否有影响。假设响应类别的客户有 110 人,其中男性和女性客户分别占 15 人和 95 人;未响应类别的客户有 90 人,其中男性客户和女性客户分别有 85 人和 5 人。根据上面 χ^2 值的计算公式,很容易计算得到 χ^2 值为

$$\chi^2=\frac{(95-55)^2}{55}+\frac{(15-55)^2}{55}+\frac{(85-45)^2}{45}+\frac{(5-45)^2}{45}=129.3$$

假设检验后说明客户性别对营销响应的影响很大。类似地,通过比较各种影响营销响应的变量的 χ^2 值,选择其中 χ^2 比较大的变量可以作为分类的特征。

CHAID 算法由 Kass 在 1980 年提出,是一种为了达到目标最优,通过目标选择、变量筛选和聚类等手段对序次等级数据和分类数据进行分析的方法。CHAID 算法的核心思想是根据结果变量与解释变量对样本进行最优分割,按照 χ^2 检验的结果进行多元列联表的自动判断分组。

CHAID 算法是在决策树算法中常用的算法之一。与 QUEST(quick unbiased efficient statistical tree,快速无偏高效统计树)等决策树算法比较,CHAID 算法可以生成非二叉树,每个树节点可以有两个以上的分支。CHAID 算法自动把数据拆分为无遗漏的、互斥的组群,最终输出一个直观的决策树。CHAID 算法细分的样本由多个属性变量共同描述,因此该方法属于多变量分析方法。此外,CHAID 算法适用于可分类数据的分析,例如地理位置的种类、工作种类等数据。

CHAID 算法的具体步骤如下。

(1) 计算决定合并类别的统计量 P 值。P 值计算的方法根据目标变量的类型确定。如果目标变量是连续型数据,则采用 F 检验方法。如果目标变量是分类数据,则建立一个交叉分类表,其中属性变量的类别作为行,目标变量的类别作为列,采用 χ^2 检验的方法,此时 P 值为 χ^2 值。

(2) 找到 P 值最小的两个属性类别,并把该 P 值与预先设定的合并临界点 α_{merge} 比较。如果 P 值小于 α_{merge},则把两个类别合并形成一个新的类别,重复该步骤。

(3) 对于那些新合并的包含了三个或三个以上原始类别的类别,通过 P 值判断是否需要再拆分成两组。把该 P 值与预先设定的分裂临界点 α_{split} 比较,如果 P 值大于 α_{split},则把该类别拆分成两个类别,返回步骤(2)。

(4) 执行步骤(2)和步骤(3),直到满足停止条件,得到决策树。停止条件包含以下几种情况:如果决策树的层数已经达到指定深度,则停止生长;对于父节点,如果节点的样本量已低于最少样本量限制,则不再分组;对于子节点,如果分组后生成的子节点中的样本量低于最小样本,则不必进行分组;当输入变量与输出变量的相关性小于一个指定值,则不必进行分组。

CHAID 算法能够较好地处理缺失值、非线性的数据,同时容易解释结果、易于掌握,所以常用作市场分析,也可用于生物学研究、居民卫生服务和人力资源分析等领域。

2. 连续属性的离散化方法

决策树一般不适合处理连续型的属性,这类变量在决策树分析前需要进行离散化。在离散化时,需要注意离散化后的属性取值对应一定的业务含义,防止信息丢失。

离散化是把连续型属性按一定标准划分为几个离散(分类型)值的过程:首先确定需要多少个离散值,再考虑如何把连续属性映射到这些离散值。因此连续属性离散化实际上是选择分割点的个数和确定分割点位置的问题。

离散化可以把连续属性分为若干区间,用不同符号映射每个区间的数值,减少连续属性的取值个数,便于后续分析。一般来说,离散化越细,得出的决策树就越复杂,预测的正确率可能越高,但会造成决策树分支过多以及计算量的增加。

如果不使用样本的类别信息,则离散化比较简单。例如,等宽法把连续属性的值域划分为相同宽度的区间,等深法则使每个区间包含相同数量的样本。前面讨论的 k-means 等聚类方法也可以用于离散化。上述这些离散化方法没有考虑样本的类别,因此离散后的区间常包含不同类的样本。为了进一步提高区间的样本纯度,可以考虑样本的类别信息,最常用的离散化方法之一是基于信息熵的方法。

在决策树的生成过程中,对于连续属性的离散化,通常是把连续属性的值域分割为两段,即二元离散化。设存在一个由 n 个样本组成的集合 S,对于某连续值属性 A,离散化的大致过程如下。

(1) 排序。按连续型属性值的增序排列,得到属性值序列 $v_1, v_2, \cdots, v_n$。

(2) 生成候选分割点。任何位于 v_i 和 v_{i+1} 之间的分割点都能把 S 中的样本划分为两类,这样有 $n-1$ 种可能的分割点。通常生成候选分割点的方法是选择上述序列中相邻每对值的中点,第 i 个候选分割点为 $(v_i + v_{i+1})/2$。当然,也有其他的候选分割点生成方法,例如,第 i 个分割点选择不超过上述中点的最大取值。

(3) 候选分割点的评价。对步骤(2)产生的候选分割点进行评价,从中选择一个最好的分割点:候选分割点把属性取值分成两个区间,每一区间都有一些不同类别的样本。参考

前面信息熵的公式,计算每一个候选分割点划分产生的信息熵,选择产生最小信息熵的点作为分割点。

(4) 取信息熵较大的区间,重复分割过程,直到区间(对应离散值)的个数达到用户指定的要求为止。

3. ID3 决策树算法

ID3、C4.5 和 CART 等都是常用的决策树算法,它们以自顶向下递归的方式构造决策树,这里主要介绍 ID3 算法。对于训练样本集,ID3 算法通过上述方法计算信息增益选择各分支属性,以信息增益最大为分支标准。再对各分支的训练样本递归建立决策树,最后得到一棵多层的决策树。这里需要注意的是,分支停止的几种条件:除算法规定的条件外,某一分支样本的数量少于设定的值,或者树的深度达到某一预设的值时,也可停止分支。

ID3 算法也存在一些问题,例如,计算信息增益时可能偏向取值种类较多的属性,一些属性在决策树构建过程中被检验多次等。C4.5 算法继承了 ID3 算法的优点,算法的基本过程与 ID3 算法相似,但在选择决策树的分支属性时用信息增益率选择属性,弥补了选择属性时信息增益偏向选择取值种类较多的属性的不足。

属性 A 信息增益率 gain_ratio(A)的定义为

$$\text{gain\_ratio}(A)=\frac{\text{gain}(A)}{-\sum_{i=1}^{v}p(a_i)\log_2 p(a_i)}$$

其中:v 为属性 A 的不同取值 a_i 的个数,从中可以看出,当 v 比较大时,就会降低增益率,从而在一定程度上解决了 ID3 算法的上述问题。

与 ID3 算法相比,C4.5 算法在效率上有了很大的提高,生成的决策树分支也较少。但 C4.5 算法在选择分支属性时仍然依据信息熵,因此生成的决策树仍然是多叉树,而不是结构较为简单的二叉树。此外,C4.5 算法没有考虑属性之间的联系。

为了简化决策树的生成,提高生成决策树的效率,可以使用其他的决策树算法,典型的有 CART 算法、SLIQ 算法和 SPRINT 算法等。

4. CART 算法

CART 是一种有效的非参数分类和回归方法,通过构建二叉树来形成,包含特征选择、树的生成和剪枝等步骤,既可以用于分类,也可以用于回归。如果待预测结果是离散型数据,则 CART 生成分类决策树;如果待预测结果是连续型数据,则 CART 生成回归决策树。数据对象的属性可以为离散型或连续型。

CART 对回归树选择能得到最小平方误差值的属性作为分裂标准,对分类树选择具有最小 Gini 指数的属性作为分裂属性。按照每个节点的分裂属性,使用二元递归分割的方法将其分割为两个子节点,由此递归下去,形成一棵结构简洁的二叉树。

1) 分类树的生成

分类树用 Gini 指数选择最优属性(特征),同时决定该属性的最优切分点。

(1) 对于有 m 个类别的样本集 S,类别集为$\{C_1, C_2, \cdots, C_m\}$,每个类别对应一个样本

子集 $S_i(1\leqslant i\leqslant m)$。该样本集的 Gini 指数为

$$\mathrm{Gini}(S)=1-\sum_{i=1}^{m}p_i^2$$

其中：$p_i=|C_i|/|S|$为样本属于类别 C_i 的概率，其中$|S|$为样本集 S 的样本数，$|C_i|$为样本集中 S 属于类 C_i 的样本数。

(2) 对于二元分类问题，根据训练样本集 S 中的属性 A 将 S 分为子集 S_1 和 S_2，给定划分样本集 S 的 Gini 指数为

$$\mathrm{Gini}_A(S)=\frac{|S_1|}{|S|}\mathrm{Gini}(S_1)+\frac{|S_2|}{|S|}\mathrm{Gini}(S_2)$$

其中：$|S_i|/|S|$为根据属性 A 划分样本集 S 时第 $i(i=1,2)$子集占整个样本的权值。

对于连续型属性，分类树的生成步骤如下。

(1) 按照分割阈值将属性集中的每个属性离散化，计算每个属性的 Gini 指数，选择 Gini 指数最小的属性作为当前节点的分裂属性。

(2) 将整个样本集分为小于或等于分割阈值和大于分割阈值两部分。

(3) 根据分类结果，对样本子集 S_1、S_2 采用与步骤(1)相同的方法递归地建立树的子节点并循环递归计算，直到所有的子节点中的样本属于同一个类别或者没有可用的分类属性为止。

对于离散型属性，分类树的生成步骤如下。

(1) 对于属性集中的每个候选属性，对其可能的取值，根据样本点对该取值为是或否将样本集分成两部分，计算每个子集的 Gini 指数，并选择 Gini 指数最小的取值对应的属性取值作为决策树的分支条件。

(2) 根据分类结果，对样本子集 S_1、S_2 采用与步骤(1)相同的方法递归地建立树的子节点，直到所有的子节点中的样本属于同一个类别或者没有可用的分类属性为止。

【例 5.7】 银行特约商户分析中交易金额的离散化

近年来，中国银行卡产业规模继续高速增长，但仍存在一定的发展瓶颈，主要体现在商家对银行卡的认同度较低，特约商户的数量少。银行既要增加特约商户的数量，也要保证特约商户的质量，发展高消费的特约商户。因此需要对特约商户的交易情况进行分析，建立数据挖掘模型，分析特约商户的信用卡交易记录，进行特约商户分类，积极发展效益高的商户。

特约商户的分类采用决策树进行，首先进行数据预处理，对连续型属性离散化。这里讨论应用 Gini 指数对连续属性进行离散化。

一条典型的信用卡交易记录包括发卡行、收单行、交易时间、交易金额、商户名称、商户行业和是否为特约商户等属性，在所有属性中，发卡行和行业是分类型属性，交易金额和交易时间需要进行概念分层，可以通过求取分割点来确定，例如，确定“顾客是否在特约商户消费”时，若选择交易金额 2000 元作为分割点，会把交易记录分成低于 2000 元和高于 2000 元两组。理想的情况是顾客低于 2000 元的消费均不是由特约商户完成，高于 2000 元的消费均是特约商户所为。但事实上，即使是最佳的分割点也很难实现这种理想情况。因此在这种情况下，属性取值会呈现一定的分布，最佳的分割点应使每组内的取值差异最小，不同组

属性取值差异越大越好。

测定属性差异程度的指标通常称为差异系数,差异系数越小,说明属性的取值越集中,分割点越理想。差异系数可以采用 Gini 指数。它的数学定义为 $1-(p_1^2+p_2^2)$,其中,p_1 是从样本集中随机抽取一个数据,属性 A 取某一离散值的概率,p_2 是取另一类值的概率,满足 $p_1+p_2=1$。p_1^2 是从训练集中随机抽取第二个数据,属性 A 仍为某类值的概率,以此类推,p_2^2 是按照同样的方法取另一类值的概率。因此确定交易金额最佳分割点的步骤如下。

(1) 计算训练集的 Gini 指数。在 13 822 条交易记录中有 9205 条是与特约商户有关的,4617 条与特约商户无关,则

$$\text{Gini 指数}=1-\left[\left(\frac{9205}{13\,822}\right)^2+\left(\frac{4617}{13\,822}\right)^2\right]=0.4449$$

(2) 指定某属性的某个取值为分割点,对样本数据进行分组后分别计算各组的 Gini 指数,并计算各 Gini 指数的加权平均值。指定交易金额分割点为 1800 元,它把顾客消费分为两组:交易金额大于 1800 元和小于 1800 元,计算两组的 Gini 指数分别为 0.4828 和 0.4438。加权平均值=(302×0.4828+13 520×0.4438)/13 822=0.4447。

(3) 计算该层的加权平均值与根节点 Gini 指数之间的差值:0.4449−0.4447=0.0002。

(4) 返回步骤(2),取分割阈值为 500 元、1000 元、1500 元,分别计算它们与总 Gini 指数的差值为 0.0015、0.0016 和 0.0004。

(5) 选差值最大的分割点为最佳分割点。从以上计算得出最佳分割阈值为 1000 元,因此选 1000 元作为交易金额的最佳分割点。由于交易金额小于 1000 元的交易数量众多,因此再对小于 1000 元的交易进行类似的离散化,如图 5.20 所示。

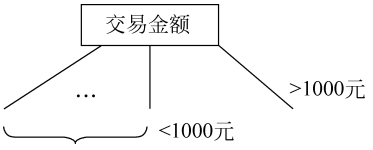


图 5.20 交易金额的离散化

2) 回归树的生成

回归树用平方误差最小化值选择最优特征,同时决定该特征的最优二值切分点。平方误差最小化值的计算过程如下。

对于样本集 S,其自变量集合为 X,因变量为 Y。把第 j 个变量 $x_j \in X$ 分成 R_1 和 R_2 两个区域:

$$R_1(j,s)=\{x \mid x_j \leqslant s\},\quad R_2(j,s)=\{x \mid x_j > s\}$$

其中 s 为 x_j 的某个取值,对应切分点,$j=1,2,\cdots,n$。寻找最优切分变量 x_j 和切分点 s 进行分支:

$$\min_{j,s}\left[\min_{c_1}\sum_{x_{ij}\in R_1(j,s),y_i}(y_i-c_1)^2+\min_{c_2}\sum_{x_{ij}\in R_2(j,s),y_i}(y_i-c_2)^2\right]$$

其中 c_1 和 c_2 分别是区域 R_1 和 R_2 上变量 x_j 所有取值对应的输出 y 的均值,x_{ij} 是变量 x_j 的第 i 个取值,y_i 是对应的因变量取值。

采用与上述步骤类似的方法递归地对 R_1 和 R_2 进行分支,直到树的预测误差或树的深度达到合适的值。

3) 树的剪枝

为了避免因噪声数据使得生成的决策树偏复杂,出现过拟合的结果,可以使用修剪的做法:使用松散的停止标准,并且在生长阶段之后,通过剪去泛化精度弱的子分支,将过度拟合

的树精简为较小的树。

大多数的决策树算法在分支时考虑一个属性，但最佳的分支标准可能是某些属性组合。有些决策树算法对此问题进行了探讨，但这种方式可能会带来算法的复杂性问题。

5. 决策树算法的可伸缩性

ID3、C4.5 等算法对规模较小、可以一次放入内存的训练样本集很有效，但实际上数以百万计样本的超大型训练集是常见的，大多数情况下无法把训练样本集全部放入内存，导致算法的有效性降低。因此需要增加可伸缩的方法以节省空间。

6. 决策树的过拟合和修剪

每种算法都有一定的局限性，过拟合(overfitting)问题是决策树的一个难题。在决策树中，有时挖掘训练样本集构造决策树无法达到较好的泛化性能，特别是当训练样本集中有异常或噪声时，或训练样本集的数量太少以至于不能产生有代表性的采样时，都可能会导致过拟合。此外，这些噪声可能导致样本冲突，例如，有两个样本具有相同的属性描述，但它们的分类却不同。当属性的描述不完备，或属性值不足以判别分类时，也会导致样本冲突。样本冲突必然会导致挖掘得到的决策树对训练样本拟合不足，即不能完全拟合数据。事实上，当训练样本集没有噪声干扰时，过拟合也有可能发生，特别是在样本集中包含的某类样本数量比较少的情况下。在决策树中，为这类样本提取的规则(对应决策树路径较长的分支)涉及很多属性，很难简化为覆盖率高的规则。此外，在决策树递归分支的过程中，一些分支样本的数量可能太少，使得进一步划分失去统计意义。可以事先给定一个阈值，当某分支的样本数少于该阈值就停止划分。很多数据挖掘软件都允许用户设置这样的阈值。例如 SPSS Modeler 允许用户设置“预期噪声”参数控制噪声数据对决策树质量的影响。

决策树的修剪是针对过拟合问题提出来的，修剪通常利用统计方法删除最不可靠的分支，以满足最小描述长度的要求，提高分类识别的鲁棒性，其实质是消除训练集中的噪声。通常采用两种方法进行决策树的修剪，即事前修剪和事后修剪。事前修剪是判断当前节点是否继续分支，而事后修剪则是在构建决策树结束后再进行修剪，但计算的工作量比较大。事前修剪需要设置阈值确定某个节点是否需要继续分支，这个阈值难以确定，通常事后修剪相对事前修剪更常用。此外，也可以交叉使用事前修剪和事后修剪两种方法，均衡决策树的复杂程度和计算量。

决策树的过拟合过程与修剪示意如图 5.21 所示。在“此处修剪”时可以使测试样本集的

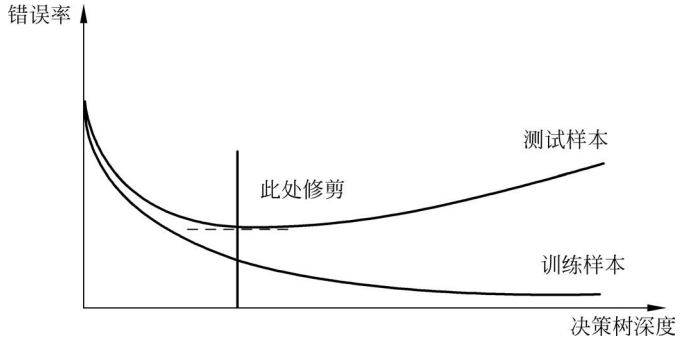


图 5.21 决策树的过拟合过程与修剪示意

错误率降至最低,而在右侧修剪时,随着产生的决策树深度的增加,可能会导致过拟合。反之,在左侧修剪时则会出现拟合不足。修剪后的分支因包含不同类的样本导致一定的分类误差。

7. 分类模型的评估

决策树等分类模型的性能需要进行评估,分类结果与样本的真实分类不一致归因于分类模型的错误率,这些错误的结果可能会造成比较大的损失。一般情况下,样本集分为训练集和检验集:训练集用于归纳分类模型,检验集用于评估模型的性能。评估的结果可以用混淆(confusion)矩阵或列联表来表示,其中列表示样本真实分类,行表示分类模型的预测分类,矩阵的元素表示真实分类与预测分类不一致的样本个数,从中可以计算各种错误的比例。混淆矩阵是真实值与预测值之间的一个交叉表格,既能识别误差的性质,也能计算错误识别率,使用户可以根据错误分类的比例评价模型。下面是三分类问题的混淆矩阵,其中 c_{ij} 表示实际样本类别是 C_i 而被预测为 C_j 的记录数,例如,c_{12} 表示属于 C_1 但被误分类为 C_2 的样本数。

真实分类 \ 预测	c_1	c_2	c_3
c_1	c_{11}	c_{12}	c_{13}
c_2	c_{21}	c_{22}	c_{23}
c_3	c_{31}	c_{32}	c_{33}

对于不存在某类样本过少的样本集(平衡样本集),可以由混淆矩阵计算分类模型正确预测的比例:

$$\text{正确率} = \sum_{i=1}^{3} c_{ii} \Big/ \sum_{i,j=1}^{3} c_{ij}$$

如果 c_{ij} 的重要性不同,例如,稀有类的正确分类比一般多数类的正确分类更有价值,那么还可以采用加权正确率评估分类器的性能:

$$\text{加权正确率} = \sum_{i=1}^{3} w_{ii} c_{ii} \Big/ \sum_{i,j=1}^{3} w_{ij} c_{ij}$$

对于两分类问题,样本可分为正例(positive)和反例(negative)两类。利用分类模型,在对样本进行分类时,可能出现 4 种情况:正例样本被分类成正例,称为真正类(true positive,TP);样本是反例但被分类成正例,称为假正类(false positive,FP);样本是反例被分类成反例,称为真负类(true negative,TN);样本是正例但被分类成反例,称为假反类(false negative,FN)。由此得到常用的分类模型准确性指标。

(1) 准确率(Accuracy)是分类模型正确分类的样本数(包括正例与反例)与样本总数的比值:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}$$

(2) 精确率(Precision)是模型正确分类的正例样本数与总的正例样本数(即正确分类的正例样本数目与错误分类的正确样本数目之和)的比值:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

(3) 召回率(Recall,也称为查全率)是模型分类正确的正例样本数与分类为正例的样本总数(分类正确的正例和分类错误的反例之和)的比值:

$$\text{Recall}=\frac{\text{TP}}{\text{TP}+\text{FN}}$$

(4) F_1 值(F-measure,又称 F-score)是精确率与召回率的调和平均,能够综合体现两种指标:

$$F_1=\frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$$

除了以上性能指标外,还可以使用 ROC 曲线等方法度量分类的质量。

ROC(receiver operating characteristic curve,接受者操作特性曲线)也是一种判断分类模型准确性的评估方法,分别用真正类率 TPR=TP/(TP+FN)和假正类率 FPR= FP/(FP+TN)作为平面坐标系的横轴和纵轴。这种图反映了分类敏感性和特异性的相互关系,图中曲线下面积(area under curve,AUC)越大,分类模型的预测准确性越高。因此,ROC 可以作为不同分类模型准确度比较的方法。如图 5.22 所示,从上到下依次是 Logistic 回归模型、Discriminant 模型和 CART 决策树模型的 ROC,从中看出 Logistic 回归模型、Discriminant 模型的预测准确度明显高于 CART 决策树模型。

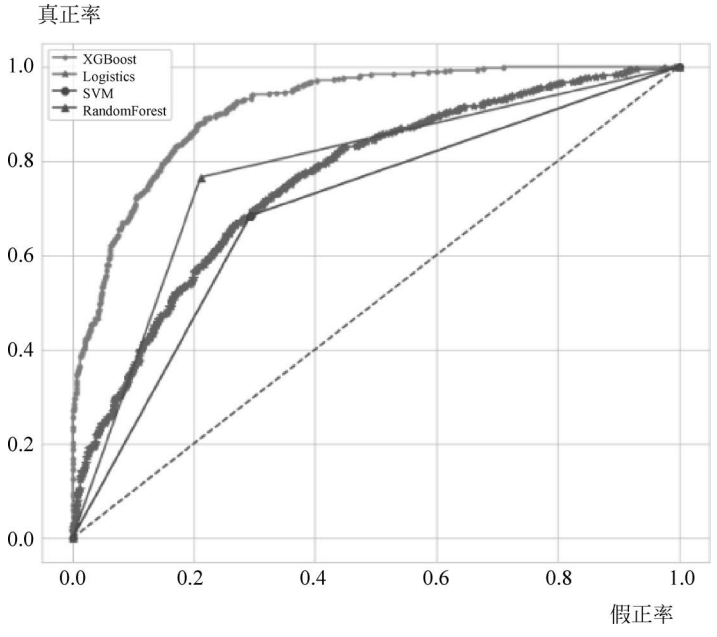


图 5.22 几种分类算法的 ROC

此外,k-折交叉校验(k-fold cross validation)也是一种常用的分类器性能评价方法:把样本集分成 k 个互不相交、大小相等的子集 $S_1,S_2,\cdots,S_k$。训练和测试进行 k 次:第 i 次训练时 S_i 用作测试集,其余的子集都用于训练。分类模型的准确率为 k 个测试集的准确率的平均值。这种方法的优点是每个样本都用于检验一次,有效地覆盖了整个数据集,但计算花费的时间比较多。

决策树的评价指标还有算法的速度、鲁棒性、可伸缩性、可解释性、计算复杂性、模型的简洁和易用性等指标。

决策树有广泛的应用,常用的场合包括在探测数据集时挑选主要的影响变量(例如作为神

经网络等算法的预处理)、预测变量的未来取值等。下面举例说明决策树在人力选择中的应用。

【例 5.8】 应用 CHAID 算法提高人才的选拔质量

人力资源是高科技行业的主要竞争力之一。人员招聘与选拔对企业员工整体素质水平有重要的影响。尤其对于半导体制造业等知识型企业,高技术、高素质人员在设计高效的半导体产品上必不可少。然而,半导体公司经常出现较高的人员流失率,也面临不易招到合适员工的困难。为了能吸引优秀的应聘者,有些半导体公司会提供吸引人的报酬和福利。但在判定应聘者能力的标准以及应聘者可能在公司工作的时间等问题上无法给出准确的判断。因此,选择优秀且能较长时间留在公司工作的工程师非常重要。

这里以中国台湾新竹科技园区的半导体制造商为例,利用 CHAID 算法,发现员工属性与工作行为之间的关联,指导人员的选拔和评价,主要的步骤如下。

(1) 问题定义。如图 5.23 所示,通过分析公司以往的人力资源管理数据预测应聘者的工作表现和保留时间。对于员工工作表现的分析,主要集中在表现"出色"员工(前 10%)和"需要改进"员工(后 5%)。根据人力资源专家的经验,如果员工没有通过三个月的试用期,这次招聘就视为失败,同时在培训上的投资也浪费了。如果员工在一年内辞职,则被视为管理问题。因此,在工作一年内辞职的以及试用期被淘汰的人员信息需要特别分析。对于应聘者在企业工作时间的分析,主要分析通过三个月试用期的员工和试用后一年内辞职的员工。

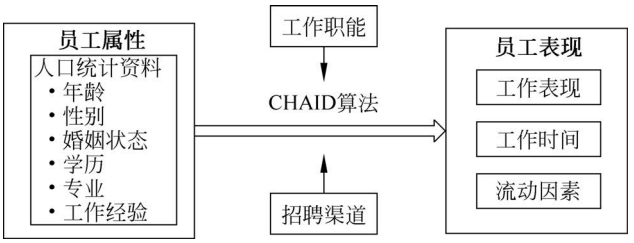


图 5.23 基于 CHAID 算法的人力资源分析

(2) 数据预处理。收集人力资源管理系统中员工的年龄、性别、工作经验、教育程度、所学专业、学校等级和招聘渠道等数据,预测变量包括工作表现(突出、成功和需要努力)、是否留在企业(考虑未通过三个月试用期和一年内辞职的情况)以及人员流失的原因。

为了保护隐私,人力资源数据大多存储在不同的数据库中。在数据挖掘之前,相关数据需要进行整合。由于数据中存在冗余和缺失的情况,需要进行格式化数据、删除冗余的数据、填补缺失的值等预处理来提升数据的质量。

(3) 分类规则的获取。选择决策树获取人员的分类规则。由于大部分员工数据都是可分类型变量,且需要对结果进行解释,这里选择决策树作为数据挖掘的方式。由于大部分人员数据都是虚拟变量,这里选择 CHAID 算法发掘员工特征变量和预测变量之间的关系。为了便于使用,可以把获得的决策树转化为规则,下面是部分生成的有关工作表现和人员流动的规则(括号内的数字为置信度):

IF 招聘渠道=外部 THEN 该员工的工作表现可能为需要改进(84%);

IF 经验=0 THEN 该员工的工作表现可能为需要改进(83%);

IF 学校等级=三等 THEN 该员工的工作表现可能为需要改进(86%);

IF 学历=硕士 AND 招聘途径=内部 AND 学校等级=一等 THEN 该员工的工作表

现为很好(63%)。

(4) 结果评估与应用。由上述步骤获得的规则经过一组人力资源专家筛选后,获得最终可用的规则。例如从招聘渠道来看,从内部招聘的员工比从外部招聘的员工表现好;内部招聘的员工相对于外部招聘的员工,更不易流失;那些从较好的学校毕业的员工和学历较高的员工表现会更好,但他们的流失率很高;具有较多工作经验的员工,工作表现更好,但他们的流失率也很高。

由于业务会发生变化,上述分析得到的规则需要定期收集新数据,递增挖掘更新规则库,以保证决策的有效性。

在本节的最后,使用 SPSS Modeler 分析选择邮购的客户特征。选择图 5.12 邮购客户的 mailshot.txt 文件,通过决策树分析可以让销售商了解哪些特征的客户是选择邮购的,便于销售商根据选择邮购用户的特征开展相应的促销活动。这里把客户的 age、sex、region、income、married、children、car、save_act、current_act 和 mortgage 作为输入,把客户是否选择邮购 mailshot_YN 作为目标。选择 C5.0 决策树算法对数据进行分析,得到的决策树如图 5.24 所示。

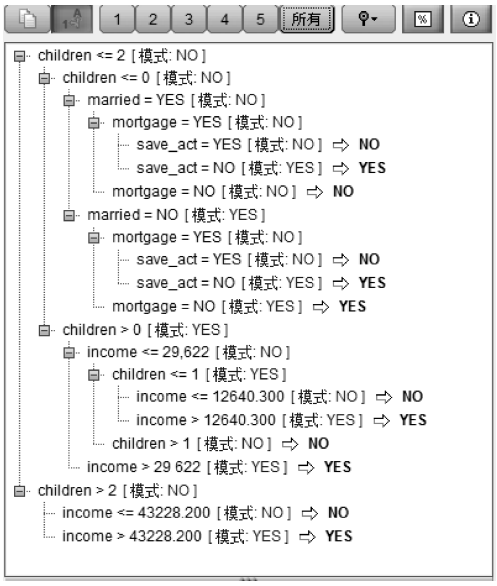


图 5.24 C5.0 决策树

从图 5.24 所示决策树中可以看出,客户的小孩数量和客户收入水平对客户选择购买方式有主要的影响。可以看出,具有以下特征的客户一般会选择邮购:小孩数量大于 2 且收入大于 43228.20 美元的客户;有一个或两个小孩,且收入大于 29622.00 美元的客户;没有小孩的,有抵押且没存款的客户;未婚也没有抵押的客户;未婚有抵押,但无存款的客户;有一个小孩且收入在 12640.00 美元到 29622.00 美元的客户。

类似地,采用 CHAID 决策树算法对上述数据进行分析,得到的决策树如图 5.25 所示。

从 CHAID 决策树模型可以得到,客户的孩子数量、客户的收入、年龄和居住地区等对 CHAID 建模有较大的影响。

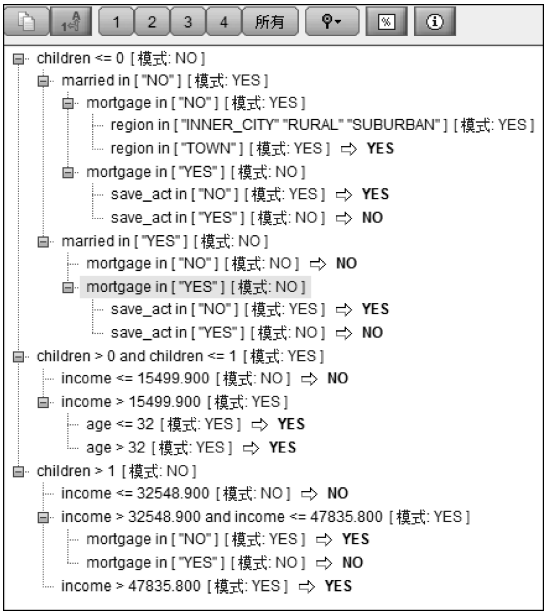


图 5.25 CHAID 决策树

5.5.3 支持向量机

支持向量机(support vector machine,SVM)具有坚实的统计学理论基础,它在解决非线性、高维模式识别问题等领域表现出许多优势,可有效处理线性和非线性可分的分类问题。目前,支持向量机已在人脸识别、文字识别、图像处理和时间序列预测等领域获得了比较广泛的应用。

支持向量机是从线性可分情况下的最优分类发展而来的,其基本思想可用两类线性可分问题来说明。如图 5.26 所示,假如两类样本是线性可分的,则可以找到一个超平面,该超平面可以把训练样本分为两类:一类样本位于超平面的上方,另一类样本位于超平面的下方。可能存在无穷多个这样的超平面,最优超平面不仅能把两类训练样本正确分开,而且使分类间隔最大。具有最大分类间隔的超平面在对测试样本进行分类时比具有较小分类间隔的超平面具有更好的泛化能力。所谓分类间隔是离最优超平面最近的样本且平行于最优超平面的两个超平面(最大边缘超平面,图 5.26 中的虚线)H_1 和 H_2 之间的距离 d。

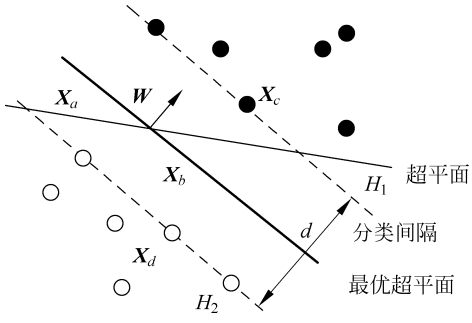


图 5.26 线性可分情况下的超平面

给定 n 个线性可分的训练样本。样本表示为 $(\boldsymbol{X}_i, y_i)(i=1,2,\cdots,n)$,其中样本的 m 个属性表示为列向量 $\boldsymbol{X}_i=(x_{i1},x_{i2},\cdots,x_{im})$,$y_i$ 是样本的类标号。这里考虑简单的二分类问题,$y_i \in \{-1,1\}$。图 5.26 所示的线性分类的最优超平面表示为

$$\boldsymbol{W}\cdot\boldsymbol{X}+b=0$$

其中：$\boldsymbol{W}$ 和 b 是超平面的参数；$\boldsymbol{W}\cdot\boldsymbol{X}$ 表示向量 $\boldsymbol{W}$ 和 $\boldsymbol{X}$ 的内积。

对于最优超平面的两个点 $\boldsymbol{X}_a$ 和 $\boldsymbol{X}_b$，满足：

$$\boldsymbol{W}\cdot\boldsymbol{X}_a+b=0$$

$$\boldsymbol{W}\cdot\boldsymbol{X}_b+b=0$$

由上面两个方程可得

$$\boldsymbol{W}\cdot(\boldsymbol{X}_a-\boldsymbol{X}_b)=0$$

从上式可以看出，系数向量 $\boldsymbol{W}$ 的方向与最优超平面垂直。

可以证明，最优超平面上的点 $\boldsymbol{X}_t$ 类标号定义为 $y_t=1$，满足：

$$\boldsymbol{W}\cdot\boldsymbol{X}_t+b=\mu>0$$

而最优超平面下的点 $\boldsymbol{X}_u$ 类标号定义为 $y_b=-1$，满足：

$$\boldsymbol{W}\cdot\boldsymbol{X}_u+b=\Psi<0$$

调整 $\boldsymbol{W}$ 和 b，两个最大边缘超平面分别表示如下：

$$H_1:\boldsymbol{W}\cdot\boldsymbol{X}+b=1$$

$$H_2:\boldsymbol{W}\cdot\boldsymbol{X}+b=-1$$

设 $\boldsymbol{X}_c$ 和 $\boldsymbol{X}_d$ 分别是超平面 H_1 和 H_2 上的点，可得

$$\boldsymbol{W}(\boldsymbol{X}_c-\boldsymbol{X}_d)=2$$

$$\|\boldsymbol{W}\|\times d=2$$

$$d=\frac{2}{\|\boldsymbol{W}\|}$$

其中：$\|\boldsymbol{W}\|$ 表示向量 $\boldsymbol{W}$ 的长度。

支持向量机的训练是为了从 n 个训练数据中估计参数 $\boldsymbol{W}$ 和 b，它们满足：

$$y_i=1,\quad \boldsymbol{W}\cdot\boldsymbol{X}_i+b\geqslant 1$$

$$y_i=-1,\quad \boldsymbol{W}\cdot\boldsymbol{X}_i+b\leqslant -1$$

即 $y_i(\boldsymbol{W}\cdot\boldsymbol{X}_i+b)\geqslant 1, i=1,2,\cdots,n$。

这样支持向量机的训练就转化为下面的最优化问题：

$$\min f(\boldsymbol{W})=\frac{\|\boldsymbol{W}\|^2}{2}$$

$$\text{s.t.}\quad y_i(\boldsymbol{W}\cdot\boldsymbol{X}_i+b)\geqslant 1,\quad i=1,2,\cdots,n$$

根据拉格朗日方程为

$$L(\boldsymbol{W},b,\lambda_i)=\frac{1}{2}\|\boldsymbol{W}\|^2-\sum_{i=1}^{n}\lambda_i[y_i(\boldsymbol{W}\cdot\boldsymbol{X}_i+b)-1]$$

其中：$\lambda_i\geqslant 0$ 为拉格朗日乘子。分别对 L 求 $\boldsymbol{W}$ 和 b 的偏导数并令它们等于 0，可得

$$\boldsymbol{W}=\sum_{i=1}^{n}\lambda_i y_i\boldsymbol{X}_i,\quad \sum_{i=1}^{n}\lambda_i y_i=0$$

利用以上两式和目标函数的 n 个不等式约束条件，难以求得 $\boldsymbol{W}$、b 和 λ_i。因此需要把拉格朗日方程转化为对偶问题，得到对偶公式：

$$L_D(\lambda_i)=\sum_{i=1}^{n}\lambda_i-\frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j\boldsymbol{X}_i\cdot\boldsymbol{X}_j$$

对偶公式仅包含拉格朗日乘子 λ_i，这是一元函数的优化问题，相对比较简单。求得 λ_i 后代入公式 $\boldsymbol{W}=\sum_{i=1}^{n}\lambda_i y_i \boldsymbol{X}_i$ 即可求出 $\boldsymbol{W}$。计算 b 需要一定的技巧，这里不做过多的阐述。

求得支持向量机的参数 $\boldsymbol{W}$ 和 b 后，检验样本 $\boldsymbol{X}_T$ 可以按下面的公式进行分类。

$$f(\boldsymbol{X}_T)=\mathrm{sign}\left(\sum_{i=1}^{n}\lambda_i y_i \boldsymbol{X}_i \cdot \boldsymbol{X}_T + b\right)$$

其中：sign( )表示符号函数，代表样本的类别。

以上讨论的是线性可分类的支持向量机。对于非线性可分类问题，很难找到区分两类样本的线性超平面，可以用非线性支持向量机来解决。非线性支持向量机的基本思想是把样本数据通过一个非线性变换，使样本数据由原来的特征空间映射到一个新的特征空间，从而把非线性可分类问题转化为线性可分类问题，因此可以用上面讨论的线性支持向量机划分样本。

【例 5.9】 支持向量机在青光眼检测中的应用

青光眼是一种眼内压增高、视神经和视功能损害的眼病。持续的高眼压可能给眼球各部分组织和视功能带来损害，如不及时治疗，视力可能会全部丧失以致失明。早期诊断是治疗青光眼的关键，多数眼科医生都能正确诊断晚期青光眼，但如何正确诊断早期青光眼是一个难题。除急性发作外，相当一部分早期青光眼患者都没有明显症状，这给眼科医生早期诊断青光眼提出了更高的要求。通过分析已搜集的原发性开角型青光眼病人的资料，得出的各危险因素对原发性开角型青光眼的影响权重，对未来医生诊断具有一定的指导意义。房角检测是检查早期青光眼的主要手段，原发性开角型青光眼患者的房角为开角还是闭角的检测，在医学诊断中主要取决于双眼眼压差、房角和前房深度等几个因素。

这里随机抽取近三年某医院眼科青光眼门诊病例 88 例，利用样本 88 例数据，建立支持向量机训练学习模型。88 例病例(原发性开、闭角型青光眼)的双眼眼压差、房角、前房深度三个变量的三维图，如图 5.27 所示。图中“+”表示开角型青光眼样本点，“○”表示闭角型青光眼样本点。从图 5.27 中可以看出，样本数据相互交叉较多，不易线性可分。

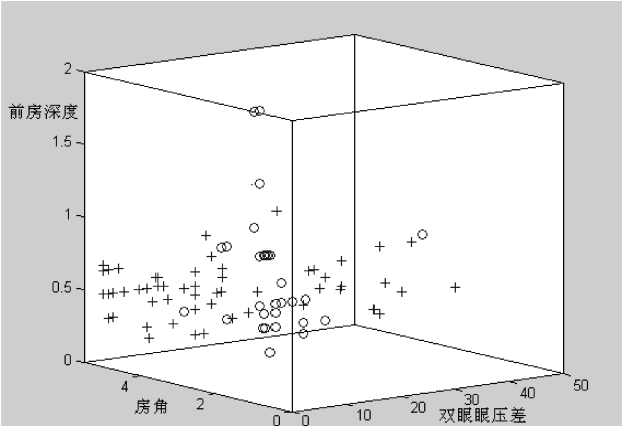


图 5.27 青光眼病例三维图

对 88 例数据利用支持向量机算法建立分类函数，得到最优平面方程。选择使房角判别分类正确率最大的组合，分类结果如表 5.8 所示，检测结果正确率为 98.86%。

表 5.8 分类结果

序号	Y	Result	序号	Y	Result	序号	Y	Result	序号	Y	Result
1	1	5.3025	23	1	1.4229	45	1	2.27	67	−1	−5.3398
2	1	1	24	1	1.2037	46	1	1.1481	68	−1	−1
3	1	1.0887	25	1	−0.6995	47	1	2.991	69	−1	−2.2929
4	1	1.0963	26	1	7.3851	48	1	1.0713	70	−1	−1.2666
5	1	1	27	1	8.4982	49	1	1.1894	71	−1	−3.7496
6	1	1	28	1	1.1156	50	1	3.1297	72	−1	−4.2329
7	1	1	29	1	1.2579	51	1	1	73	−1	−3.5738
8	1	7.3374	30	1	5.3573	52	1	1	74	−1	−1.3298
9	1	5.3318	31	1	3.0704	53	1	3.0705	75	−1	−0.7429
10	1	4.0522	32	1	4.0671	54	1	1.1018	76	−1	−3.0232
11	1	7.4321	33	1	4.0644	55	1	1	77	−1	−5.7684
12	1	1	34	1	3.0705	56	1	1.0004	78	−1	−2.3637
13	1	4.045	35	1	2.3337	57	1	5.3573	79	−1	−2.0486
14	1	8.0019	36	1	1.4229	58	1	3.0003	80	−1	−1.4389
15	1	2.2736	37	1	1	59	−1	−3.77	81	−1	−4.8653
16	1	5.6996	38	1	1	60	−1	−3.6162	82	−1	−1
17	1	6.2962	39	1	5.7072	61	−1	−6.309	83	−1	−1
18	1	2.6828	40	1	1.1016	62	−1	−5.1213	84	−1	−3.0938
19	1	5.8982	41	1	1	63	−1	−1	85	−1	−3.0938
20	1	1.2575	42	1	1.0713	64	−1	−1.0605	86	−1	−1
21	1	1.3245	43	1	1.6394	65	−1	−1	87	−1	−2.1881
22	1	3.1297	44	1	1	66	−1	−1.9405	88	−1	−5.2856

表 5.8 中 Y 为确诊后的临床诊断结果，$Y=1$ 为原发性开角型青光眼，$Y=-1$ 为原发性闭角型青光眼。Result 为模型数据回代结果。Result＞0 代表房角为开角，否则为闭角。得到结果后，可以把 26 个新数据用支持向量机模型检测，结果如表 5.9 所示。

表 5.9 测试结果

序号	Y	双眼眼压差 a	房角 b	前房深度 c	Test	序号	Y	双眼眼压差 a	房角 b	前房深度 c	Test
1	1	1	4	0.66	1.5403	14	1	6	4	0.7	7.3175
2	1	15	5	0.66	1.8277	15	1	35	2	0.66	0.3472
3	1	36	4	0.33	1.2016	16	1	1	4	1	1.6438
4	1	16	4	0.33	0.9513	17	1	22	5	0.5	1.2037
5	1	1	4	0.33	1.4229	18	−1	4	1	1	−1.4554
6	1	0	3	0.5	−2.8798	19	−1	12	1	1	−2.2692
7	1	4	3	1	5.3568	20	−1	6	1	0.5	−1.0596
8	1	10	3	0.33	1.0142	21	−1	5	1	1	−1.1438
9	1	20	4	1	0.9939	22	−1	5	1	0.5	−0.9292
10	1	20	4	0.33	1	23	−1	3	1	1	−2.3637
11	1	9	4	0.33	3.3436	24	−1	4	1	1	−1.4554
12	1	22	5	0.3	1.2041	25	−1	1	1	0.5	−5.3485
13	1	9	4	0.25	3.3432	26	−1	28	1	1.5	−0.1068

表 5.9 中 Y 为确诊后的临床诊断结果，$Y=1$ 为原发性开角型青光眼，$Y=-1$ 为原发性闭角型青光眼。Test 为房角判别模型检测结果。如果以 Y 为衡量标准，那么 Test 检测结

果与临床确诊吻合率达到96.15%。

支持向量机用于预测银行客户流失的Python代码如下：

```
from sklearn.svm import SVC
from sklearn.externals import joblib
from sklearn.utils import shuffle
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold

df = pd.read_csv("./data/select-data.csv")
train = []
target = []
for i in range(0, len(df["EstimatedSalary"])):
    mid = []
    mid.append(df["Geography"][i])
    mid.append(df["Gender"][i])
    mid.append(df["EB"][i])
    mid.append(df["Age"][i])
    mid.append(df["EstimatedSalary"][i])
    mid.append(df["NumOfProducts"][i])
    mid.append(df["CreditScore"][i])
    mid.append(df["Tenure"][i])
    mid.append(df["HasCrCard"][i])
    mid.append(df["IsActiveMember"][i])
    target.append(df["Exited"][i])
    train.append(mid)
train = np.array(train)
target = np.array(target)
train, target = shuffle(train, target)
#构建10折交叉验证
kf = KFold(n_splits=10)
for train_index, test_index in kf.split(train):
    trainx = train[train_index]
    trainy = target[train_index]
    testx = train[test_index]
    testy = target[test_index]
    svc = SVC(kernel='linear', C=0.1)
    clf = svc.fit(trainx, trainy)
    sc = svc.score(test, test_target)
    print('%.7f' % sc)
```

5.5.4 反向传播神经网络

神经网络最初来自通过计算机模型来模仿人类智能的实践。1943年，神经生理学家McCulloch和逻辑学家Pitts设计了神经活动的逻辑运算模型，以解释生物神经元的工作机理，为人工神经网络的研究奠定了理论基础。在20世纪50年代，计算机科学家在McCulloch和Pitts工作的基础上，提出了感知器(perceptron)的模型，可解决手推车上的扫帚平衡等简单问题。1969年Minsky等学者指出感知器仅能解决线性划分，而对于非线性问题会遇到很大困难。为解决感知器非线性可分类问题，1986年美国的一个并行计算研究小组提出了著名的反向传播(back propagation，BP)算法，引发了神经网络的研究热点。此

后有关神经网络的研究逐渐从实验室转向商用。目前，神经网络经常用于分类和聚类，作为一种重要的数据挖掘方法它已在医学诊断、信用卡欺诈识别、手写体数字识别以及发动机的故障诊断等领域得到了广泛的应用。

图 5.28 所示为一个常见的 BP 神经网络结构，包括输入层、隐层和输出层。各层神经元可接收前一层神经元的信号，经激活函数产生新信号后传递给下一层。当输入 x_1、x_2、x_3 等数据时，经过隐层的计算，最终输出结果。

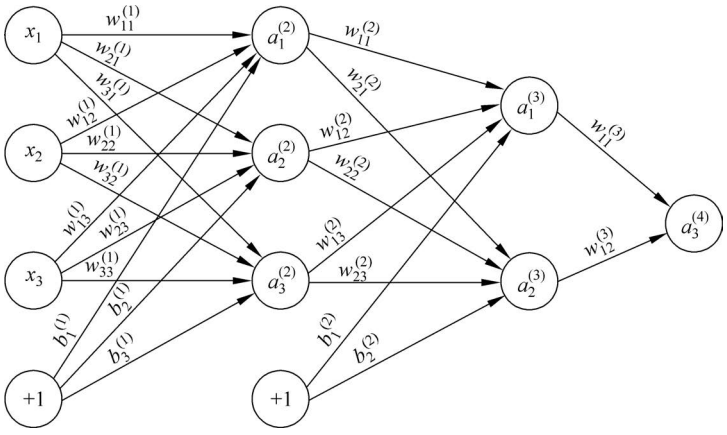


图 5.28　BP 神经网络结构

在神经网络中，将前一层的输出加权求和后，需要使用一个激活函数，目的是增加网络模型的非线性。引入了激活函数后，则可以让神经网络逼近任何非线性函数。常用的激活函数包括 Sigmoid 函数、双曲正切(tanh)函数和 ReLU 函数。

Sigmoid 函数为

$$f(x)=\frac{1}{1+\mathrm{e}^{-x}}$$

其值域在 0 至 1，导数为 $f'(x)=f(x)\cdot(1-f(x))$，易于计算。但 Sigmoid 函数的缺点在于输出靠近 0 和 1 时，曲线过于平坦，容易出现梯度消失的问题，且输出的值域不对称。

双曲正切函数为

$$\tanh(x)=\frac{\mathrm{e}^{x}-\mathrm{e}^{-x}}{\mathrm{e}^{x}+\mathrm{e}^{-x}}$$

可将数据映射到 -1 至 1，但其值域两端依然过于平坦，依然存在梯度消失的问题。

ReLU 函数为

$$f(x)=\max(0,x)$$

其收敛速度比 Sigmoid 和双曲正切函数更快，缺点是负值被截断为 0，导致特征消失，有些神经元可能永远无法被激活。

在图 5.28 中，$w_{ij}^{(l)}$ 表示第 $l-1$ 隐层($l-1=0$ 时表示输入层)第 j 个神经元与第 l 隐层第 i 个神经元间的权值，$b_i^{(l)}$ 表示第 l 隐层第 i 个神经元的偏置项。在下文中，将使用 S_l 表示第 l 隐层神经元的个数(未计入偏置项)，n_l 表示神经网络总层数(包括输入、输出层)。

1. 前向传播

使用 $a_i^{(l+1)}$ 表示第 l 隐层第 i 单元的激活值(输出值)。当 $l=0$(即输入层)时,$a_i^{(1)}=x_i$,表示第 i 个输入值。使用 $z_i^{(l+1)}$ 表示第 l 隐层第 i 单元输入加权和(包括偏置项),例如:

$$z_i^{(l+1)}=\sum_{j=1}^{S_l} w_{ij}^{(l)} a_j^{(l)}+b_i^{(l)}$$

并选择函数 $f(\cdot)$作为该层的激活函数,则 $a_i^{(l+1)}=f(z_i^{(l+1)})$

对于给定参数集合(W,b)的神经网络,就可以按照函数 $h_{W,b}(x)$来计算输出结果。本例中神经网络的前向传播计算过程如下:

$$a_1^{(2)}=f(w_{11}^{(1)}x_1+w_{12}^{(1)}x_2+w_{13}^{(1)}x_3+b_1^{(1)})$$
$$a_2^{(2)}=f(w_{21}^{(1)}x_1+w_{22}^{(1)}x_2+w_{23}^{(1)}x_3+b_2^{(1)})$$
$$a_3^{(2)}=f(w_{31}^{(1)}x_1+w_{32}^{(1)}x_2+w_{33}^{(1)}x_3+b_3^{(1)})$$
$$a_1^{(3)}=f(w_{11}^{(2)}a_1^{(2)}+w_{12}^{(2)}a_2^{(2)}+w_{13}^{(2)}a_3^{(2)}+b_1^{(2)})$$

将激活函数 $f(\cdot)$扩展为用向量(分量的形式)来表示,即 $f([z_1,z_2,z_3])=[f(z_1),f(z_2),f(z_3)]$,则上面的等式可以简化为

$$z^{(2)}=W^{(1)}x+b^{(1)}$$
$$a^{(2)}=f(z^{(2)})$$
$$z^{(3)}=W^{(2)}a^{(2)}+b^{(2)}$$
$$a^{(3)}=f(z^{(3)})$$

上述计算步骤称为前向传播。在给定第 $l-1$ 隐层的激活值 $a^{(l)}$后($a^{(1)}=x$ 表示输入层的激活值),就可以按照下面步骤计算得到第 l 层的激活值 $a^{(l+1)}$:

$$z^{(l+1)}=W^{(l)}a^{(l)}+b^{(l)}$$
$$a^{(l+1)}=f(z^{(l+1)})$$

2. 计算总误差

假设有一个包含 m 个样例的固定样本集$\{(x^{(1)},y^{(1)}),\cdots,(x^{(m)},y^{(m)})\}$,可以用批量梯度下降法来训练神经网络。对于单个样例$(x^{(i)},y^{(i)})$,其代价函数为

$$J(W,b;x^{(i)},y^{(i)})=\frac{1}{2}\|h_{W,b}(x^{(i)})-y^{(i)}\|^2$$

对于包含 m 个样例的数据集,可将整体代价函数定义为

$$J(W,b)=\left[\frac{1}{m}\sum_{i=1}^{m}J(W,b;x^{(i)},y^{(i)})\right]+\frac{\lambda}{2}\sum_{l=0}^{n_{l-2}}\sum_{i=1}^{s_{l+1}}\sum_{j=1}^{s_l}(w_{ij}^{(l+1)})^2$$
$$=\left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\|h_{W,b}(x^{(i)})-y^{(i)}\|^2\right)\right]+\frac{\lambda}{2}\sum_{l=0}^{n_{l-2}}\sum_{i=1}^{S_{l+1}}\sum_{j=1}^{S_l}(w_{ij}^{(l+1)})^2$$

其中:$J(W,b)$定义中的第一项是一个均方差项,第二项是一个正则化项(也可称为权重衰减项),其目的是减小权重的幅度,防止过度拟合。

3. 反向传播

反向传播算法的思路如下:给定一个样例(x,y),首先进行“前向传播”运算,计算网络

中所有的激活值，包括 $h_{W,b}(x)$ 的输出值。然后，针对第 l 层的每一个节点 i，计算出其残差 $\delta_i^{(l+1)}$，该残差表明了该节点对最终输出值的残差的影响程度。对于最终的输出层节点，可以直接算出神经网络的输出与实际值之间的差距，可将这个差距定义为 $\delta_i^{(n_l)}$。对于隐藏单元将基于第 l 层节点残差的加权平均值计算 $\delta_i^{(l)}$，这些节点以 $a_i^{(l)}$ 作为输入。下面将给出反向传播算法的细节。

梯度下降法中每一次迭代都按照如下公式对参数 W 和 b 进行更新：

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \alpha \frac{\partial}{\partial w_{ij}^{(l)}} J(W,b)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W,b)$$

其中：α 是学习率。

1）计算残差

对于第 n_{l-1} 层（输出层）的每个输出单元 i，可根据以下公式计算残差：

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} J(W,b;x,y) = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \| h_{W,b}(x) - y \|^2$$

$$= \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \sum_{j=1}^{S_{n_{l-1}}} (y_j - a_j^{(n_l)})^2 = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \sum_{j=1}^{S_{n_{l-1}}} (y_j - f(z_j^{(n_l)}))^2$$

其中：

$$\sum_{j=1}^{S_{n_{l-1}}} (y_j - f(z_j^{(n_l)}))^2 = (y_1 - f(z_1^{(n_l)}))^2 + (y_2 - f(z_2^{(n_l)}))^2 + \cdots +$$

$$(y_i - f(z_i^{(n_l)}))^2 + \cdots + (y_{S_{n_{l-1}}} - f(z_{S_{n_l}}^{(n_l)}))^2$$

因此上式可得

$$\delta_i^{(n_l)} = -(y_i - f(z_i^{(n_l)})) \cdot f'(z_i^{(n_l)}) = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})$$

对 $l = n_l - 2, n_l - 3, n_l - 4, \cdots, 1$ 的各层，其第 i 个节点的残差计算方法如下：

$$\delta_i^{(n_l-1)} = \frac{\partial}{\partial z_i^{(n_l-1)}} J(W,b;x,y) = \frac{\partial}{\partial z_i^{(n_l-1)}} \frac{1}{2} ||h_{W,b}(x) - y||^2 = \frac{1}{2} \frac{\partial}{\partial z_i^{(n_l-1)}} \sum_{j=1}^{S_{n_{l-1}}} (y_j - a_j^{(n_l)})^2$$

$$= \frac{1}{2} \sum_{j=1}^{S_{n_{l-1}}} \frac{\partial}{\partial z_i^{(n_l-1)}} (y_j - a_j^{(n_l)})^2 = \frac{1}{2} \sum_{j=1}^{S_{n_{l-1}}} \frac{\partial}{\partial z_i^{(n_l-1)}} (y_j - f(z_j^{(n_l)}))^2$$

$$= \sum_{j=1}^{S_{n_{l-1}}} -(y_j - f(z_j^{(n_l)})) \frac{\partial}{\partial z_i^{(n_l-1)}} f(z_j^{(n_l)}) = \sum_{j=1}^{S_{n_{l-1}}} -(y_j - f(z_j^{(n_l)})) f'(z_j^{(n_l)}) \frac{\partial z_j^{(n_l)}}{\partial z_i^{(n_l-1)}}$$

$$= \sum_{j=1}^{S_{n_{l-1}}} \delta_j^{(n_l)} \frac{\partial z_j^{(n_l)}}{\partial z_i^{(n_l-1)}} = \sum_{j=1}^{S_{n_{l-1}}} \left(\delta_j^{(n_l)} \frac{\partial}{\partial z_i^{(n_l-1)}} \sum_{k=1}^{S_{n_{l-2}}} f(z_k^{(n_l-1)}) \right) \cdot w_{jk}^{(n_l-1)})$$

其中：

$$\sum_{k=1}^{S_{n_l-2}} f(z_k^{(n_l-1)}) \cdot w_{jk}^{(n_l-1)} = f(z_1^{(n_l-1)}) \cdot w_{j1}^{(n_l-1)} + f(z_2^{(n_l-1)}) \cdot w_{j2}^{(n_l-1)} + \cdots +$$

$$f(z_i^{(n_l-1)})\cdot w_{ji}^{(n_l-1)}+\cdots+f(z_{s_{n_l-2}}^{(n_l-1)})\cdot w_{js_{n_l-1}}^{(n_l-1)}$$

所以

$$\delta_i^{(n_{l-1})}=\sum_{j=1}^{S_{n_{l-1}}}\delta_j^{(n_l)}\cdot w_{ji}^{(n_l-1)}\cdot f'(z_i^{(n_l-1)})=\left(\sum_{j=1}^{S_{n_{l-1}}}w_{ji}^{(n_l-1)}\cdot\delta_j^{(n_l)}\right)\cdot f'(z_i^{(n_l-1)})$$

将上式中的 n_{l-1} 与 n_l 替换为 l 和 $l+1$,可得

$$\delta_i^{(l)}=\left(\sum_{j=1}^{S_l}w_{ji}^{(l)}\cdot\delta_j^{(l+1)}\right)\cdot f'(z_i^{(l)})$$

上述步骤即为“反向传播”的残差计算过程。

2) 计算 $w_{ij}^{(l)}=w_{ij}^{(l)}-\alpha\frac{\partial}{\partial w_{ij}^{(l)}}J(W,b)$

$$\frac{\partial}{\partial w_{ij}^{(l)}}J(W,b;x,y)=\left[\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial w_{ij}^{(l)}}J(W,b;x^{(i)},y^{(i)})\right]+\lambda w_{ij}^{(l)}$$

因为

$$z_i^{(l+1)}=\sum_{j=1}^{S_l}w_{ij}^{(l)}\cdot a_j^{(l)}+b_i^{(l)}\ \text{且}\frac{\partial}{\partial z_i^{(l+1)}}J(W,b;x,y)=\delta_i^{(l+1)}$$

所以

$$\frac{\partial}{\partial w_{ij}^{(l)}}J(W,b;x,y)=\frac{\partial J(W,b;x,y)}{\partial z_i^{(l+1)}}\cdot\frac{\partial z_i^{(l+1)}}{\partial w_{ij}^{(l)}}=\delta_i^{(l+1)}\cdot a_j^{(l)}$$

即

$$w_{ij}^{(l)}=w_{ij}^{(l)}-\alpha\frac{\partial}{\partial w_{ij}^{(l)}}J(W,b)=w_{ij}^{(l)}-\alpha(\delta_i^{(l+1)}\cdot a_j^{(l)}+\lambda w_{ij}^{(l)})$$

3) 计算 $b_i^{(l)}=b_i^{(l)}-\alpha\frac{\partial}{\partial b_i^{(l)}}J(W,b)$

$$\frac{\partial}{\partial b_i^{(l)}}J(W,b)=\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial b_i^{(l)}}J(W,b;x^{(i)},y^{(i)})$$

因为

$$z_i^{(l+1)}=\sum_{j=1}^{S_l}w_{ij}^{(l)}\cdot a_j^{(l)}+b_i^{(l)}\ \text{且}\frac{\partial}{\partial z_i^{(l+1)}}J(W,b;x,y)=\delta_i^{(l+1)}$$

所以

$$\frac{\partial}{\partial b_i^{(l)}}J(W,b;x,y)=\frac{\partial J(W,b;x,y)}{\partial z_i^{(l+1)}}\cdot\frac{\partial z_i^{(l+1)}}{\partial b_i^{(l)}}=\delta_i^{(l+1)}$$

即

$$b_i^{(l)}=b_i^{(l)}-\alpha\frac{\partial}{\partial w_{ij}^{(l)}}J(W,b)=b_i^{(l)}-\alpha\cdot\delta_i^{(l+1)}$$

4. 参数更新

在计算出各层的误差项后,还需设置合适的学习率。学习率控制每次更新参数的幅度,若学习率过高或过低都有可能对模型的结果产生不良影响,而合适的学习率则可以加快模

型的训练速度。

学习率太大会导致参数更新幅度过大，可能跳过损失函数的极小值，导致参数值不断在极值点两端震荡，而无法继续减小损失函数。学习率太小则会导致参数更新太慢，需要消耗大量训练资源才能保证获取参数最优值。对于学习率的设置，可在刚开始更新时选择较大的学习率，当参数逐渐接近最优值时，逐步减小学习率，使得参数最终能够达到极优值。

以下为实现梯度下降法的具体训练过程。

(1) 设定(W,b)初值。

(2) 对于样本集$\{(x^{(1)},y^{(1)}),\cdots,(x^{(m)},y^{(m)})\}$进行前向传播，计算出每一层的激活值$a_i^{(l)}$。

(3) 计算每一层的残差$\delta_i^{(l+1)}$，求得各层参数的导数：

$$\frac{\partial}{\partial w_{ij}^{(l)}}J(W,b;x,y)=\frac{\partial J(W,b;x,y)}{\partial z_i^{(l+1)}}\cdot\frac{\partial z_i^{(l+1)}}{\partial w_{ij}^{(l)}}=\delta_i^{(l+1)}\cdot a_j^{(l)}$$

$$\frac{\partial}{\partial b_i^{(l)}}J(W,b;x,y)=\frac{\partial J(W,b;x,y)}{\partial z_i^{(l+1)}}\cdot\frac{\partial z_i^{(l+1)}}{\partial b_i^{(l)}}=\delta_i^{(l+1)}$$

$$\frac{\partial}{\partial w_{ij}^{(l)}}\frac{\lambda}{2}\sum_{l=0}^{n_{l-2}}\sum_{i=1}^{s_{l+1}}\sum_{j=1}^{s_l}(w_{ij}^{(l+1)})^2=\lambda w_{ij}^{(l)}$$

(4) 更新参数：

$$w_{ij}^{(l)}=w_{ij}^{(l)}-\alpha\frac{\partial}{\partial w_{ij}^{(l)}}J(W,b)=w_{ij}^{(l)}-\alpha(\delta_i^{(l+1)}\cdot a_j^{(l)}+\lambda w_{ij}^{(l)})$$

$$b_i^{(l)}=b_i^{(l)}-\alpha\frac{\partial}{\partial b_i^{(l)}}J(W,b)=b_i^{(l)}-\alpha\cdot\delta_i^{(l+1)}$$

重复步骤(2)至(4)，直至损失函数值不再下降。

网络训练的停止标准是训练次数达到设定的次数或者训练误差小于某阈值。值得注意的问题是，梯度下降法是一种贪心算法，网络在训练时可能陷入局部极小，影响神经网络的应用效果。常用的解决方法是控制学习率或者在权的更新中增加冲量项α，使网络的学习可能跳出局部最小。与学习率类似，α也需要多次实验，才能确定合理的取值。由上可见，需要设置的经验参数太多也是神经网络的不足之处。

下面举个简单的例子来说明BP神经网络在分类中的应用。例如，VISA公司曾用神经网络探测顾客的信用卡欺诈行为，如图5.29所示。此网络的输入是影响顾客信誉的几个属性：收入、负债、年龄和付款记录等，输出用于预测顾客的信用。

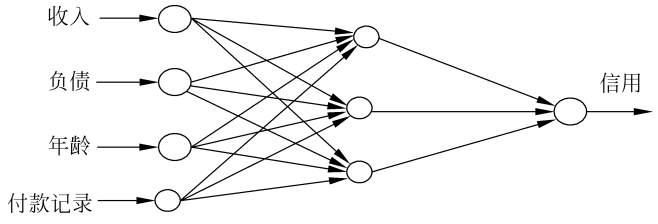


图5.29 信用卡欺诈检测

神经网络具有并行性、分布存储、容错性和学习能力等优点，但其从训练样本中挖掘出的模式却表现在网络的权和偏置上，这种知识很难被人理解，因此有学者研究用网络提取规

则的方法来提高神经网络的可解释性。

BP 神经网络用于预测银行客户流失的 Python 代码如下:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
# 加载数据
data = pd.read_csv('scalar.csv')
# 分离特征和目标变量
X = data.drop(columns = ['Exited'])                                    # 特征列
y = data['Exited']                                                     # 目标列
# 数据预处理:标准化特征值
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.2, random_
state = 42)
# 构建 BP 神经网络模型
model = Sequential()
model.add(Dense(64, input_dim = X_train.shape[1], activation = 'relu'))  # 输入层和隐藏层
model.add(Dense(32, activation = 'relu'))                              # 隐藏层
model.add(Dense(1, activation = 'sigmoid'))                            # 输出层
# 编译模型
model.compile(optimizer = Adam(learning_rate = 0.001), loss = 'binary_crossentropy', metrics =
['accuracy'])
# 训练模型
model.fit(X_train, y_train, epochs = 50, batch_size = 32, validation_split = 0.2, verbose = 1)
# 评估模型
loss, accuracy = model.evaluate(X_test, y_test, verbose = 0)
print(f'Test Accuracy: {accuracy:.4f}')
```

【例 5.10】 神经网络在外出就餐预测中的应用

外出就餐已成为很多人的生活习惯,是什么因素影响人们外出就餐也引起了一些学者的注意。研究表明,人们外出就餐的习惯与其性别、社会阶层、激励因素以及餐厅的特色有关系。例如,缺少时间、方便、环境的改变、多样的食物等是新加坡人外出就餐的主要原因。那么什么因素影响了中国台湾消费者外出就餐呢?

这里通过问卷调查收集了来自中国台湾不同地区的 800 个问卷,使用神经网络把经常出去和那些不经常出去就餐的人区分开来。受访消费者中大部分人是快餐食品的消费者,年龄在 15 岁至 64 岁,他们有不同的收入水平,来自不同的地区。此外,受访者的个人特征,包括社会人口统计、选择餐馆时考虑的因素、受访者的生活方式等数据也被记录下来。为了提高神经网络模型的简约性,把调查中得到的多个单个指标进行了归纳,得到了综合的变量。

把样本数据分成两个集合:随机选择的 534 个样本作为训练样本,剩下的样本作为测试样本。如果受访者外出就餐的频率平均每月不少于 25 次,则标记为类 1;否则标记为类 2。训练样本集中有 125 个类 1 的样本和 409 个类 2 的样品。在测试样本中,相应的数字分别

为 63 和 203。

使用的神经网络包含 55 个输入节点、1 个隐藏节点和 1 个输出节点。神经网络的训练使用改进的 BP 算法，这种方法比标准的反向传播方法收敛更快。一旦神经网络被训练好，可以对低权值的网络连接进行修剪，同时被修剪后的网络在训练样本集和测试样本集的准确率都能得到保证。

从修剪后的神经网络抽取得到那些经常外出就餐的消费者和不经常外出就餐的消费者，并在测试样本集上检验神经网络预测的准确率。

结果发现，那些不太内向并且更自信的受访者可能外出就餐。收入比较低且经常使用计算机的学生在选择就餐时更看重优越的地理位置。相比之下，那些不使用互联网的可能是年龄比较大的市民，他们往往对味道更看重。对于那些外出就餐的人群可以分成不同的消费阶层，不同的因素对于预测外出就餐的频率发挥着作用。因此，了解市场的异质性的营销人员针对细分市场的不同消费群体，可以采取不同的营销策略。

5.5.5 其他分类方法

除了贝叶斯分类器、决策树、支持向量机和神经网络等分类方法外，k 最近邻(k-nearest neighbor)分类器、基于案例的推理、组合分类器、模糊集(fuzzy set)、遗传算法和粗糙集理论等方法也常用于分类。

1. k 最近邻分类器

k 最近邻分类器是一种比较简单的、基于实例的分类学习方法，不需要通过复杂的训练过程建立分类模型，既可以用于可分类属性，也可用于连续属性的分类。它已在欺诈检测、顾客响应预测和协同过滤(collaborative filtering)等领域得到应用。

k 最近邻分类器的基本思想是给定一个未确定类别的样本 x，在样本空间搜索，找出与未确定类别样本距离最近的 k 个样本 $x_i(i=1,2,\cdots,k)$，待分类的样本属于哪一类由 k 个近邻中的样本大多数所属的类别确定。从中可以看出，k 最近邻分类主要的问题是确定合适的样本集、距离函数、组合函数和 k 值。对于多种类型的属性，距离函数可参照聚类分析中样本相似性的度量公式，而组合函数可以用简单无加权投票(voting)或加权投票的方法。在简单无加权投票中，每个近邻 x_i 对 x 分类的影响都被认为是相同的。通过对 k 个近邻 x_i 所属的类别计数，把 x 归为计数最多的类。

$$\max_{C_j} \sum_{i=1}^{k} \eta(x_i \in C_j)$$

其中：η 表示计数函数，如果 $x_i \in C_j$，则 $\eta(x_i \in C_j)=1$；否则 $\eta(x_i \in C_j)=0$。

当所属分类计数相同时，为 x 随机选取一个类别。加权投票对每个计数加权。

$$\max_{C_j} \sum_{i=1}^{k} w_i \eta(x_i \in C_j)$$

其中：权值一般定义为 $w_i=1/d(x,x_i)^2$，$d(x,x_i)$表示样本 x 与近邻 x_i 的距离。

k 最近邻分类器基于局部的数据进行预测，对噪声比较敏感。k 值的选择与数据有关。过大的 k 值可以减小噪声的影响，但使未确定类别样本点的近邻样本数量很大，可能导致分类错误。而过小的 k 值可能导致投票失效或者受噪声影响。一个较好的 k 值可通过各

种启发式技术来获取。

找出某样本的最近邻样本可能计算所有样本对之间的距离。为有效地发现最近邻,可以利用聚类算法对训练样本集进行聚类,如果两个簇的中心相距比较远,则对应簇中的样本一般不可能成为近邻。只要计算相邻簇的样本之间的距离即可寻找某样本的近邻。

2. 基于案例的推理

基于案例的推理(case-based reasoning)通过调整已解决的相似案例(源案例)的解作为新问题(目标案例)的解。基于案例的推理模仿人类的思维过程,以案例库作为知识的表示形式,搜索与新问题相似的案例,然后通过调整相似案例的解决方法得到新问题的解。作为一种典型的 k 最近邻分类器,基于案例的推理也需要确定合适的距离函数和组合函数来度量目标案例与源案例的相似性。目前,基于案例的推理已在医疗诊断、法律和房地产评估等领域得到了成功应用。

案例可以表示为特征向量的形式,由案例的特征及其解(solution)组成。例如,一部数码相机的价格由其品牌、像素、光学变焦倍数等因素决定,那么这些因素可以组成特征向量。对于一个新问题,首先表示成案例的形式,然后按照一定的相似度搜索方法在案例库中搜索与目标案例相似度高的案例。基于案例的推理方法的重要步骤是相似度的评价,一般比较目标案例与源案例的相似程度。最后把最相似的案例集的解决方案进行一定的调整、组合作为目标案例的解,这里的调整可根据具体的应用情景设置调整规则。新问题及其解也会被加入案例库。案例库的案例数量对以后的推理效果影响很大。基于案例的推理过程如图 5.30 所示。

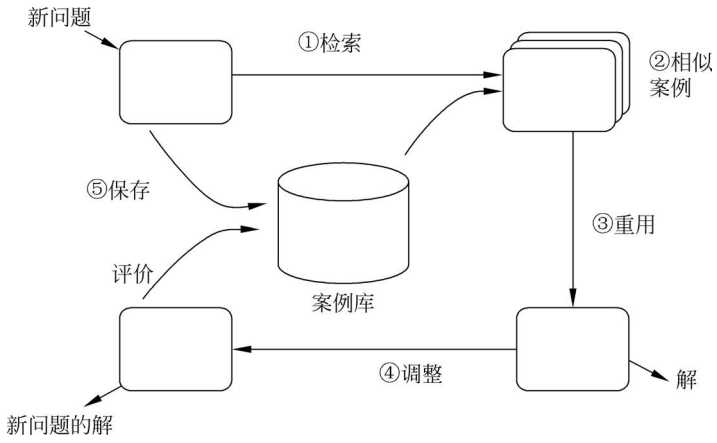


图 5.30 基于案例的推理过程

3. 遗传算法

遗传算法最初是由美国密歇根大学的 J. Holland 教授提出的,借鉴了生物进化中的遗传、基因变异、杂交和自然选择的思想。这种算法一般用于解决全局优化问题,已被广泛地用于组合优化、机器学习、信号处理、自适应控制和人工生命等领域。

遗传算法具有很多优越的性能。首先它是一个高效并行并且有全局搜索能力的算法,

可以大大减少计算时间。遗传算法操作的对象是一组可行解(种群),而不是单个解(个体),有良好的并行性和全局搜索能力,不易陷入局部最优解。遗传算法按概率搜索,在有噪声的情况下仍然能够有很大的概率找到最优解。此外,因为遗传算法以适应度作为唯一的搜索信息,所以很适合解决高复杂度的非线性优化问题。

遗传算法是从可能潜在解集的一个种群开始的,种群由经过基因编码的一定数目的个体组成。在遗传算法中,基因是杂交、变异操作的最基本的单位。遗传算法的搜索策略是通过个体基因的交换来实现的。

首先通过基因编码将初始种群表示成计算机能够处理的二进制字符串。初始化种群有两种策略:根据已有的知识,有选择性地构造一些解;也可以随机生成一定数目的个体。例如,长度为 8 的编码方式可能为 00010010,每个编码都是问题的可行解。如果要求解函数 $y=(x-25)^2+1$ 的最小值,二进制编码 00010010 等价于 $x=18$。初始种群通常不是最优解。适应度函数用于评价个体的适应程度,如果不满足优化准则,则产生新一代。按照适应度选择父代中的个体,通过杂交产生后代,然后按一定概率变异。后代适应度被重新计算,子代替换父代构成新的种群。上述过程循环执行,直至满足优化准则为止。

下面简要介绍选择、杂交和变异等操作。

(1) 选择是为了从当前群体中选出优良的个体,常用的方法是轮盘选择方法。其中个体被选中的概率与它们的适应度成正比,适应度越高的个体,被选中的概率也越大。优秀个体有更多的机会繁殖后代,但适应度低的染色体则容易被淘汰,以产生局部最优解。

(2) 杂交是两个个体的基因部分交换产生后代的过程。通过这种方式,后代继承了父代基因中的优秀部分,使子女个体的适应度提高。杂交概率通常取 0.7。下面介绍几种常见的杂交算子。

① 单点杂交算子:等概率随机确定一个基因的位置作为杂交点,取第一个染色体的杂交点前半部分,取第二个染色体的杂交点后半部分,生成一对新的染色体。举个单点杂交算法的例子:

$$\begin{matrix}00000010 \\ 00001111\end{matrix} \xrightarrow{\text{随机选择一个杂交点,如第 5 个基因}} \begin{matrix}00000111 \\ 00001010\end{matrix}$$

② 单点随机杂交算子:等概率随机确定一个基因的位置作为杂交点,以一定概率交换两个个体的后半部分,取第一个个体作为杂交结果。

③ 均匀随机杂交算子:独立地以一定的概率把父代个体的相应分量交换为另一父代个体的相应分量。

(3) 变异是将个体的某些基因变化后形成新的个体,例如,个体 00000010 的第二个基因发生变异,形成新的个体 01000010。变异使群体有多样性,有助于突破局部最优解。

杂交概率一般较大,介于 0.65～0.9。变异概率较小,一般介于 0.001～0.01。

除了典型的遗传算法外,很多学者还进行了优化研究,目前已经衍生出多种不同的新版本,其中许多方法对个体的编码方式、适用度函数、遗传算子、控制参数以及执行策略等进行了改进。

【例 5.11】 应用遗传算法解决旅行商问题

下面以旅行商问题(traveling salesman problem,TSP)为例来介绍遗传算法的运算过程。TSP 是一种常见的优化问题。假设有一个旅行商人要拜访 n 个城市,他需要选择路径

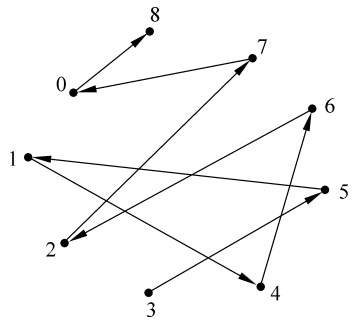


图 5.31 一条跨越 9 个城市的路线

长度最小的路径。路径的限制是每个城市只能拜访一次,最后还要回到原来出发的城市。这里假设 $n=9$,已知 9 个城市(编号为 0,1,…,8)的横纵坐标,可以计算出一条路线的长度,一条可能的路线如图 5.31 所示。

遗传算法应用于 TSP 的主要步骤如下。

(1) 个体编码。经典遗传算法的运算对象是表示个体的字符串。为了方便起见,这里采用城市的编号成串进行编码(符号编码),一个个体例子为 351462708。

(2) 初始群体。初始群体用随机方法产生。

(3) 适应度计算。遗传算法用个体的适应度来评价其优劣程度。这里用每代个体路线长度的倒数计算适应度,个体路线长度的倒数越大,说明路线越短,其适应度越高。

(4) 选择。把当前群体中适应度较高的个体通过一定的概率遗传到下一代群体中,这里用轮盘赌选择法。

(5) 交叉。由于传统的单点随机交叉可能会使城市重复访问,因此这里采用了部分映射交叉(部分映射交叉首先对个体编码串双点交叉,然后根据交叉区域内基因值之间的映射关系来修改未交叉区域的各个基因值)。例如,假设两个可能的父代路径 Parent1 与 Parent2 分别为 245103678 与 031276854,选取第 4～6 个基因后进行部分映射交叉后得到两个新个体。

Child1: 145276308

Child2: 762103854

(6) 变异。变异函数随机选择两个基因进行逆序变异,例如,对于个体 532174806,则变异后的新个体为 512374806。

对群体 $P(t)$进行一轮选择、交叉、变异运算之后可得到新一代的群体 $P(t+1)$,通常新一代的群体的适应度最值和均值都能得到改进,如此不断迭代后,就可以得到一个最佳的路线。

4. 组合方法

组合(ensemble)方法或集成方法是多种基分类算法的组合,结论是由所有基算法进行投票(用于分类问题)或者求加权平均(用于回归分析)。这些基分类算法往往是弱学习算法,即分类正确率仅比随机猜测略高的学习算法。它们组合后的效果可能优于强学习算法(识别准确率很高并能在多项式时间内完成的学习算法),因此受到了人们的关注。组合方法的主要问题是选择哪些独立的较弱学习模型以及如何把它们的学习结果整合起来。

不同分类方法的分类结果以某种方式(如多数表决或加权投票)组合起来。实验表明,组合方法得到的结果通常比单个分类方法得到的结果更加准确。常用的组合方法是对同一训练样本集用多种分类方法归纳不同的分类模型。此外,也可以对样本集按一定方式多次抽样得到多个训练样本集,在此基础上选择某分类方法归纳多个分类模型。这类方法包括装袋(bagging)法、提升(boosting)法和堆叠(stacking)法等。

(1) 装袋法。假设 S 为 n 个样本的集合,装袋法的过程大致如下:首先从样本集 S 中

采用多次放回抽样训练集 S_t，在每一个训练集 S_t 上选择特定的学习算法，都可以建立一个分类模型。对于一个未知类别的测试样本，每个分类模型都会返回一个预测结果(投票)，根据多数表决就可以确定测试样本可能的类。从上面过程可见，装袋法可以改善分类模型的泛化能力，对于噪声数据也不会过拟合。

(2) 提升法。提升法的基本思想是给每一个训练样本都分配一个权重，来确定它们在训练集的抽样分布，开始时所有样本的权重相同。然后，选择一个分类方法训练，归纳出一个分类模型。利用这个分类模型对样本集的所有样本进行分类，按下面方法更新训练样本的权重：增大错误分类的样本权重，减少正确分类的样本权重，使分类方法在随后的迭代中关注错误分类的样本。根据更新后的样本权重提升选择训练样本集，进入下一轮训练。如此迭代，得到一系列分类模型。对于测试样本，把每轮训练得到的分类模型的预测结果加权平均，即可完成组合预测。

(3) 堆叠法。堆叠法主要是训练一个组合多个弱学习器的模型，首先训练多个不同的弱学习器，然后以这些弱学习器的输出作为输入来训练一个模型，从而得到一个最终的输出。首先在整个训练数据集上通过重采样方法得到多个训练子集，然后分别在这些训练子集上进行训练，将这些弱分类器预测得到的结果作为下一层分类器(元分类器)的输入，最后将元分类器得到的结果作为最终的预测结果。

下面简要介绍梯度提升决策树(gradient boosting decision tree，GBDT)、AdaBoost 和随机森林(random forest)等几种常用的组合分类方法。

1) GBDT 算法

GBDT 是一种迭代的回归树算法，该算法由多棵决策树组成，每一棵决策树是从训练前面所有决策树的残差中来学习，所有决策树的输出结果综合得到最终结果。为了防止过拟合，GBDT 支持提升法。

提升树(boosting tree)采用加法模型，主要思想是不断拟合残差。GBDT 的含义就是用梯度提升的策略训练出来的决策树模型，即使用损失函数的负梯度在当前模型的值作为回归问题提升树算法的残差近似值。

GBDT 算法的输出是一组回归分类树，组合各个树输出结果：

$$\hat{y} = \sum_{k=1}^{K} f_k(x), \quad f_k \in F$$

其中：$\hat{y}$ 即对于输入 f_k 的预测值；f_k 表示样本到树的映射；F 表示所有树组成的函数空间。

$$F^{(t)}(x) = F^{(t-1)}(x) + f_t(x)$$

在训练 GBDT 模型时，每一步只学习一个基函数及其参数，得到一个决策树，优化损失函数：

$$\hat{y}_i^t = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{t-1} + f_t(x_i)$$

在第 t 步的学习中，对 x_i 的预测为 $\hat{y}_i^t = \hat{y}_i^{t-1} + f_t(x_i)$，$f_t(x_i)$ 为这一步中学习生成的决策树，则损失函数的形式可以写成

$$\sum_{i=1}^{n} l(y_i, \hat{y}_i^t) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{t-1} + f_t(x_i))$$

当损失函数使用平方损失函数时，可以写为

$$\sum_{i=1}^{n}(y_i-(\hat{y}_i^{t-1}+f_t(x_i)))^2=\sum_{i=1}^{n}[2(\hat{y}_i^{t-1}-y_i)f_t(x_i)+f_t(x_i)^2]+C$$

其中：$(\hat{y}_i^{t-1}-y_i)$为残差，即上一步 x_i 的预测值 $\hat{y}_i^{t-1}$ 与实际值 y_i 的差值；C 为常数项。因此当 GBDT 算法的损失函数为平方损失函数时，每一步生成的决策树是对上次决策树模型残差的拟合。计算负梯度：

$$r_{it}=\left[\frac{\partial L(y_i,f_{t-1}(x_i))}{\partial f_{t-1}(x_i)}\right]$$

然后按照梯度下降法修正其中的参数，得到一棵新的回归树 $f_t(x)$，寻找 $f_t(x)$的最佳划分点需要遍历每个特征的每个可能值，使得损失函数最小的那个划分点即为最佳划分点。再更新模型 $F^{(t)}(x)=F^{(t-1)}(x)+f_t(x)$。如此迭代，直至输出最终的回归树 $\hat{y}=\sum_{k=1}^{K}f_k(x)$，$f_k\in F$。

基于 GBDT，XGboost 算法在计算损失函数时引入了二阶泰勒公式，在损失函数上引入了正则化。有兴趣的读者可以查阅相关资料。

```
# 导入必要的库
import xgboost as xgb
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from xgboost import plot_importance

# 加载数据
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns = data.feature_names)
y = data.target

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.2, random_state = 42, stratify = y)

# 转换为 DMatrix 格式(XGBoost 的高效数据结构)
dtrain = xgb.DMatrix(X_train, label = y_train)
dtest = xgb.DMatrix(X_test, label = y_test)

# 设置模型参数
params = {
    'objective': 'binary:logistic',          # 二元分类问题
    'eval_metric': 'logloss',                # 评估指标
    'eta': 0.1,                              # 学习率
    'max_depth': 6,                          # 树的最大深度
    'subsample': 0.8,                        # 每棵树随机采样的比例
    'colsample_bytree': 0.8,                 # 每棵树随机采样的列比例
    'seed': 42,                              # 随机种子
    'nthread': 4                             # 并行线程数
}

# 训练模型
```

```
num_round = 100                              # 迭代次数
evals = [(dtrain, 'train'), (dtest, 'eval')]
model = xgb.train(
    params,
    dtrain,
    num_round,
    evals = evals,
    early_stopping_rounds = 10,              # 早停轮数
    verbose_eval = 10                        # 每 10 轮打印一次评估结果
)

#模型评估
y_pred_prob = model.predict(dtest)
y_pred = np.round(y_pred_prob)               # 将概率转换为类别

print("\n模型评估:")
print(f"准确率: {accuracy_score(y_test, y_pred):.4f}")
print("\n分类报告:")
print(classification_report(y_test, y_pred))
print("\n混淆矩阵:")
print(confusion_matrix(y_test, y_pred))

# 使用 scikit-learn API 进行网格搜索优化
print("\n开始网格搜索优化...")
# 转换为 scikit-learn API 的模型
sk_model = xgb.XGBClassifier(**params)

# 定义参数网格
param_grid = {
    'max_depth': [3, 6, 9],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'n_estimators': [50, 100, 200]
}
# 网格搜索
grid_search = GridSearchCV(
    estimator = sk_model,
    param_grid = param_grid,
    cv = 5,
    scoring = 'accuracy',
    n_jobs = -1,
    verbose = 1
)
grid_search.fit(X_train, y_train)

print(f"\n最佳参数: {grid_search.best_params_}")
print(f"最佳准确率: {grid_search.best_score_:.4f}")

# 使用最佳模型进行预测
best_model = grid_search.best_estimator_
y_pred_best = best_model.predict(X_test)
print("\n优化后模型评估:")
print(f"准确率: {accuracy_score(y_test, y_pred_best):.4f}")
```

```
print("\n优化后分类报告:")
print(classification_report(y_test, y_pred_best))
```

2) AdaBoost 算法

AdaBoost 算法是一种适用于数值型和非数值型数据的分类算法,属于提升算法。与决策树、最邻近分类、贝叶斯分类等算法不同,AdaBoost 算法顺序建立多棵决策树,通过训练效果改变样本的权重,不断提升决策树的训练性能。AdaBoost 算法对于大的数据量准确率会达到比较高的水平。

AdaBoost 算法具有明显的提升算法特征:依次训练多个弱分类器,通过多轮训练,不断提升对训练样本的拟合质量。第一次迭代时,每个训练样本的权值是相同的。然后经过多次迭代,每次迭代计算弱分类器的错误率,根据错误率调整训练样本的权值,对于错分的样本会增加其权重。如此,每次训练都会产生一个弱分类器,直到弱分类器的错误率较小或者达到最大迭代次数。当训练完成时,AdaBoost 算法将上述训练得到的多个弱分类器按照一定的方法组合成一个强分类器,可以预测新样本的类别。

对于二分类问题,已知训练样本 $X_i=(x_1,x_2,\cdots,x_n)(i=1,2,\cdots,N)$,$Y_i=[-1,+1]$。$X_i$ 为输入属性向量,Y_i 为输出向量。D_t 表示训练样本集的权值分布,t 表示算法迭代次数。w_{ti} 表示第 t 次迭代中训练样本集的权值。H_t 表示基分类器,H_{final} 表示最终强分类器。AdaBoost 算法的具体流程如下。

(1) 确定训练数据的初始权重。开始训练样本有相同的权值,N 个训练样本则权值为 $1/N$。w_{1i} 表示第一次迭代训练样本集的权值 $D_1=(w_{11},w_{12},\cdots,w_{1i},\cdots,w_{1N})=(1/N,1/N,\cdots,1/N,\cdots,1/N)$。

(2) 迭代 T 次,第 t 次的计算过程如下。

① 选择一个误差率最低的阈值来训练基分类器 H_t。

$$H_t(X)\rightarrow[-1,+1]$$

② 计算 $H_t(X)$在权值分布 D_t 上的误差。$H_t(X)$在训练集上的误差率 e_t 是 $H_t(X)$被错误分类样本的权值和。

③ 计算 $H_t(X)$的系数 α_t,表示 $H_t(X)$在强分类器中的重要程度:

$$\alpha_t=\frac{1}{2}\ln\frac{1-e_t}{e_t}$$

当 e_t 减小时,α_t 增大,表示基分类器的分类误差率越小,在强分类器中作用越大。

④ 计算训练样本集下一次的权值分布 D_{t+1}。

$$D_{t+1}=(W_{t+1,1},W_{t+1,2},\cdots,W_{t+1,i},\cdots,W_{t+1,N})$$

$$w_{t+1,i}=\frac{w_{ti}}{Z_t}\exp(-\alpha_t y_i H_t(x_i))\quad(i=1,2,\cdots,N)$$

其中:Z_t 是一个规范化常数。

$$Z_t=\sum_{i=1}^{N}w_{ti}\exp(-\alpha_t y_i H_t(x_i))$$

组合上述弱分类器得到最终的强分类器:

$$f(x)=\sum_{t=1}^{T}\alpha_t H_t(x)$$

根据 $f(x)$函数的符号可以确定新样本的类别。

对于多分类问题，AdaBoost 算法按照上述步骤类似计算，但计算基本分类器在最终强分类器中所占的比重时有所不同。

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets

dataset_all = datasets.load_breast_cancer()
X = dataset_all.data
Y = dataset_all.target

seed = 42
kfold = KFold(n_splits = 10, random_state = seed, shuffle = True)
dtree = DecisionTreeClassifier(criterion = 'gini',max_depth = 3)
dtree = dtree.fit(X, Y)
pre = dtree.predict(X)
result = cross_val_score(dtree, X, Y, cv = kfold)
print("决策树结果:",result.mean())

model = AdaBoostClassifier(base_estimator = dtree, n_estimators = 100,random_state = seed)
result = cross_val_score(model, X, Y, cv = kfold)
print("提升法改进结果:",result.mean())
```

3）随机森林

随机森林是 Leo Breiman 在 2001 年提出的一种集成学习方法，扩展了传统决策树方法，提高了学习的泛化能力和精度。这种方法采用随机的方式进行抽样，然后利用决策树算法得到多个决策树（森林），预测时由这些树的预测结果组合确定。随机森林常利用分类回归树（CART）作为基学习器。

随机森林首先对输入的数据进行行、列的随机采样，其中行采样采用有放回的方式，这样每棵树都进行独立的随机抽样，不容易出现过拟合现象。而列采样是从众多条件属性中，随机选择适当数量的部分属性，通常这个数量取属性总个数的平方根或者 $\log_2$（属性数）。在随机森林中，还可以使用一个随机性，在决策树分支时，在最好的几个属性中随机选择一个进行分支，这样就可以保证基算法之间的独立性。

随机采样后使用完全分裂的方式建立决策树。当给定一个新样本时，用森林中的每一棵决策树分别对这个样本进行分类，森林中多个决策树给出类别最多的分类是样本的可能类别。有学者曾比较了 179 种不同的分类学习算法在著名的机器学习数据库 UCI 中 121 个数据集的表现，发现随机森林的准确度最高。

由于随机森林多棵树之间是独立构建的，因此在大数据环境下可以并行处理。

随机森林用于生存预测的 Python 代码如下：

```
from sklearn.model_selection import train_test_split #for split the data
from sklearn.metrics import accuracy_score #for accuracy_score
from sklearn.model_selection import KFold #for K-fold cross validation
from sklearn.model_selection import cross_val_score #score evaluation
from sklearn.model_selection import cross_val_predict #prediction
from sklearn.metrics import confusion_matrix #for confusion matrix
from sklearn.ensemble import RandomForestClassifier
```

```
all_features = traindf.drop("Survived",axis = 1)
Targeted_feature = traindf["Survived"]
X_train,X_test,y_train,y_test = train_test_split(all_features,Targeted_feature,test_size =
0.3,random_state = 42)
# X_train.shape,X_test.shape,y_train.shape,y_test.shape
model = RandomForestClassifier(criterion = 'gini',n_estimators = 700,min_samples_split = 10,
min_samples_leaf = 1, max_features = 'auto',oob_score = True,random_state = 1,n_jobs =- 1)
model.fit(X_train,y_train)
prediction_rm = model.predict(X_test)
print('随机森林分类的准确度是', round(accuracy_score(prediction_rm,y_test) * 100,2))
```

对于较大的数据集,可以按照 8∶1∶1 或 7∶2∶1 的比例样本集分成训练集、验证集和测试集。训练集用来训练模型;验证集用于在每一步测试模型;测试集用于训练后评估模型。

5.6 关联分析

关联分析是数据挖掘的一个重要方法,最近几年已被业界广泛关注。关联是指在两个或者多个变量之间存在某种规律性,但关联并不一定意味着因果关系。关联规则是寻找在同一事件中出现的不同项目的相关性,关联分析是挖掘关联规则的过程。早期关联分析用于分析零售企业顾客的购物行为模式,所以关联分析又被称为购物篮分析。关联规则也可用于商品货架布置、销售配货、存货安排、购物路线设计、商品陈列设计、交叉销售以及根据购买模式对用户进行分类等方面。比较经典的关联规则是啤酒和尿布的故事,Wal-Mart 超市通过关联分析发现,一定比例的顾客在购买尿布的同时也有购买啤酒的行为特点,于是把尿布和啤酒摆在一起出售,从而使两者的销量都增加了。

关联规则也广泛应用于其他各个领域,例如,信用卡公司、银行和股票交易所可以通过关联规则识别欺诈行为,通信行业可以通过关联规则确定顾客的主要来源。此外,关联规则在医学领域也有应用。

5.6.1 关联规则

关联规则是由 Agrawal 等在 1993 年提出的。关联规则的一个最简单的应用是购物篮分析,例如,购买山地车的顾客可能会同时购买其他商品:头盔还是手套? 顾客在购买面包(bread)的同时也会购买奶制品(dairy)的购物模式,可以用关联规则来描述:bread→dairy [support=3%,confidence=60%]。通过关联分析也可以找出顾客群的特征。例如,年龄小于 24 岁且没有住房的顾客,倾向于同时购买鱼和水果蔬菜。

下面先介绍关联规则的几个基本定义。

1. 项目

集合 $I=\{i_1,i_2,\cdots,i_m\}$ 称为项集合,其中 m 为正整数,$i_k(k=1,2,\cdots,m)$ 称为项目。项目是从具体问题中抽象出来的一个概念。在超市的关联规则挖掘中,项目表示顾客购买的各种商品,如牛奶、面包和旅游鞋等。

2. 事务

在购物篮分析中,一般不关心顾客购买的商品数量和价格等因素,因此顾客的一次购物

可以用该顾客购买的商品表示，称为事务，所有事务的集合构成关联规则挖掘的数据集，称为事务数据库。关联规则挖掘的事务数据库记为 D，其中的每个元组称为事务。事务 T 是项目的集合，每一个事务都有唯一的标识，如表 5.10 所示。

表 5.10 事务数据集

事务	购买商品(项集)	事务	购买商品(项集)
10	i_1、i_2、i_3	30	i_1、i_4
20	i_1、i_3	40	i_2、i_5、i_6

在进行关联规则挖掘时，若把超市所有销售的商品作为一个集合，每个商品均用一个布尔值描述是否被购买，则每个顾客购物清单就可以用一个布尔向量来表示，分析相应布尔向量就可得到哪些商品会在一起被购买的购物模式。

除上面的表示方法外，事务数据集也可表示成矩阵的形式 $\boldsymbol{D}=(d_{ij})_{n\times m}$，此矩阵的行表示事务，列表示项目，$d_{ij}=1$ 或 0，表示某事务包含或不包含某项目。这种方式比较容易计算项目或项集的支持度，但会导致矩阵过于稀疏。表 5.11 是表 5.10 事务数据集对应的矩阵表示形式。

表 5.11 事务数据集的矩阵表示形式

事务	项 目					
	i_1	i_2	i_3	i_4	i_5	i_6
10	1	1	1	0	0	0
20	1	0	1	0	0	0
30	1	0	0	1	0	0
40	0	1	0	0	1	1

3. 项集

项集是由 I 中项目构成的集合。若项集包含的项目数为 k，则称此项集为 k-项集。任意的项集 U 和事务 T 若满足 $T\supseteq U$，则称事务 T 包含项集 U。

4. 频繁项集

对任意的项集 U，数据库 D 中的事务包含项集 U 的比例为 ε，则项集的支持度为 $\varepsilon=\text{support}(U)$，其中包含项集 U 的事务数称为项集 U 的支持数，记为 support_count(U)。在购物篮分析中，项集的支持度反映了其流行程度(prevalence)。若项集 U 的支持度大于或等于用户指定的最小支持度(minsupport)，则项集 U 称为频繁项集(或大项集)，否则项集 U 称为非频繁项集(或小项集)。这里的最小支持度较难确定，太大或太小都会影响关联分析的结果。如果频繁项集有 k 个项目，那么称此频繁项集为 k 频繁项集。数据库 D 中的事务数记为 $|D|$，频繁项集是至少被 $\varepsilon|D|$ 条事务包含的项集。

5. 支持度、置信度和提升度

关联规则是形如 $U\rightarrow V$ 的规则，其中 U、V 为项集，且 $U\cap V=\varnothing$。项集 U、V 不一定包含于同一个项集合 I，例如分析具有某些特征 U 的顾客有消费某些商品 V 的关联行为，这里

的 U 是顾客的性别、职业、住址等属性集合,而 V 是商品集合。关联规则的支持度 $s(U\rightarrow V)$ 和置信度 $c(U\rightarrow V)$是度量项目关联的重要指标,它们分别描述了一个关联规则的有用性和确定性。支持度是用户兴趣的重要度量,支持度很低的关联规则表示随机现象。置信度反映了关联规则的正确程度(accuracy),即购买了项集 U 中的商品的顾客同时也购买了 V 中商品的可能性(条件概率)有多大。

在事务数据库 D 中,关联规则 $U\rightarrow V$ 的支持度为

$$s(U\rightarrow V)=\frac{\text{support\_count}(U\cup V)}{|D|}$$

关联规则 $U\rightarrow V$ 的置信度为

$$c(U\rightarrow V)=\frac{\text{support\_count}(U\cup V)}{\text{support\_count}(U)}$$

其中:support_count($U\cup V$)是包含项集 $U\cup V$ 的事务数;support_count(U)是包含项集 U 的事务数。

关联规则 $U\rightarrow V$ 的提升度(lift)是一个关于该规则有效性的度量,表示为

$$\text{lift}(U\rightarrow V)=\frac{c(U\rightarrow V)}{\text{support}(V)}$$

当提升度大于 1 时,与单纯依赖项集 V 在事务数据库的支持度相比,关联规则能更好地预测项目 V 是否会出现。例如,一般有 30%的顾客会购买打印机,而通过关联分析发现,购买台式计算机的顾客在此次购物中有 60%的顾客也会购买打印机。因此对于购买了台式计算机的顾客,他购买打印机的可能性会增加 60%/30%=2 倍。

6. 强关联规则

若关联规则 $U\rightarrow V$ 的支持度和置信度分别大于或等于用户指定的最小支持度(minsupport)和最小置信度(minconfidence),则称关联规则 $U\rightarrow V$ 为强关联规则,否则称关联规则 $U\rightarrow V$ 为弱关联规则。关联规则挖掘的核心是找出事务数据库 D 中的所有强关联规则。这里的最小支持度是指项集在事务数据库中出现的次数占总事务数的最低比率,最小置信度是指两个项集 U 和 V,在所有出现的 U 项集中,同时出现的最小比例。

下面讨论关联规则分类。

(1) 根据处理的项目类别,关联规则可以分为布尔型和数值型。布尔型关联规则处理的项目都是离散的,它显示了这些变量之间的关系。例如,性别=“女”→职业=“秘书”,而数值型关联规则可以和多维关联或多层关联规则结合起来。对数值型属性进行处理,参考连续属性离散化方法或统计方法对其进行分割,确定划分的区间个数和区间宽度。数值型关联规则中也可以包含可分类型变量。例如,性别=“女”→平均收入>2300,这里的收入是数值类型,所以是一个数值型关联规则。

(2) 基于规则中项目的抽象层次,可以分为单层关联规则和多层关联规则。项目(概念)通常具有层次性,如图 5.32 所示。

位于项目层次较低层的项目通常支持度也较低,而泛化到高层概念可能获取一些有趣的模式,但也要注意,在概念分层的较高层发现的规则可能没有实用价值。例如,下面的多层关联规则:

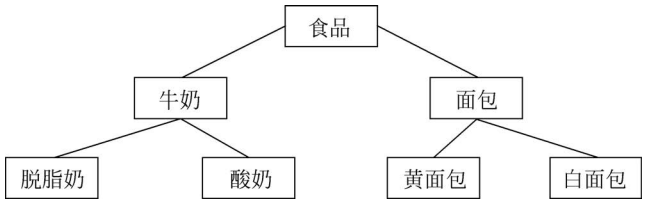


图 5.32 项目层次

① 牛奶→面包[20%,60%];

② 酸奶→黄面包[6%,50%]。

(3) 基于规则包含的项目维数,关联规则可以分为单维和多维两种。单维关联规则处理单个项目的关系,多维关联规则处理多个项目之间的某些关系。例如,下面的关联规则分别是单维和多维关联规则。

① buys(x,"diapers")→buys(x,"beers")[0.5%,60%];

② major(x,"CS")^takes(x,"DB")→grade(x,"A")[1%,75%]。

这里的关联规则不限于顾客购买的商品之间的关联,顾客的年龄、职业、信誉度、收入和地址等也可能作为项目。不同的项目可以在不同的概念层次,这种关联规则也称为交叉关联规则,从而能发现更一般的关联规则,但应避免由此带来的冗余关联规则问题。此外,有些谓词需要通过对连续属性离散化实现概念分层,如顾客的年龄、收入等变量:age(x,[30,39]) ^ income(x,[42,48]) →buys(x,"PC")[1%,75%]。这里的项目用谓词表示,其中 x 是变量,泛指顾客,^表示逻辑与。

(4) 分离关联规则(dissociation rule),也称为负相关模式。分离关联规则与一般的关联规则相似,只是在关联规则中出现项目的反转项,在购物篮分析中可发现不在一起购买的商品。例如,购买牛奶的顾客一般不购买汽水。这里请读者思考一下,如何发现分离关联规则?

5.6.2 Apriori 算法

Apriori 算法的基本思想是先找出所有的频繁项集,然后由频繁项集产生强关联规则,这些规则必须满足最小支持度和最小置信度。

搜索所有的频繁项集需要多次搜索事务数据库 D,这是影响关联算法性能的主要因素。Apriori 算法是用 $k-1$ 频繁项集生成候选的 k 频繁项集,但候选频繁项集通常是很大的,例如,在购物篮分析中,m 个项目组成的项集可能产生 2^m-1 个候选频繁项集,以及 $3^m-2^{m+1}+1$ 个关联规则。但在一般情况下,这些规则大部分不满足强关联规则的条件,这个问题成为关联规则挖掘的瓶颈。因此,减少候选项集的大小,然后再扫描事务数据库,计算候选项集的支持度是必要的。如果最长的频繁项集是 n,那么需要 $n+1$ 次事务数据库扫描。因此,如何高效地找出频繁项集是关联规则挖掘的关键问题。

Apriori 算法利用"频繁项集的任何子集也一定是频繁的,或者非频繁项集的超集一定是非频繁的"先验性质减少频繁项集的搜索空间。图 5.33 所示为$\{i_1,i_2,i_3,i_4\}$的项集格(lattice),这种结构能枚举所有可能的项集。假设$\{i_2,i_3,i_4\}$是频繁项集,那么它的所有子集$\{i_2\}$、$\{i_3\}$、$\{i_4\}$、$\{i_2,i_3\}$、$\{i_2,i_4\}$和$\{i_3,i_4\}$都是频繁的。反之,如果$\{i_1,i_2\}$是非频繁的,那它的所有超集$\{i_1,i_2,i_3\}$、$\{i_1,i_2,i_4\}$和$\{i_1,i_2,i_3,i_4\}$都是非频繁的。

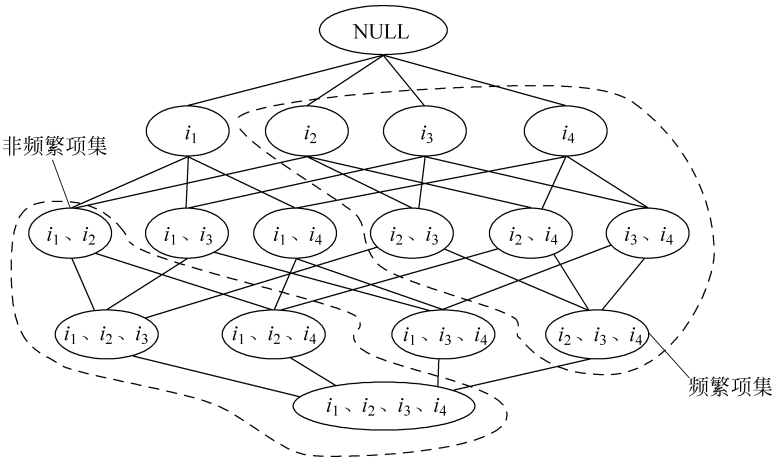


图 5.33 项集格

假定频繁项集 L_{k-1} 中的项目按英文字典顺序排列，由 $k-1$ 频繁项集生成候选的 k 频繁项集 apriori-gen(L_{k-1})需要进行下面操作：候选频繁项集的产生和修剪。这个步骤需要避免产生过多不必要的、重复的候选频繁项集，也不能遗漏候选频繁项集。下面是一种常用的候选频繁项集的产生和修剪方法。

由前 $k-2$ 项相同的一对 $k-1$ 频繁项集 L_{k-1} 连接生成候选 k 频繁项集 I_k。其中 $k=2,\cdots,n+1$，这里假设由 Apriori 算法得到的频繁项集的长度最大值为 n。这种方法既可保证候选频繁项集的完全性(不遗漏)，又可避免重复地产生候选频繁项集。

```
Insert into I_k select u.item_1, u.item_2, …, u.item_{k-1}, v.item_{k-1}
from L_{k-1} u, L_{k-1} v where u.item_1 = v.item_1, …, u.item_{k-2} = v.item_{k-2}, u.item_{k-1} ≠ v.item_{k-1}
```

然后计算候选频繁项集 I_k 的支持度或利用上面先验性质进行修剪得到频繁项集。有关提高频繁项目集的挖掘效率，目前已经有多种改进算法。第 13 章还介绍了分布并行 Apriori 算法。读者可以查阅资料，讨论如何进一步提高 Apriori 算法的效率。

下面讨论由频繁项集 U 产生强关联规则的问题。对于 $V \subset U$ 且 $V \neq \varnothing$，如果 support(U)/support(V)≥minconfidence，由定义知 $V \rightarrow (U-V)$ 为强关联规则。对 k 频繁项集，可能产生的强关联规则多达 2^k-2 个。

为减少强关联规则的搜索时间，Apriori 算法中关联规则的产生可应用下面定理：如果关联规则 $V \rightarrow (U-V)$ 不满足最小置信度要求，那么 $V' \rightarrow (U-V')$ 也一定不满足最小置信度要求，其中 $V' \subset V$。例如，假设 $\{i_1,i_2,i_3,i_4\}$ 是频繁项集，关联规则 $\{i_1,i_2,i_3\} \rightarrow \{i_4\}$ 不满足最小置信度要求，那么 $\{i_1,i_2\} \rightarrow \{i_3,i_4\}$、$\{i_1,i_3\} \rightarrow \{i_2,i_4\}$、$\{i_2,i_3\} \rightarrow \{i_1,i_4\}$、$\{i_1\} \rightarrow \{i_2,i_3,i_4\}$、$\{i_2\} \rightarrow \{i_1,i_3,i_4\}$ 和 $\{i_3\} \rightarrow \{i_1,i_2,i_4\}$ 等都不是强关联规则。

需要注意的是，强关联规则只是用户兴趣度的客观度量，可以排除一些无趣的规则，但并不一定反映用户主观的兴趣评价。有兴趣的读者可以查阅相关文献深入探讨。

这里利用 SPSS Modeler 对上述 mailshot.txt 文件选择 Apriori 算法进行关联分析，得到的部分关联规则如图 5.34 所示。

后项	前项	支持度 %	置信度 %
current_act	mailshot_YN car married	12.333	91.892
current_act	mortgage car		
current_act	mortgage region = INNER_CITY save_act	11.0	90.909
current_act	region = INNER_CITY car married save_act	10.333	90.323
current_act	mortgage car married	10.0	90.0
current_act	region = INNER_CITY car married	14.0	88.095
current_act	mortgage mailshot_YN	13.667	87.805
current_act	region = INNER_CITY married save_act	23.333	87.143
current_act	region = TOWN car save_act	10.333	87.097
current_act	car married save_act	23.0	86.957

图 5.34 Apriori 算法进行关联分析

【例 5.12】 基于关联分析的服装缺陷管理

客户在寻求低价格产品的基础上，对产品的质量提出了更高的要求。制造商为了保证自己在市场中的竞争力，严格进行质量管理。不幸的是，缺陷产品的存在是不可避免的。就服装制造业这种受人为因素影响较大的行业而言，工人缝纫技能之间的差别直接导致产品质量的差别。为了生产高质量、低成本的产品，质量管控工作显得尤为重要。在传统的服装缺陷检测过程中，检测人员对每一件商品认真检查，判断是否需要返工。但检测人员没有考虑到不同缺陷之间的联系和对缺陷的预测，增加了劳动强度，降低了检测的准确性。

服装制造通常采取流水线的方式生产，这是一个复杂的过程，其中牵涉多个部门。因而很难确定每个缺陷的责任部门，缺陷的根源无从查询。一件产品身上的缺陷可能多达上百处，这么大规模的缺陷信息缺少一套有效的管理和分析机制。低效率的缺陷识别将给服装行业带来诸多不良后果，例如客户满意度降低、返工成本高、生产周期长。这里利用关联分析挖掘服装缺陷关联规则，达到有效管理质量数据，确定重复发生缺陷的根源的目的。

把不同部门存储的有关制造流程和产品质量的数据从不同的数据库中提取出来。因为不同部门使用的数据结构不一致，需要把这些数据预处理，统一格式后存储到数据仓库中。然后对类似表 5.12 所示的多维数据利用 Apriori 算法进行关联分析，得到产品缺陷的关联规则，识别隐藏的产品缺陷模式。其中 i_1、i_2、i_3、i_4、i_5 和 i_6 分别表示断线、开线、裤腿不对称、掉色、口袋变形和明缝。支持度设置为 25%，置信度为 90%。产生的部分关联规则如表 5.13 所示。

表 5.12 缺陷记录

缺陷记录	产品编号	缺陷 i_1	缺陷 i_2	缺陷 i_3	缺陷 i_4	缺陷 i_5	缺陷 i_6
1	005	i_1				i_5	
2	028	i_1	i_2				i_6
3	032	i_1	i_2				i_6
4	058		i_2	i_3			i_5
5	098						
6	105		i_2	i_3		i_5	
7	110	i_1	i_2	i_3		i_5	i_6
8	153			i_3	i_4		i_6
9	170	i_1	i_2				i_6
10	190		i_2	i_3		i_5	
11	197	i_1			i_4		
12	199	i_1	i_2				i_6

表 5.13 产生的部分关联规则

条件项	结果项	支持度/%	置信度/%
$i_1 i_2$	i_6	41.70	100
$i_1 i_6$	i_2	41.70	100
$i_2 i_6$	i_1	41.70	100
i_1	$i_2 i_6$	41.70	71.53
i_2	$i_1 i_6$	41.70	62.52
i_6	$i_1 i_2$	41.70	83.4
$i_2 i_3$	i_5	33.30	100
$i_2 i_5$	i_3	33.30	100
$i_3 i_5$	i_2	33.30	100
i_2	$i_3 i_5$	33.30	49.93
i_3	$i_2 i_5$	33.30	79.86
i_5	$i_2 i_3$	33.30	79.86

利用上面关联分析得到的关联规则,制订一套有效的质量改善计划。

(1) 利用服装缺陷的关联规则,可以看出存在一种服装缺陷,那么另一些缺陷也可能同时存在。利用这种关联模式,质量管理小组就可以在一种缺陷存在的情况下,有效预测潜在的其他缺陷。

(2) 找出关联规则中缺陷同时存在的原因。可以看出,某些缺陷同时存在,这种缺陷之间的联系可能是由特定的原因导致。为了降低这种缺陷发生的概率,需要从根源解决问题。例如缺陷 i_1、i_2 发生时,i_6 也可能会同时发生,经过追查,根本原因为缝纫工人做工差。

(3) 在找出导致缺陷发生的根源后,质量管理人员应把更多的人力、设备等资源投入解决这些问题的根源。例如针对缝纫工人做工差的问题,可以为缝纫工人提供更多的培训,从而降低缺陷发生的概率。

5.6.3 FP 增长算法

尽管 Apriori 算法利用频繁项集的任何子集也是频繁的启发式，减少了候选频繁项集的大小，但仍然会产生大量的候选频繁项集，对事务数据库的重复扫描带来很大的开销。

与 Apriori 算法不同，频繁模式增长(frequent pattern growth)算法，简称 FP 增长算法，使用一种称为 FP 树的数据结构，并且采用分而治之的策略，无须产生候选频繁项集就能得到全部的频繁项集。

1. 构造 FP 树

FP 树是事务数据库的压缩表示，每个事务都映射到 FP 树中的一条路径。不同的事务可能包含若干相同的项目，因此这些路径会有所重叠，使得事务数据能得到一定程度的压缩。FP 增长算法挖掘频繁项集的过程如下。

(1) 首先搜索事务数据库 D，找到 1 频繁项集及其支持数。继续考虑表 5.13，假设最小支持数仍为 2，按支持数递减排序，其结果记为 $L=[i_2:8, i_1:7, i_3:7, i_4:2, i_5:2]$。

(2) 构造 FP 树。创建 FP 树的根节点，用符号 null 标记。第二次搜索事务数据库 D，按 L 中的次序排列每个事务的项集，并对每个事务创建由根节点 null 出发的路径。

例如，对表 5.14 事务数据库按 L 重新排序，第一个事务按 L 的次序为$\{i_2, i_1, i_4\}$。构造 FP 树的第一个分支$<(i_2:1),(i_1:1),(i_4:1)>$，其中的数字表示节点的计数。读取第二个事务时会产生第二个分支，然而该分支与第一个事务共享前缀分别为 i_2 和 i_1，这时把共享前缀的节点计数加 1。扫描所有事务后得到的 FP 树如图 5.33 所示。为了方便遍历，FP 树还包含连接具有相同节点的指针列表，在图 5.35 中用虚线表示。

表 5.14 某日超市的购物记录

交易时间	购买商品	交易时间	购买商品
14:25	$i_1、i_2、i_4$	18:55	$i_2、i_3$
15:07	$i_1、i_2、i_3$	19:26	$i_1 i_2、i_5$
16:33	$i_2、i_3$	19:52	$i_2、i_4$
17:05	$i_1、i_3$	20:03	$i_1、i_2、i_3$
18:40	$i_1、i_2、i_3、i_5$	20:16	$i_1、i_3$

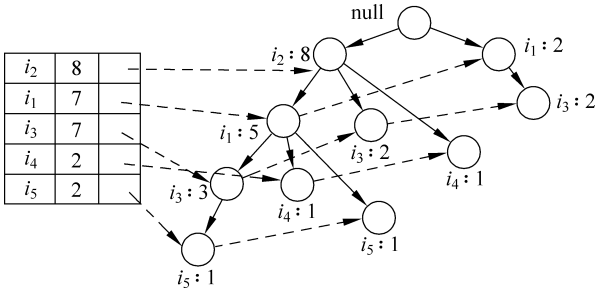


图 5.35 构造 FP 树

2. 利用 FP 树产生频繁项集

FP 增长算法采用自底向上的方式搜索 FP 树,由 L 的倒序开始,对每个 1 频繁项集构造条件 FP 树,然后递归地对该条件 FP 树进行挖掘:首先考虑 i_5,i_5 出现在 FP 树的两个分支$<(i_2i_1i_5:1)>$和$<(i_2i_1i_3i_5:1)>$中,如图 5.36 所示。此处 i_5 的前缀路径有两条:$<(i_2i_1)>$和$<(i_2i_1i_3:1)>$。由于 i_3 的支持数只有 1,因此生成的条件 FP 树只包含单个路径,即$<(i_2:2,i_1:2)>$,因此该路径产生的频繁项集的所有组合为$<(i_2,i_5:2)>$、$<(i_1,i_5:2)>$、$<\{i_2,i_1,i_5:2\}>$。类似地,表 5.15 描述了挖掘 FP 树得到的条件 FP 树及其产生的频繁项集。

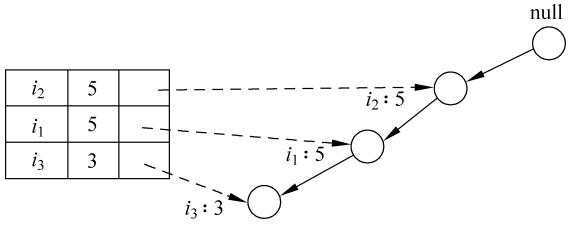


图 5.36 i_5 的条件 FP 树

表 5.15 挖掘条件 FP 树

项	前缀路径	条件 FP 树	产生的频繁项集
i_5	$\{(i_2,i_1,i_3:1),(i_2,i_1:1)\}$	$<i_2:2,i_1:2>$	$i_2i_5:2$、$i_1i_5:2$、$i_2i_1i_5:2$
i_4	$\{(i_2,i_1:1)(i_2:1)\}$	$<i_2:2>$	$i_2i_4:2$
i_3	$\{(i_2,i_1:3),(i_2:2)(i_1:2)\}$	$<i_2:3,i_1:3><i_2:2>,<i_1:2>$	$i_2,i_1,i_3:3$、$i_2i_3:5$、$i_1i_3:5$
i_1	$\{i_2:5\}$	$<i_2:5>$	$i_2i_1:5$

研究表明,FP 增长算法对不同长度的频繁模式有很好的适应性,同时在效率上比 Apriori 算法有较大的提高。如果 FP 树存储在内存中,那么就可以直接从内存中提取频繁项集,而不必重复扫描硬盘上的事务数据。

Apriori 和 FP 增长关联算法 Python 代码如下:

```
import pandas as pd
from apyori import apriori
import pyfpgrowth

def loadData():
    # 加载数据
    data = pd.read_excel('超市销售关联.xlsx', encoding = 'utf - 8')
    inputList = data.values.tolist()
    # 将表格中的 T 和 F 转换为商品名称,A~F 为商品编号
    header = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K']
    itemList = []
    for item in inputList:
        cur = []
        for i in range(len(item)):
            if item[i] == 'T':
```

```
                cur.append(header[i])
            itemList.append(cur)
        return itemList

def apriori_method(data, min_support, min_confidence, min_lift, max_length):
    associate_rules = apriori(data, min_support = min_support,
                              min_confidence = min_confidence, min_lift = min_lift,
max_length = max_length)

    for rule in associate_rules:
        print("频繁项集 %s,置信度 %f" % (rule.items, rule.support))
        for item in rule.ordered_statistics:
            print("%s -> %s, 置信度 %f 提升度 %f" %
                  (item.items_base, item.items_add, item.confidence, item.lift))
        print()

def fpgrowth_method(data, min_support, min_confidence):
    # 频繁项集
    patterns = pyfpgrowth.find_frequent_patterns(data, min_support)
    # 规则
    rules = pyfpgrowth.generate_association_rules(patterns, min_confidence)
    print(rules)
    for i in rules:
        print("%s -> %s 置信度 %f" % (i, rules[i][0], rules[i][1]))

if __name__ == "__main__":
    data = loadData()
    min_support = 0.1                          # 最小支持度
    min_confidence = 0.5                       # 最小置信度
    min_lift = 0.0                             # 最小提升度
    max_length = 3                             # 最长关系长度
    print('Apriori 得到的关联规则')
   # apriori_method(data, min_support, min_confidence, min_lift, max_length)
    print('FP-growth 树得到的关联规则')
    fpgrowth_method(data, min_support, min_confidence)
```

5.6.4 其他关联规则挖掘算法

除了常用的 Apriori 算法、FP 增长算法等以外，还有其他的关联分析算法。

1. 约束性关联规则挖掘算法

关联规则的挖掘过程包括用户指定数据源与阈值、选择挖掘算法以及返回关联规则等步骤。这是一种无监督的学习过程，由于缺乏用户控制，可能导致产生过多的规则，实际效果可能并不好，而且用户关心的是某些特定关联规则，用单一的阈值不能体现用户的需求，所以挖掘的结果往往不能令用户满意。约束性关联规则挖掘算法是一个有监督(supervised)的学习过程，它把约束条件引入挖掘算法中，从大量的频繁项集中筛选出符合约束条件的有用规则，可以提高算法的运行效率和用户满意度。

2. 增量式关联规则挖掘算法

在实际应用中,很多数据集都是不断增长的,每当有新的数据加入后,重新挖掘是很费时的。因此,需要对已发现的关联规则进行更新。一些学者在研究关联规则的高效更新时,提出了增量式关联规则挖掘算法。增量式的挖掘方法是当事务(交易)数据库变化后,在原有挖掘结果的基础上生成新的关联规则,删除过时关联规则的过程。目前已有多种增量关联规则更新算法,这些算法充分利用已有的挖掘结果高效地发现新的关联规则。

Carma 算法也是一种比较常用的关联分析算法,与 Apriori 算法相比,具有占用内存少、允许在算法执行过程中按需要重新设置支持度以及只需要对数据集少数几次扫描就可以得到关联规则集等优点。Carma 算法也包括寻找频繁项集、在频繁项集的基础上产生关联规则两个阶段。在寻找频繁项集阶段,动态调整最小支持度产生候选频繁项集,然后对候选频繁项集进行删减得到最终的频繁项集。在计算频繁项集的过程中,新数据的读入只对已有的数据做局部调整,而不是对整个数据库重新扫描,因此 Carma 算法的执行效率较高。

3. 多层关联规则挖掘

目前,大多数研究者对于单层关联规则挖掘算法研究相对较多。由于多维数据空间数据的稀疏性,因此在低层的数据项之间很难找出强关联规则。而在较高概念层发现的强关联规则可能有应用价值。

目前出现了几种多层关联规则挖掘算法:一种多层关联挖掘算法采用了人工自定义不同层最小支持度的方式,并对经典的 Apriori 算法进行了改进。例如,在交易数据库中,可能某种食品的支持度不能达到要求,但在较高层的概念中,却可以达到更高的支持度以满足最小支持度。例如,“果汁,花生”的支持度小于“饮料,干果”的支持度,虽然前者不能形成一条强关联规则,但后者可能形成用户感兴趣的强关联规则。在这种情况下,可以为不同的层次定义不同的最小支持度阈值。另一种基于遗传算法的多层关联挖掘算法则利用不同层的先验知识,采用启发式的阈值自定义方法,解决了第一种算法层数较多,需要人工定义大量的阈值,随意性较大的问题。

5.7 序列模式挖掘

序列(sequence)模式挖掘也称为序列分析,由 Agrawal 在 1995 年提出。序列模式挖掘是从序列数据库中发现事件之间在时序上的规律。序列模式挖掘是关联规则挖掘的推广,挖掘序列数据库中项集间的时序关联。最初在带有交易时间属性的交易数据库中发现频繁项目序列,以发现一个时间段内顾客的购买行为规律,例如,购买氧气瓶的顾客一年内会回来充气多少次?序列模式挖掘应用领域包括顾客购买行为模式预测、Web 访问模式预测、疾病诊断、自然灾害预测和 DNA 序列分析等。例如,Fingerhut 公司通过序列分析发现,那些改变住所的顾客在搬家后 12 周内购买力增加 3 倍,其中前 4 周尤为明显。

5.7.1 基本概念

设 $I=\{i_1,i_2,\cdots,i_n\}$是一个项集,序列就是若干事件(元素)组成的有序列表。一个序

列 Se 可表示为$\langle s_1, s_2, \cdots, s_n \rangle$，其中 $s_j (j=1,2,\cdots,n)$为事件，也称为 Se 的元素。元素由不同的项组成。当元素只包含一项时，一般省去括号，例如，$\{i_2\}$一般表示为 i_2。元素之间是有顺序的，但元素内的项是无序的，一般定义为词典序。序列包含项的个数称为序列的长度，长度为 L 的序列记为 L-序列。序列数据库就是元组<sid，Se>的集合，即有序事件序列组成的数据库，其中 Se 是序列，sid 是该序列的序列号。

存在两个序列 $\alpha=<a_1, a_2, \cdots, a_n>$，$\beta=<b_1, b_2, \cdots, b_m>$，如果存在整数 $1 \leqslant i_1 < i_2 < \cdots < i_n \leqslant m$ 且 $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \cdots, a_n \subseteq b_{i_n}$，那么称序列 α 是 β 的子序列(subsequence)，或者序列 β 包含 α，记作 $\alpha \subseteq \beta$。序列 α 在序列数据库 Se 中的支持度为序列数据库 Se 中包含序列 α 的序列个数除以总的序列数，记为 support(α)。给定支持度阈值 τ，如果序列 α 在序列数据库中的支持度不低于 τ，则称序列 α 为序列模式(频繁序列)。

事务数据库如表 5.16(表中数字是项目编号)所示，交易中不考虑顾客购买物品(项目)的数量，只考虑物品有没有被购买。整理后可得到顾客购物序列库，如表 5.17 所示。

表 5.16 事务数据库

事务发生的时间	顾客 ID	购买项集
2004.12.10	2	10、20
2004.12.12	5	90
2004.12.15	2	30
2004.12.20	2	40、60、70
2004.12.25	4	30
2004.12.25	3	30、50、70
2004.12.25	1	30
2004.12.30	1	90
2004.12.30	4	40、70
2004.12.31	4	90

表 5.17 顾客购物序列库

顾客标识	顾客购物序列
1	<30,90>
2	<{10,20},30,{40,60,70}>
3	<{30,50,70}>
4	<30,{40,70},90>
5	<90>

设最小支持度为 25%，从表 5.16 中可以看出，<30,90>是<30,{40,70},90>的子序列。两个序列<30,90>、<30,{40,70}>的支持度都为 40%，因此是序列模式。

5.7.2 类 Apriori 算法

序列模式挖掘是在给定序列数据库中找出满足最小支持度阈值的序列模式的过程。其中类 Apriori 算法是由 Apriori 算法引申而来的，其基本过程类似 Apriori 算法：首先扫描序列数据库，得到长度为 1 的序列模式 L_1。然后对长度为 i 的种子集 $L_i (i \geqslant 1)$通过连接操

作,例如,<1,2,3>与<2,3,4>连接得到<1,2,3,4>,生成长度为 $i+1$ 的候选序列模式 I_{i+1}。扫描序列数据库,计算每个候选序列模式的支持数,产生长度为 $i+1$ 的序列模式 L_{i+1}。重复上述步骤,直到没有新的候选序列模式产生为止。

$$L_1 \rightarrow I_2 \rightarrow L_2 \rightarrow I_3 \rightarrow L_3 \rightarrow \cdots ?$$

在类 Apriori 算法中,两个长度小的序列模式连接需要满足一定的条件:如果去掉序列模式 α 的第一个项目与去掉序列模式 β 的最后一个项目得到的序列相同,则可以把序列模式 α 和 β 连接,把 β 的最后一个项目放到 α 中。至于 β 的最后一个项目是作为一个项目合并到 α 的最后一个元素,还是作为一个不同的元素,取决于 β 的最后两个项目是否属于同一个元素。为提高搜索的性能,类 Apriori 算法也在每次连接操作后应用了启发式:如果某候选序列模式的某个子序列不是序列模式,那么此候选序列模式不可能是序列模式,可以把此候选序列模式删除。

下面以表 5.18 的原始序列为例来说明如何利用类 Apriori 产生序列模式集,大致过程如表 5.19 所示,其中假设最小支持度为 40%。这里需要注意的是,<1,2,3,4>与<1,2,4,3>是两个不同的序列模式,应区分对待。

表 5.18 产生候选序列模式

原始序列	序列模式	连接结果
<1,2,3,4>	<1,2,3>	
<{1,5},2,3,4>	<1,2,4>	
<1,3,4,{3,5}>	<1,3,4>	<1,2,3,4>
<1,3,5>	<1,3,5>	
<4,5>	<2,3,4>	

表 5.19 类 Apriori 算法过程

序列	支持度
<1>	0.8
<2>	0.4
<3>	0.8
<4>	0.8
<5>	0.8
大于 1 的序列模式	
<1,2>	0.4
<1,3>	0.8
<1,4>	0.6
<1,5>	0.4
<2,3>	0.4
<2,4>	0.4
<3,4>	0.6
<4,5>	0.4
大于 2 的序列模式	
<1,2,3>	0.4

续表

序　　列	支　持　度
<1,2,4>	0.4
<1,3,4>	0.6
<1,3,5>	0.4
<2,3,4>	0.4
大于3的序列模式	
<1,2,3,4>	0.4

上述分析对元素的时间间隔没有特殊的要求。如果对元素施加一定的时限约束，那么就需要对上述序列分析算法进行修改，才能得到符合要求的序列模式。

除类 Apriori 算法外，序列模式的挖掘算法还有 SPADE 等算法，它们都直接或间接地利用了 Apriori 算法的思想。这些算法普遍存在的缺陷是扫描序列数据库次数过多，可能会产生很大的候选序列集。针对此问题，一些序列模型挖掘算法采用基于投影的方法，充分利用投影数据库的原理和当前挖掘的频繁序列集把序列数据库递归地投影到一组更小的投影数据库上，以减小搜索空间。

5.8 回归分析

在数据挖掘中经常要分析变量之间的关系。回归分析(regression analysis)是一种基本的统计分析方法，它已被广泛地应用于数据挖掘领域。在现实应用中，变量之间存在着某种关系，这些变量之间的关系一般可以分为两类：一类是变量之间存在着完全确定的关系，即一个变量能被其他变量确定；另一类是变量之间存在某种程度的不确定关系，统计学把这种不确定关系称为相关关系。例如，制造企业产品质量与各个生产因素之间存在一定的关系，可以分析这些关系以做出预测或确定最佳的作业条件。确定性关系和相关关系之间没有严格的界限。一方面，由于测量误差等原因，确定性关系可以通过相关关系表现；另一方面，通过对事物内部发展规律的深刻认识，相关关系又可能转化为确定性关系。两个变量之间的相关关系是不确定的，但可以通过不断观察，得到它们之间的统计规律。分析一个变量与其他一个(或几个)变量之间的相关关系的统计方法就称为回归分析。常见的回归分析包括线性回归、多元回归、非线性回归、广义线性回归(对数回归、泊松回归)等。回归分析的主要内容包括确定连续值变量之间的相关关系，建立回归模型，检验变量之间的相关程度，应用回归模型对变量进行预测等。

根据回归分析涉及的自变量个数，可把回归分析分为一元回归分析和多元回归分析。而按照自变量和因变量之间的关系类型，回归分析可分为线性回归分析和非线性回归分析。一般来说，回归分析的步骤如下。

(1) 确定因变量和影响因素(自变量)。

(2) 绘制散点图，观察变量的大致关系。

(3) 求回归系数，并建立回归模型。这一步试图比较真实数据与回归模型输出之间的误差来探索变量之间的关系。

(4) 检验回归模型。

(5) 进行预测。

5.8.1 一元回归分析

一元线性回归是描述两个变量之间线性相关关系的最简单的回归模型,如图 5.37 所示。散点图中两个变量呈线性关系。一元线性回归模型表示为 $y=a+bx+\varepsilon$,其中 a 和 b 是系数,ε 是随机变量。在这个线性模型中,自变量 x 是非随机变量。随机变量 ε 要求服从正态分布,即 $\varepsilon \sim N(0,\sigma^2)$。

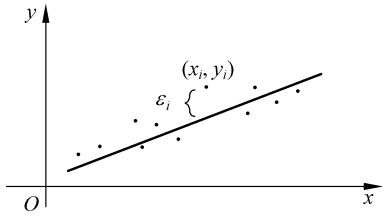


图 5.37 一元线性回归

回归模型中的参数 a 与 b 在一般情况下都是未知数,必须根据样本数据 (x_i, y_i) 进行估计(ε_i 相互独立)。确定参数 a 与 b(分别记作 $\hat{a}$ 和 $\hat{b}$)值的原则是使样本的回归直线同观察值的拟合状态最好,即使偏差 $|\varepsilon_i|$ 较小。为此,可以采用最小二乘法计算。对应于每一个 x_i,根据回归方程都可以求出一个 $\hat{y}_i$,它就是 y_i 的一个估计值。有 n 个观察值就有相应的 n 个偏差。要使模型的拟合状态最好,即要使 n 个偏差的总和最小。为了计算方便,以误差的平方和最小为标准确定回归模型。

$$Q=\sum_{i=1}^{n}(y_i-\hat{y}_i)^2=\sum_{i=1}^{n}(y_i-a-bx_i)^2$$

$$\frac{\partial Q}{\partial a}=2\sum_{i=1}^{n}[y_i-(a+bx_i)]\cdot(-1)=0$$

$$\frac{\partial Q}{\partial b}=2\sum_{i=1}^{n}[y_i-(a+bx_i)]\cdot(-x_i)=0$$

得到参数 a 和 b 的最小二乘估计:

$$\hat{b}=S_{xy}/S_{xx},\hat{a}=\bar{y}-\hat{b}\bar{x}$$

其中:$\bar{x}$、$\bar{y}$ 分别是变量 x、y 的 n 个样本的平均值;$S_{xy}=\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})$;$S_{xx}=\sum_{i=1}^{n}(x_i-\bar{x})^2$。

求出参数 $\hat{a}$ 和 $\hat{b}$ 以后,就可以得到回归方程 $\hat{y}=\hat{a}+\hat{b}x$,因此只要给定了一个 x 值,就可以根据回归方程求得一个 $\hat{y}$ 作为实际值 y 的预测值。但是用 $\hat{y}_i$ 预测 y 的精度如何?统计学用估计平均误差的方法度量回归方程的可靠性。一个回归方程的估计平均误差可定义为

$$S_e=\sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}$$

运用估计平均误差可以对回归方程的预测结果进行区间估计。若观察值围绕回归直线服从正态分布,且方差相等,则有 68.27%的点落在 $\pm S_e$ 的范围内,有 95.45%的点落在 $\pm 2S_e$ 的范围内,有 99.73%的点落在 $\pm 3S_e$ 的范围内。

【例 5.13】 一元回归分析

表 5.20 给出了某种产品 2000 年在 8 个地区的销售数据，试建立该种产品的月平均销售收入 y 对月平均广告支出 x 的线性回归方程。

表 5.20 销售数据表

地区编号	1	2	3	4	5	6	7	8
月平均销售收入 y/万元	31	40	30	34	25	20	35	40
月平均广告支出 x/万元	5	10	5	7	4	3	7	9

图 5.38 是表 5.20 中 8 个样本点对应的散点图，从中可见月平均销售收入 y 与月平均广告支出 x 之间呈一定的线性关系。

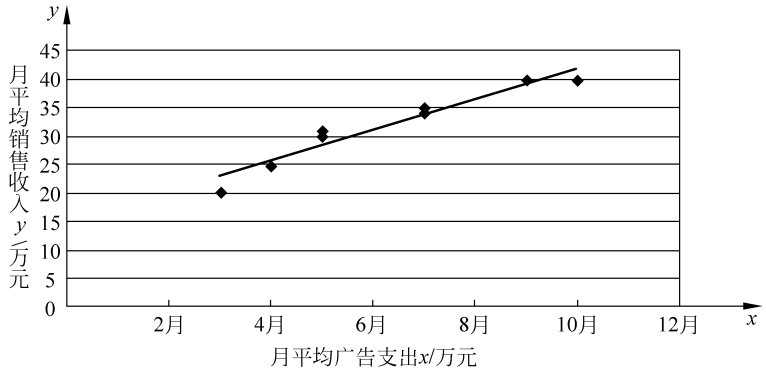


图 5.38 月平均广告支出 x 与月平均销售收入 y

计算得到 $\sum_{i=1}^{8} x_i = 50$，$\sum_{i=1}^{8} x_i^2 = 354$，$\sum_{i=1}^{8} y_i = 255$，$\sum_{i=1}^{8} x_i y_i = 1708$。

代入上面参数 $\hat{a}$ 和 $\hat{b}$ 的计算公式可得

$$\hat{b} = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} = 2.753$$

$$\hat{a} = \frac{\sum_{i=1}^{n} y_i}{n} - \hat{b}\,\frac{\sum_{i=1}^{n} x_i}{n} = 14.669$$

月平均销售收入 y 对月平均广告支出 x 的线性回归方程为

$$\hat{y} = 14.669 + 2.753x$$

ε^2 的无偏估计为

$$\hat{\varepsilon} = S_e^2 = \frac{1}{6}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = 4.076\,273$$

回归方程建立后还需要检验变量之间是否确实存在线性关系，因为对回归方程的求解过程事先并不知道两个变量是否存在线性相关关系。一元线性回归模型的统计检验可以用

F 检验法、t-检验法和 r 检验法等进行检验。

x 与 y 之间的线性相关估计模型 $\hat{y}=\hat{a}+\hat{b}x$ 在估计 y 时所产生的误差,称为回归中的方差分析。若没有利用 x 与 y 之间的相关关系估计总体的均值,则会选择 y_i 的平均值 $\bar{y}$ 作为总体的估计值。由此而产生的误差是 $\sum(y_i-\bar{y})^2$,数据总的变动称为总离差平方和,记为 SS_T。若利用 x 与 y 之间的线性相关估计模型去估计总体均值,则所产生的误差是 $\sum(y_i-\hat{y}_i)^2$,称为残差平方和 SS_E,它是未被回归方程解释的部分。被回归方程解释的部分,称为回归平方和 SS_R。

$$SS_T=\sum(y_i-\bar{y})^2,\quad SS_E=\sum(y_i-\hat{y}_i)^2,\quad SS_R=\sum_{i=1}^{n}(\hat{y}_i-\bar{y})^2$$

它们的相互关系为

$$\begin{aligned}SS_T&=\sum(y_i-\bar{y})^2\\&=\sum[(\hat{y}_i-\bar{y})+(y_i-\hat{y}_i)]^2\\&=\sum(\hat{y}_i-\bar{y})^2+\sum(y_i-\hat{y}_i)^2+2\sum(\hat{y}_i-\bar{y})(y_i-\hat{y}_i)\\&=\sum(\hat{y}_i-\bar{y})^2+\sum(y_i-\hat{y}_i)^2\end{aligned}$$

则 $SS_T=SS_R+SS_E$。

由回归平方和与残差平方和的意义可知,在总的离差平方和中,回归平方和所占比重越大,线性的回归效果就越好。相反,如果残差平方和所占比重越大,则线性回归效果越差。定义样本决定系数 R^2 和修正样本决定系数 $\bar{R}^2$ 分别如下。

$$R^2=\frac{SS_R}{SS_T}=1-\frac{SS_E}{SS_T}$$

$$\bar{R}^2=1-\frac{SS_E/(n-k-1)}{SS_T/(n-1)}$$

其中:k 表示自变量个数;R^2 可以作为回归值与实际观测值拟合程度的度量。R^2 越接近 1,说明两者的拟合程度越好。例 5.13 中 $R^2=0.928$,说明该产品的月平均销售收入与月平均广告支出的回归效果非常显著。

使用 SPSS Statistics 对表 5.20 的数据进行线性回归分析,得到的结果如图 5.39 所示。R^2 取值、F 检验和 t 检验也说明了本次线性回归模型的合理性。

预测是回归模型最重要的应用,回归预测包括点预测和区间预测。回归点预测是指对于给定的变量值 x_0,用回归值 $\hat{y}_0=\hat{a}x_0+\hat{b}$ 作为变量 y 的预测值 y_0。然而在现实中,实际值与预测值总会产生偏移,因此还需要得到可能偏离的范围以提高预测的可靠程度,这称为区间预测,即以一定的概率来预测 y_0 附近的变动范围。

5.8.2 多元线性回归分析

多元线性回归分析是研究一个变量 y 与多个其他变量 $x_1,x_2,\cdots,x_k$ 之间关系的统计分析方法。假设因变量 y 与相互独立的自变量 $x_1,x_2,\cdots,x_k$ 之间有线性关系 $y=\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_kx_k+u$,其中 $\beta_0,\beta_1,\beta_2,\cdots,\beta_k$ 是回归系数,u 为随机误差。上面的公式一般称

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.963[a]	.928	.916	2.01426

a. Predictors: (Constant), 月平均广告支出

ANOVA[b]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	314.532	1	314.532	77.524	.000[a]
	Residual	24.343	6	4.057		
	Total	338.875	7			

a. Predictors: (Constant), 月平均广告支出
b. Dependent Variable: 月平均销售收入

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	14.669	2.080		7.053	.000
	月平均广告支出	2.753	.313	.963	8.805	.000

a. Dependent Variable: 月平均销售收入

图 5.39　一元线性回归分析结果

为多元线性回归模型。由于 $\beta_0,\beta_1,\beta_2,\cdots,\beta_k$ 可以利用已知样本数据进行估计。设 $\hat{\beta}_0,\hat{\beta}_1,\hat{\beta}_2,\cdots,\hat{\beta}_k$ 是利用一组简单随机样本经计算得到的样本统计量，把它们作为未知参数 $\beta_0,\beta_1,\beta_2,\cdots,\beta_k$ 的估计值，得到估计的回归方程 $\hat{y}=\hat{\beta}_0+\hat{\beta}_1x_1+\hat{\beta}_2x_2+\cdots+\hat{\beta}_kx_k$，该方程称为样本回归方程或经验回归方程，$\hat{y}$ 称为 y 的样本估计值或样本回归值。

设 $(x_{1i},x_{2i},\cdots,x_{ki};y_i)$，其中 $i=1,2,\cdots,n$ 是对因变量 y 和自变量 $x_1,x_2,\cdots,x_k$ 的 n 次独立样本观测值，代入多元线性回归模型得到 $y_i=\beta_0+\beta_1x_{1i}+\beta_2x_{2i}+\cdots+\beta_kx_{ki}+u_i$，$i=1,2,\cdots,n$，它是由 n 个方程组成的一个线性方程组。

$$\begin{cases} y_1=\beta_0+\beta_1x_{11}+\beta_2x_{21}+\cdots+\beta_kx_{k1}+u_1 \\ y_2=\beta_0+\beta_1x_{12}+\beta_2x_{22}+\cdots+\beta_kx_{k2}+u_2 \\ \vdots \\ y_n=\beta_0+\beta_1x_{1n}+\beta_2x_{2n}+\cdots+\beta_kx_{kn}+u_n \end{cases}$$

表示成矩阵形式为 $\boldsymbol{Y}=\boldsymbol{X\beta}+\boldsymbol{u}$，其中：

$$\boldsymbol{Y}=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n\times1},\quad \boldsymbol{X}=\begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{k1} \\ 1 & x_{12} & x_{22} & \cdots & x_{k2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{1n} & x_{2n} & \cdots & x_{kn} \end{bmatrix}_{n\times(k+1)}$$

$$\boldsymbol{\beta}=\begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}_{(k+1)\times1},\quad \boldsymbol{u}=\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}_{n\times1}$$

这里 $\boldsymbol{Y}$ 是因变量样本观测值的 $n\times1$ 阶列向量，$\boldsymbol{X}$ 是自变量样本观测值的 $n\times(k+1)$ 阶

矩阵,它的每个元素 x_{ij} 都有两个下标,第一个下标 i 表示相应的列(第 i 个变量),第二个下标 j 表示相应的行(第 j 个观测值)。$\boldsymbol{X}$ 的每一列表示一个自变量的 n 个观测值向量,$\boldsymbol{\beta}$ 为未知参数的 $(k+1)\times 1$ 阶列向量,$\boldsymbol{u}$ 为随机误差项的 $n\times 1$ 阶列向量。假设自变量 x_i 之间是相互独立的,把样本数据代入 $\boldsymbol{Y}=\boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{u}$,得到 $\hat{\boldsymbol{\beta}}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y}$,式中 $\boldsymbol{X}^{\mathrm{T}}$ 表示 $\boldsymbol{X}$ 的转置,而 $(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}$ 表示 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 的逆操作。

样本回归方程的矩阵形式为 $\hat{\boldsymbol{y}}=\boldsymbol{X}\hat{\boldsymbol{\beta}}$,其中:

$$\hat{\boldsymbol{y}}=\begin{bmatrix}\hat{Y}_1\\ \hat{Y}_2\\ \vdots\\ \hat{Y}_n\end{bmatrix}_{n\times 1},\quad \hat{\boldsymbol{\beta}}=\begin{bmatrix}\hat{\beta}_0\\ \hat{\beta}_1\\ \vdots\\ \hat{\beta}_k\end{bmatrix}_{(k+1)\times 1}$$

在这里,$\hat{\boldsymbol{y}}$ 被解释为变量样本观测值 $\boldsymbol{Y}$ 的 $n\times 1$ 阶估计值列向量,$\hat{\boldsymbol{\beta}}$ 是未知参数 $\boldsymbol{\beta}$ 的 $(k+1)\times 1$ 阶估计值列向量。多元回归分析中常见的方法包括最小二乘法、拟合优度检验等。

回归模型也可以使用 F 检验,即方差齐性检验。从两个研究总体随机抽取样本,首先判断这两个总体的方差是否相等(方差齐性),可以用 F 检验。回归方程线性是否显著的分析过程如下:原假设 $H_0,\beta_1=\beta_2=\cdots=\beta_k=0$;备择假设 $H_1,\beta_1,\beta_2,\cdots,\beta_k$ 至少一个不为 0。$F=(\mathrm{SS_R}/k)/[\mathrm{SS_E}/(n-k-1)]$。如果 $F<F_\alpha$,那么自变量系数在 $1-\alpha$ 的置信度内服从原假设(α 为显著性水平);如果 $F>F_\alpha$,那么放弃原假设 H_0,有 $1-\alpha$ 的置信度选择备择假设,回归系数至少一个不为 0,回归方程是线性的。

例 5.14 建立了多元回归线性模型并使用拟合优度检验和预测。

【例 5.14】 多元回归分析案例

表 5.21 所示为我国 1988—1998 年的城镇居民人均全年耐用消费品支出、人均全年可支配收入和耐用消费品价格指数的统计资料(取自《中国统计年鉴》)。试建立城镇居民人均全年耐用消费品支出 y 关于人均全年可支配收入 x_1 和耐用消费品价格指数 x_2 的回归模型。

表 5.21 1988—1998 年城镇居民人均统计资料

年份	人均耐用消费品支出 y	人均全年可支配收入 x_1	耐用消费品价格指数 x_2
1988	137.16	1181.4	115.96
1989	124.56	1375.7	133.35
1990	107.91	1510.2	128.21
1991	102.96	1700.6	124.85
1992	125.24	2026.6	122.49
1993	162.45	2577.4	129.86
1994	217.43	3496.2	139.52
1995	253.42	4283.0	140.44
1996	251.07	4838.9	139.12
1997	285.85	5160.3	133.35
1998	327.26	5425.1	126.39

图 5.40 是表 5.21 中 11 个样本对应的散点图，从中可见城镇居民人均全年耐用消费品支出 y 与人均全年可支配收入 x_1 和耐用消费品价格指数 x_2 之间呈一定的线性关系。

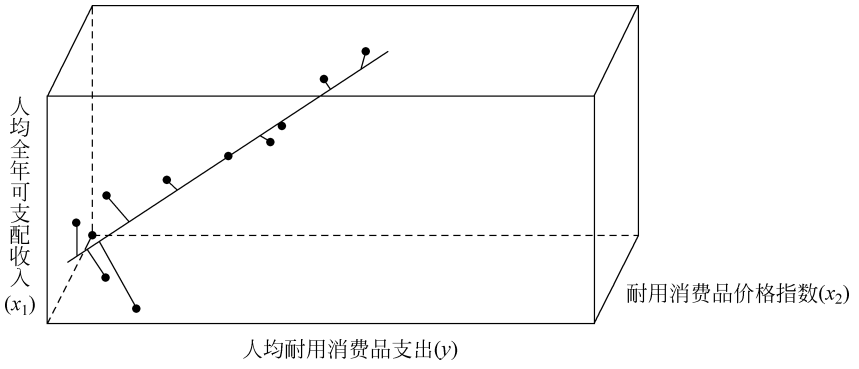


图 5.40　二元线性回归

根据经济理论和对实际情况的分析，城镇居民人均全年耐用消费品支出 y 依赖于可支配收入 x_1 和耐用消费品价格指数 x_2 的变化：$y=\beta_0+\beta_1 x_1+\beta_2 x_2+u$。

1. 估计模型未知参数

由表 5.21 中的数据，计算如下：

$\sum x_{1i}=33\,575.4$，$\sum x_{2i}=1433.54$，$\sum y_i=2095.31$；

$\bar{x}_1=3052.309\,091$，$\bar{x}_2=130.3218$，$\bar{y}=190.4827$；

$\sum x_{1i}^2=129\,253\,961.9$，$\sum x_{2i}^2=187\,421.9434$，$\sum x_{1i}x_{2i}=4\,445\,613.295$；

$\sum y_i^2=461\,991.4253$，$\sum x_{1i}y_i=7\,654\,936.718$，$\sum x_{2i}y_i=275\,976.737$。

将结果代入上面的公式，可得

$$\hat{\boldsymbol{\beta}}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y}=\begin{bmatrix}11 & 33\,575.4 & 1433.54\\ 33\,575.4 & 129\,253\,961.9 & 4\,445\,613.295\\ 1433.54 & 4\,445\,613.295 & 187\,421.9434\end{bmatrix}^{-1}\begin{bmatrix}2095.31\\ 7\,654\,936.718\\ 275\,976.737\end{bmatrix}$$

$$=\begin{bmatrix}158.6251\\ 0.0494\\ -0.9133\end{bmatrix}$$

$\hat{\beta}_0=158.6251$，$\hat{\beta}_1=0.0494$，$\hat{\beta}_2=-0.9133$。

估计的回归方程为 $\hat{y}=158.6251+0.0494x_1-0.9133x_2$。

计算残差平方和为 $\mathrm{SS}_E=\sum e_i^2=3277.198$。

σ^2 的无偏估计量为 $\hat{\sigma}^2=\dfrac{\sum e_i^2}{n-k-1}=\dfrac{3277.198}{11-2-1}=409.6497$。

回归估计标准误差为 $\hat{\sigma}=\sqrt{409.6497}=20.2398$。

2. 经济意义检验

$\hat{\beta}_1=0.0494$,表示城镇居民全年人均耐用消费品支出随着人均全年可支配收入的增长而增加,并且介于0～1,因此该回归系数的符号、大小都与经济理论和人们的经验期望值相符合。$\hat{\beta}_2=-0.9133$,表示城镇居民全年人均耐用消费品支出随着耐用消费品价格指数的降低而增加,虽然我国在1988—1998年,耐用消费品价格指数经历了由高到低,又由低到高,再由高到低的剧烈变化,但总的走势呈下降趋势,因此该回归系数的符号和大小也与经济理论和人们的经验期望值一致。

3. 统计检验

1) 拟合优度检验

$$SS_T=\sum(y_i-\bar{y})^2=\sum y_i^2-n\bar{y}^2=\sum y_i^2-\frac{1}{n}\left(\sum y_i\right)^2$$

$$=461\,991.4253-\frac{1}{11}\times(2095.31)^2=62\,871.0620$$

$$SS_E=\sum e_i^2=3277.198$$

$$SS_R=SS_T-SS_E=59\,593.864$$

把上述结果分别代入样本决定系数公式和修正样本决定系数公式可得

$$R^2=\frac{SS_R}{SS_T}=0.948$$

$$\bar{R}^2=1-\frac{SS_E/(n-k-1)}{SS_T/(n-1)}=0.929$$

结果表明估计的样本回归方程很好地拟合了样本观测值。

2) 预测

如果在2000年,我国城镇居民家庭人均可支配收入达到5800元,耐用消费品价格指数为135,下面对2000年我国城镇居民家庭人均耐用消费品的支出进行预测。

(1) 点预测。

把$x_0=(5800,135)$代入估计的样本回归方程$\hat{y}_i=158.6251+0.0494x_1-0.9133x_2$中,得到2000年我国城镇居民家庭人均耐用消费品支出的点估计值为$\hat{y}_{2000}=158.6251+0.0494\times5800-0.9133\times135=321.85$。

(2) 区间预测。

计算预测误差e_0方差的估计值$S^2(e_0)=\hat{\sigma}^2[1+\boldsymbol{X}_0(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}_0^{\mathrm{T}}]=4114.1$。

得到标准差的估计值$S(e_0)=\sqrt{610.4232}=64.14$。

对于给定的显著性水平$\alpha=0.05$,从t分布表中查出自由度为8的t分布双侧分位数$t_{0.05/2.8}=2.306$,得到y_{2000}的置信度为95%,预测区间为$(\hat{y}_{2000}-t_{a/2}S(e_0),\hat{y}_{2000}+t_{a/2}S(e_0))=(321.85-2.306\times64.14,321.85+2.306\times64.14)$。

图5.41是使用SPSS Statistics对表5.20的数据进行二元线性回归的结果,R^2和F检验的结果表明,上述线性回归模型是合理的。

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.974[a]	.948	.935	20.23981

a. Predictors: (Constant), 耐用消费品价格指数, 人均全年可支配收入

ANOVA[b]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	59593.864	2	29796.932	72.738	.000[a]
	Residual	3277.198	8	409.650		
	Total	62871.062	10			

a. Predictors: (Constant), 耐用消费品价格指数, 人均全年可支配收入
b. Dependent Variable: 人均耐用消费品支出

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	158.625	121.947		1.301	.230
	人均全年可支配收入	.049	.005	1.020	10.535	.000
	耐用消费品价格指数	-.913	.991	-.089	-.922	.384

a. Dependent Variable: 人均耐用消费品支出

图 5.41 二元线性回归分析结果

几种常见的回归分析 Python 程序如下。

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model

# 读取 Excel 文件
df = pd.read_excel('data.xlsx')

# 提取数据
x = df[['a','b']].values  # 假设 a,b 是其中的两个自变量字段,可以根据具体数据选择自变量
y = df['c'].values        # 假设 c 是因变量字段

# 分割数据为训练集和测试集
train_size = int(0.8 * len(x))
x_train, x_test = x[:train_size], x[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# 建立线性回归
regressor = LinearRegression()
regressor.fit(x_train, y_train)
linear_pred = regressor.predict(x_test)

# 定义多项式回归
poly_reg = PolynomialFeatures(degree = 5)

# 特征处理
x_train_poly = poly_reg.fit_transform(x_train)
```

```
# 定义回归模型
lin_reg = LinearRegression()

# 训练模型
lin_reg.fit(x_train_poly, y_train)
poly_pred = lin_reg.predict(poly_reg.transform(x_test))

# 创建 LASSO 回归模型
model = linear_model.LassoCV()
model.fit(x_train, y_train)

# lasso 系数
print("LASSO alpha:", model.alpha_)

# 相关系数
print("LASSO coefficients:", model.coef_)

lasso_pred = model.predict(x_test)

# 创建岭回归模型
alphas_to_test = [0.1, 1.0, 10.0]
model = linear_model.RidgeCV(alphas = alphas_to_test, store_cv_values = True)
model.fit(x_train, y_train)

# 岭系数
print("Ridge alpha:", model.alpha_)

# loss 值
print("Ridge CV values shape:", model.cv_values_.shape)

ridge_pred = model.predict(x_test)

# 输出预测结果
print("Linear Regression Predictions:", linear_pred)
print("Polynomial Regression Predictions:", poly_pred)
print("LASSO Regression Predictions:", lasso_pred)
print("Ridge Regression Predictions:", ridge_pred)
```

5.8.3 其他回归分析

事实上,现实中的大多数问题都是非线性的,需要对变量进行变换,把非线性问题转换为线性问题来解决。在线性回归问题中,变量一般是独立的。但在很多情况下,高次多项式可以更好地反映变量之间的关系,这就需要引入非线性回归来预测未知变量。非线性回归的思想是通过对变量进行转换,把非线性模型转换为线性模型,然后按上述方法再求解线性模型,求出其中参数后代入原非线性模型。例如,对多项式回归 $y=c_0+c_1x+c_2x^2+c_3x^3+c_4x^4$,先把此方程转换成线性方程,需要定义如下几个新变量:$x_1=x$、$x_2=x^2$、$x_3=x^3$、$x_4=x^4$。代入原来的多项式方程中,得到 $y=c_0+c_1x_1+c_2x_2+c_3x_3+c_4x_4$,多项式回归问题就转化为一个多元线性回归问题了。

对于双曲线函数 $y=\dfrac{x}{ax+b}$,进行线性转换 $y_1=1/y$,$x_1=1/x$,则有 $y_1=a+bx_1$。

对于指数函数等比较复杂的非线性函数,需要通过更复杂的转换。例如,对 $y=\alpha x^{\beta}$ 可

以做如下变换：$\ln y=\ln\alpha+\beta\ln x$，定义 $y_1=\ln y$，$x_1=\ln x$，得到 $y_1=\ln\alpha+\beta x_1$。

Logistic 回归建立了一个多项式对数回归模型，用于预测二值变量的值(0 或 1)。相对于独立变量 $x_1,x_2,\cdots,x_n$，变量 y 等于 1 的概率定义如下。

$$p(y=1\mid x_1,x_2,\cdots,x_n)=\frac{e^{-(a_1x_1+a_2x_2+\cdots+a_nx_n+\mu)}}{1+e^{-(a_1x_1+a_2x_2+\cdots+a_nx_n+\mu)}}$$

Logistic 回归在数据挖掘中很有用，特别是解决两类的数据概率打分问题，如顾客流失风险打分等。这个模型也可以转化为线性模型，下面举例说明。

【例 5.15】 应用 Logistic 回归模型预测银行顾客是否会拖欠贷款

根据历史数据识别银行拖欠顾客的特征，预测潜在信贷顾客是否拖欠贷款。这里选取 700 个信贷顾客的历史记录，其中 21.5%是拖欠顾客。选择顾客 sex(性别)、income(收入)、age(年龄)、education(文化程度)、employ(现单位工作年数)、debtinc(负债率)和 creddebt(信用卡债务)等作为自变量，顾客是否拖欠贷款作为因变量：1 代表拖欠，0 代表正常。选择 70%历史记录进行训练，剩下 30%历史数据用于验证，建立一个预测因变量取 1 的概率的 Logistic 回归模型，以对新的潜在顾客是否拖欠贷款进行预测。

影响顾客拖欠的自变量比较多，这里采用 Forward/Backward 方式用于剔除不重要的自变量，例如，income、education 和 age 等对顾客信用的影响不显著，拖欠概率的回归方程如下：

$$\ln\frac{p}{1-p}=-0.76-0.249\text{employ}-0.069\text{address}+0.08\text{debtinc}+0.594\text{creddebt}$$

对模型进行显著性检验，并对回归模型与样本数据的拟合程度以及模型预测精度进行评价，回归模型满足一定要求即可部署使用。从中可以发现拖欠贷款概率较大的客户的特征为：工作不稳定、住址经常变动、债务比率高、信用卡债务多。

二元 Logistic 回归分析还可用于风险预测的其他场合。图 5.42 是对电信用户的输入数据自动筛选后，使用 Logistic 回归对这些用户的流失进行预测的结果。

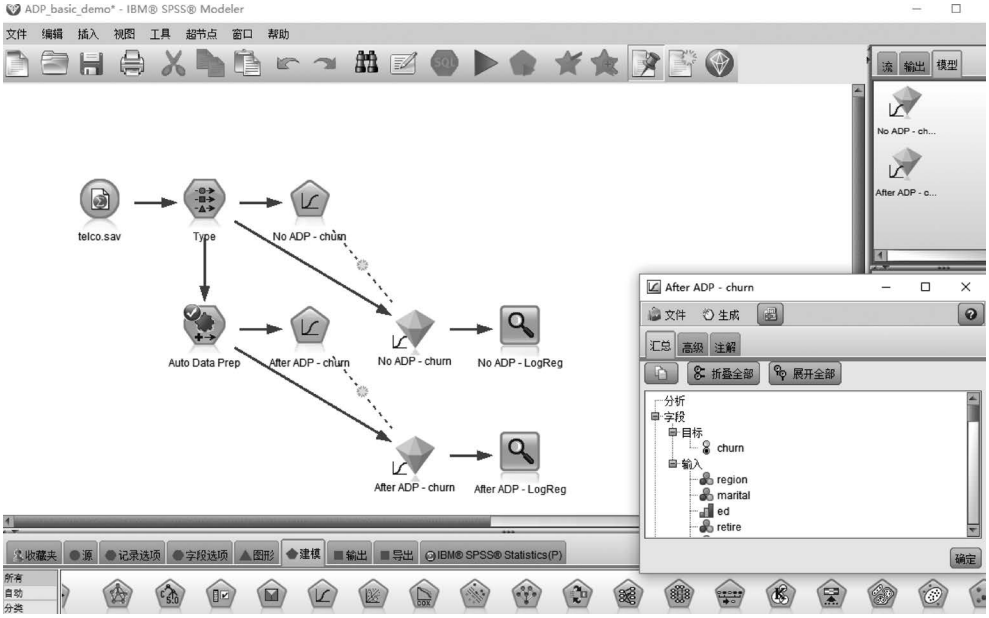


图 5.42　Logistic 回归在电信用户流失预测中的应用

【例 5.16】 基于移动通信社交网络分析的客户流失预警模型

随着市场份额趋于饱和以及竞争激化,移动运营商的用户数和收入增长十分缓慢,客户流失管理是移动运营商关注的主要问题之一。该移动运营商每个月面临着平均 1.5%左右的客户流失率,传统客户关怀的做法是打电话给本月花费少或连续几个月延迟缴费的客户,这种方法成本高、准确度低,还可能会打扰到正常的客户。这里以南方某三线城市的某移动运营商的运营数据为基础,利用 Logistic 回归模型,设计了一个客户流失预警模型,能够快速、高效并且较低成本地识别高风险的流失客户。

在利用 Logistic 回归模型前,需要确定显著影响客户流失的因素,并做预处理,然后计算每个客户的流失概率,以便按照流失概率寻找可能的高风险流失客户进行针对营销。随机选取 5 万个左右平均每月 ARPPU(average revenue per paying user,每付费用户平均收入)值大于 80 元的客户的数据(从 2014 年 3 月到 8 月共六个月),包括客户的入网时间、当月花费、话费情况以及按月份统计的客户通话详单等数据,对上述数据的空值、异常值、重复记录等数据进行预处理。为了更有效地使用 Logistic 回归模型,从上述数据衍生出三个与用户通信网络有关的新变量:用户的度、联系的强度以及用户的信息熵,其中用户的度是与其有过通话记录(包括呼入与呼出)的不同用户的总数,联系的强度表示用户平均通话时长,用户的信息熵表示与其通话的所有客户的平均通话时长的分布。上述三个新变量越大,客户的流失可能性越小,因此这些变量更能体现用户的流失特征。

以入网时长、当月花费、个体的度、联系的强度、个体的信息熵、本月相比上月花费的变化、本月相比上月通话人数的变化等作为自变量,客户的流失与否作为因变量,构建 Logistic 回归模型,采取覆盖率—捕获率(覆盖率是客户抽样比例,捕获率类似召回率)评判模型的预测精度。实验证明,上述方法只需较少的覆盖率,就可以得到比较高的捕获率。

此外,前面讨论的 BP 等神经网络也可以解决任意非线性回归问题,只不过其获得的模型难以解释。

Logistic 回归分析的典型 Python 代码如下。

```
#导入必要的库
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler
import seaborn as sns

#加载数据
data = load_breast_cancer()
X = data.data
y = data.target
feature_names = data.feature_names
target_names = data.target_names

#数据预处理
# 标准化特征
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42, stratify=y)

# 创建并训练 Logistic 回归模型
# 使用 L2 正则化,设置较高的 C 值减少正则化强度
model = LogisticRegression(
    penalty='l2',
    C=1.0,
    solver='lbfgs',
    max_iter=1000,
    random_state=42
)
model.fit(X_train, y_train)

# 模型评估
# 训练集和测试集预测
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)

# 计算准确率
train_accuracy = accuracy_score(y_train, y_train_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)
print("\n模型评估结果:")
print(f"训练集准确率: {train_accuracy:.4f}")
print(f"测试集准确率: {test_accuracy:.4f}")
```

5.9 时间序列分析

时间序列(time series)是指由在不同时间上的观察值或事件组成的序列。时序数据库则是一种有时间标记的序列数据库。现实中这些时间序列数据都是通过数据收集工具自动获取的,数据量非常大。时间序列数据是包含时间属性的序列数据的一种特殊形式,与Web访问序列可能不同,股票涨停序列是时间序列数据。

时间序列分析的基础是惯性原则,即在一定条件下,被预测事物的过去变化趋势会延续到未来。时间序列分析运用统计分析和数据挖掘技术从时间序列数据库中找到系统的发展趋势等,有助于对系统的分析或者系统变化预测,例如,利用某地区近几年月平均降雨量对未来的月降雨量进行预测。此外,时间序列分析还可以发现突变以及离群点。其主要的应用包括股票市场分析、销售预测、自然灾害预测、过程与质量控制等。

1. 时间序列分析方法

时间序列分析方法包括确定型的时间序列分析方法和随机型的时间序列分析方法两种。确定型的时间序列可以用一个确定的时间函数$Y=F(t)$来拟合,图5.43中的虚线为某公司季度净利润的趋势。

一般地,时间序列分析主要包括两方面:时间序列建模和时间序列预测。前者用于分

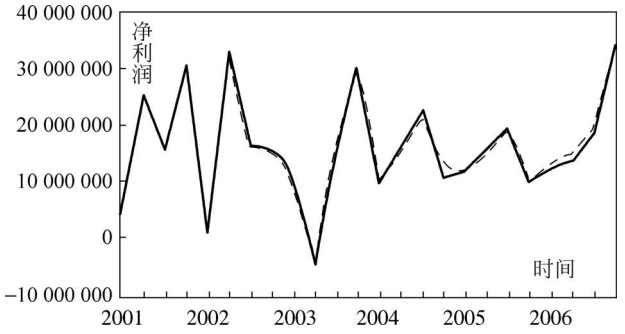


图 5.43 季度净利润的时间序列数据

析产生时间序列的机制,后者用于预测时间序列变量的未来值。时间序列数据的变化特征可以分为以下几类,时间序列通常是由以下几种基本运动合成的。

(1) 趋势。趋势是时间序列在较长时间内呈现出的某种上升或下降的大体方向。确定趋势的典型方法包括加权移动平均法和最小二乘法等。

(2) 周期运动。周期运动是时间序列呈现出的围绕长期趋势的一种"波浪形"周期性变动。

(3) 季节性变化。时间序列在一年内重复出现的周期运动称为季节性变化。这里的季节不限于一年中的四季,可以广义地表示周期性的变化。

(4) 不规则运动。由各种偶然、突发或不可预见的因素引起的时间序列变动,称为不规则运动,如自然灾害等。

目前主要的时间序列分析模型如下:自回归(auto-regressive,AR)、移动平均(moving average,MA)以及自回归综合移动平均(auto-regressive integrated moving average,ARIMA)等模型。ARIMA 模型是常用的时间序列算法,一般用于对动态数据的预测。移动平均模型用于消除时间序列数据中的波动,降低其变差,因此也称为时间序列的光滑,n 阶移动平均的计算公式如下。

$$\frac{x_1+x_2+\cdots+x_n}{n},\quad \frac{x_2+x_3+\cdots+x_{n+1}}{n},\quad \frac{x_3+x_4+\cdots+x_{n+2}}{n},\quad \cdots$$

如果对 n 阶移动平均使用加权的算术平均,则可以得到 n 阶加权移动平均,通常加权权值的中心元素会取相对较大的值以抵消光滑带来的影响,例如,给定一个包含 7 个值的序列,并使用权值(1,4,1),对应的三阶移动平均与加权三阶移动平均如下。

原始序列:	7	9	2	1	6	8	4
三阶移动平均:		6	4	3	5	6	
加权三阶移动平均:		7.5	3	2	5.5	7	

对于三阶移动平均,第一个值为(7+9+2)/3=6,对于加权三阶移动平均,第一个值为(1×7+4×9+1×2)/6=7.5。

移动平均模型也存在一些不足,例如,移动平均有时可能产生原始数据中没有出现的周期变动,而且可能因为异常值的存在而受到较大的影响。

p 阶自回归模型可表示为

$$X_t=\varphi_1X_{t-1}+\varphi_2X_{t-2}+\cdots+\varphi_pX_{t-p}+u_t$$

其中：$\varphi_1,\varphi_2,\cdots,\varphi_p$ 是待估计参数，可以用已知历史数据估计，称为自回归系数；u_t 为随机误差项，是由相互独立的白噪声序列组成，且服从均值为 0，方差为 σ^2 的正态分布。

q 阶移动平均模型可表示为

$$X_t = u_t - \theta_1 u_{t-1} - \theta_2 u_{t-2} - \cdots - \theta_q u_{t-q}$$

其中：$\theta_1,\theta_2,\cdots,\theta_q$ 是待估计参数，可以用已知历史数据估计，称为移动平均系数。

将随机误差项为白噪声的纯 p 阶自回归模型与随机误差项不是白噪声的 q 阶纯移动平均模型组合，得到一个自回归移动平均模型，简称 ARMA(p,q)模型。该模型可表示为

$$X_t = \varphi_1 X_{t-1} + \varphi_2 X_{t-2} + \cdots + \varphi_p X_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q}$$

应用 ARIMA 模型做预测的 Python 如下。

```
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from numpy.random import randn
data = randn(90)       #data 可以替换成具体的时间序列变量
data = pd.Series(data)
data.index = pd.Index(sm.tsa.datetools.dates_from_range('2001','2090'))
data.plot(figsize = (14,7))
fig = plt.figure(figsize = (14,7))
diff_1 = data.diff(1)     #一阶差分
#自相关图和偏自相关图
fig = sm.graphics.tsa.plot_acf(data,lags = 40,ax = fig.add_subplot(211))
fig = sm.graphics.tsa.plot_pacf(data,lags = 40,ax = fig.add_subplot(212))
#ARMA(7,0)模型
arma_mod = sm.tsa.ARMA(data,(7,0)).fit()
#计算模型的 AIC、BIC 和 HQIC
print(arma_mod.aic,arma_mod.bic,arma_mod.hqic)
predict_values = arma_mod.predict('2090', '2100', dynamic = True)
fig, ax = plt.subplots(figsize = (14,7))
ax = data.ix['2001':].plot(ax = ax)
predict_values.plot(ax = ax)
```

2. 相似性搜索

在时间序列数据库中，相似性搜索是指在允许微小差别的条件下，寻找与给定查询序列相似的数据序列。相似性搜索在诸如股票市场分析和心电图分析中相当有用。子序列匹配和全序列匹配是相似性搜索的两种类型。其中，子序列匹配在应用中更加常见。

在进行时间序列分析时，首先需要使用 5.3 节介绍的数据归约，尤其是维归约技术以缩小时间序列数据的存储空间，提高处理速度。这些技术主要包括离散傅里叶变换(discrete Fourier transform，DFT)、离散小波变换(discrete wavelet transform，DWT)和基于主成分分析的奇异值分解(singular value decomposition，SVD)等。时间序列数据经过数据归约和变换后，可以使用一些索引方法来提高相似性查询的速度。在进行相似性分析，尤其是子序列匹配时，需要考虑两个子序列数据集之间的距离。距离越小，则两个序列越相似。匹配过程中需要对两个子序列进行调整，处理两者的间隙、偏移量和振幅等差别，然后就可能找到相似序列。

3. 应用时间序列分析需要注意的问题

时间序列分析也存在一些问题：首先，时间序列分析的前提是假定事物过去的变化规律会延续到未来。然而，并不是所有事物都是连续性发展变化的，有时它们的发展轨迹复杂多样。因此，在应用时间序列分析法时需要注意事物未来的发展变化规律，考虑随着时间的推移是否会出现一些新的特点。其次，时间序列分析突出了时间因素在预测中的作用，而忽略了外界因素的影响，因而存在预测误差。如果外界因素发生比较大的变化，预测可能有较大的偏差。因此时间序列分析对于中短期预测的效果一般比长期预测的效果好。

5.10 数据挖掘技术与应用的发展方向

在大数据时代，这些主题仍有一定的研究价值。考虑到数据正朝着大规模、类型多样、分布异构以及产生速度快等方向发展，因此对数据挖掘算法的性能将有更高的要求。数据挖掘的应用领域也越来越广泛，成功的案例也越来越多。

1. 模式挖掘

模式挖掘(pattern mining)在最近的很多年里一直是数据挖掘的热点之一，它包含的内容很丰富，包括频繁模式挖掘、结构模式挖掘和关联规则挖掘等，目前已经出现了很多有效的算法。在现实应用中，总的发展趋势是需要挖掘的模式长度更长且更加复杂，如生物数据量大、维度高，传统的 Apriori 算法及其变种不能有效地处理这类数据。当前研究的主要方向是针对较长模式和复杂模式设计有效的算法，经常采用模式压缩、情景分析和语义分析等技术辅助模式挖掘，以提高模式挖掘的效率和准确性。

2. 信息网络分析

信息网络包括社会网络、生物网络和恐怖组织网络等，总的来说，信息网络的研究还是一个新领域。在信息网络挖掘中的一个方向是把信息网络看作一个图，用图挖掘的方法来研究信息网络。从图论的角度来看，基于图的很多聚类、分类和索引技术有助于对信息网络的分析。同时，信息网络存在大量的节点与链接，对信息网络的深入研究也是很有必要的。社会网络分析(social network analysis)作为信息网络分析的一个重要分支，在最近几年得到广泛关注，它在社会舆情分析、自然灾害监测、股价预测等领域都有成功的应用。

3. 流数据挖掘

流数据是大量流入系统、不断变化的多维数据，这些数据很难存储于传统的数据库中。由于流数据的速度快、规模大等特点，必须使用单遍扫描、联机和多维方法对流数据进行挖掘，目前针对流数据的聚类、分类和异常分析等已经进行了很多工作。流数据的应用很多，如网络异常监控、网络路由、传感器网络、股票分析、社交网站分析等，如何提高流数据分析的精度和实时性，扩大流数据挖掘的应用范围是未来研究的重点。

4. 针对移动数据和 RFID 数据的挖掘

随着传感器网络、手机、GPS、其他移动设备以及无线射频识别(radio frequency

idenfication，RFID)技术的广泛使用，每天产生大量的移动对象数据，移动数据往往是多维的，包括时间、速度和位置等信息。如何为这些移动数据构建数据仓库并进一步挖掘知识是数据挖掘的一个热点，例如，利用手机的位置信息进行商业推荐，根据城市交通流量数据进行智能调度，分析气象数据进行风力发电站智能选址，或者利用摄像头的录像信息检查非法入侵等。RFID 在产品生产、海关通关和商品零售等领域有着巨大的应用价值，用数据挖掘的方法监控整个生产、销售流程产生的 RFID 数据能够给企业带来可观的效益。

5. 时空和多媒体数据挖掘

现实生活中的很多数据都涉及时间、空间，例如，地图搜索服务或者天气预报服务，人们更习惯使用图片、视频等多媒体方式进行交流，针对多媒体数据的挖掘可以发现相当丰富的知识。时空数据和多媒体数据的挖掘已经成为一个研究热点。目前，对时空和多媒体数据的挖掘总体上还不能满足应用的需求，尤其是对于多维的、复杂的以及需要大量计算的数据，挖掘结果还不够准确。

6. 生物信息挖掘

生命科学领域会产生大量的数据，包括生物数据集成、基因序列、生物网络和生物图像等，生物信息挖掘已成为一个活跃的领域。目前，很多研究者都建立了生物信息数据库，来解决生物信息多而杂乱的问题，在生物信息数据库的基础上可以方便地进行更加复杂的数据分析。例如利用序列分析生物基因密码，使用聚类分析方法对住院患病人群进行分类，利用时序分析进行流感疫情预测，采用关联分析进行患者并发症分析等。

7. 文本挖掘和 Web 挖掘

自从 Web 出现以后，Web 替代了传统媒体，成为目前最流行、使用最广泛、信息量最大的信息平台，针对 Web 的内容挖掘、结构挖掘和使用挖掘已经有了相当大的进展。随着 Web 2.0 的发展，在 Web 挖掘中增加了动态内容挖掘和个性化信息挖掘的要求，促进了 Web 挖掘进一步的发展。文本挖掘是数据挖掘中研究比较早的一个领域，文本挖掘的方法也可以应用于半结构化和非结构化数据中，如数字图书馆或生物数据。Web 挖掘和文本挖掘是数据挖掘中发展很快的一个领域，基于语义分析或适应个性化需求的挖掘方法是未来研究的热点。

8. 软件工程和系统分析中的数据挖掘

在软件的构建和运行过程中，都会积累大量的数据，如何利用这些数据从而提高系统的运行效率是一个值得关注的问题。例如，在工作流系统中，可以通过工作流挖掘的方法找出系统中的瓶颈或异常，从而提高系统性能。这类数据挖掘工作按照是否实时收集分析数据可以分为动态挖掘和静态挖掘。目前这个领域仍存在很多不足。

9. 面向数据立方体的多维 OLAP 挖掘

基于数据仓库的数据立方体计算和 OLAP 可以提高对多维、大型数据集的分析能力。除了传统的数据立方体外，很多基于复杂的统计数据立方体，如回归立方体、预测立方体等

都已被应用于多维分析中,促使 OLAP 和数据挖掘的结合,即 OLAP 挖掘。

此外,大数据时代产生的海量数据对传统数据挖掘算法提出了挑战,解决实时性需求迫在眉睫。而实时性对推荐系统、营销分析等应用是非常重要的。目前比较流行的方法是使用云计算等分布式计算平台,采用 Map Reduce 技术把计算任务分解计算后汇总,同时利用并行计算提高数据挖掘算法的效率。实时性数据挖掘在社交网络分析、网络安全监测、社会网络情感分析等方面有广泛的应用前景。

思考题

1. 简述数据挖掘的发展史?

2. 数据挖掘有哪些步骤? 以电信运营商的顾客细分为例,分析每一步骤关键的问题。

3. 作为一种数据挖掘方法和展示工具,举例说明可视化技术的应用。

4. 举例说明数据挖掘在银行、保险、电信、零售或政府管理中的应用。

5. 数据预处理在数据挖掘过程中有什么作用? 常见的预处理方法有哪些? 数据降维和特征获取常用哪些方法? 请举例说明。

6. 聚类算法的实质是什么? 常用的几种聚类算法分别适用于什么场合? 请举例说明某种聚类算法的应用。

7. 分别取 $k=2$ 和 3,利用 k-means 聚类算法对以下的点聚类:(2,1)、(1,2)、(2,2)、(3,2)、(2,3)、(3,3)、(2,4)、(3,5)、(4,4)、(5,3),并讨论 k 值以及初始聚类中心对聚类结果的影响。

8. 分类问题的实质是什么? 有哪些常用的方法? 分类算法的性能如何评价?

9. 表 5.22 是购买汽车的顾客分类训练样本集。假设顾客的属性集中家庭经济状况、信用级别和月收入之间条件独立,则对于某顾客(测试样本),已知其属性集 $X=$(一般,优秀,12k),利用朴素贝叶斯分类器计算这位顾客购买汽车的概率。

表 5.22 购买汽车的顾客分类训练样本集

序号	家庭经济状况	信用级别	月收入/元	购买汽车
1	一般	优秀	10k	是
2	好	优秀	12k	是
3	一般	优秀	6k	是
4	一般	良好	8.5k	否
5	一般	良好	9k	否
6	一般	优秀	7.5k	是
7	好	一般	22k	是
8	一般	一般	9.5k	否
9	一般	良好	7k	是
10	好	良好	12.5k	是

10. 决策树算法的实质是什么? 以机器学习数据库中 splice 数据集为例,回答下面的问题。

(1) 分别计算信息增益和 Gini 指数,分析应选择哪个属性为决策树根节点的分支

属性？

(2) 使用 ID3 算法构造决策树。

注：splice 数据集下载地址为 http://archive.ics.uci.edu/ml/datasets。

11. 连续型属性如何离散化？请用 ID3 算法或 C4.5 算法举例说明。

12. 决策树算法的过拟合问题如何解决？

13. 选择合适的数据，应用 CART、C4.5 算法挖掘决策树，并与 ID3 算法的结果进行比较。

14. 支持向量机的基本思想是什么？请举例说明支持向量机的应用。

15. 讨论 BP 神经网络处理分类问题的原理，并举例说明此网络的应用。

16. 考虑表 5.23 中的一维数据集，分别根据 1 最近邻、3 最近邻、5 最近邻和 8 最近邻，使用多数表决投票对数据点 5.0 分类，讨论 k 最近邻分类中 k 的取值对分类结果的影响(表中“+”和“−”表示类别)。

表 5.23 k 最近邻分类数据集

数据点	0.6	3.1	4.4	4.6	4.7	4.9	5.3	5.6	7.2	9.8
类别	−	−	+	+	+	−	−	+	−	−

17. 关联规则挖掘的基本思想是什么？

18. 对于表 5.24 所示的数据集，假设最小支持数和最小置信度分别为 2 和 65%，考虑下面问题。

表 5.24 购物篮事务

事务	购买商品	事务	购买商品
1	{牛奶，啤酒，尿布}	6	{牛奶，尿布，面包，黄油}
2	{面包，黄油，牛奶}	7	{面包，黄油，尿布}
3	{牛奶，尿布，饼干}	8	{啤酒，尿布}
4	{面包，黄油，饼干}	9	{牛奶，尿布，面包，黄油}
5	{啤酒，饼干，尿布}	10	{啤酒，饼干}

(1) 画出该数据集的项集格，判断每个节点是否为频繁项集。

(2) 分别用 Apriori 算法和 FP 增长算法挖掘表中数据集，提取所有的强关联规则。

19. 序列分析与关联规则挖掘有什么关系？请举例讨论。

20. 对于表 5.15 的序列数据库，假设最小支持度为 20%，利用类 Apriori 算法提取所有的序列模式。

21. 时间序列分析与序列分析有什么关系？

22. 表 5.25 是某商品多次价格变动与相应销售量的数据，请利用回归分析求出价格 x 与销售量 y 的关系(提示：x 与 y 的关系大致为抛物线，先变换为线性回归问题再求解)。

表 5.25 价格变动与相应销售量的数据

价格 x	1.2	1.8	3.1	4.9	5.7	7.1	8.6	9.8
销售量 y	4.5	5.9	7.0	7.8	7.2	6.8	4.5	2.7

23. 哪些数据挖掘算法之间可以组合使用？请举例说明。

24. 查阅最新资料，讨论目前数据挖掘面临的一些挑战。

25. 组合分类方法的基本思想是什么？举例说明其应用。

26. 阅读以下论文，讨论如何使用 Logistics 回归、随机森林和多层前向神经网络等多种分类算法对 P2P 平台的用户信用进行预测。

MOSCATO V，PICARIELLO A，SPERLÍ G. A benchmark of machine learning approaches for credit score prediction[J]. Expert Systems with Applications，2021，165.

27. 讨论数据挖掘中的隐私保护方法。

28. 以某个企业的促销项目为例，讨论利用数据挖掘选择目标客户的过程。

29. 某产销一体化企业经过多年的信息化建设，已经建立了比较完善的 CRM、ERP、OA 等基础信息系统，并积累了大量的历史数据。但这些大量、分散、独立存在的数据对于业务人员、管理人员来说，很难充分利用。如何知道什么客户对公司的价值贡献最大，他们的特征是什么？如何了解哪些产品之间关联程度比较强，以便给出合理的定价策略？怎么分析一部分顾客购买了 A 商品一段时间后还会购买其他某类商品，以便主动推荐？如何分析广告投入对未来的销售量的影响？该公司最近几年还开展了电子商务业务，对零售网站的顾客访问日志、购物篮数据可以做哪些分析？举例说明。

30. 讨论下面的数据分析需要使用何种数据挖掘方法？并给出简单的分析思路。

(1) 给出某电商平台前几个月一些客户的浏览和交易日志数据，预测未来一个月客户可能的行为。

(2) 某汽车制造商为了推广新的车型，计划请某社交平台有影响的人物试驾，并在该社交平台发布试驾报告。

(3) 某个体户想开一家川菜馆，请利用大众点评网的餐馆介绍、点评等相关数据，分析菜馆的选址以及配套设施，给出理由。

(4) 某银行在客户分析中，需要了解年龄和收入对客户价值的影响。

(5) 某保险公司欲推出面向农民的自然灾害险，需要预测今年的灾害发生情况。