多属性数据挖掘研究中的关联规则应用

The Association Rules' Application In Research Of Multi – Dimensional Data Mining

宋　欣　（天津职业大学电子信息工程学院 天津 300402）
王志航　（北京 NEC 公司 北京 100876）
廉明欢　（北京邮电大学 计算机科学与技术学院 北京 100876）

摘要：数据挖掘是一个崭新的计算机应用领域，它将极大地促进信息对于人类社会进步所起的作用。在对数据挖掘中关联规则进行初步讨论后，提出了一些基本的概念和算法，并对于找频繁项集的问题进行了程序实现上的尝试，取得了一些有用的结果。

关键词：数据挖掘 关联规则 算法

1　数据挖掘产生的必然性

随着数据库技术的迅速发展以及数据库管理系统的广泛应用，人们积累的数据越来越多。激增的数据背后隐藏着许多重要的信息，人们希望能够对其进行更高层次的分析，以便更好地利用这些数据。目前的数据库系统可以高效地实现数据的录入、查询、统计等功能，但无法发现数据中存在的关系和规则，无法根据现有的数据预测未来的发展趋势。缺乏挖掘数据背后隐藏的知识的手段，导致了"数据爆炸但知识贫乏"的现象。

2　数据挖掘的介绍

数据挖掘（Data Mining）就是从大量的、不完全的、有噪声的、模糊的、随机的实际应用数据中，提取隐含在其中的、人们事先不知道的、但又是潜在有用的信息和知识的过程。这个定义包括好几层含义：数据源必须是真实的、大量的、含噪声的；发现的是用户感兴趣的知识；发现的知识要可接受、可理解、可运用；并不要求发现放之四海皆准的知识，仅支持特定的发现问题。

数据挖掘是一种新的商业信息处理技术，其主要特点是对商业数据库中的大量业务数据进行抽取、转换、分析和其他模型化处理，从中提取辅助商业决策的关键性数据。数据挖掘可以描述为：按企业既定业务目标，对大量的企业数据进行探索和分析，揭示隐藏的、未知的或验证已知的规律性，并进一步将其模型化的先进有效的方法。

目前数据挖掘的热点主要是网站的数据挖掘（Web site data mining）、生物信息或基因的数据挖掘、文本的数据挖掘（Textual mining）等几个方面。

3　数据挖掘算法——关联规则

关联规则是寻找在同一个事件中出现的不同项的相关性，比如在一次购买活动中所买不同商品的相关性。要计算包含某个特定项或几个项的事务在数据库中出现的概率只要在数据库中直接统计即可。某一特定关联在数据库中出现的频率称为支持度。比如在总共 1000 个事务中有 15 个事务同时包含了"锤子和钉子"，则此关联的支持度为 1.5%。

要找到有意义的规则，我们还要考察规则中项及其组合出现的相对频率。当已有 A 时，B 发生的概率是多少？也即概率论中的条件概率。这个条件概率在数据挖掘中也称为可信度，计算方法是求百分比：（A 与 B 同时出现的频率）/（A 出现的频率）。

对关联规则按不同的情况进行分类可以有多种不同的分类方法。其中一种分类方法是按照规则中涉及的数据的维数，关联规则可以分为单维的和多维的。

由于关联规则的分类方法较多，在这里不再一一

介绍，只介绍涉及多维数据的关联规则，即：应用于多属性的关联规则。

（1）多属性挖掘简述。在实际生活中，我们经常要了解不同事物之间存在着哪种必然的联系。比如有这样一条规律：年龄在 20－30 岁的学生在所有购买电脑的人群当中占有相当大的比重。我们将这一规律用公式来表达：

年龄(X,"20...30")∪(职业(X,"学生")⇒ 购买(X,"笔记本电脑")

由此我们可以看出，这里涉及到三个维上的数据：年龄、职业、购买。

（2）多属性关联规则挖掘算法的关键。由于多属性的关联规则数据挖掘要涉及到的属性不止一个，很显然，这一算法实现的复杂程度比事务数据库的挖掘要困难一些。这些困难包括数据项的预处理、产生规则维数的不可知性等等。下面我们先从单属性挖掘的一个经典算法看起，首先掌握关联规则实现的基本原理，再来解决刚才提出的一系列问题。

（3）经典的 Apriori 算法。Apriori 算法是一种最有影响的挖掘布尔关联规则频繁项集的算法。它使用一种称作逐层搜索的迭代方法，k－项集用于探索(k+1)－项集。首先，找出频繁 1－项集的集合。该集合记作 L1。L1 用于找频繁 2－项集的集合 L2，而 L2 用于找 L3，如此下去，直到不能找到频繁 k－项集。

为提高频繁项集项集逐层产生的效率，一种称作 Apriori 性质的重要性质用于压缩搜索空间。

Apriori 性质：频繁项集的所有非空子集都必须也是频繁的。Apriori 性质基于如下观察：根据定义，如果项集 I 不满足最小支持度阈值 min_sup，则 I 不是频繁的，即 P(I) < min_sup。如果项 A 添加到 I，则结果项集(即 I∪A)不可能比 I 更频繁出现。因此，I∪A 也不是频繁的，即 P(I∪A) < min_sup。

4　多属性关联规则挖掘算法的实现

4.1　数据结构的设计

利用 Apriori 算法，我们可以得到多维关联规则，但是产生规则维数的不确定性为规则在计算机内的存储造成了不便。

下面给出一个能解决这一问题的数据结构。

该数据结构是由三个结构体通过指针串连在一起的，每一个结构体都存储了表中不同层次上的信息。

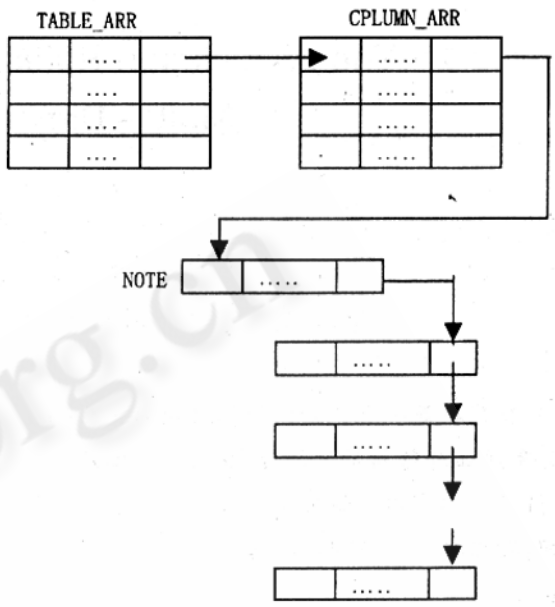

图 1　数据结构图

TABLE_ARR：

该结构体存储了每一个属性的简要信息，它的元素个数就是要挖掘表的属性个数。TABLE_ARR 的数据成员如下：

int nTableTag

表标志项，由于如果规则中所有涉及的属性全部是来自推销员或者客户的信息，这些规则在实际分析过程中是毫无意义的。设立这个标志可以避免这种情况的发生。如：

将推销员年龄、工作时间、推销员性别这三项的标志项设为 1，涉及客户信息的几个属性的标志项设为 2。

int nLengthColumnArr

指向的 COLUMN_ARR 数组的长度，代表该属性的频繁一项集中不同属性的个数。例如推销员年龄这一项，如果只有 30－40 和 40－50 年龄段的人数满足最小支持度的要求，而其他年龄段的人数过少，那么 nLengthColumnArr 的值为 2。

int nColumnId

由于 TABLE_ARR 中不记录属性名，所以用这一项作为代号区分不同的属性。

COLUMN_ARR * pColumn

指向 COLUMN_ARR 数组的指针。

由此可见，TABLE_ARR 结构存储了表中横向的信息。

COLUMN_ARR：

该结构体存储了每一个属性的频繁一项集中属性的名称、计数等相关信息。它的元素个数就是指向它的 TABLE_ARR 表中相应项的 nLengthColumnArr 的值。它的数据成员如下：

CString cColumnValue

属性名称

int nCount

该属性在频繁一项集中的计数

int nColumnId

属性 id，对应于 TABLE_ARR 表中相应项的 nColumnId。

int nSelfId

自身 id，恒为 0

NODE * pNode

指向 NODE 结点的指针

由此可见，COLUMN_ARR 结构存储了表中每一个属性纵向的信息。

NODE：

NODE 结构是三个结构体当中最重要，也是存储信息最多、存储结构最复杂的结构体。所有的规则都在这一结构中产生。

NODE 结构体虽然在计算机中是以链表的形式存在，但是它本身要说明的其实是树型结构。NODE 链中结点的组织是按照从根结点往下逐层从左到右遍历的顺序存储的。根结点是 NODE 链中的第一个结点，也是 COLUMN_ARR 结构中 pNode 指针指向的结点。我们也称之为头结点。它的实际作用是指向它的 COLUMN_ARR 结构相应项的一个副本，在初始化 COLUMN_ARR 结构时生成。

NODE 结构的数据成员如下：

NODE * pNextNode;

后链

CString cNodeValue;

属性名

int nCount;

从根结点到当前结点遍历所经过属性组在数据库中的计数

int nColumnId;

属性 id，对应于 TABLE_ARR 表中相应项的 nColumnId。

int nParentId;

双亲节点的 id

int nSelfId;

自身 id

int nTreeHeight;

结点在树中所在的层，根结点为 1，根的孩子结点为 2……以此类推。

规则的读取实际上是完成从根结点到其他结点遍历的过程。为了保证产生规则中的值来自不同的属性以及不重复存储规则信息，在程序中应用了相关的机制和数据结构。

4.2 程序流程设计

程序的基本流程主要由四个步骤组成：

(1) 初始化 TABLE_ARR 结构的信息和其他相关信息。这一步完成一些准备工作。首先扫描数据库，完成属性个数计数和记录条数的计数，其中要挖掘的属性个数就是 TABLE_ARR 数组的长度。为 TABLE_ARR 分配空间，因为要挖掘的属性是 6 个，所以定义全局变量 TABLE_ARR array[6]，并完成相应的初始化工作。

(2) 寻找频繁一项集(即初始化 COLUMN_ARR 结构并生成 NODE 结构头结点)。扫描数据库中每一个待挖掘的属性，查询该属性下的每一种值的计数是否满足最小支持度。若满足，则将它加入到该属性的频繁一项集之中。

(3) 寻找频繁 n 项集(即完成 NODE 链结构)。设置一个循环计数 l，l 从 2 开始到待扫描的属性个数结束，每一次循环结束就完成 l 维关联规则的产生。循环结束后，所有规则全部产生完毕。为使效率提高，设置一个规则变化标志。当每一次循环执行完时检查这一标志，如果在这次循环的过程中没有产生一条新规则，则根据 Apriori 性质，不可能再产生更高维数的规则，直接返回。

前面已经提到过，规则的生成实际上是对 NODE 链的构造，NODE 链所代表的树型结构的高度就是能产生规则的最大维数。利用这一点，在执行第 l 次循环

时,先查看一下 NODE 链最后一个结点的 nTreeHeight 值(代表树的高度),如果该值小于 I-1,说明在执行上一次循环时,没有产生新的规则,因此就没有必要再次扫描这一条 NODE 链了,转而去扫描其他的 NODE 链。

(4) 扫描 NODE 结构(遍历树),读取合理规则并输出 。通过给出自身 id、双亲结点 id 和所在树的层数三个值,完全可以构造一颗逻辑意义上的树。规则的读取就是从根结点到每个叶子结点的遍历过程。

下面我介绍一下规则产生的过程。

借助一个数组类型的数据结构,我将它起名叫做追踪数组。该数组的每一个值是指向 NODE 类型的指针,存放 NODE 链中相应结点的地址。在产生规则时,需要扫描每一条 NODE 链,每扫描一条 NODE 链时,都要对追踪数组进行初始化。

在输出规则的时候应该注意放弃那些没有实际意义的干扰规则。比如,一条规则只包括了推销人员的信息而没有设计到客户的任何的信息,这些规则是没有任何意义的,不应输出。因此,利用前面提到的 TABLE_ARR 结构的 nTableTag 标志可以做到这一点。

5 结论

程序中数据结构的定义是比较简单的,程序的流程也比较清晰和单一。作为数据挖掘中的一个模块,本程序实现的关联规则算法有着重要的应用。

本文给出了对于数据挖掘中关联规则的初步讨论,以及一些基本的概念和算法,并对于找频繁项集的问题进行了程序实现上的实现,取得了一些有用的结果。

数据挖掘是一个崭新的计算机应用领域,它将极大地促进信息对于人类社会进步所起的作用。在对大量的已知信息进行知识发现和整理的过程中,我们可以提供一定程度的决策支持,在生产,销售,保险,事故分析等等领域发挥巨大的经济和社会效益。

参考文献

1 http://www. almaden. ibm. com/cs/quest/demo/assoc/general. html.

2 http://www. sgi. com/software/mineset. html.

3 B Hetzler, W M Harris, S Harvre, and P Whitney, Visualizing the full spectrum of document relationships. In: Proceedings of the Fifth International Society for Knowledge Organization Conference, 1998.

4 M Hao, M Hsu, U Dayal, S F Wei, T Sprenger, T Holenstein. Market basket analysis visualization on a spherical surface. HP Labs Technical Report, HPL-2001-3, 2001.

5 Jiawei Han and Micheline Kamber, Data Mining: Concepts and Techniques, Simon Fraser University, 2000.

6 朱扬勇、周欣、施伯乐,规则型数据采掘工具集 AMINER,高技术通讯,Vol.10 No.3 2000.