

一种高效的关联规则维护算法研究与实现

郭有强

(蚌埠学院 计算机科学与技术系, 安徽 蚌埠 233030)

摘要:关联规则挖掘是数据挖掘领域中的重要研究内容之一。由于数据挖掘的过程是动态交互的,因此对已经发现的关联规则进行维护更新显得非常重要。提出了一种实用的在支持度和置信度不变的情况下数据集规模减小的负增量关联规则维护算法。算法在如何减少数据集的扫描次数,如何充分利用现有的信息减少候选集的规模等方面进行了研究,给出了算法的具体实现。理论分析和实验结果表明算法是有效的。

关键词:数据挖掘;关联规则;增量维护算法;剪枝

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2007)10-0123-04

Study and Implementation of High-Performance Maintenance Algorithm for Mining Association Rules

GUO You-qiang

(Computer Sci-tech Department, Bengbu College, Bengbu 233030, China)

Abstract: Mining of association rules is an important research topic among the various data mining problems. The issue of maintaining discovered association rules is paid more attention in the same way. This paper provides a practical maintenance algorithm for negative incremental association rules in which the size of data sets is reduced, with the supporting and confidence limits unchanged. The algorithm explores how to diminish the number of scanning data sets, and how to make the best use of known information to shorten the size of candidate sets and so on. The concretization of the algorithm is also given. Theoretical analysis and experimental results suggest that is effective.

Key words: data mining; association rules; incremental maintenance algorithm; pruning

0 引言

随着数据库技术广泛应用,数据库中存储的数据量急剧增大,而数据挖掘技术就是从这些大量的数据中发现有效的、新颖的、具有潜在应用价值的知识,以及最终可理解模式的非平凡过程。它的目的在于提高市场决策能力,检测异常模式,在过去的经验基础上预言未来趋势。关联规则^[1]是数据挖掘中一种重要的模式,也称为关联模式。

关联规则算法是数据挖掘的一个重要研究方向,其侧重于确定数据集中不同领域的联系,找出满足给定支持度和可信度的多个域之间的相互关系。由于数据库极其庞大,不仅需要设计高效的算法来挖掘关联规则,而且也迫切需要设计高效的算法来更新、维护和管理已经挖掘出来的关联规则。目前的挖掘算法很

多,如 Apriori^[2], AIS^[3], DHP^[4], Partition^[5]和抽样算法等。维护更新算法也相继出现不少,大多数的算法都是以 Apriori 算法为核心进行的改进与优化,如 FUP^[6], FUP2^[7], IUA^[8]和 UWEP^[9]算法等,但是它们仍旧不能高效地解决维护更新的问题。文中研究的是利用已经获得的频繁项目集的结果,解决在支持度和置信度不变的情况下数据集规模减小的模式维护问题,并通过实例验证了该算法的正确性和高效性。

1 问题描述和相关性质

关联规则的挖掘问题可以分解成以下两个子问题。

(1) 找出事务数据库 D 中所有具有用户指定最小支持度的项集的一个非空子集。具有最小支持度的项集称为频繁项集(frequent itemset),反之就称为非频繁项集。

(2) 利用频繁项集生成所需要的关联规则。这些规则必须满足最小支持度和最小置信度。

第二个问题较为容易和直观,主要的工作集中在

收稿日期:2006-12-22

基金项目:安徽省科技厅自然科学基金项目(050420207)

作者简介:郭有强(1966-),男,江苏邳江人,硕士,副教授,从事数据挖掘和可视化程序设计教学与研究。

第一个问题上。

维护更新算法的核心就是利用已挖掘的关联规则为基础,在变化了的数据库/参数上发现新的关联规则,删除失效的关联规则,进一步解决关联规则的维护更新问题。

原事务数据库 D , 已知当前 D 的频繁项集 L_D , 现保持最小支持度 s 不变, 欲减少事务数据集为 d , 由于事务数据库的变化, 导致原来的频繁项集 L_D 对于变化后的数据库 $D-d$ 已无意义。

$|D|$ 、 $|d|$ 、 $|D-d|$ 分别为原数据库、欲减数据集、减后数据库的总事务条数; L_D 为原数据库 D 的频繁项集, L_d 为欲减数据集 d 的频繁项集, L_{D-d} 为减后数据库 $D-d$ 的频繁项集; D 、 d 和 $D-d$ 的最小支持数分别为 support_D 、 support_d 、 support_{D-d} ; 项目集 X 在数据集 D 、 d 、 $D-d$ 中的支持数目分别记为 $X.\text{count}_D$ 、 $X.\text{count}_d$ 、 $X.\text{count}_{D-d}$ 。

性质 1: 当 $X.\text{count}_{D-d} \geq s \times |D-d|$ 时, $X \in L_{D-d}$ 。

证明: 由最低支持度和频繁项目集的定义可直接推导出。

性质 2: 保持减后数据库 $D-d$ 的支持度不变(s), 原数据库 D 的支持数为 $\text{support}_D = s \times |D|$, d 的支持数为 $\text{support}_d = s \times |d|$, 减后数据库的支持数应为 $\text{support}_{D-d} = s \times |D-d|$ 。

证明: 因为 D 的支持度为 s

$$s = \text{support}_D / |D| = (\text{support}_{D-d} + \text{support}_d) / |D-d + d|$$

$$D-d + d = (\text{support}_{D-d} + s \times |d|) / |D-d + d|$$

所以 $\text{support}_{D-d} = s \times |D-d|$

性质 3: $X \in L_D \cup L_d - L_D$ (图 1 中阴影部分), 则 X 在 $D-d$ 中一定是不频繁的。

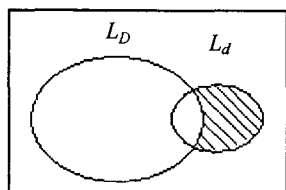


图 1 $X \in L_D \cup L_d - L_D$ 示意图

反证法: 显然 $X \in L_D$, $X \in L_d$ 所以 $X.\text{count}_d \geq s \times |d|$ ①

假设 $X \in L_{D-d}$ 则 $X.\text{count}_{D-d} \geq s \times |D-d|$ ②

$$\textcircled{1} + \textcircled{2} \quad X.\text{count}_{D-d} + X.\text{count}_d \geq s \times |D-d| + s \times |d|$$

$$X.\text{count}_D \geq s \times |D|$$

得: $X \in L_D$ 与题意相反, 所以假设不成立

性质 4: $X \in L_D \cup L_d - L_d$ (图 2 中阴影部分), 则

X 在 $D-d$ 中一定是频繁的。

反证法: 显然 $X \in L_D$, $X \in L_d$ 所以 $X.\text{count}_d < s \times |d|$ ①

假设 $X \in L_{D-d}$ 则 $X.\text{count}_{D-d} < s \times |D-d|$ ②

$$\textcircled{1} + \textcircled{2} \quad X.\text{count}_D < s \times |D|$$

得: $X \in L_D$ 与题意相反, 所以假设不成立

性质 5: $X \in L_D \cap L_d$, 则 X 在 $D-d$ 中不一定是频繁的。

证明: 因为 $X \in L_D$ 所以 $X.\text{count}_D \geq s \times |D|$

$$X.\text{count}_{D-d} + X.\text{count}_d \geq s \times |D|$$

$$X.\text{count}_{D-d} \geq s \times |D| - X.\text{count}_d$$

又因为 $X \in L_d$ $X.\text{count}_d \geq s \times |d| = s \times |d| + n$ ($n = 0$ 或正整数)

$$X.\text{count}_{D-d} \geq s \times |D| - s \times |d| - n = s \times |D-d| - n$$

所以不能保证 $X.\text{count}_{D-d} \geq s \times |D-d|$

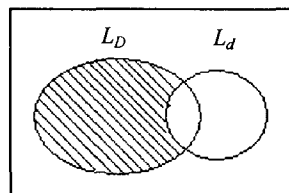


图 2 $X \in L_D \cup L_d - L_d$ 示意图

2 维护算法基本思想及算法实现

2.1 维护算法基本思想

(1) 在分离欲减数据集 d 的过程中, 得到 d 中所有 1 项目的相关信息(项目支持事务 ID 集合, 每项目支持事务总计数), 得到频繁 1-项集; 在此基础上使用连接项集(即当前的频繁 1-项集)中的项目对频繁 1-项集进行增长, 分别得到候选 2-项集(此方法称为“项目增长法”), 从而求得频繁 2-项集, 将频繁 2-项集中的所有项包含的单一项目构成的集合, 与连接项集求交集, 动态更新连接项集中的项目(即产生新的连接项集, 为以后增长做准备); ……; 在得到频繁 $K-1$ 项集后, 利用求解过程中更新的连接项集中的项目进行增长, 分别得到候选 K 项集, 从而求得频繁 K 项集。依据此方法只需要扫描数据集一次, 便得到 d 的所有频繁项集。

(2) 扫描减后的数据集 $D-d$, 得到 $D-d$ 中所有 1 项目的相关信息(项目支持事务 ID 集合, 每项目支持事务总计数)。

(3) 所有项目集合由两部分组成: $L_D \cup L_d$ 和不属于 $L_D \cup L_d$ 部分的项目集合。

a. 考虑 $L_D \cup L_d - L_D \cap L_d$:

① $L_D \cup L_d - L_D$: 即 X 在 d 中是频繁的, 但在 D

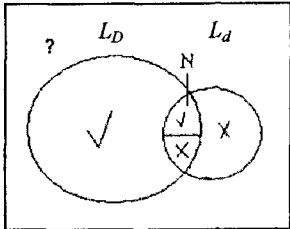
中是不频繁的,则在 $D-d$ 中一定是不频繁的(性质 3)。

② $L_D \cup L_d - L_d$:即 X 在 D 中是频繁的,但在 d 中是不频繁的,则在 $D-d$ 中一定是频繁的,即 $X \in L_{D-d}$ (性质 4)。

b.考虑 $L_D \cap L_d$:此部分的项集在 $D-d$ 中可能频繁,也可能不频繁(性质 5)。

列出所有项目包含的单一项目,在 $D-d$ 中查看单一项目支持度,将所有 $\geq s * |D-d|$ 的作为一个集合 M , $M \in N$;在 $L_D \cap L_d$ 中去除有子项不包含在 M 中的项,利用子项交集的方法在 $D-d$ 中计算其他项目的事务支持数,若 $X.count_{D-d} \geq s * |D-d|$, $X \in N$, $N \in L_{D-d}$ 。

综上所述,利用已知信息,得到在 $D-d$ 中项集是否频繁的分布如图 3 所示。



(注:图中 ✓ 为频繁, × 为不频繁, ? 为不一定)

图 3 $L_D \cup L_d$ 中的项集在 $D-d$ 中分布图

c.考虑不属于集合 $L_D \cup L_d$ 的部分:

$D-d$ 中所有 1 项集的支持度 $\geq s * |D-d|$ 的为频繁 1 项集 L_1 , $L_1 \in L_{D-d}$;将 L_1 中的项进行两两组合得到二项集,二项集中的项目若在 L_{D-d} 中,将其放入 C ,若不在,则进一步判断是否在 $L_d - N$ 中,若不在,则是候选二项集,计算候选二项集的支持数,得到频繁二项集 H ,则 $H \in L_{D-d}$, $L_2 = H \cup C$,清空 H 、 C ;将 L_2 中的单一项目集合 L 作为连接项集,与 L_2 组合,得到三项集,三项集中的项目若在 L_{D-d} 中,将其放入 C ,若不在,则进一步判断是否在 $L_d - N$ 中,若不在,则是候选三项集,计算其支持数,得到频繁三项集 H ,则 $H \in L_{D-d}$, $L_3 = H \cup C$,清空 H 、 C ;……。

d.返回 L_{D-d} 。

2.2 算法实现

第一步:分离出欲减数据集 d ,计算 d 的频繁项集 L_d 。扫描 d ,得到 d 中每个单一项目的事务支持 ID 集合和记录总数 $|d|$,得到频繁 1 项集 L_1 。

$L_1 \in L_d$; $L = L_1$;

for ($k = 2$; $C_k \neq \emptyset$; $k++$) // C_k : L_{k-1} 与 L 连接得到的候选 k 项集

{

for all $X \in C_k$

得到频繁 k 项集 L_k ;

$L_k \in L_d$;

$L = \{L_k \text{ 中项目的单项集合} \}$ // 更新 L

}

第二步:考虑 $L_D \cup L_d - L_D \cap L_d$ 中的项集。

for all $X \in (L_D \cup L_d - L_d)$

$X \in L_{D-d}$;

第三步:考虑 $L_D \cap L_d$ 。扫描 $D-d$,得到 $D-d$ 中每个单一项目的事务支持 ID 集合和记录总数 $|D-d|$,得到频繁 1 项集 L_1 。

for all $X \in (L_D \cap L_d)$

{

for each 所有项目包含的单一项目 x

{

if ($X.x.count_{D-d} \geq s * |D-d|$) // 项目 X 中的某单一项目 x 在 $D-d$ 中的支持事务计数

$x \in M$;

}

$M \in L_{D-d}$;

// 单一项目被包含在 M 中的所有项目,去掉有单一项目不被包含在 M 中的项目集

for each $X \mid X.x \in M$ // $X.x$: X 中的单一项目

if ($X.count_{D-d} \geq s * |D-d|$)

$X \in N$; // D 和 d 交集部分,在 $(D-d)$ 中频繁的项目集合

$N \in L_{D-d}$;

}

第四步:考虑不属于集合 $L_D \cup L_d$ 的部分。

$L_1 \in L_{D-d}$; $L = L_1$;

for ($k = 2$; $C_k \neq \emptyset$; $k++$) // C_k : L_{k-1} 与 L 连接得到的项目

{

for all $X \in C_k$

{

if ($X \in L_{D-d}$)

$X \in C$; // C : 已经确定过的频繁项集,剪掉

else if ($X \notin L_d - N$)

$X \in C'_k$; // C'_k : 剪枝后的候选集

}

for all $X \in C'_k$ // 确定候选集中的频繁项

if ($X.count_{D-d} \geq s * |D-d|$)

$X \in H$;

$H \in L_{D-d}$;

$L_k = H \cup C$; // 频繁 k 项集由 H 和 C 两部分构成

$L = \{L_k \text{ 中项目的单项集合} \}$ // 更新 L

}

Answer = L_{D-d}

3 实验及性能分析

采用的测试机为台式 PC 机,其配置是:CPU 为 Pentium43.06GHZ,512MB 内存,操作系统为 Windows

2003 服务器,选用 VC++ 6.0 编程环境。数据来源为蚌埠学院教务处学籍管理数据,数据库采用 SQL server 2000,属性有 15 个。

支持度 5%,用项目增长法分别求得不同事务数的频集测试结果(见表 1)。

表 1 使用项目增长法测试数据描述(时间单位:s)

记录数	4000	5000	6000	7000	8000	9000	10000
支持度 5%	8.12	9.69	11.09	12.5	13.91	14.06	16.72

支持度 5%,先用项目增长法求得事务数为 10000 的频繁项集,然后利用负增量维护方法计算每递减 1000 事务的频繁项集,测试结果(见表 2)。

表 2 利用负增量维护方法测试结果(时间单位:s)

记录数	9000	8000	7000	6000	5000	4000
支持度 5%	12.36	12.2	11	10.5	9.1	8

两种测试结果的性能对比如图 4 所示。

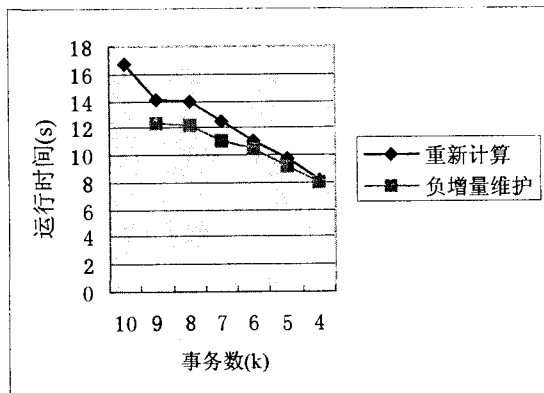


图 4 两种测试结果的性能对比

可以看出,利用负增量维护方法计算变化后的数据集的频集时,所用的时耗比重重新计算要小,故效率高。从而验证了文中讨论的方法的正确性、可行性和有效性。

4 结束语

关联规则的更新是数据挖掘技术中的一个重要内容。笔者在对正负增量式关联规则更新技术^[8,10,11]深入研究基础上,提出了相应的算法。通过分析可以看出:整个更新过程只扫描一次数据集,避免了反复扫描事务数据集的 I/O 负载;通过连接项集产生候选项,增强了产生候选项的针对性和有效性,提高了候选项的支持事务计数的效率;充分利用已知挖掘结果和正在挖掘过程中得到的信息,获得无需计算支持数便已知

的 L_{D-d} 中的项目,剪枝过程贯穿于整个算法,大大减少了需要在 $D-d$ 中计算支持事务计数的候选集的规模。在算法的运行时耗上具有很明显的优势。历史数据集越大,剪枝效果越加明显,越能显现出本算法的优越性。

参考文献:

- [1] Han Jiawei, Kamber M. 数据挖掘概念与技术[M]. 北京:机械工业出版社,2001.
- [2] Agrawal R, Imielinski T, Swami A. Mining Association Rules between Sets of Items in Large Databases[C]//Proc. of the ACM SIGMOD Int. Conf. on Management of Data (ACM SIGMOD'93). Washington, USA: [s. n.], 1993:207-216.
- [3] Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules[C]//Proc. of the 20th Int. Conf. on Very Large Databases (VLDB'94). Santiago, Chile: [s. n.], 1994:487-499.
- [4] Park J S, Chen M S, Yu P S. An Effective Hash-based Algorithm for Mining Association Rules[C]//Proc. of the ACM SIGMOD Int. Conf. on Management of Data (ACM SIGMOD'95). San Jose, California: [s. n.], 1995:175-186.
- [5] Savasere A, Omiecinski E, Navathe A. An Efficient Algorithm for Mining Association Rules in Large Databases[C]//Proc. of '95 Int. Conf. Very Large Database (VLDB'95). Zurich, Switzerland: [s. n.], 1995.
- [6] Cheung D W, Han J. Maintenance of Discovered Association Rules in Large Databases: An Incremental Update Technique [C]//Proc. of the 12th Int. Conf. on Data Engineering (ICDE'96). New Orleans, Louisiana: [s. n.], 1996.
- [7] Cheung D W, Lee S D, Kao B. A General Incremental Technique for Maintaining Discovered Association Rules [C]//Proc. of the 5th Int. Conf. on Database Systems for Advanced Applications. Melbourne, Australia: [s. n.], 1997:185-194.
- [8] 冯玉才,冯剑琳.关联规则的增量式更新算法[J]. 软件学报,1998,9(4):301-306.
- [9] Ayan N F. An Efficient Algorithm To Update Large Itemsets with Early Pruning [C]//Proc. of the 5th Int. Conf. on Knowledge Discovery and Data Mining (KDD'99). San Diego, California, USA: [s. n.], 1999.
- [10] 朱红蕾,李 明.关联规则挖掘的维护算法研究[J]. 微机发展,2004,14(2):37-40.
- [11] 朱玉全,宋余庆.关联规则挖掘中增量式更新算法的研究[J]. 计算机工程与应用,2005(15):186-187.

(上接第 122 页)

morphosis of two-dimensional curves[J]. The Visual Computer, 1998, 14:415-428.

- [13] Surazhsky T, Elber G. Metamorphosis of Planar Parametric Curves via Curvature Interpolation[J]. International Journal of Shape Modeling, 2002, 8(2):201-216.