基于 Hash 表的关联规则挖掘算法的改进

卢云彬,曹汉强

(华中科技大学 电子与信息工程系,湖北 武汉 430074)

摘 要:经典的 Apriori 算法在大项目集的挖掘过程中因为重复搜索导致效率低下。提出一种改进的 Hash 表结构应用于 DHP 算法中的项目集存放,定义新的 Hash 函数确定项目集的存放地址,并基于新的 Hash 表结构,以并行挖掘的方式优化关联规则算法的剪枝过程。实验结果表明,与 Apriori 算法相比,文中的方法可以更好地节省存储空间,提高挖掘效率。

关键词:数据挖掘;关联规则;Apriori 算法;DHP 算法;Hash 表

中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2007)06-0012-03

Improvement of Association Rules Mining Algorithm Based on Hash Table

LU Yun-bin, CAO Han-qiang

(Dept. of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: With classical Apriori algorithm, mining large itemsets is inefficient because of repeated scanning. In this paper, develop an algorithm DHP with improved Hash table for efficient large itemset generation. The stored address of itemsets is determined by a new Hash function. Based on the new Hash table, can use parallel mining to improve pruning process in association rules algorithm. From the experiment results, the method in this paper can save more storing space and enhance mining efficiency compared with Apriori algorithm.

Key words: data mining; association rules; Apriori algorithm; DHP algorithm; Hash table

0 引 言

近年来,数据挖掘技术引起了越多越多的关注。关联规则算法是数据挖掘的一个重要研究方向,自其提出后就一直受到重视,它直接反映出项目集中数据间的潜在关联,通过概率统计得出最常见的情况,用以指导以后的决策,以及判断新的行为是否合法,后者应用在入侵检测中有十分重要的意义。

关联规则算法有很多,Agrawal R. 最早提出了经典的 Apriori 算法[1],该算法解决了关联规则的基本定义和要求,但瓶颈是效率低;其后,他又作了很多改进效率的研究,提出候选集是大项目集的充要条件是当且仅当它的所有子集都是大项目集[2],该性质决定了大项目集生成的迭代步骤,可以提前去掉无效的 k - 项目集,避免生成无用的 $(k+1)$ - 项目集,改进了候选集的筛选过程。J. S. Park 等人针对 Apriori 算法挖掘过程中导致重复搜索的问题,首先提出了 DHP(Direct Hashing and Pruning)算法[3],DHP 算法利用直接哈希修剪技术,快速发现频繁项集,提高搜索效率。John D. Holt 和 Soon M. Chung 对上述两种经典算法及其改进算法做了研究综述[4],并应用到文本数据库中进行比较,发现在数据量增大后,DHP 算法相对于传统 Apriori 算法效率有很大提高。

文中提出了一种新的 DHP 算法 Hash 表结构,通过选择有序 Hash 树结构存放频繁项目集,能有效提高挖掘效率,优化了 Apriori 算法生成大项目集的剪枝过程,同时,极大地节省了大项目集存储空间,在时间和空间两方面同时改进了频繁项目集的挖掘效率。

1 Apriori 算法和 DHP 算法的分析

Apriori 算法是一种先验概率算法,它利用了频集特性的先验知识,采取层次顺序搜索的循环方法来完成频繁项集的挖掘工作[5],这一循环方法主要利用 k - 项集来产生 $(k+1)$ - 项集。具体描述如下:首先找出频繁 1 - 项集,记为 L_1;然后利用 L_1 来挖掘 L_2,即频繁 2 - 项集;不断迭代循环,直至无法发现更多的频繁

收稿日期:2006-09-11

基金项目:国家科技攻关项目(2004BA811B06)

作者简介:卢云彬(1982-),男,湖北大冶人,硕士研究生,研究方向为信息安全、数据库系统及其应用;曹汉强,博士,教授,博士生导师,研究方向为数字图像处理、信息安全、激光全息防伪技术。

$(k+1)$ - 项集 L_{k+1} 为止。Agrawal R 证明了大项目集性质:大项目集的任一子集也一定是大的[6]。在这个性质下,迭代过程中增加了候选集 C_k,由大项目集 L_k 产生下一层候选集 C_{k+1},再通过剪枝过程筛选出大项目集形成 L_{k+1}。

Aprirori 算法存在一个重复扫描数据库的问题,即为了对候选集 C_k 剪枝,不得不对 C_k 中的每一项依次作判断,要回到数据库搜索其支持度是否满足大于判决门限 s 的要求,这个重复扫描占用了大量的时间开销。

DHP 算法是利用直接哈希修剪技术的算法,用来减少数据传输量。对候选项目集构建 Hash 表后,直接从 Hash 树中获取各项目集的支持度,而不必重复搜索数据库,能有效减小大项目集的体积,有效减少数据传输量,有效减少数据库扫描趟数。

2　基于有序 Hash 表的 DHP 算法

如前所述,由于候选集中每一项的判断都需要到前一层查询,所以前一层频繁集在 Hash 表中的结构会对查询工作有很大的影响,有序快捷的存储能有效改进查询效率。因此,文中提出了一种有序 Hash 表结构,能够直接通过 Hash 函数定位所需要的项目是否存在,而不必逐项搜寻。

有序 Hash 表结构设计如下:为了在 Hash 表中书写方便起见,项目集采用简写形式,如{*B C*} 简写为 *BC*。

(1) 同层项目集存放在连续的位置,即在某一段或某一列表格中,存放的均为 C_k,在另一段或者另一列表格中存放 C_{k+1};

(2) 在同层项目集中,用 Hash 函数决定其存放位置,这样可以直接得到项目集的存储地址进行读取,而不需要逐项寻找。

Hash 函数定义为:$H(X,Y) = (|(X*2 \pm Y)| \bmod 11) + 1$

其中,X 为项目集的首项,Y 为后面所有项,将各项按照字母顺序对应如表 1 所示。

表 1　各数据项对应的转换数值

A	*B*	*C*	*D*	*E*	*F*	…	*W*	*X*	*Y*	*Z*
1	2	3	4	5	6	…	23	24	25	26

例如,$H(A,B) = (|(1*2+2)| \bmod 11) + 1 = 5$,$H(B,CD) = (|(2*2+3+4)| \bmod 11) + 1 = 1$

不同项可能得到相同 Hash 地址,需要用链表结构来存储冲突项。为了减少因避免冲突而预留太多的存储空间,文中采用了修正 Hash 函数值与链表结合的方法,链表中存放冲突项新的 Hash 地址,此新的 Hash 地址由修正的 Hash 函数得到,例如,项目集 *BC* 按 Hash 函数 $H(X,Y) = (|(X*2+Y)| \bmod 11) + 1$ 计算得到地址 8,填充表格时发现地址 8 已经填入了项目集 *AE*,那么再按 Hash 函数 $H(X,Y) = (|(X*2-Y)| \bmod 11) + 1$ 计算得到地址 2,如发现该地址为空,则将项目集 *BC* 填入,同时在地址 8 的链表中填入 2,表示有一个冲突项已经填入到新的地址 2 中。如果地址 2 又被另一个项 *AJ* 占据,因为已无法从 Hash 函数再得到新的值,此时用链表结构将 *BC* 直接填入地址 8 后面的空位,同时在地址指针中填入地址 8,表示该冲突项填在了地址 8 的后面链表中。

对于数据项远超过 11 的情况,会导致很多冲突数据项在此公式下找不到空位置,在这种情况下应该先修正 Hash 公式,即除数 11 改为略大于数据项总量一半的第一个质数。文中采用的 Hash 地址由 $H(X,Y) = (|(X*2+Y)| \bmod 11) + 1$ 计算得到,修正函数只是用来查找一个新地址存放冲突数据项,以减少空间浪费。

3　实验仿真与数据分析

3.1　文中算法得到的 Hash 表结构

对于包含 6 个事务的 1 - 项集 $L_1 = \{\{A\},\{B\},\{C\},\{D\},\{E\},\{F\}\}$,经过连接得到全部的 2 - 项候选集 C_2,根据算法,得到 C_2 各子项存放的 Hash 表仿真结果如表 2 所示。

表 2　全部 2 - 项集存放的 Hash 表结构

Hash 地址	1	2	3	4	5	6	7	8	9	10	11
2 - 项集	*BD*	*BC*	*CD*	*DE*	*AB*	*AC*	*AD*	*AE*	*AF*	*BE*	*BF*
链表地址	1	2	4	4	0	6	0	2	1	0	3
2 - 项集	*CE*	*CF*		*DF*		*EF*					

从该 Hash 表结构可以看出,该存储方式合理地利用了存储空间,Hash 函数的 2 次地址选择能够让每一层空间尽量用满,而不会因为部分冲突导致链表过长而浪费空间。

该表格的生成时间很短,而且随项目集的增多变化并不大,因为 Hash 地址是由函数直接计算获得,不存在遍历过程,只需要处理冲突情况,所以时间开销很少,但对搜索效果的提高却十分显著。在查找数据项时,只需根据此函数确定地址,冲突项获取的新地址必然在对应链表中,所以从该地址能判断一个数据项是否存在于整个 Hash 表中,需要搜索的只是一个链表,而不必对整个表格遍历,所以节省了大量时间。

3.2　采用 Hash 表结构对搜索的改进

文中实验的硬件条件为:P4 3.0G 的 CPU,512M

内存,80G 并口硬盘。系统采用 Windows 2003 Server,算法采用的编程语言是 Matlab。通过进行 2 - 项集搜索测试,得到遍历时间与项目集数量的变化曲线,如图 1 所示。

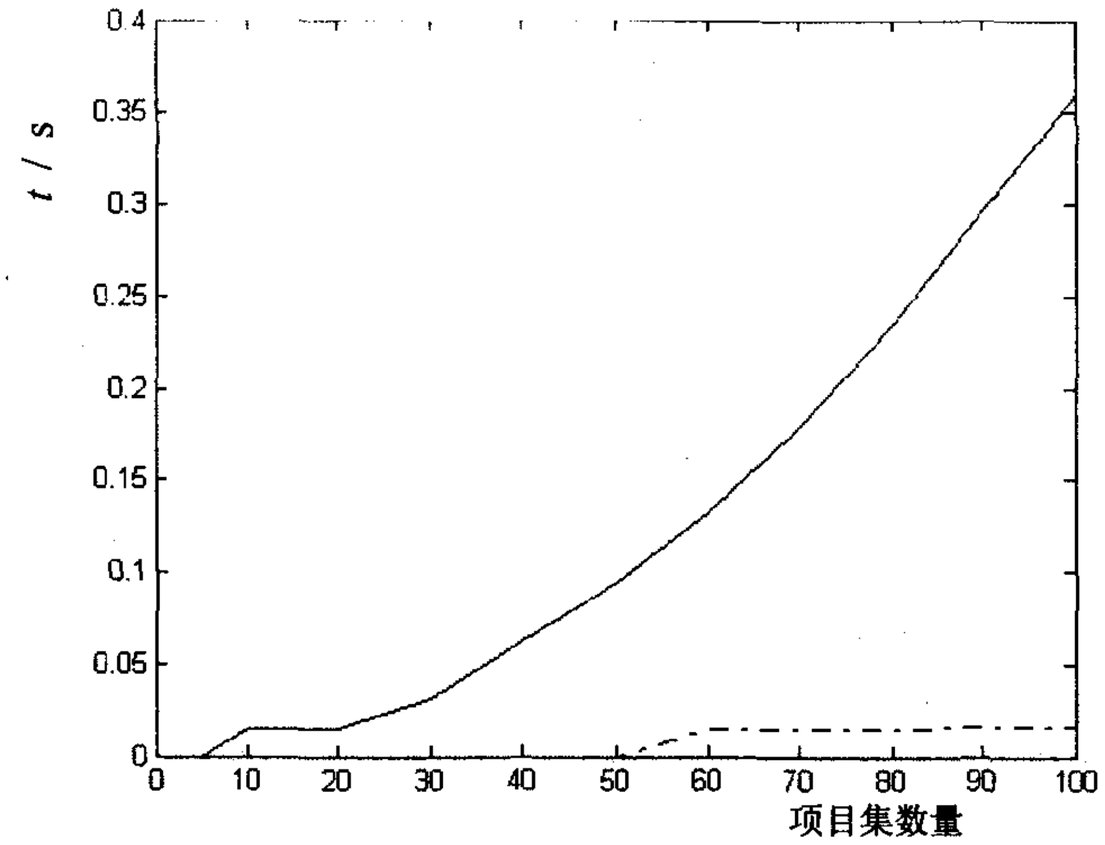

图 1 搜索时间随项目增多的变化

图 1 横坐标表示项目集数量的变化,纵坐标表示运行时间的变化。2 - 项集的数量从 5 开始变化到 100,采用存放在普通表格和文中提出的 Hash 表结构分别进行了两次实验。前者实验结果为实线,后者实验结果为点划线。

在对全部的 2 - 项集挖掘后,得到如图 2 所示的仿真结果,其中横坐标表示设定的最小支持度,纵坐标表示运行时间,单位为秒。本实验的测试环境是在 2 - 项集为 500 时作的大项目集挖掘,其中虚线为候选集的输入时间,实线为 Apriori 算法下进行的顺序遍历搜索时间,点划线为文中 DHP 算法下进行的 Hash 寻址搜索时间。

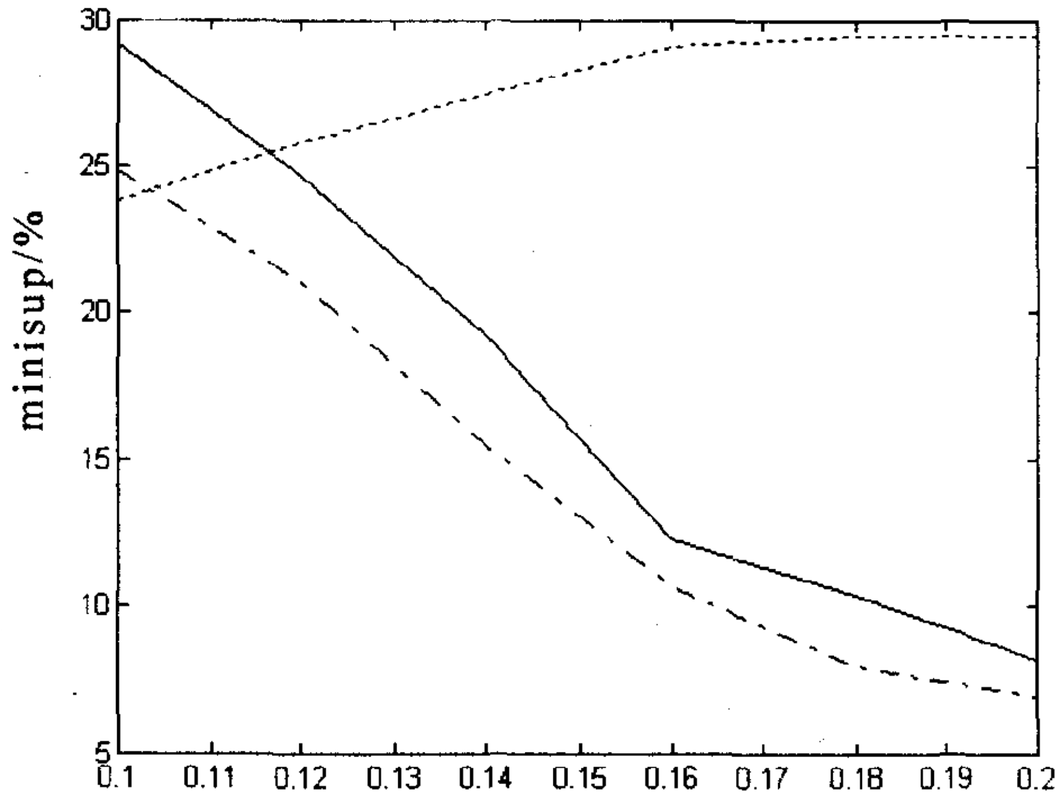

图 2 全部 2 - 项集的挖掘时间

从图 1 可以看到,项目集中事务数量较少时,增加事务导致时间开销的增大不明显。但是在项目集变得很大时,遍历时间开销的增大幅度将呈非线性的幂函数曲线变化,而采用 Hash 表存储结构后的搜索时间依然保持较低的平缓变化。

从图 2 可以看出,采用 Hash 表结构的 DHP 算法可以改进 Apriori 算法挖掘大项目集的效率,不论设定多少支持度,都能保证在更短的时间内完成全部项目集的挖掘。

4 结 论

文中对 DHP 算法中的 Hash 表进行了改进,并通过实验仿真证明了文中的 Hash 表结构能够有效地提高项目集的搜索效率,更快地确定项目是否属于频繁项。同时,采用 Hash 函数能够高效地利用存储空间,减少因为冲突项而出现的预留空间浪费。不同层的项目集将存放在不同表格中,可以进一步减少冲突项的出现。这些改进进一步提升了 DHP 算法的挖掘效率,更快地产生全部大项目集,为生成全部关联规则节省了大量的时间。

参考文献:

[1] Agrawal R, Imielinski T, Swami A. Database mining: a performance perspective[J]. Knowledge and Data Engineering, IEEE Transactions, 1993, 5(6): 914 - 925.

[2] Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases[J]. ACM SIGMOD Record, 1993, 22(2): 207 - 216.

[3] Park Jong Soo, Chen Ming - Syan, Yu P S. Using a Hash - Based Method with Transaction Trimming for Mining Association Rules [J]. Knowledge and Data Engineering, IEEE Transactions, 1997, 9(5): 813 - 825.

[4] Holt J D, Chung S M. Efficient mining of association rules in text databases[C]//Proceedings of the eighth international conference on Information and knowledge management. [s. l.]: ACM Press, 1999: 234 - 242.

[5] Han Jiawei, Kamber M. DATA MINING Concepts and Techniques[M]. 北京: 高等教育出版社, 2001.

[6] Dunham M H. 数据挖掘教程[M]. 郭崇慧, 田凤占, 靳晓明译. 北京: 清华大学出版社, 2005.

(上接第 11 页)

社, 1990.

[4] 何迎晖, 钱伟民. 随机过程简明教程[M]. 上海: 同济大学出版社, 2004.

[5] Castleman K R. Digital image processing[M]. 北京: 清华大学出版社, 2003.

[6] Sonka M, Hlavac V. 图像处理、分析与机器视觉[M]. 北京: 人民邮电出版社, 2002.