

基于 HASH 表的多谓词约束下频繁项集挖掘

张 婕¹, 张 燕^{1,2}, 李广水^{1,2}

(1 金陵科技学院 信息技术学院, 江苏 南京 211169; 2 江苏省信息分析工程实验室, 江苏 南京 211169)

摘 要: 针对在交易数据库中挖掘出指定顾客相关属性的频繁项集这一问题, 提出了基于维约束进行求解的构想, 采用模式增长的挖掘方法, 但传统的模式树不同的是将原先每一节点频繁计数值设为在所有可能的谓词约束下该项的计数形成的向量, 并利用 HASH 表进行向量值及项所在层的位置映射, 因此, 在不同的约束组合下的频繁项集挖掘将不再需要扫描数据库. 仿真实验表明该挖掘算法的完备性, 通过与先筛选再挖掘的算法进行比较, 证明该挖掘算法具有更高的效率.

关键词: 多谓词约束; 频繁项集; 模式增长; HASH 表

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2011)10-0056-04

Multi-predicate Constrained Frequent Itemsets Mining Based on Hash Table

ZHANG Jie¹, ZHANG Yan^{1,2}, LI Guang-shui^{1,2}

(1 School of IT, Jinling Institute of Technology, Nanjing 211169, China;

2 Jiangsu Information Analysis Engineering Laboratory, Nanjing 211169, China)

Abstract: Aiming at how to mine frequent itemsets from affair database after specifying customer characters, the conception of mining with multidimensional constrained is brought forward. The FP_Growth algorithm is employed, but the way for constructing FP_Tree is different in every item-node, it's not the count of the item occurring in the database, but using a vector that makes of counts of the item under every constructing, the node vector and level structure are saved in HASH table, it can support mining the frequent itemsets in facultative constructing without scanning the database again. The result of experiment proves this algorithm is self-contained and has higher efficiency by comparing to what mine after selecting.

Key words: Multi-predicate constraints; frequent itemsets; FP_Growth, hash table

1 引言

对多谓词约束下的频繁项集挖掘, 一个直接的方法是首先对交易数据库 B 进行约束筛选之后再挖掘频繁项集^[1-4], 但无论是采用基于 Apriori^[5-7]挖掘或基于 FP-Growth、图挖掘方法, 在约束更新的情况下, 都需要重头再进行一遍, 这显然不能保证高效的频繁项集挖掘要求.

本研究基于模式增长方式, 但传统的模式树不同的是将原先每一节点频繁计数值设计为谓词约

束向量, 并利用 HASH 表进行向量值映射, 以方便频繁项搜索, 我们称之为 HashFP-Growth 挖掘算法.

2 基于模式增长的频繁项集挖掘

模式增长(FP_growth)是由 Han J、Pei J 等^[8]在 2000 年首次提出的频繁项集挖掘算法, 其主要过程分为两步: FP-tree 构建和频繁项集挖掘. FP-tree 构建过程中, 首先扫描事务数据库 D 一遍, 获取所有 1-频繁项及其支持度, 并最终形成以支持度递减

收稿日期: 2011-04-28; 修回日期: 2011-06-30

基金项目: 金陵科技学院学生科技创新立项课题; 金陵科技学院博士启动资金(JIT-B-201101)

排序的列表 L,在对 D 进行第二次扫描过程中,将依据 L 中的顺序对每一事务中的频繁项构建 FP-tree: FP-tree 根结点设为 null,某一事务项中的所有频繁 1-项集,即列表 L 中有对应数据的项被添加到 FP-tree 中,具体过程是如果树中已存在一个对应项的节点,则该节点频繁值加 1,否则创建一个新的节点,并将其频繁值设为 1,该节点位置依据 L 中的顺序链接到其父节点,并且通过节点链结构将其链接到具有相同项名的节点。

对 FP-tree 挖掘采用了模式增长算法,从叶节点开始直至根形成 FP-tree 的一条路径,对路径中每一节点的组合计算其支持度,得到该路径下的所有频繁模式,遍历所有叶节点之后获得事务数据库 D 中的所有频繁模式。

相比于经典的 Apriori 方法,模式增长挖掘方法极大地减少了对海量数据的扫描次数,在频繁挖掘领域得到了广泛研究^[9-10]。

3 构建带谓词约束向量的模式树

为有效实现对多谓词约束的频繁项集挖掘,需将约束信息包含在所构建的数据结构中,在基于模式树的挖掘算法中,将原先每个节点频繁计数值设置为一个向量,向量中的各个数值分别代表该项满足对应约束的计数值,并约定其第一个值表示该项总的出现次数,即不考虑约束下的频繁总计数,在数据存储过程中,利用两级 HASH 表的方式保存节点信息,一级 HASH 表存储节点向量,二级 HASH 表存储各节点对应的 HASH 表。我们称此结构为 HashFP_Tree。

为清楚描述该方式,以下用一个交易数据库进行 HashFP_Tree 构建举例。

交易数据库 B 数据如图 1 所示。

则扫描一次 B 得到各 1 项的计数由 1 至 10 分别为 5,6,7,3,2,5,0,2,8,3,假设支持数设为 5,则有频繁 1-项集分别为 I1, I2, I3, I6, I9, 依其降序排序的 L 为 I9, I3, I2, I1, I6。

第二次扫描 B 之前,需要确立交易数据库所有可能的约束项,对于 B 而言,其约束项分别为男、女、工程师、教师、售货员,加上在没有约束下的频繁值,每一节点向量共有 6 个值,第一个值是无约束情况下该节点的频繁总计数,第二个值是满足性别为“男”时该节点的频繁值,第三个值是满足性别为“女”时该节点的频繁值,以此类推。其时,二次扫描 B 之后,建立的 HashFP_Tree 如图 2 所示。

id	sex	job	item
1	男	工程师	I1 I2 I3 I5 I8 I9 I10
2	男	教师	I1 I3 I6 I9
3	女	售货员	I2 I3 I9 I10
4	女	教师	I1 I3 I8 I9
5	男	工程师	I2 I5
6	女	工程师	I1 I3 I6 I9
7	男	售货员	I3 I4 I6 I9
8	男	售货员	I2 I4 I6 I9 I10
9	女	教师	I1 I2 I3 I4 I9
10	男	工程师	I2 I6

图 1 交易数据库 B

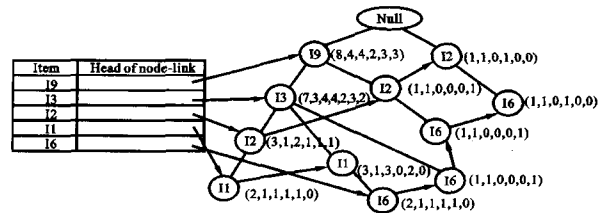


图 2 基于交易数据库 B 创建的 HashFP_Tree

在第二次扫描 B 过程中,先创建 HashFP_Tree 的根节点,以“NULL”标记它。对于 B 中每个事务 T,获得它相对应的约束条件 C,执行如下过程:

(1)设排序后的频繁项表为 $[p|P]$,其中 p 是第一个元素,而 P 是剩余元素的表;

(2)调用 insert_hashtree($[p|P]$, T, C). 该函数执行过程为:如果 T 有子女 N 使得 $N.item_name = p.item_name$,则 N 在满足 C 约束的各频繁计数以及频繁总计数各增加 1,否则创建一个新节点 N, 将其在满足 C 约束的各频繁计数以及频繁总计数设置为 1,链接到它的父节点 T,并且通过节点链结构将其链接到具有相同 item_name 的节点,同时创建 Hashtable 对象 $treeHash$,用来保存当前的结点信息。因为各节点在树中的出现不止一次,又 Hash-table 的 key 值必须是唯一的,因此必须对不同节点的同项进行标识,我们定义变量 i 作为节点名的后缀, i 的定义为:所在层 X_{i0} + 从左至右该层该项出现的次数。如果 P 非空,递归地调用该函数。insert_hashtree($[p|P]$, T, C)的具体实现如下:

```

{ If T 有子女 N 使得  $N.item\_name = p.item\_name$ 
{ 创建一个 Hashtable 对象  $treeHash$ ;
 $treeHash = (Hashtable)_treeHash[N.item\_name]$ ; 将
结点 N 的信息赋值给  $treeHash$ 
将  $treeHash$  中 C 包含的所有条件下的 value 值加 1;

```

_treeHash.Remove(N.item_name);将原来结点信息从_treeHash中移除

调用_treeHash.Add(N.item_name,_itemHash);

计算 i

Else {创建新结点 N,将其链接到父结点,并通过节点链结构将其链接到具有相同 item-name 的节点

创建一个 Hashtable 对象 _itemHash,并初始化, key 值为各约束属性名, value 值全部赋值为 0;

将 _itemHash 中 C 包含的所有条件下的 value 值赋值为 1;

调用 change(N.item_name,i);生成新的结点名 N.item_name? 作为 _treeHash 的 key 值

调用_treeHash.Add(N.item_name,_itemHash);将结点信息添加至 _treeHash

计算 i

If P 非空 递归调用 insert_hashtree([p|P],T,C);}

4 对 HashFP_Tree 进行有谓词约束频繁项集挖掘

对_HashFP_Tree 进行的主要过程挖掘类似于传统的 FP-Growth 方式,不同之处在于_HashFP_Tree 是基于条件的挖掘,模式基的产生首先纵向沿着头表 L 向上,优先遍历头表,在遍历头表的过程中同时横向遍历每个频繁 1-项集对应的链表域,得到节点在约束下对应的条件模式基,模式基的产生过程如下:

创建 Hashtable 类型的对象 _tempHash; 其作为中间变量,用来保存节点的信息

_tempHash=_treeHash[节点名+i.Tostring]; 其中 i 为如前定义的节点的位置

count=min(_tempHash[约束 x],_tempHash[约束 y],_tempHash[约束 Z]……)

则某节点在某一条树分支上的模式基可表示为:<节点的后缀路径;count>

在得到频繁 1-项集各个约束组合下对应的条件模式基之后,再以每个头表中的频繁 1-项集的条件模式基作为输入源来构造条件模式基的 FP-tree,即条件树.以对图 2 中的 I1 在约束为“男”、“工程师”组合情况下进行条件树的构造为例,得到如图 3 所示的条件树.

通过对条件树的挖掘,得到最终频繁项集,该过程与传统方法一致.

由以上挖掘过程可以看出,一次构造出 HashFP-Tree 之后,对给定的不同约束,总是对该树进行挖掘,不需要重新扫描交易数据库 B.

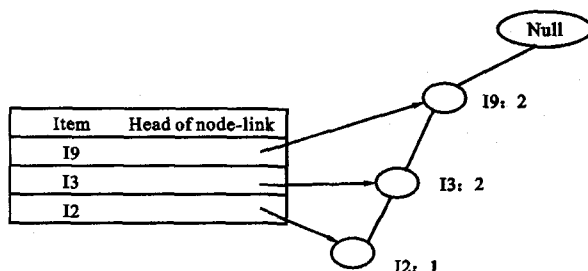


图 3 节点 I1 的条件树

5 仿真实验

5.1 实验设计

仿真实验利用 C# 进行代码编写.

设 $A=\{a_1,a_2,\dots,a_m\}$ 是描述交易者身份特征的相关属性,可以量化为不同的数值范围, $I=\{i_1,i_2,\dots,i_n\}$ 是不同的交易项集.我们在 SQL Server 中产生随机的交易数据库,如表 1 所示.具体过程如下:

表 1 交易数据库结构

Age	Income	Career	sex	Buy_items
Young	Low	Teacher	Male	I11, I20
Old	High	Manager	Male	I2, I7, I89, I90
...	...	...	...	...

(1)对于 A 中的每个约束以及 I ,我们分别定义对应的枚举对象 enum;

(2)在 for 循环中用 Randomde 对象 rd 调用方法 rd.Next(int a),在 $0\sim a$ 内产生随机数,即随机产生不同的约束值;

(3)将产生的数据整理后插入到数据库中.

我们预设年龄(Age)取值范围为:青年、中年、老年;收入(income)取值范围为:高、中、低;职业的取值范围为:教师、经理、工程师、农民;性别的取值范围为:男、女,而事务数据库中的交易项设为 I1~I100,共 100 项,生成实验记录 2 000 条.

哈希表(Hashtable)是 C# 中 System.Collections 命名空间提供的一个容器,用于处理和表现类似 key-value 的键值对,其中 key 通常可用来快速查找,value 用于存储对应于 key 的值.Hashtable 中 key-value 键值对均为 object 类型,所以 Hashtable 可以支持任何类型的数据,可以满足实验要求的字符型、数值型两种类型.

5.2 实验结果及分析

实验主要回答两个问题:(1)该算法的完备性;

(2)算法效率.

为回答第一个问题,我们预设支持度为 5%,首先给出年龄为 Young 的约束条件,得到 206 频繁项集,将每一条记录依 I1~I100 排序得到一个数据集 D1,再给出年龄为 Young 且收入为 Low 的约束条件,得到 83 个记录的数据集 D2,比较 D2 和 D1,发现 D2 的每一条记录是 D1 中某记录的子集.另一方面,我们首先利用 SELECT 查询语句对仿真数据集进行筛选得到目标数据集 B1,再对 B1 的交易项部分进行传统频繁项集挖掘,将挖掘结果与我们相同约束条件下的挖掘结果进行比较,得到完全一样的频繁项集.由此可以得到本算法具有完备性.

对算法效率的检验,我们采用与 C# 编写的传统 FP-Growth 挖掘算法进行比较,具体方法为:首先利用 SELECT 查询语句将整个目标数据集引入到内存表中(DataSet 对象中的一个表),针对某一指定约束组合,从 FP-Growth 对该内存表进行筛选进行计时直到挖掘结束为一个挖掘时间,由于采用本文的挖掘方法在首次构建 HashFP-Tree 过程中,需要耗费链接 SQL_server 时间,因此,没有对无约束情况下两种挖掘算法的挖掘比较,表 2 所示为不同约束组合下两种挖掘方法的耗时.

表 2 FP-Growth 与 HashFP-Growth 挖掘时间比较 ms

约束 算法	T1	T1 AND T2	T1 AND T2 AND T3	T1 AND T2 AND T3 AND T4
FP-Growth	360	330	280	230
HashFP-Growth	220	200	180	180

其中 T1 代表 Age="Young", T2 代表 Income="Low", T3 代表 Career="Teacher", T4 代表 Sex="Male"

由表 2 可以看出,HashFP-Growth 方法较先筛选再挖掘方法有一定的优势,特别在数据集较大或约束因子较少的情况下尤为明显.另一方面,在实际情况下,FP-Growth 方法针对不同的谓词约束总是需要去访问数据库,因此,效率将会更差.

6 结束语

针对多谓词属性约束下挖掘事务数据库中频繁项集问题,提出了基于模式树的 HashFP-Growth 挖掘算法,该算法一个主要的优点在于对不同的约束组合,只需构建一个模式树,因此,挖掘过程中不再需要扫描数据库.

当然,在有更多的属性以及某些属性拥有更多的属性值情况下,约束集合会变得很大,因而,

HashFP-Tree 对内存要求将会更高.

另一方面,本文针对的是不同属性值之间“AND”情况,在现实中,也存在“OR”情况,特别在同一属性之间,如希望挖掘出“Age”在“Young”或“Old”下的所有频繁项集,应对该问题的一个直接方法是预设所有可能的约束组合,但这会导致约束集合数量“爆炸式增加”,从而导致 HashFP-Tree 在常规内存中难以实现.

如何在多谓词属性约束下挖掘出最大频繁项集、闭项集以及在数据集更新的情况下对频繁项集的高效挖掘,所有这些,都将是我们将要研究的主题.

参考文献:

- [1] Micheline Kamber, Jiawei Han, Jenny Y Chiang. Meta-rule-guided mining of multi-dimensional association rules using data cubes[C]// Proceeding of the 3rd International Conference on Knowledge Discovery and Data Mining, Newport Beach, California, 1997.
- [2] 刘君强,潘云鹤. 挖掘空间关联规则的前缀树算法设计与实现[J]. 中国图象图形学报, 2003, 8(4): 476-480.
- [3] 钟勇,秦小麟,包磊. 一种基于多维集的关联模式挖掘算法[J]. 计算机研究与发展, 2006, 43(12): 2117-2123.
- [4] 刘波,潘久辉. 基于频繁模式图的多维关联规则挖掘算法研究[J]. 电子学报, 2007, 35(8): 1612-1617.
- [5] 周永生,熊结青,沙宗尧. 带确定性决策项的关联规则挖掘及其在生物化工生产中的应用[J]. 武汉大学学报: 信息科学版, 2010, 35(9): 1125-1128.
- [6] Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases [C]// Proceedings of the 1993 ACM-SIGMOD international conference on management of data (SIGMOD'93). Washington, DC, 1993: 207-228.
- [7] 彭硕,吴昊. 基于 MDPI 的多维关联规则算法的研究[J]. 微电子学与计算机, 2011, 28(1): 78-82.
- [8] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation[C]// Proceeding of the 2000 ACM-SIGMOD international conference on management of data (SIGMOD'00). Dallas, TX, 2000: 1-12.
- [9] 朱玉全,孙志挥,季小俊. 基于频繁模式树的关联规则增量式更新算法[J]. 计算机学报, 2003, 26(1): 91-96.
- [10] 吉根林,杨明,宋余庆,等. 最大频繁项目集的快速更新[J]. 计算机学报, 2005, 28(1): 127-135.

作者简介:

张燕女, (1969-), 博士研究生, 副教授.

李广水男, (1965-), 博士, 教授.