基于COFI-Tree的N-最有兴趣项目集挖掘算法

肖继海[1],崔晓红[1],陈俊杰[2]
(1. 太原理工大学,山西 晋中 030600;
2. 太原理工大学,山西 太原 030024)

摘 要:BOMO算法采用递归构造条件子树,在挖掘大数据集时耗时较长,执行效率低,为了解决这一不足,文中给出一种基于COFI-Tree的挖掘N-最有兴趣项目集算法。算法采用COFI-Tree结构,无需递归构造条件子树FP-Tree,在同一时间内只有一个COFI-Tree在内存,并且有效地减少了其运算时间。通过对两种算法进行对比分析,实验结果得出:该算法比BOMO算法程序执行时间明显缩短;在挖掘大数据集时执行效率显著提高,尤其是k<4时,性能最好。由此可见,改进后的算法是可行有效的。

关键词:数据挖掘;关联规则;N-最有兴趣项目集;FP-Tree;COFI-Tree
中图分类号:TP311.13 **文献标识码**:A **文章编号**:1673-629X(2012)03-0099-04

Mining N-Most Interesting Itemsets Algorithm Based on COFI-Tree

XIAO Ji-hai[1],CUI Xiao-hong[1],CHEN Jun-jie[2]
(1. Taiyuan University of Technology,Jinzhong 030600,China;
2. Taiyuan University of Technology,Taiyuan 030024,China)

Abstract:*BOMO algorithm constructs conditional FP-Tree recursively so that it requires more memory and CPU resources. To solve this problem,an algorithm for mining N-most interesting itemsets based on COFI-Tree is presented. This algorithm adopts COFI-Tree. COFI-Tree doesn't need to construct conditional FP-Tree recursively and there is only one COFI-Tree in memory at a time. Experiment shows that the algorithm based on COFI-Tree performs faster than current best algorithm BOMO;The algorithm has good performance for large data set,especially it shows the best when for k value is smaller than 4. This shows that the improved algorithm is feasible and effective.*

Key words:data mining;association rules;N-most interesting itemsets;FP-Tree;COFI-Tree

0 引 言

1994年,文献[1]最早提出了挖掘频繁项目集的算法Apriori,它是利用给定的阈值,来找频繁度大于给定阈值的项目集,但是要生成大量的候选项目集。2000年,文献[2]中提出了采用FP-Tree结构的FP-Growth算法来挖掘频繁项目集,该算法无需生成候选项目集。2003年,文献[3,4]中提出了与FP-Tree相似的COFI-Tree结构,解决了文献[2]中递归构造"条件FP-Tree"的问题,并且证明COFI-Tree的性能比FP-Tree好[5]。2004年,文献[6,7]中提出了基于FP-Tree的BOMO算法来挖掘N-最有兴趣项目集,该算法需要递归地构造"条件FP-Tree",增加了CPU和内存的开销。迄今为止已提出了许多高效的关联规则挖掘算法,其中以Agawal提出的Apriori算法最为著名,大多数挖掘算法都是建立在Apriori算法基础之上,包括IUA[8]、PIUA算法[9]、IUAR算法[10]和IIUA[11]算法等等。但是Apriori算法无论在时间效率还是空间伸缩性上都面临着挑战,因此研究人员探索出很多新的挖掘方法[12]。

文中介绍的算法是在文献[6]的基础上进行改进,采用COFI-Tree结构,无需递归构造"条件FP-Tree",大大提高了算法的效率。

定义1 N-最有兴趣的K-项目集[13]。

将K-项目集按支持度降序排列,其中设S为第N

收稿日期:2011-05-25;**修回日期**:2011-08-27
基金项目:山西省自然科学基金资助项目(2007011050)
作者简介:肖继海(1978-),男,山西朔州人,讲师,硕士,研究方向为数据库、网络多媒体;陈俊杰,教授,博士生导师,研究方向为数据库、网络多媒体。

个 K - 项目集的支持度，N - 最有兴趣的 K - 项目集是指支持度大于等于 S 的 K - 项目集的集合。

定义 2 N - 最有兴趣项目集是指 N - 最有兴趣的 K - 项目集($1 \leqslant K \leqslant K_{max}$) 的并集。其中 K_{max} 是项目集的上限。

为了便于说明，设有交易数据库如表 1 所示。

表 1 交易数据库

T_1	T_2	T_3	T_4	T_5	T_6
ABCD	BCDE	ABDE	ACEF	AB	AC
T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
AC	BEF	AF	CF	ABD	BDE
T_{13}	T_{14}	T_{15}	T_{16}	T_{17}	T_{18}
CD	CF	BDEF	ABDE	ACEF	ABCD

1 基于 COFI-Tree 的 N-最有兴趣项目集挖掘算法

1.1 构建 FP-Tree

生成 FP-Tree 的算法步骤在文献[5]中有详细的介绍，分两步完成。其中，频繁项头的每个表项由三个域组成：item-name，support，node-link。node-link 是指针域，node-link 指向下一个具有同样的 item-name 域的节点的第一个节点。

树中的每个节点由 3 个域组成：item-name，count，node-link。item-name 储存该节点的数据项名称。count 记录所在路径代表的交易中达到此节点的交易个数。node-link 指向下一个与 item-name 域值相同的节点，若无这样的节点，则置为 null。

1.2 构造 COFI-Tree

生成 COFI-Tree 的算法步骤在文献[5]中有详细的介绍，同样也是分两步完成。其中，COFI-Tree 与 FP-Tree 的结构相似，但是，在 COFI-Tree 中每个节点不仅包含 item-name，count，而且增加了参与量 participation counter 域。

1.3 挖掘 COFI-Tree 得到 N-最有兴趣项目集

挖掘 COFI-Tree 的算法：

将某个 COFI-Tree 的频繁项头表中的节点按照频繁度降序排列。

第 1 步：取频繁项头表中频繁度最大的节点，设为 A。

第 2 步：

(1) 在这个 COFI-Tree 中从 A 到根节点，将经过的节点存放在 D 中，其中，各节点的频繁度 F 等于节点 A 的频繁度与参与量之差。

(2) 将 D 中产生的所有候选式保存在 X 中，其中将不包含 A 的模式删掉。

(3) 逐个检查 X 中的每一个模式在 A - 候选列表中是否存在，若不存在，则将这个模式插入 A - 候选列表中，并将其频繁度设为 F，否则，频繁度仅增加 F。

(4) 将 D 中所有项目的参与量都增加 F。

第 3 步：循环第 1，2 步，将这个 COFI - Tree 中的所有节点都生成相应的模式插入 A - 候选列表中。

第 4 步：根据 ξ，将 A - 候选列表中的非频繁模式删掉。

第 5 步：对于 A - 候选列表中的每一个模式 S：

(1) 将 S 中的节点个数保存在 L 中；

(2) 若 $L < K_{max}$，① 若 S 的频繁度 $\geqslant \xi_L$，则将 S 插入 R_L，修改 ξ_L，修改 $\xi(\xi = \min(\xi_1, \xi_2, \xi_3, \cdots, \xi_L))$。② 若 $\xi_L \neq 0$ 并且 R_L 中的 X 的支持度小于 S 的频繁度，则将 X 从 R_L 中删掉。

2 算法举例

接下来，为了加深理解，以表 1 所示的交易数据库来挖掘 2-最有兴趣的 2-项目集。根据算法有：$N = 2$，$K_{max} = 2$。起初，设结果集为空，阈值为 0，即：$R_1 = \{\}$，$R_2 = \{\}$，$\xi_1 = 0$，$\xi_2 = 0$，$\xi = \min(\xi_1, \xi_2)$。

2.1 生成 FP-Tree

按照生成 FP-Tree 的算法，表 1 数据库所构成的 FP-Tree 如图 1 所示。

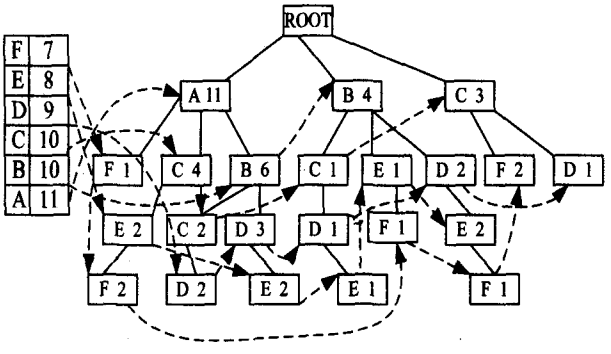

图 1 FP-Tree

2.2 生成并且挖掘 COFI-Tree 得到 2-最有兴趣的 2-项目集

所有项目的 COFI-Tree 不是同时生成的，而是一个一个生成，当一个项目的 COFI-Tree 删掉后，才构造另一个项目的 COFI-Tree。所以，该算法将按照频繁度的降序来一个一个构造 COFI-Tree。

对于频繁度最大的 A，无需构造它的 COFI-Tree，只需修改结果集 R_1 和阈值 ξ 就可以了。所以，$R_1 = \{ < A:11 > \}$，$R_2 = \{\}$，$\xi_1 = 0$，$\xi_2 = 0$，$\xi = \min(\xi_1, \xi_2) = 0$。

2.2.1 B COFI - Tree

按照 2.2 所示的算法，构造的 B COFI - Tree 如图 2 所示。在图 2 中，B 为频繁度最大的节点，所以首先找到 $(B,10)$，其次找到 $(AB,6)$，最后修改结果集 R_1 和

阈值 ξ:R_1 = { < A:11 >, < B:10 > }, R_2 = { < AB:6 > }, ξ_1 = 10, ξ_2 = 6, ξ = min(ξ_1, ξ_2) = 6。

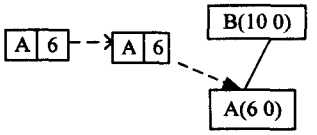

图2　B COFI - Tree

2.2.2　C COFI - Tree

构造的 C COFI - Tree 如图3所示。在图3中,C 为频繁度最大的节点,所以找到(C,10),接着 A 为频繁度最大的节点,找到(AC,6),(ABC,2),最后 B 为频繁度最大的节点,找到(BC,3),由于(ABC,2) 的长度为3超过了 K_{max} = 2,所以删除(ABC,2),又由于(BC,3) 的频繁度为3小于6,所以删掉(BC,3)。最后修改结果集 R_1 和阈值 ξ:R_1 = { < A:11 >, < B:10 >, < C:10 > }, R_2 = { < AB:6 >, < AC:6 > }, ξ_1 = 10, ξ_2 = 6, ξ = min(ξ_1, ξ_2) = 6。

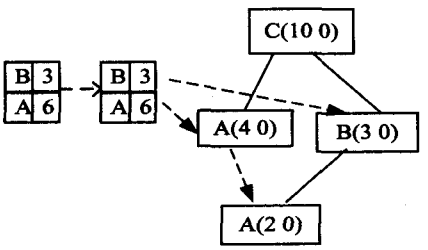

图3　C COFI - Tree

2.2.3　D COFI - Tree

构造的 D COFI - Tree 如图4所示。在图4中,D 为频繁度最大的节点,所以首先找到(D,9),其次找到(BD,8),由于(D,9) 的频繁度小于10,所以删掉(D,9)。最后修改结果集 R_1 和阈值 ξ:R_1 = { < A:11 >, < B:10 >, < C:10 > }, R_2 = { < BD:8 >, < AB:6 >, < AC:6 > }, ξ_1 = 10, ξ_2 = 6, ξ = min(ξ_1, ξ_2) = 6。

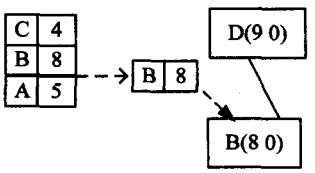

图4　D COFI - Tree

2.2.4　E COFI - Tree

构造的 E COFI - Tree 如图5所示。在图5中,E 为频繁度最大的节点,所以首先找到(E,8),其次找到(BE,6),由于(E,8) 的频繁度小于10,所以删掉(E,8)。最后修改结果集 R_1 和阈值 ξ:R_1 = { < A:11 >, < B:10 >, < C:10 > }, R_2 = { < BD:8 >, < AB:6 >, < AC:6 >, < BE:6 > }, ξ_1 = 10, ξ_2 = 6, ξ = min(ξ_1, ξ_2) = 6。

2.2.5　F COFI - Tree

构造的 F COFI - Tree 如图6所示。在图6中,仅有 F 节点,所以得到(F,7),由于(F,7) 的频繁度小于10,所以删掉(F,7)。最后修改结果集 R_1 和阈值 ξ:R_1 = { < A:11 >, < B:10 >, < C:10 > }, R_2 = { < BD:8 >, < AB:6 >, < AC:6 >, < BE:6 > }, ξ_1 = 10, ξ_2 = 6, ξ = min(ξ_1, ξ_2) = 6。

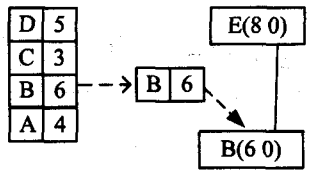

图5　E COFI - Tree

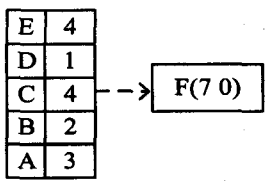

图6　F COFI - Tree

到此,就得到 R_2 = { < BD:8 >, < AB:6 >, < AC:6 >, < BE:6 > } 为表1所示的交易数据库的2 - 最有兴趣的2 - 项目集。

3　实验论证

3.1　实验环境

(1)数据:使用IBM的合成数据集生成器生成。数据集标记为:Dx. Ty. Mz。其中,D 表示事务的个数,T 表示每个事务中项的平均个数,M 表示不同项的个数。

(2)软硬件环境:VC++6.0,CPU 2.8GHz,内存4.0G,硬盘400G。

3.2　算法比较与分析

将该算法与目前最好的算法 BOMO 进行比较。BOMO 的源代码可以在网上免费获得,网址:http://appsrv.cse.cuhk.edu.hk/~kdd/program.html。图7表明,该算法在挖掘大数据集时性能最好,执行速度比 BOMO 快。这是因为:基于 COFI-Tree 的挖掘算法不需递归地构建条件子树,并且在某一时间段内只有一个项的 COFI-Tree 在内存。

3.3　算法性能的测试

图7表明,对于固定的 k 值,挖掘所需要的总时间随 N 的增大线性增加。图8表明,对于 k < 4 时,挖掘所需要的总时间随 N 的增大而线性增加,但是当 k 增大时,尤其当 k > 6 时,挖掘所需要的总时间急速增加。

4　结束语

文中研究了 N - 最有兴趣项目集挖掘算法,根据 N - 最有兴趣项目集的定义,提出基于 COFI - Tree 挖

掘 N - 最有兴趣项目集。通过对两种算法进行对比，分析实验结果得出：该算法在挖掘大数据集时性能比 BOMO 好，尤其是 $k < 4$ 时，性能最好。

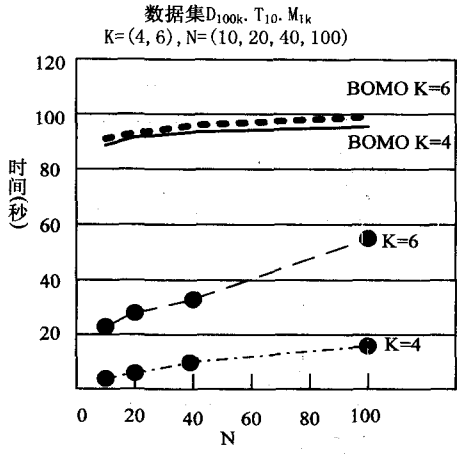

图 7 算法比较

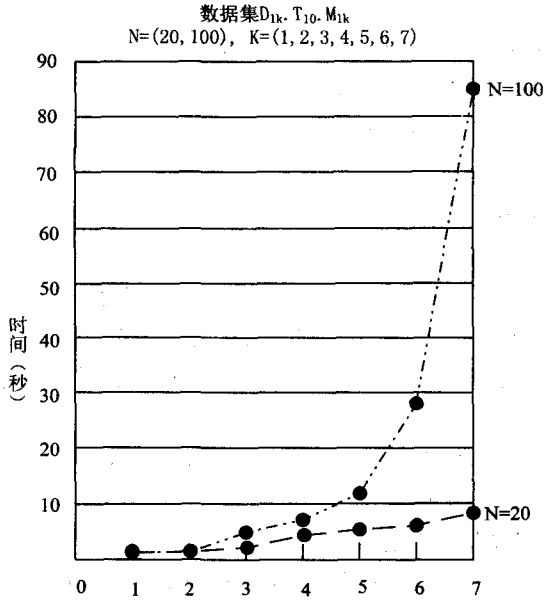

图 8 实验结果

参考文献：

[1] Agrawal R, Srikant R. Fast algorithms for mining association rules[C] // Proceedings of the 20th VLDB Conference. [s. l.]:[s. n.],1994:487-499.

[2] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation[C]//2000 ACM SIGMOD Intl Conference onManagement of Data. [s. l.]:[s. n.],2000:1-12.

[3] El-Hajj M, Zaïane O R. COFI-tree mining:a new approach to pattern growth with reduced candidacy generation[C]// Workshop on Frequent Itemset Mining Implementations (FIMI'03) in Conjunction with IEEE-ICDM. [s. l.]:[s. n.], 2003.

[4] El-Hajj M, Zaiane O R. Non recursive generation of frequent k-itemsets from frequent pattern tree representations[C]// Proceeding of 5th International Conference on Data Warehousing and Knowledge Discovery (DaWak'2003). [s. l.]:[s. n.],2003.

[5] 陈俊杰，崔晓红. 基于 FP-Tree 的频繁闭合项目集挖掘算法的研究[J]. 计算机工程与应用，2006，42(34)：169-171.

[6] Cheung Y L, Fu A W. An FP-tree Approach for Mining N-most Interesting Itemsets[C]//Proceedings of the SPIE Conference on Data Mining. [s. l.]:[s. n.],2002.

[7] Cheung Y L, Fu A W. Mining association rules without support threshold:with and without item constraints[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(9): 1052-1069.

[8] 陈 煜，徐维祥. 基于逆向搜索的关联规则更新算法[J]. 计算机工程，2011，37(8)：25-27.

[9] 丁祥武. 挖掘时态关联规则[J]. 武汉交通科技大学学报，1999，23(4)：365-367.

[10] 高 峰，谢剑英. 发现关联规则的增量式更新算法[J]. 计算机工程，2000，26(12)：49-50.

[11] 钟晓桢. 基于 Apriori 和 IUA 的改进算法[J]. 江汉大学学报，2007，35(3)：59-63.

[12] 王爱平，王占凤，陶嗣干，等. 数据挖掘中常用关联规则挖掘算法[J]. 计算机技术与发展，2010，20(4)：105-108.

[13] Fu A W C, Kwong R W W, Tang J. Mining N-most Interesting Itemsets[C]//Proceedings of International Symposium on Methodologies for Intelligent Systems (ISMIS). [s. l.]:[s. n.],2000.

(上接第 98 页)

[6] Shen Yi, Bu Yunfeng, Yuan Mingxin. Study on Weigh-in-Motion System Based on Chaos Immune Algorithm and RBF Network[J]. Computational Intelligence and Industrial Application, 2008(2):502-506.

[7] Yang Z R. A novel radial basis function neural network for discriminant analysis[J]. IEEE Transactions on Neural Networks, 2006, 17(3):604-612.

[8] 黎志刚，蔡 萍，周志峰. 基于 BP 网络的汽车动态称重数据处理方法[J]. 微计算机信息，2006(23)：251-253.

[9] 袁明新，张 勇，张 雨. 基于 RBF 网络的动态称重系统设计[J]. 交通与计算机，2003，21(2)：60-63.

[10] 张义超，卢 英，李 炜. RBF 网络隐含层节点的优化[J]. 计算机技术与发展，2009，19(1)：103-105.

[11] 姚恩涛，张 君，倪国芬，等. 基于图像特征分类和 RBF 网络的两轴车辆动态称重技术[J]. 南京航空航天大学学报，2007，39(1)：99-102.

[12] 储岳中. 改进的 RBF 神经网络在非线性系统中的应用[J]. 计算机技术与发展，2008，18(3)：196-199.