一种基于 MFP 树的快速关联规则挖掘算法

李志云,周国祥

(合肥工业大学 计算机与信息学院,安徽 合肥 230009)

摘 要:在关联规则挖掘 FP_Growth 算法的基础上,提出一种基于 MFP 树的快速关联规则挖掘算法。文中给出了 MFP 算法的工作原理。MFP 算法能在一次扫描事务数据库的过程中,把该数据库转换成 MFP 树,然后对 MFP 树进行关联规则挖掘。MFP 算法比 FP_Growth 算法减少一次对事务数据的扫描,因此具有较高的时间效率。

关键词:关联规则挖掘;MFP 树;MFP 算法

中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2007)06-0094-03

A Fast Association Rule Mining Algorithm Based on MFP Tree

LI Zhi-yun, ZHOU Guo-xiang

(School of Computer & Information, Hefei University of Technology, Hefei 230009, China)

Abstract: Based on FP_Growth algorithm of association rule mining, this paper presents a new association rule mining algorithm called MFP Tree. The MFP algorithm can convert a transaction database into an MFP tree through scanning the database only once, and then do the mining of the tree. Bacause the MFP algorithm scans a transaction database one time less the FP_growth algorithm, the MFP algorithm is more efficient with time.

Key words: association rule mining; MFP tree; MFP algorithm

1 概 述

关联规则的挖掘是数据挖掘研究的重要内容之一,它反映了大量数据中项目集之间有趣的关联或相关联系[1]。关联规则挖掘中最重要的是进行频繁模式的挖掘[2]。挖掘频繁模式的经典算法是 Apriori 算法和 FP_Growth 算法[3]。

Apriori 算法使用广泛,大家比较熟悉。FP_Growth 算法是用于从事务数据库中挖掘布尔型关联规则的频繁模式。它的挖掘过程比较复杂,但是可以简单地划分成三个基本步骤[4]:首先是扫描事务数据库,根据给出的 min_sup(最小支持度)建立 L 表;然后第二次扫描事务数据库,依据 L 表,构建 Fp-tree;最后对构建的 Fp-tree 进行挖掘,找出所有的频繁模式。有了频繁模式就可以根据行业背景方便地建立所需的关联规则[5]。

尽管 FP_Growth 算法的关联规则挖掘效率比 Apriori 算法高,但是它仍然需要扫描二次事务数据库,得到 L 表;第二次扫描事务数据库,构造出 Fp-tree。由于扫描实际的事务数据库的开销很大,若能在此基础上再减少挖掘算法对事务数据库的扫描次数,则能进一步有效地提高关联规则挖掘效率。为此,设计了一种称之为 MFP 的快速关联规则挖掘算法。MFP 算法有 2 个基本步骤:一是扫描事务数据库,在扫描过程中就把事务数据库转换成类似于 Fp-tree 的树(下面称为 MFP 树),并且保留了所有事务数据库中 item 间的关联信息;二是挖掘 MFP 树,从中找出所有可能的关联规则。与 FP_Growth 算法相比,MFP 算法只需对事务数据库扫描 1 次,因而可提高关联规则挖掘的时间效率[6]。

2 MFP 算法的工作原理

2.1 相关定义

定义 1:事务数据库 D。用于存储交易记录的数据库,数据库中的每一个交易记录都有惟一的标识。一个交易记录包括了一笔交易涉及到的所有 item 的有序排列。表 1 为一简化的事务数据库 D[7]。其中 T001 为事务数据库的首记录标号,T001 相对应的交易包括了 I1,I2 和 I5,记录中的 item 按标号的顺序排列。

收稿日期:2006-08-16

作者简介:李志云(1969-),女,山东昌邑人,硕士研究生,研究方向为计算机软件与理论;周国祥,教授,研究方向为计算机软件与理论。

表 1　事务数据库

TID	List of item_ID's
T001	I1,I2,I5
T002	I2,I4
T003	I2,I3
T004	I1,I2,I4
T005	I1,I3
T006	I2,I3
T007	I1,I2
T008	I1,I2,I3,I5
T009	I1,I2,I3

定义 2:MFP 树由一个标号为 NULL 的根结点和数个树结点构成,每一个结点可带有 n 个树结点(n = 0,1,2…),当 n = 0 时,称该结点为叶结点。由于 MFP 树用于表示一事务数据库,树的结点用数据库中的 item 的标号表示。除根结点外,每一个结点由 Ii.count 和 Ii.pointer 2 个域构成。其中 Ii.count 为该结点上出现的相同 item 个数,标示在结点标号 Ii 右侧一括号中;Ii.pointer 为指向其父结点的指针。

定义 3:路径结点组合 C。它是 MFP 树的一叶结点,由构成 C 的结点标识,在标识符右侧用括号中的一数值表示该组合的记数值,该记数值等于叶结点的 Ii.count 值。设一 MFP 树,如图 1 所示[6]。

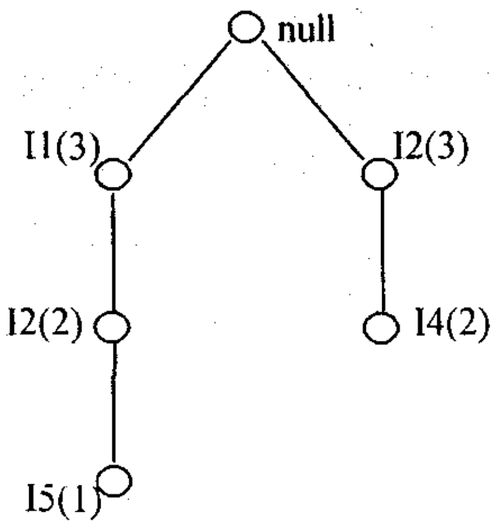

图 1　MFP 树及路径组合

I4,I5 是叶结点,分别形成二条到根结点的路径。由于 I5.count=1,I5 路径上的结点为 I1,I2,I5,因此 3 个结点形成 4 个组合,4 个组合分别是:{I5,I2}(1),{I5,I1}(1),{I5,I2,I1}(1),{I2,I1}(1);I4 路径上的结点为 I2、I4,I4.Count=2,则 2 个结点形成的组合是:{I4,I2}(2)。

定义 4:表 TL 是一指针队列,表中每一个元素均指向一个 MFP 树的叶结点,指针与叶结点一一对应。

定义 5:候选频繁模式集 CF,集合元素为 C。

2.2　算法的工作原理

MFP 算法分为构造 MFP 树和挖掘 MFP 树两部分。

(1)构造 MFP 树。

首先创建 MFP 树的根结点 NULL,然后取事务数据库 D 的第一条记录,并将其插入到 MFP 树中,插入过程为:取出记录中的第一个 item,若 MFP 树的根结点存在与该 item 标号相同的子结点,则在该子结点的记数值(count 域)上加 1,否则在根结点下创建一新的子结点,以该 item 的标号作为新创建子结点的标识;然后以被插入的结点为子树的根,将记录中的下一个 item 按上述步骤插入到子树中;如此重复,使得记录中的所有 item 都被插入到 MFP 树中,完成一条记录插入过程。接着,再从事务数据库 D 中取下一条记录,重复上述记录插入过程,直到数据库中的记录全部被插入到 MFP 树中。由此,经过一次扫描把事务数据库 D 转换成 MFP 树,从树叶结点到根结点的一条路径对应了数据库的一条或多条记录,并且保留了原来记录中 item 之间的关联信息。为便于后续的关联规则的挖掘,在构建树的过程中,使 TL 表的指针指向 MFP 树的叶结点,指针与叶结点一一对应。

按上述过程,可把表 1 所示的事务数据库 D 转换成图 2 所示的 MFP 树,结点标号 Ii 右侧括号中的数值为该结点的 Ii.count 值[6]。

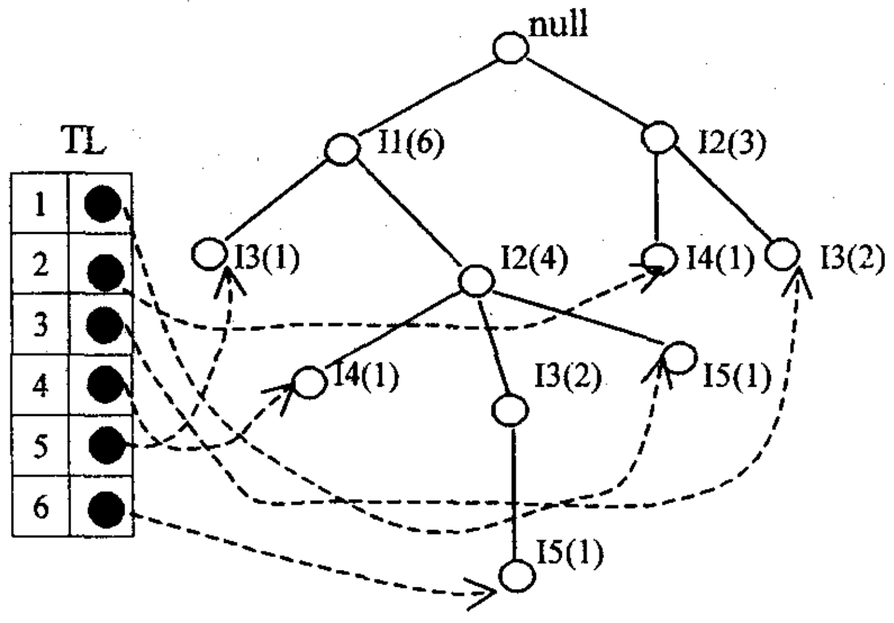

图 2　事务数据库 D 转换成的 MFP 树

(2)挖掘 MFP 树。

取 TL 表中的首元素,得到该元素指向的叶结点和从叶结点到根 NULL 的路径。对该路径上所有结点进行组合(根结点 NULL 除外),按定义 3 设置各组合的标识和记数值。然后将所有的组合送入候选频繁模式集 CF,若 CF 中已经存在相同的组合(即组合的标识相同),则进行合并。合并操作为:组合标识保持不变,记数值为二者之和。该路径上所有的组合进入候选频繁模式集 CF 后,对该路径上的结点进行修正。修正过程为:使该路径上的所有结点的 Count 值减去叶结点的 Count 值;减去后,若该路径上的叶结点存在不带其他子结点的父结点且父结点不是根 NULL,则将指向该叶结点的指针转向其父结点,以它为叶结点,对新的路径进行同样的操作。否则,删除 TL 表中的这一指针。

此后,再取表 TL 的下一元素,重复上述过程,直到表中的元素都被取出,完成所有 MFP 树叶结点的处理。

最后,用给出的 min_sup 值(最小支持度)剔除 CF 中记数值小于 min_sup 的组合。这样,留在 CF 中的就是所要找的频繁模式,依次可构造出所要的候选关联规则,并可用给出的 min_conf(最小置信度)筛选出所需要的关联规则。

下面是对已构造的 MFP 树(图 2)进行挖掘的过程。

由 TL 表中的首元素指针找到叶结点 I5,从而得到路径结点 I5,I2,I1。对该 3 结点进行组合得{I5,I2}(1),{I5,I1}(1),{I5,I2,I1}(1),{I2,I1}(1)。此时 CF 为空,因而四组合直接进入 CF。然后对该路径上的结点进行修正,将 I5,I2,I1 的 Count 值减 1。由于 I5 的父结点 I2 带有其他子结点,因此删除 TL 表中的这一指针元素。TL 表中最后的指针指向叶结点 I5,从而得到路径结点 I5,I3,I2,I1 和该 4 结点的各个组合,把组合送入 CF,并修正该路径上的结点。此时由于 I5 的父结点 I3 无其他的子结点,所以不删除 TL 表中该指针,只是将指向 I5 的指针改为指向其父结点 I3。然后再对以 I3 为叶结点的路径(由结点 I3,I2,I1 构成)进行同样的操作。最后,从 CF 中剔除记数值小于 min_sup(本例中 min_sup=2)的组合。对图 2 所示的 MFP 树挖掘后得到的 CF 如下所示:

CF={{I3,I1}(3),{I2,I1}(4),{I3,I2}(4),{I5,I2}(2),{I5,I1}(2),{I4,I2}(2),{I5,I2,I1}(2),{I3,I2,I1}(2)}

3 MFP 算法

3.1 算 法

MFP 算法的伪代码如下所示:

```
输入:事务数据库 D,min_sup
输出:频繁模式集 CF
方法:
    (1)构建 MFP 树。
创建根结点 NULL;
打开 D;
While(not eof)
{R=当前记录;
call insert_tree(R,NULL);
记录指针下移 1 位;}
Procedure insert_tree(record,root_nod)
  {while (record<>Φ)
  {first=record 中首 Item;
  从 record 中删除该 Item;
  if (root_nod 中存在标号为 first 的子结点 a)
    a.count=a.count+1;
  else
    {创建标号为 first 的结点 a;
    a.count=1;
    a.pointer 指向 root_nod;}
if(record=Φ)
{删除 TL 表中指向该路径上其他结点的指针;在 TL 表中创建指向结点 a 的指针;}
```

(2)挖掘 MFP 树(tree,TL)。

```
While(TL<>Φ)
  {得到 TL 首元素指向的叶结点 a;
  组合 a 到 NULL 路径上的结点形成 C1,C2,…Cm;
    For (I=1;I<=m;I++)
      {Ci.count=a.count;
      Ci 标号设为构成该组合的结点的有序排列;
        If (Ci∈CF)
      CF.Ci.count=CF.Ci.count+Ci.count;
      Else
      Ci 送入 CF;}
  该路径上其他结点的 count 值减去叶结点的 count 值;
  If (a 的父结点不存在其他子结点且不是根结点)
    TL 中指向 a 结点的指针指向 a 的父结点;
  Else
    删除 TL 中指向 a 结点的指针;}
  删除 CF 中 count 值<min_sup 的组合;
```

3.2 算法的实验运行结果

为了测试 MFP 算法的实际运行效果,在同样的软硬件环境下,用 MFP 算法和 FP_Growth 算法分别对一实验商业事务数据库进行布尔关联规则挖掘测试。反复测试表明:两种算法挖掘出来的候选频繁集相同,而 MFP 算法的挖掘速度约提高 20%。由 MFP 算法的工作原理可知:事务数据库增大时,MFP 算法的时间效率将进一步增加。

当事务数据库很大,以至于难以在给定的计算机内存中构建一个与整个数据库对应的 MFP 树时,可将事务数据库按一定条件横向分割成几个子数据库,然后逐一在内存中构造对应的 MFP 树。将各自 MFP 树的路径结点组合送入一个总的 FC。最后再对 FC 进行处理,得到所有的频繁模式。因此,MFP 算法特别适用于对大型事务数据库的关联规则挖掘。

4 结束语

关联规则的挖掘具有重要的实用价值,在数据挖掘领域应用广泛[8]。由于被挖掘的事务数据库规模较大,关联规则挖掘算法的运行效率就显得特别重要。

(下转第 100 页)

表 1 的第 6 个对象矛盾,文献[13]中 LEM2 算法也可得到 7 条规则;但以上两种算法得到规则集的数目均比表 2 所示规则集多。对于不一致决策表而言,由算法 1 得到的决策树,不一致对象对应的决策属性值有两个,且简化后得到的不确定性决策规则的可信度均小于 1。

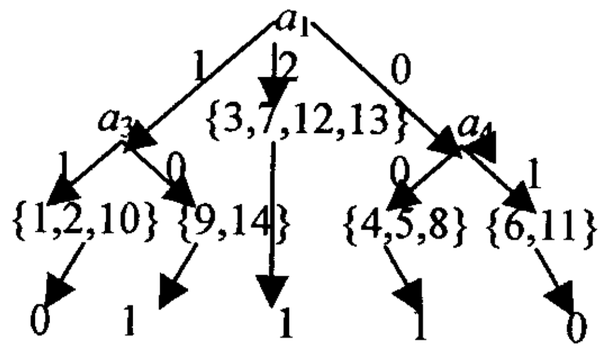

图 1 决策树

表 2 最小确定性决策规则集

序号	决策规则集	可信度	覆盖度
1	$(a_1,1) \wedge (a_3,1) \rightarrow (d,0)$	1	3/14
2	$(a_1,1) \wedge (a_3,0) \rightarrow (d,1)$	1	2/14
3	$(a_1,2) \rightarrow (d,1)$	1	4/14
4	$(a_1,0) \wedge (a_4,0) \rightarrow (d,1)$	1	3/14
5	$(a_1,0) \wedge (a_4,1) \rightarrow (d,0)$	1	2/14

4 结束语

在决策表中,以知识决策熵的属性重要性为启发信息,自顶向下递归构造决策树,然后遍历决策树,并简化所获得的决策规则。实例分析的结果表明,该算法为从决策表中搜索最小决策规则提供了一种有效的方法,并且该研究可进一步扩展粗糙集理论的应用领域。

参考文献:

[1] Wu X D, Urpani D. Induction by attribute elimination[J]. IEEE Transaction on Knowledge and Data Engineering, 1999,11(5):805-812.

[2] 林嘉宜,彭 宏,郑启伦.一种新的基于粗糙集的值约简算法[J].计算机工程,2003,29(4):70-71.

[3] 常犁云,王国胤,吴 渝.一种基于 Rough Set 理论的属性约简及规则提取方法[J].软件学报,1999,10(11):1206-1211.

[4] 黄 兵,周献中.不一致决策表中规则提取的矩阵算法[J].系统工程与电子技术,2005,27(3):441-445.

[5] 苗夺谦,王 珏.基于粗糙集的多变量决策树构造方法[J].软件学报,1997,8(6):425-431.

[6] 刘小虎,李 生.决策树的优化算法[J].软件学报,1998,9(10):797-800.

[7] Hong J R. AE1:An extension matrix approximate method for general covering problem[J]. International Journal of Computer and Information Science,1985,14(6):421-437.

[8] 王国胤.决策表核属性的计算方法[J].计算机学报,2003,26(5):611-615.

[9] Liang J Y, Shi Z Z. The information entropy, rough entropy and knowledge granulation in rough set theory[J]. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,2004,12(1):37-46.

[10] Guan J W, Bell D A. Rough computational methods for information systems[J]. Artificial Intelligences, 1998,105:77-103.

[11] 郑 芳,吴云志,杭小树.粗集理论中知识的粗糙性研究[J].计算机工程与应用,2002,38(4):98-101.

[12] 徐章艳,刘作鹏,杨炳儒,等.一个复杂度为 max($O(|C||U|)$,$O(|C|^2|U|)$)的快速属性约简算法[J].计算机学报,2006,29(3):391-399.

[13] Grzymala-Bausse D M, Grzymala-Busse J W. The usefulness of a machine learning approach to knowledge acquisition [J]. Computational Intelligence,1995,11(2):268-279.

(上接第 96 页)

MFP 算法只需对事务数据库扫描一次,就可把事务数据库转换成 MFP 树,然后对 MFP 树进行挖掘。实验表明:MFP 算法的关联规则挖掘时间效率较高,是一种切实高效的挖掘方法。

参考文献:

[1] 刘乃丽,李玉忱.一种基于 FP_tree 的最大频繁项目集挖掘算法[J].计算机应用,2005,25(5):999-1000.

[2] 毛国君,段立娟.数据挖掘原理与算法[M].北京:清华大学出版社,2005.

[3] Goebel M, Gruenwald L. A Survey of Data Mining and Knowledge Discovery Software Tools[J]. SIGKDD Explorations,1999,1(5):20-23.

[4] 蒋良孝.一种基于 FP-增长的决策规则挖掘算法[J].计算机科学,2003,32(6):23-25.

[5] Han J,Pei J. Freespan:Frequent pattern-projected Sequential pattern Mining[R]. In Technical Report CMPT2000-06,Simon Fraser University,2000:6-12.

[6] 高 俊,何守才.布尔型关联规则挖掘算法[J].计算机工程,2006,32(1):116-118.

[7] Han Jiawei. Data Mining:Concepts and Techniques[D]. Burnaby:Simon Fraser University,2000:155-163.

[8] 张云涛,龚 玲.数据挖掘原理与技术[M].北京:电子工业出版社,2004.