文章编号: 1001-9081(2006)08-1898-02

基于日历的时序关联规则挖掘算法

崔晓军[1,2], 薛永生[2]
(1.襄樊职业技术学院 信息技术系, 湖北 襄樊 441050; 2.厦门大学 计算机科学系, 福建 厦门 361005)
通讯作者: xjxhy@yeah.net

摘 要: 以日历格作为框架来研究时序关联规则, 提出了一个有效的挖掘算法。在用户指定的日历模式下, 首先通过一次扫描产生所有的频繁 2项集及相应的 1*日历模式, 在此基础上产生 k*日历模式, 并利用聚集性质产生候选 K项集及相应的日历模式, 最后扫描事务数据库产生所有的频繁项集及其日历模式。实验证明, 该算法具有较好的性能。

关键词: 日历格; 时序数据挖掘; 关联规则
中图分类号: TP311.13 文献标识码: A

Algorithm for mining calendar-based temporal association rules

CUI Xiao-Jun[1,2], XUE Yong-sheng[2]
(1.*Engineering Faculty, Ningbo Polytechnic College, Ningbo Zhejiang* 315800, *China*;
2.*Department of Infomation Technology, Xiangfan Vocational and Technical College, Xiangfan Hubei* 441050, *China*)

Abstract: An efficient algorithm for temporal association rules based on calendar patterns was presented. A user-given calendar schema was adopted to specify the interesting time intervals as calendar patterns. Then database was scanned once to find all frequent 2-itemsets and their 1-star calendar patterns. Aggregation property and Apriori property were utilized to find all candidate patterns. Finally, calendar based temporal association rules were obtained through scanning. The experimental results indicate that this proposed algorithm is feasible and efficient.

Key words: calendar lattice; temporal data mining; association rule

0 引言

关联规则挖掘发现大量数据中项集之间有趣的关联或相关联系, 首先由 Agrawal[1]等人提出, 由于其内在的计算复杂性[2], 发现更快速的挖掘方法一直是众多研究的目标, 然而大量研究着重于算法效率的提高, 而很少考虑时间模式的影响。例如, 人们在每年 2月 14号购买鲜花和巧克力, 项集(鲜花, 巧克力)在传统关联挖掘中会被认为非频繁的(不满足最小支持度阈值), 然而如果考察周期为 2月 14号, 则是频繁的。因此, 考虑时间模式[3]的关联规则挖掘更有实际意义。

关于时序模式的关联规则挖掘的研究, 文献[4]提出周期性关联规则的挖掘, 文献[5]提出循环关联规则挖掘, 文献[6,7]提出日历关联规则挖掘。由于周期性和循环模式建立在单一的时间粒度上, 而日历模式建立在多时间粒度上, 这与实际生活中的年、月、日, 时、分、秒等多粒度时间表示更吻合, 因此基于日历的时序关联规则挖掘更有实用价值。

1 问题定义

1.1 日历模式

定义 1 日历格 C, 是一个带合法性约束 v的关系格 $C=(A_1: D_1, A_2: D_2, \cdots, A_n: D_n)v$, 每个属性 A_i是一个时间粒度名称, 如年、月; 每个域 D_i是一个有穷的正整数的子集; 约束 v是一个 $D_1\times D_2\times\cdots\times D_n$上的布尔函数, 保证连接后的结果是合法的时间间隔, 例如, 若想排除周末的事务, 则可将相应的日期约束设为 F。为简化描述, 下文对 C的描述忽略 D_i和 v, 即 C表示为$(A_1, A_2, \cdots, A_n)$。

定义 2 日历模式 $p=(d_1, d_2, \cdots, d_n)$, 是日历格 C上的一个元组。对于 $\forall d_i(1\leqslant i\leqslant n)$, 有 $d_i\in D_i$或 $d_i=*$(不指定 d_i的值)。

例 1 给定日历格(年, 月, 日):

日历模式 $p_1=(2005, 5, 1)$表示 2005年的 5月 1日; 日历模式 $p_2=(*, 2, 14)$表示每年的 2月 14日。

通常, 一个日历模式可以从日历格中通过固定部分日历单位的值而得到, 很显然, 一个日历模式定义了一个时间间隔的集合。

定义 3 日历模式 p的子模式 sp, 在给定的日历格 C下, 日历模式 $sp=(d_1', d_2', \cdots, d_n')$是日历模式 $p=(d_1, d_2, \cdots, d_n)$的子模式, 当且仅当对于任意的 $i(1\leqslant i\leqslant n)$, $d_i=d_i'$或 $d_i=*$, 即 sp所表示的时间间隔是 p所表示时间间隔的子集。

设定在一个日历格 $C=(A_1, A_2, \cdots, A_n)$中, 对于任意的 $i(1\leqslant i<n)$, A_i单元均包含 A_{i+1}的所有单元, 如(年, 月, 日)中, 每个月均包含在某一年中, 而(年, 月, 周)中每个周并不完全包含在月中, 这在本文中是不予考虑的格。

定义 4 k-star日历模式 p_k, 指含有 k个 * 的日历模式; *star*日历模式 p*指至少含有一个 * 的日历模式; 基本时间间隔指不包含 * 的日历模式。

性质 1 k-star日历模式 $p_k=(*, *, *, \cdots, *, A_{k+1}, A_{k+2}, \cdots, A_n)$可以通过所有的$(k-1)$-star日历模式 $p_{k-1}=(*, *, *, \cdots, *, A_k, A_{k+1}, \cdots, A_n)$合并得到, 即 $p_k=\cup p_{k-1}$。

证明: 假定 D_k共由 m个不同的时间间隔组成, 即 A_k有 m

收稿日期: 2006-02-16; 修订日期: 2006-05-08
基金项目: 福建省自然科学基金资助项目(A0310008); 福建省高新技术研究开放计划重点资助项目(2003H043)
作者简介: 崔晓军(1972-), 男, 湖北襄樊人, 副教授, 硕士研究生, 主要研究方向: 数据仓库、数据挖掘; 薛永生(1946-), 男, 福建福清人, 教授, 主要研究方向: 数据库理论与应用、分布式数据库、数据仓库、数据挖掘、网络技术。

个不同的取值 A_k^j($1 \leqslant j \leqslant m$), 则每个(k – 1)– star日历模式可表示为:

$$X \xrightarrow{t_i} Y \quad |D_i(I)\frac{|P_i(X \cup Y)|}{|P_i(X)|} =$$

$$(*, *, *, \cdots, *, A_k^j, A_{k+1}, \cdots, A_n)$$

p_{k-1}的并可表示为:

$$\bigcup p_{k-1} = \sum_{j=1}^{m} p_{k-1}$$

$$= \sum_{j=1}^{m} (*, *, \cdots, *, A_k^j, A_{k+1}, \cdots, A_n)$$

$$= (*, *, \cdots, *, *, A_{k+1}, \cdots, A_n)$$

$$= p_k$$

性质 1表明, 如果每个日历模式 p_{k-1}的信息已知, 则日历模式 p_k也可推导出来。

1.2　时序关联规则

根据关联规则的定义[1], 设 $I = \{i_1, i_2, \cdots, i_n\}$是数据项的集合, 任务相关的数据 D是数据库事务集合, D中的每个事务 $T \subseteq I$, A是一个项集, 当且仅当 $A \subseteq T$时 T包含 A。关联规则 $X \rightarrow Y (X \subset I, Y \subset I, X \cap Y = \varnothing)$成立, 当且仅当其支持度 $supp(A \rightarrow B) \geqslant$ 用户设定的最小支持度 min_sup, 且其置信度 $conf(A \rightarrow B) \geqslant$ 用户设定的最小置信度 min_conf。

假定每个事务 T都具有一个时间戳 TID(确定事务发生的时间), 利用日历格的基本时间间隔 t或日历模式 p, 可以将满足基本时间间隔 t的事务集合表示为 $T(t)$, 而将满足日历模式 p的事务集表示为 $T(p)$。假定事务数据库 D被划分为 n个集合 $D_1, D_2, \cdots, D_n$, 每个 D_i包含在相应的时间间隔 t_i内发生的所有事务。时序关联规则的挖掘任务旨在发现基于给定的日历格的所有的时序关联规则, 即发现所有的关联规则 r和日历模式 p的匹配(r, p), 其中关联规则 r满足最小支持度阈值和最小置信度阈值, 如(鲜花 巧克力, 每年的 2月 14日)。

在给定时间间隔 t_i内的关联规则可以表示为 $X \xrightarrow{t_i} Y$, 其中 $X \subset I, Y \subset I, X \cap Y = \varnothing$。令 $|D_i|$表示划分 D_i中的事务总数, $|D_i(I)|$表示在划分 D_i中包含项集 I的事务数, 时序关联规则 $X \xrightarrow{t_i} Y$在划分 D_i中的支持度 $S_i = \frac{|P_i(X \cup Y)|}{|P_i|}$, 其置信度 $C_i = \frac{|P_i(X \cup Y)|}{|P_i(X)|}$, 关联规则 $X \rightarrow Y$在时间间隔 t_i内成立, 当且仅当 $S_i \geqslant min_sup$且 $C_i \geqslant min\_conf$。

2　基于日历的时序关联挖掘算法

2.1　基本思想

发现全部基于日历的时序关联规则的工作可以分为三步: 1) 通过分区扫描事务数据库, 发现满足 1* 日历模式(V^1)的所有的频繁 2项集 $L2$; 2) 利用 $L2$生成满足所有 K * 日历模式(V^k)的候选项集; 3) 通过一次扫描事务数据库, 发现所有的频繁项集及相应的频繁日历模式。

2.2　算法描述

算法: BCTAR(基于日历的时序关联规则挖掘算法)

输入: 事务数据库 D; 最小支持度阈值 min_sup; 日历模式 $p = (d_1, d_2, \cdots, d_m)$

输出: D中的频繁项集 L及相应的频繁日历模式集 V

方法:

```
将 D 划分为 n 个不相交的分区 P1, P2, …, Pn;
L2 = ∅; V1 = ∅;
∀2-项集 I∈ P1
IF I是频繁的                          //大于最小支持度计数
  {L2 = L2 ∪ { I};
  V1(I) = V1(I) ∪ V11;
                          //V1(I) 为项集 I的 1* 日历模式集合
  V11. count = 1; }
                  /N11. count为项集 I的第 1个 1* 日历模式的计数
FOR k = 2 to n
  ∀2-项集 I∈ Pk
  IF I是频繁的
    IF I∉ L2
      {L2 = L2 ∪ { I};
      V1(I) = V1(I) ∪ V11;
      V11. count = 1; }
    ELSE IF I∈ L2 and Vi1 ∉ V1(I)
      {V1(I) = V1(I) ∪ Vi1;
      Vi1. count = 1; }
  ELSE
    Vi1. count++;
ENDFOR
∀2-项集 I∈ L2
FOR k = 2 to m – 1
  Vk(I) = ∪_{j≥1} Vjk-1                //Vk(I) 为项 I的 K * 日历模式
VI = ∪_{k≥1} (∪_{j≥1} Vjk)             /NI为项集 I的候选日历模式集合
C3 = L2∞L2                        /由频繁 2项集连接生成候选 3项集
FOR k > 3
  Ck = Ck-1∞Ck-1                  /由候选 k – 1项集连接生成候选 k项集
FORK ≥ 3
  ∀k-项集 Iik ∈ Ck
  VIik = ∩ ({VIjk-1 ⊂ Iik})
FOR each t∈ T
  /扫描事务数据库一次, 产生频繁项集 I及相应的日历模式 V
  ∀I∈ C
  IF TID ∈ VI and I∈ t
    I. count++;
  ∀I∈ C
  IF I. count ≥ |T| * min_sup L = L ∪ {I}; V' = V' ∪ {VI}
    RETURN L, V';
```

3　实验与性能分析

为了验证算法的优越性, 给出了此算法与文献[6]中的时序 Apriori算法的比较分析。实验数据采用文献[1]中介绍的 T10.I4.D400K 和 T10.I4.D1000K, 其中 T代表事务平均大小, I代表可能的最大频繁项集的项数, D代表事务数据库的事务数, 以 K为单位。实验采用的日历模式为: 年, 月, 日。两个事务数据库分别划分为 400和 1 000个分区, 每个分区中的 1 000条记录对应于一个基本的时间间隔。实验环境中的硬件平台为 DELL OPTIPLEX GX270(P4 2.60GHz CPU, 512M RAM), 操作系统 Windows 2003 Server, 算法用 Java实现。

实验结果如图 1、图 2所示, 可以看出, 尽管 BCTAR算法比时序 Apriori算法产生的候选项集多一些, 但在支持度阈值较小时性能更优越。这是由于在较小的支持度阈值时, 最大的频繁项集的长度将更长, 时序 Apriori算法扫描数据库的次数将会增加, 而 BCTAR算法则最多扫描数据库两次。因此,

(下转第 1903页)

右分别是 $L(K_2)$ 插入 $L(K_1)$ 中耗费的时间和 $L(K_1)$ 插入 $L(K_2)$ 中耗费的时间。

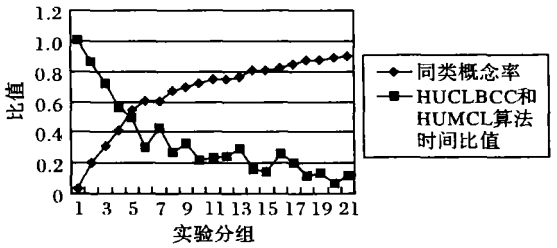

图 3　时间性能随概念同类率趋势图

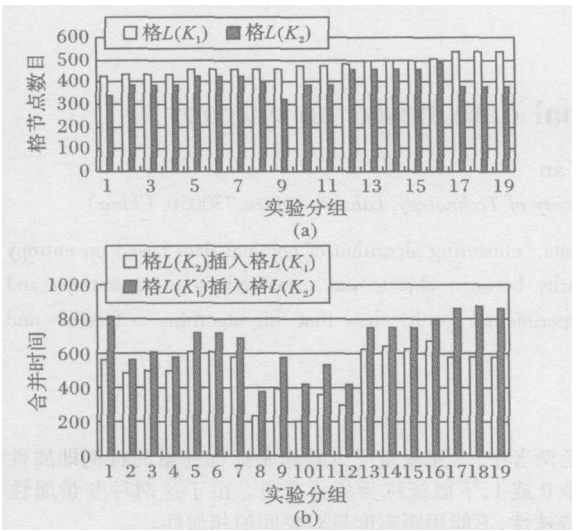

图 4　$L(K_1)$ 和 $L(K_2)$ 插入顺序不同对算法 3性能的影响

图 4表明:把节点数量较少的格插入到节点多的格中比反着做更省时间。例如,实验中第 7个分组,格 $L(K_1)$ 的节点个数比较多,将格$L(K_2)$插入$L(K_1)$进行合并耗费的时间要少于将格 $L(K_1)$ 插入 $L(K_2)$ 耗费的时间,说明把节点数量较少的格插入到节点多的格中比反着插入更省时间。

4　结语

本算法利用了待插入节点的同类概念仅需更新自身及其子节点即可完成插入的特点,节省大量的比较时间,实验表明,本文所提出的基于同类概念的概念格横向合并算法是有效的。并且,该算法也很适用于对概念格进行并行构造,将本算法真正地用于概念格分布并行构造是我们将要进行的下一步工作。

参考文献:

[1] GANTER B, WILLE R. Formal Concept Analysis: Mathematical Foundations[M]. Berlin: Springer-Verlag, 1999.

[2] KROHN U, DAVIES N J, WEEKS R. Concept lattices for knowledge management[J]. BT Technology Journal, 1999, 17(4): 108 - 113.

[3] KUZNETSOV S O. Machine learning on the basis of formal concept analysis[J]. Automation and Remote Control, 2001, 62(10): 1543 - 1564.

[4] TILLEY T, COLE R, BECKER P, *et al*. A Survey of Formal Concept Analysis Support for Software Engineering Activities[A]. Proceedings of 1st International Conference on Formal Concept Analysis[C]. 2003.

[5] 李云, 刘宗田, 陈　, 等. 多概念格的横向合并算法[J]. 电子学报, 2004, 11: 1849 - 1854.

[6] NJIWOUA P, NGUIFO E M. A parallel algorithm to build Concept Lattice[A]. Proceedings of the 4th Groningen International Information Technology Conference for Students[C]. University of Groningen, The Netherlands, Fevrier 1997. 103 - 107.

[7] FU H G, NGUIFO E M. A Parallel Algorithm to Generate Formal Concepts for Large Data[A]. Second International Conference on Formal Concept Analysis, ICFCA 2004[C]. Sydney, Australia: Springer, 2004. 394 - 401.

(上接第 1899页)

BCTAR算法在不同的支持度阈值下均表现出较好的性能,而时序 Apriori算法则随着阈值的不同差异很大。

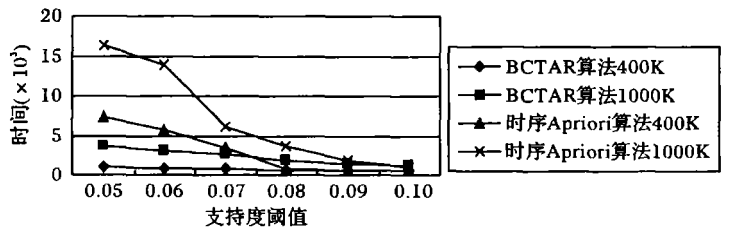

图 1　算法在不同的事务数据库和不同的支持度阈值下的执行时间比较

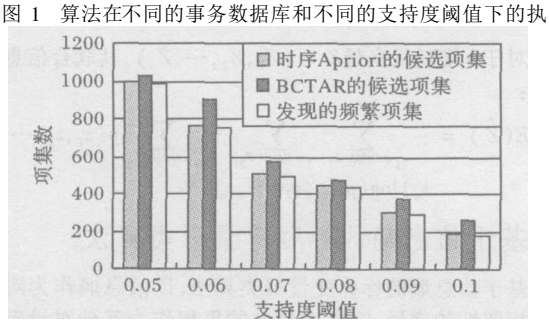

图 2　算法所产生的候选项集的比较

实验结果表明,本文提出的基于日历格的时序关联规则挖掘算法能够用于发现时序数据库中任意时间间隔内的时序关联规则,且当支持度阈值较小时,该算法具有较好的性能。

参考文献:

[1] AGRAWAL R, MIELINSKI T, SWAMI A N. Mining association rules between sets of items in large databases[A]. Proceedings of the ACM-SIGMOD 1993 International Conference on Management of Data[C]. 1993. 207 - 216.

[2] AGRAWAL R, SRIKANT R. Fast algorithms for mining association rules in large databases[A]. Proceedings of the 1994 International Conference on Very Large Data Bases[C]. 1994. 487 - 499.

[3] ALE J M, ROSSI G H. An approach to discovering temporal association rules[A]. Proceedings of the 2000 ACM Symposium on Applied Computing[C]. 2000. 294 - 300.

[4] HAN J, DONG G, YIN Y. Efficient Mining of Partial Periodic Patterns in Time Series Database[A]. Proceedings of the International Conference on Data Engineering[C]. 1999. 106 - 115.

[5] OZDEN B, RAMASWAMY S, SILBERSCHATZ A. Cyclic Association Rules[A]. Proceedings of the 15th International Conference on Data Engineering[C]. 1998. 412 - 421.

[6] LI Y, NING P, WANG X S, *et al*. Discovering Calendar-based Temporal Association Rules[J]. Data and Knowledge Engineering, 2003, 44(2): 193 - 218.

[7] RAMASWAMY S, MAHAJAN S, SILBERSCHATZ A. On the Discovery of Interesting Patterns in Association Rules[A]. Proceedings of the International Very Large Database Conference[C]. 1998. 368 - 379.