第32卷 第23期 计 算 机 工 程 2006年12月
Vol.32 No.23 Computer Engineering December 2006
·软件技术与数据库· 文章编号：1000—3428(2006)23—0043—02 文献标识码：A 中图分类号：TP311

一种新的关联规则增量式挖掘算法

张健沛，杨 悦，刘 卓
(哈尔滨工程大学计算机科学与技术学院，哈尔滨 150001)

摘 要：针对数据库不断更新变化及现实生活中大多只对近期数据感兴趣的特点，该文提出了一种基于滑动窗口过滤器的关联规则增量式挖掘算法（SWFAI 算法）。该算法通过分组及时舍弃挖掘过程中生成的非频繁项目集，有效降低主存压力，减少对数据库的扫描次数，能够对时变数据库进行高效地关联规则挖掘。通过实验证明了该算法能够有效地进行关联规则的挖掘，并在效率上有较大提高。

关键词：关联规则；增量式挖掘；滑动窗口；过滤器；频繁项目集

A Novel Incremental Mining Algorithm of Association Rules

ZHANG Jianpei, YANG Yue, LIU Zhuo
(College of Computer Science and Technology, Harbin Engineering University, Harbin 150001)

【Abstract】 With the continual change and update of data in database, and the character of interesting in recent data in real life, an incremental mining algorithm of association rules based on sliding window filter(SWFAI) is presented. In the executing process of SWFAI algorithm, non-frequent item sets are given up in time by the way of dividing groups. The stress of main memory is abated, the times of scan of database are cut down, and the algorithm executes more efficient mining of association rules in time-variant database. An experiment is designed to prove that SWFAI algorithm can perform the mining of association rules availably, and the efficiency is improved at a certain extent.

【Key words】 Associational rules; Incremental mining; Sliding window; Filter; Frequent item sets

1 概述

关联规则是Agrawal等人提出的一个重要的数据挖掘方法，其核心技术是基于频繁项目集理论的递推方法[1]。目前，关联规则方法已在附加邮递、目录设计、追加销售、仓储规划以及基于购买模式对客户进行划分等方面得到了广泛的应用。基于关联规则的数据挖掘方法能够发现大量数据中项集之间有趣的关联或相关联系。Apriori算法是关联规则挖掘算法的代表，使用逐层搜索的迭代方法，k-项集用于搜索(k+1)-项集，用候选项集 C_k 找频繁项集 L_k 。诸多研究对原有的Apriori类算法进行优化，如引入随机采样、并行思想等以提高算法挖掘的效率。另外，Han等人提出了基于FP-Tree的关联挖掘算法FP-growth，其对不同长度的规则有很好的适应性，同时在效率上较之Apriori类算法有很大的提高[2]。但是，无论哪种算法其本身都无法更新、维护和管理已挖掘出来的关联规则。针对更新问题，增量式挖掘算法被提出了。

所谓增量式挖掘算法是指针对动态变化的数据库或数据仓库，当某些情况发生变化时，并不需要重新扫描整个数据库，而是在原来挖掘结果的基础上，仅作由新情况所引起的更新。增量式挖掘使得关联规则库处于动态更新状态，既具有动态的学习能力，又有相对较优的时间特性。关联规则的增量式挖掘大体包含两种情况[3]：

(1)最小支持度与最小置信度保持不变，数据库或数据仓库随时间不断变化，引起关联规则发生变化，包括以下两种情况：

1)当一个新增数据集db添加到原始数据库 DB 中时，生成新数据库 $DB \cup db$ 的关联规则。针对该问题，Cheung等人提出了FUP算法及FUP2 算法[4]；

2)当一个数据集 db 从原始数据库 DB 中删除时，生成新数据库 $DB \cup db$ 的关联规则，主要解决方法是 FUP2 算法。

(2)数据库或数据仓库保持不变，调整最小支持度与最小置信度，引起关联规则发生变化，针对这种情况，冯玉才等提出了IUA[5]，朱玉全等提出了FIUA等关联规则的增量式更新算法[6]。

本文主要就第 1)类问题如何生成关联规则进行研究。目前解决这一问题主要是应用FUP算法，一般情况下原始数据库 DB 是很大的，FUP算法需要对数据库($DB \cup db$)进行k次扫描使得算法的效率明显降低[7]。同时，很多增量式挖掘问题只对近期数据感兴趣，需要得出部分旧数据后的挖掘结果，而不是整个数据库的挖掘结果。针对这一问题，本文提出了基于滑动窗过滤器的关联规则增量式挖掘算法，简称SWFAI算法。

2 SWFAI 关联规则的增量式挖掘算法

SWFAI 算法的基本思想是将事务数据库分割为若干部分{P1，P2，…，Pn}，算法为每一部分生成的候选项目集分配一个过滤器 F 进行过滤，前一挖掘阶段生成的候选项目被选择性地带入下一阶段的候选项目集，后续的挖掘过程生成的候选项目集包含两种类型：

(1) α 型候选项目集：由前一阶段生成被选择带到下一阶段，继续作为候选项目集参与当前阶段的候选项目集；

基金项目：黑龙江省自然科学基金资助项目(F2005-02)；哈尔滨工程大学基础研究基金资助项目
作者简介：张健沛(1956－)，男，教授、博导，主研方向：数据库与知识库；杨 悦，博士生；刘 卓，硕士生
收稿日期：2006-03-24 **E-mail：**candyyangyue@yahoo.com.cn

(2) β 型候选项目集：不包含在前一阶段生成的候选项目集当中，而由当前阶段新产生的候选项目集。

在挖掘过程中，SWFAI 算法考查候选项目集中每个项目的起始位置和出现次数，以此来判断 α 型、β 型候选项目集，随后通过对出现次数和相应支持度的比较决定该项目是否作为挖掘结果保留在过滤器中。

现将 SWFAI 算法描述中涉及到的变量介绍如下：

$db^{i,j}$ ：表示由 P_i 到 P_j 的连续的部分组成的部分事务数据库；$|P_k|$ ：表示 P_k 中包含的事务数目；s ：表示最小支持度；$N_{P_k}(I)$ ：P_k 部分生成项目集中项目 I 的数目；$C_v^{i,j}$ ：由 $db^{i,j}$ 生成的候选 v-项集；$C^{i,j}$ ：由 $db^{i,j}$ 生成的候选项目集的集合；L：最终得到的频繁项目集；Δ^- ：在对时变数据库的挖掘过程中要去除的数据部分；D^- ：在对时变数据库的挖掘过程中保持不变的数据部分；Δ^+ ：在对时变数据库的挖掘过程中新增加的数据部分。

SWFAI 算法如下：

输入：事务数据库 DB，最小支持度 s 以及新增数据部分 db；

输出：新增数据加入后，增量式挖掘得出的频繁项目集结果 L。

方法：首先将原事务数据库分为若干部分，P_1、P_2、P_3、…、P_n，该数据库可表示为 $db^{1,n}$，为每一部分分配一个过滤器 F 并置空。

(1)对 P_1 部分进行扫描，记录生成的 2-项集中各项目的起始位置 I.start 和出现次数 I.count。将 I.count 与 $\lceil s * |P_1| \rceil$ 进行比较，若不小于，则将该项目 I 置入过滤器 F，否则舍弃。

(2)对 $db^{1,n}$ 的 P_k ($2 \le k \le n$)按顺序进行扫描，对生成的 2-项集记录其起始位置 I.start 及出现次数 I.count。

1)如果该项目不在 F 中，正常记录其 I.start，I.count 值，比较 I.count 与该部分的最小支持度 $\lceil s * |P_k| \rceil$ 的大小，若不小于，则将该项目置入 F 中，执行 F=F$\bigcup$I。

2)如果该项目已经在 F 中，则计算其总的出现次数，即当前部分的出现次数与原来的出现次数相累加，计算其起始部分到当前部分的最小支持度 $\left\lceil s * \sum_{m=I.start,k} |P_m| \right\rceil$，若前者小于后者，则在 F 中移除该项目，否则继续保留。

(3)重复(2)直至所有 P_k 执行扫描完毕，将 F 中的所有项目及其 I.start 和 I.count 置入 $C_2^{1,n}$，存入主存中。

(4)新的数据 Δ^+ 进入数据库，计算原始数据库去掉数据 Δ^- 后保留的 D^- 部分的 2-项集。无需扫描数据库，只需考查主存中现有的 $C_2^{1,n}$ 项目的 I.start 和 I.count，若项目的 I.start 值小于当前的起始点，则将其设置为当前起始点，并将该项目的 I.count 作相应改变；否则 I.start 和 I.count 均保持不变，与相应的最小支持度进行比较，得到 D^- 部分的 2-项集。

(5)利用 D^- 部分的 2-项集，从(2)中的方法得到 $D^- + \Delta^+$ 部分即数据库 $db^{i,j}$ 的 2-项集，存入主存中。

(6) $C_{v+1}^{i,j} = C_v^{i,j} * C_v^{i,j}$ ($2 \le v \le j$)生成所有的频繁项目集，将所有生成的频繁项目集的 I.count 设置为 0。

(7)对 $db^{i,j}$ 进行一次整体扫描，对每个 $I \in C^{i,j}$ 考查其出现次数，当其出现次数不小于其对应的最小支持度 $\lceil s * |db^{i,j}| \rceil$，则保留该项目 $L = L \cup I$，否则移除。L 为最后得到的增量式挖掘的频繁项目集结果。

从以上的算法描述可以看出，SWFAI 算法的执行过程只需完成 3 次扫描，有效降低了扫描次数，一般情况下，其消耗的时间要远小于 FUP 算法中的 k 次扫描。该算法又采用了过滤器来增强算法的剪枝效果，使得非频繁项集在其出现时就被及时移除，不会带入并影响下一阶段的算法执行过程，有效缓解了 CPU 和主存的压力，减轻了算法的计算负担并提高了准确度。

3 实验结果分析

在常规计算机环境下用 VC++实现 SWFAI 算法，使用与文献[8]同样的数据生成程序来生成测试数据，与目前相对成熟的 FUP2 算法进行性能比较。

实验中涉及到的参数有：$|D|$ 表示数据库中的事务数目；$|T|$ 表示事务的平均大小；$|\Delta^+|$ 表示增加的事务数目；$|\Delta^-|$ 表示减少的事务数目；$|d|$ 表示新增的事务总数目；$|I|$ 表示最大的潜在频繁项目集的平均大小；$|L|$ 表示最大的潜在频繁项目集的数目；N 表示项目的总数。虽然 SWFAI 算法中 $|\Delta^+|$ 与 $|\Delta^-|$ 的大小可以是不同的，但在实验中为了简单起见，设 $|d| = |\Delta^+| = |\Delta^-|$，原数据库为 $db^{1,n}$，新增数据库为 $db^{i,j}$，则有 $|db^{i,j}| = |db^{1,n} - \Delta^- + \Delta^+| = |D|$，这里 $\Delta^- = db^{1,i-1}$，$\Delta^+ = db^{n+1,j}$。在实验中，为了描述方便，用 Tm-In-Dp-dq 来表示一个 $|T|$=m、$|I|$=n、$|D|$=p、$|d|$=q 的数据库。受空间的限制，实验中测试数据库生成时的参数取值为 N=10 000，$|L|$=2 000，生成的测试数据分别为 T10-I4-D100K-d10K，T10-I4-D100K-d20K 以及 T10-I4-D200K-d20K。实验分为两种情况，分别为：(1)相同测试数据库，不同的最小支持度条件下的测试，结果如图 1 所示。(2)保持最小支持度不变，对 3 种不同的测试数据库运行算法，测试结果如图 2 所示。

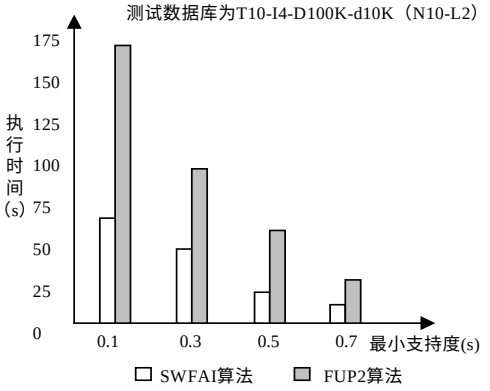

图 1 不同最小支持度下的实验结果

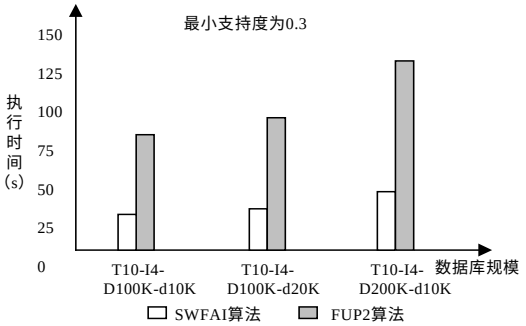

图 2 不同数据库规模下的实验结果

4 结论

在实际应用中，关联规则增量式挖掘大多只对近期数据感兴趣，本文针对这一特点，将滑动窗口过滤器技术应用到关联规则的增量式挖掘中，提出了基于滑动窗口过滤器的关联规则增量式挖掘算法(SWFAI 算法)。通过实验证明了

（下转第 60 页）