渐进式关联规则挖掘算法研究与实现①

Study and Implementation of Mining Algorithm for Association Rules in Advancing Patterns

郭有强　(蚌埠学院 计算机科学与技术系 安徽 蚌埠 233030)

摘　要： 针对大型数据库(或新增数据集)，提出了一种渐进式挖掘算法。算法充分利用以往挖掘过程中的信息，无需再次扫描基础数据集，对新增数据集部分也只扫描一次，能够有效地降低更新挖掘所需的时间成本。给出了算法的具体实现。通过对实验结果的性能对比分析，表明算法是可行的，有效的。本算法的设计思想可以应用到并行关联规则挖掘或实现分布式数据挖掘。

关键词： 关联规则 渐进式挖掘 频繁项目集 剪枝

1　引言

关联规则算法[1]是数据挖掘的一个重要研究方向。关联规则挖掘[2,3]可以分解成两个问题：找出事务数据库中所有支持度不小于最小支持度的项目集，称为频集；在频集中产生所有置信度不小于最小置信度的关联规则。由于第二个问题的求解比较容易和直接，因此目前的关联规则算法都是针对第一个问题提出的。现实中数据库是极其庞大的，挖掘过程需要对数据库进行大量的扫描工作，挖掘效率较低；而且挖掘出来的关联规则只是相对于当前的数据库有效，数据库发生变化，则要重新运行关联规则发现算法，这意味着要重新处理已经处理过的数据，不能有效的利用已经获得的信息。如果在以往工作的基础上,分阶段分时期对数据分块进行渐进式挖掘，尽可能只处理新增部分的数据集，不仅可以提高挖掘效率，而且可以分析出规则的变化趋势(如有些规则逐步淘汰等)。

在对关联规则算法及更新算法研究[4-6]的基础上，提出了基于项目增长原理的挖掘算法，并针对支持度和置信度不变，数据集规模增大(大型数据库先进行数据分块处理，先挖掘一块数据，然后逐步增加)的情况，提出了一种渐进式更新算法。提出的算法充分利用以往挖掘过程中产生的相关信息，只完整扫描一次数据集。

2　问题描述和相关性质

2.1 问题描述

有大型数据库 D，假定 D 已经被挖掘过，并保留了相关信息和频繁项集 L_D,现保持最小支持度 s 不变，添加事务数据集 d，由于事务数据集的变化，将导致 L_D 对于变化后的数据集 D∪d 已无意义，需要重新挖掘 $L_{D\cup d}$。

2.2 相关性质

|D|、|d|、|D∪d|分别为基础数据集、新增数据集、和新增后的全部数据集的总事务条数；L_D 为基础数据集 D 的频繁项目集的集合，L_d 为新增的数据集 d 的频繁项目的集合，$L_{D\cup d}$ 为 D∪d 的频繁项目的集合；D、d 和 D∪d 的最小支持数分别为 $support_D$、$support_d$、$support_{Dud}$；项目集 X 在数据库 D、d 和 D∪d 中的支持数目分别记为 $X.support_D$、$X.support_d$ 和 $X.support_{Dud}$。

性质 1：当 $X.support_{Dud}\geqslant s\times(|D\cup d|)$时，$X\in L_{D\cup d}$

证明：由最低支持度和频繁项目集的定义可直接推导出。

性质 2：保持增长后数据集(D∪d)的支持度不变(为 s)，原数据集 D 的支持数为 $support_D=s\times|D|$，D∪d 的支持数为 $support_{Dud}=s\times|D\cup d|$，新增数

① 基金项目:安徽省科技厅自然科学研究项目(050420207);蚌埠学院自然科学重点项目(BBXY2007204A)
收稿时间:2008-10-28

据集的支持数应为 $support_d=s\times|d|$

证明：因为 $D\cup d$ 的支持度为 s

$s=support_{Dud}/|D\cup d|$

$=(support_D+support_d)/|D\cup d|$

$=(s\times|D|+support_d)/|D\cup d|$

所以 $support_d=s\times|d|$

性质 3：$L_D\cap L_d\in L_{D\cup d}$

证明：因为存在项目集 $X\in L_D$ 所以 $X.support_D\geqslant|D|\times s$

又因为 $X\in L_d$ 所以 $X.support_d\geqslant|d|\times s$

令 X 在 $D\cup d$ 中的支持度为 s′，于是可得：

$s'=X.support_{DUd}/(|D\cup d|)$

$=(X.support_D+X.support_d)/(|D|+|d|)$

$\geqslant(|D|\times s+|d|\times s)/(|D|+|d|)=s$

得 $s'\geqslant s$ 得证。

性质 4 若某一项目集 X 在 D 中为非频繁项目集，在 d 中也为非频繁项目集，则在 $D\cup d$ 中也为非频繁项目集。

证明：因为存在项目集 X 在 D 中为非频繁项目集，在 d 中也为非频繁项目集

所以 $X.support_D<|D|\times s$，$X.support_d<|d|\times s$

令 X 在 $D\cup d$ 中的支持度为 s′

于是可得

$s'=X.support_{DUd}/(|D\cup d|)$

$=(X.support_D+X.support_d)/(|D|+|d|)$

$<(|D|\times s+|d|\times s)/(|D|+|d|)=s$

得 $s'<s$ 得证。

由性质 3、4 可得推理：$D\cup d$ 中的频繁项目集一定来自 D 的频繁项目集或 d 的频繁项目集。

性质 5：事务项 $X\in L_D\cap X\notin L_d$，若 $X.support_D\geqslant support_D\times|D\cup d|/|D|$，或 $X.support_D+X.support_d\geqslant support_{Dud}$ 则 $X\in L_{D\cup d}$。

证明：由最低支持度和频繁项目集的定义可直接推导出。

性质 6：事务项 $X\in L_d\cap X\notin L_D$，若 $X.support_d\geqslant support_d\times|D\cup d|/|d|$，或 $X.support_D+X.support_d\geqslant support_{Dud}$ 则 $X\in L_{D\cup d}$。

证明：由最低支持度和频繁项目集的定义可直接推导出。

3 算法基本思想及算法实现

3.1 算法基本思想

(1)扫描 d 得到每个项目的支持事务 ID 集合以及记录总数|d|，d 的支持数为 $s\times|d|$，合并后 $D\cup d$ 的支持数为：$s\times|D\cup d|$。

(2)采用项目增长法的算法，得到 d 的频繁项集 Ld。

(3)考虑两频集的重复部分，将两频集相与，由定理 3 将结果存入 $LD\cup d$ 。

(4)考虑两频集的不重复部分。

①原属于 D 中的频集：在 D 中的支持数大于等于合并后所需要的支持数，则应包含在 $L_{D\cup d}$ 中；如在 D 中的支持数小于合并后所需要的支持数，则需要进一步考虑在 d 中的支持数，两者之和若大于等于合并后所需要的支持数，则应包含在 $L_{D\cup d}$ 中；否则，不应该是合并后的频繁项集中的成员。

②原属于 d 中的频集：在 d 中的支持数大于等于合并后所需要的支持数，则应包含在 $L_{D\cup d}$ 中；如在 d 中的支持数小于合并后所需要的支持数，则需要进一步考虑在 D 中的支持数，两者之和若大于等于合并后所需要的支持数，则应包含在 $L_{D\cup d}$ 中；否则，不应该是合并后的频繁项集中的成员。

(5)经过以上处理得到的 $L_{D\cup d}$ 即为合并后 $D\cup d$ 的频繁项集。

综上所述，利用已知信息，得到在 $D\cup d$ 中项集是否频繁的分布如图 1 所示。

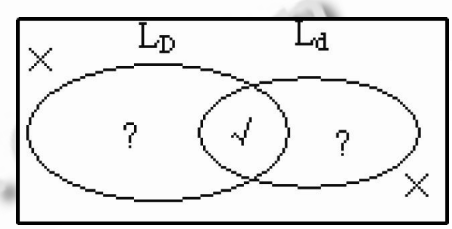

图 1　项目集在 $D\cup d$ 中是否频繁分布图

(图中√为频繁，×为不频繁，？为不确定)

(6)合并保存原数据集 D 和新增数据集 d 中的所有 1 项目的支持事务集合及事务记录总数|D∪d|，以便以后事务动态再增长时，可以不再扫描原数据集，只要扫描新增数据集，重复 1-6 步骤，即可动态更新频繁项集。

项目增长法(为提高挖掘数据集频集的效率，本文提出的一种算法)：对于数据集 D，得到 D 中所有 1 项目的相关信息(每项目支持事务集合和支持事务计数)，得到频繁 1-项集；在此基础上使用连接项集 L(即当前的频繁 1-项集)中的项目对频繁 1-项集进行增

长，分别得到侯选 2-项集，从而求得频繁 2-项集，将频繁 2-项集中的所有项包含的单一项构成的集合，与连接项集求交集，动态更新连接项集 L 中的项目(即产生新的连接项集，为以后增长做准备)；.....；在得到频繁 K－1 项集后，利用求解过程中更新的连接项集中的项目进行增长，分别得到侯选 K 项集，从而求得频繁 K 项集。

3.2 算法实现

for all T∈d do　　//T：数据集中的事务记录
begin
得到 d 中每个项目的事务支持 ID 集合和记录总数|d|;
end;
得到 d 的频繁项集 L_d;　　//采用项目增长法
$L_c=L_D\cap L_d$;
if($L_c\neq\phi$)
　　$L_c\in L_{D\cup d}$;　　//L_c 应被包含在 $L_{D\cup d}$ 中
for all　$X\in L_D-L_c$　do　　//X：项目集
begin
if($X.support_D\geq support_D\times|D\cup d|/|D|$
or($X.support_D+X.support_d)\geq support_{Dud}$)
　　$X\in L_{D\cup d}$;
end;
for all　$X\in L_d-L_c$ do
begin
if($X.support_d\geq support_d\times|D\cup d|/|d|$
or($X.support_d+X.support_D)\geq support_{Dud}$)
　　$X\in L_{D\cup d}$;
end;
合并保存 D 和 d 中的所有 1 项目的支持事务集合及事务记录总数|D∪d|;
Answer=$L_{D\cup d}$;

4　实验及性能对比分析

本文所采用的测试机为台式 PC 机，其配置是：CPU 为 Pentium® 4 3.06GHZ,512MB 内存，操作系统为 Windows2003 服务器，选用 VC++6.0 编程环境。数据来源为某学院教务处学籍管理数据，数据库采用 SQL server2000，属性有 15 个。在支持度 5%，使用项目增长法在不同数据集下的测试结果，用项目增长法求得事务数为 4000 的频繁项集，然后利用渐进式挖掘方法计算每递增 1000 事务的频繁项集，形成渐进式挖掘，测试结果如表 1 所示。

表 1　使用项目增长法测试数据描述(时间单位为 S)

记录数	4000	5000	6000	7000	8000	9000	10000
项目增长法	8.12	9.69	11.09	12.5	13.91	14.06	16.72
渐进式挖掘	8.12	9	10.5	11	11.9	12.16	13.56

两种方法性能对比如图 2 所示。

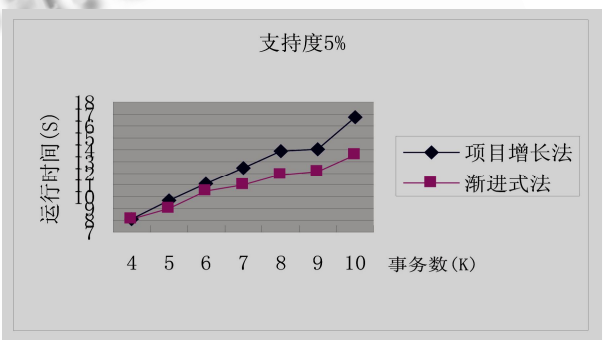

图 2　两种测试结果的性能对比

可以看出，利用渐进式挖掘方法计算新增事务后数据集的频集，用时比重新运算效率略高。验证了本文讨论的方法的正确性、可行性和有效性。随着历史数据集规模的增大，更加能够显现出本算法的优越性和实用性。

5　结束语

在对增量式关联规则更新技术深入研究基础上，提出了本算法。通过分析可以看出：整个更新过程无需再次扫描原数据集，对新增数据集也只扫描一次；通过连接项集产生候选项，增强了产生候选项的针对性和有效性，提高了候选项的支持事务计数的效率；充分利用以往挖掘过程中的结果，竭力获得无需计算支持数便已知的 LD∪d 中的项目，剪枝过程贯穿于整个算法，大大减少了候选集的规模，在算法的运行时耗上具有很明显的优势。但也存在明显的不足，就是

(下转第 60 页)

(上接第 52 页)

始终要保存前面挖掘的相关信息，这样就增加了额外的存储空间。牺牲一定的存储空间，以换取算法的执行速度，在各种存储备份方案优化的今天，在很多环境下是值得的。本算法的设计思想可以应用到并行(Parallel)关联规则挖掘或实现分布式(in Distributed System)数据挖掘。下一步工作准备考虑更合适优化的存储方案，以使本算法不仅在算法运行时间上而且在数据存储空间上都得到优化处理。

参考文献

1 Han Jw, Kamber M.数据挖掘概念与技术,北京:机械工业出版社, 2001.

2 David W,Cheung J, Lee SD ,et al.A general Incremental Technique for Maintaining Discovered Association Rules. Proceedings of the Fifth International Conference on Database Systemsf or Advanced Applications, Melbourne, Australia, 1997:185－194.

3 闫炜,崔杜武,付长龙.基于幂集的关联规则挖掘算法研究.计算机工程与应用, 2004,40(1):192－193.

4 李超,余昭平.基于最大模式的关联规则挖掘算法研究.微计算机信息, 2006,22(6):164－165.

5 陈劲松,施小英.一种关联规则增量更新算法,计算机工程, 2002,28(7):106－107.

6 宋欣,王志航,廉明欢.多属性数据挖掘研究中的关联规则应用.计算机系统应用, 2007,16(8):99－102.