增量式频集快速维护算法研究

郭有强
(蚌埠学院 计算机科学与技术系,安徽 蚌埠 233030)

摘 要:算法充分利用以往挖掘过程中的结果,无需再次扫描原数据集,对新增数据集也只扫描一次,即可得到事务更新后的数据集的频繁项集。避免了重新处理已经处理过的数据和多次扫描新增数据集,与其他相关算法相比,减少了算法运行时间,提高了挖掘效率。随着历史数据集的增大,更加显现出本算法的优越性。算法还可以用于解决由于数据集过大而导致的内存不够的 Apriori 算法的挖掘问题。

关键词:关联规则;增量式更新;1 项集;频繁项目集
中图分类号:TP311.13 **文献标识码**:A **文章编号**:1673-629X(2007)11-0074-03

Study of Incremental Rapid Maintenance Algorithm for Frequent Itemsets

GUO You-qiang
(Computer Sci-tech Department, Bengbu College, Bengbu 233030, China)

Abstract: This algorithm makes full use of the results of mining and will get frequent itemsets of the item updated data set by scanning the newly-added data set only once without rescanning the original one. The algorithm avoids re-dealing with the data which has been dealt with and repeatedly scanning newly-added data set. Compared with other associating algorithm, it reduces the run-time and improves the mining efficiency. With the enlarging of the historical data set, the superiority will be shown more obviously. And it can also resolve the mining problems of the Apriori algorithm which is due to the too large data set leading to the insufficient memory.

Key words: association rule; incremental updating; one itemsets; frequent itemsets

0 引 言

模式维护是数据挖掘中一个具有挑战性的任务。关联规则[1]更新算法就是在数据集规模、支持度 s、置信度 c 发生变化后,重新获得新的关联规则的高效算法。由于获得频繁项目集的计算量占关联规则挖掘的大部分计算量,所以大部分算法都是解决如何获得新的频繁项目集的问题。

对大型数据库的数据挖掘需要对数据库进行大量重复的扫描工作,代价十分巨大,而随着新数据的积累,以往发现的规则有可能变得没有意义,为了在新的数据集上发现关联规则,一种方法是对新的数据集重新运行关联规则发现算法,这样做意味着要重新处理已经处理过的数据,不能有效地利用已经获得的结果。另一种方法是在以往工作的基础上,利用已经获得的结果,发现新的数据集的规则,尽可能只处理新增加的数据集。显然,后者效率较高。D. W. Cheung 等提出了增量式更新关联规则的方法 FUP 和 FUP3 算法[2,3],是解决在支持度和置信度不变的情况下数据集规模增大,即向原数据集添加新的事务记录的情况下,确定新的频繁项目集的算法,其方法能有效地利用已经获得的频繁项目集的结果,产生一个小得多的候选项目集,大大降低了关联规则的更新代价。但其缺点是需要反复多次扫描数据库,代价很大。考虑到一般新增数据集的规模都较小而原数据集规模较大,文中提出的高效动态更新算法在支持度和置信度不变、数据集规模增大时,无需再次扫描原数据集,只是充分利用以往挖掘过程中产生的相关信息,对新增数据集也只扫描一次,即可得到更新后的数据集的频繁项集。与其他相关算法相比,极大地减少了算法运行时间,提高了算法的效率。

收稿日期:2007-01-24
基金项目:安徽省教育厅自然科学研究项目(2006KJ302ZC);蚌埠学院自然科学重点项目(BBXY2007204A)
作者简介:郭有强(1966-),男,江苏邗江人,硕士,副教授,主要从事数据挖掘和可视化程序设计教学与研究。

1 相关定理

原数据集为 D,新增的数据集为 d,支持度为 s,

$|D|$、$|d|$、$|D \cup d|$ 分别为原数据集、新增数据集和新增后的全部数据集的总事务条数;L_D 为原数据集 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 $D \cup d$ 的频繁项目的集合;D、d 和 $D \cup d$ 的最小支持数分别为 $support_D$、$support_d$、$support_{D\cup d}$;项目集 X 在数据库 D、d 和 $D \cup d$ 中的支持数目分别记为 $X.support_D$、$X.support_d$ 和 $X.support_{D\cup d}$。

定理 1　$L_D \cap L_d \in L_{D\cup d}$

证明:因为存在任意项目集 $X \in L_D$,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:

$s' = X.support_{D\cup d}/(|D \cup d|) = (X.support_D + X.support_d)/(|D|+|d|) \geqslant (|D| \times s + |d| \times s)/(|D|+|d|) = s$,得 $s' \geqslant s$,定理得证。

定理 2　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:因为存在任意项目集 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:

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

$(X.support_D + X.support_d)/(|D|+|d|) < (|D| \times s + |d| \times s)/(|D|+|d|) = s$

得 $s' < s$,定理得证。

由定理 1、2 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。

定理 3　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 $X.support_D + X.support_d \geqslant support_{D\cup d}$,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。

定理 4　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 $X.support_D + X.support_d \geqslant support_{D\cup d}$,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。

2　更新算法基本思想及算法实现

2.1　更新算法基本思想

(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) 采用降幂子集[4]的算法(只扫描数据集一次),得到 d 的频繁项集 L_d

(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 $L_{D\cup d}$。

(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。

(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 $|D \cup d|$,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。

2.2　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 Ld; //采用降幂子集的算法
Lc = LD∩Ld; // LD 和 Ld 的交集
if (Lc≠∅) Lc∈LDUd; // Lc 应被包含在 LDUd 中
for all X∈LD - Lc do
begin
  if (X.supportD≥supportD × |DUd|/|D| or (X.supportD + X.supportd)≥supportDUd)
    X∈LDUd;
end ;
for all X∈Ld - Lc do
begin
  if (X.supportd≥supportd × |DUd|/|d| or (X.supportd + X.supportD)≥supportDUd)
    X∈LDUd;
end;
合并保存原数据 D 和新增数据库 d 中的所有 1 项目的支持事务集合及事务记录总数|DUd| ;
Answer = LDUd;
```

3 具体应用分析

假定有事务数据库 D(支持度 $s = 40\%$) 和追加事务数据库 d,如表 1 所示。

表 1 事务数据库 D 和追加事务数据库 d

D	
TID	项目集
1	$ABCE$
2	BCE
3	CDE
4	ACD
5	$ABCE$
6	ABC
7	ABD
8	CE
9	ABC
10	ACE

d	
TID	项目集
11	$ACDE$
12	BCE
13	AE
14	ACD
15	AEF

原数据库在以往一次扫描得到的每个项目的支持事务 ID(如表 2 所示)以及记录总数|D|。

表 2 事务数据库 D 每个项目的支持事务 ID

D		
项目集	支持事务 TID 集合	支持数
A	{1,4,5,6,7,9,10}	7
B	{1,2,5,6,7,9}	6
C	{1,2,3,5,6,8,9,10}	8
D	{3,4,7}	3
E	{1,2,3,5,8,10}	6

以往挖掘得到的 D 的频繁项集为:

$L_D = \{\{A\},\{B\},\{C\},\{E\},\{A,B\},\{A,C\},\{B,C\},\{C,E\},\{A,B,C\}\}$

(1)扫描 d 得到每个项目的支持事务 ID集合(如表3所示) 以及记录总数 $|d|$,d 的支持数为 $s \times |d| = 2$。

表 3 事务数据库 d 每个项目的支持事务 ID

d		
项目集	支持事务 TID 集合	支持数
A	{11,13,14,15}	4
B	{12}	1
C	{11 ,12,14}	3
D	{11,14}	2
E	{11,12,13,15}	4
F	{15}	1

合并后 $D \cup d$ 的支持数为:$s \times |D \cup d| = 6$。使用降幂子集的算法得到 d 中频繁项集为:

$L_d = \{\{A\},\{C\},\{D\},\{E\},\{A,C\},\{A,D\},\{A,E\},\{C,D\},\{C,E\},\{A,C,D\}\}$。

(2) 考虑重复部分:D 与 d 中的频繁项集相与得 $\{\{A\},\{C\},\{E\},\{A,C\},\{C ,E\}\}$,由定理1可知,这些项集还是 $D \cup d$ 中的频繁项集。

(3) 考虑不重复部分:

由原来是 D 中的频繁项集 $\{\{B\},\{A,B\},\{B,C\},\{A,B,C\}\}$ 和 d 中的频繁项集 $\{\{D\},\{A,D\},\{A,E\},\{C,D\},\{A,C,D\}\}$ 两部分组成:

① 考虑第一部分的情况,由定理 3 可得:

$*\{B\}$:在 D 中的支持数为 $\{B\}.\text{support}_D = 6$,所以 $B \in L_{D\cup d}$。

$*\{A,B\}$:在 D 中的支持数为 $5 < 6$,所以需进一步考虑 d 中的支持数,$\{A,B\}.\text{support}_d = 0$,$\{A,B\}.\text{support}_D + \{A,B\}.\text{support}_d = 5 < 6$,所以 $\{A,B\} \notin L_{D\cup d}$。

$*\{B,C\}$:在 D 中的支持数为 $5 < 6$,所以需进一步考虑 d 中的支持数,$\{B,C\}.\text{support}_d = 1$,$\{A,B\}.\text{support}_D + \{A,B\}.\text{support}_d = 6$,所以 $\{B,C\} \in L_{D\cup d}$。

同理可得:$\{A,B,C\} \notin L_{D\cup d}$。

② 考虑第二部分的情况,由定理 4 可得:

$\{D\} \notin L_{D\cup d}$,$\{A,D\} \notin L_{D\cup d}$,$\{A,E\} \in L_{D\cup d}$,$\{C,D\} \notin L_{D\cup d}$,$\{A,C,D\} \notin L_{D\cup d}$

(4) 由定理 1、2 推理及综合 (2)、(3) 所述,合并后 $D \cup d$ 的频繁项集为 $L_{D\cup d} = \{\{A\},\{B\},\{C\},\{E\},\{A,C\},\{A,E\},\{B,C\},\{C,E\}\}$

(5) 合并保存原数据库 D 和新增数据库 d 中的所有 1 项目的支持事务集合(如表 4 所示) 及事务记录总数 $|D|+|d|$。

表 4 $D \cup d$ 所有 1 项目的支持事务集合

$D \cup d$		
项目集	支持事务 TID 集合	支持数
A	{1,4,5,6,7,9,10,11,13,14,15}	11
B	{1,2,5,6,7,9,12}	7
C	{1,2,3,5,6,8,9,10,11 ,12,14}	11
D	{3,4,7,11,14}	5
E	{1,2,3,5,8,10,11,12,13,15}	10
F	{10}	1

4 结束语

在对增量式关联规则维护技术深入研究[5,6]的基础上,提出了本算法。通过以上分析可以看出,整个维护过程无需再次扫描原数据集,对新增数据集也只扫描一次;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 $L_{D\cup d}$ 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。本算法还可以用于解决由于数据集过大而导致的内存不够的 Apriori 算法的挖掘问题,相当于数据集分组挖掘。但也存在明显的不足,就是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。实践表明本算法是可行的。下

(下转第 83 页)

6 小　结

主要介绍一种自动的科研论文搜索和评价方法,利用元搜索引擎实现一个基于搜索的科技论文自动评价过程,该评价过程不仅适用于已发表的论文,还可以用于尚未发表论文的评价。

参考文献:

[1] 叶继元,朱　强.论文评价与期刊评价——兼及核心期刊的概念[J].学术界,2001(3):63－71.

[2] 陈江帆.Web of science数据库检索及其科学研究价值试析[J].情报探索,2002(3):41－42.

[3] 樊怡菁.SCIE 和 Scopus 引文功能的比较分析[J].现代情报,2006(3):80－82.

[4] 岑俏玲.学者专用型搜索引擎——Google Scholar[J].科技情报开发与经济,2005,15(22):56－57.

[5] 李明伟.免费学术数据库 GoogleScholar 浅析[J].情报探索,2005(9):78－79.

[6] 李志荣,沈利华.站在巨人的肩膀上——Google Scholar 搜索引擎的评介[J].现代情报,2005(10):205－206.

[7] Jacso P. Comparison and analysis of the citedness scores in Web of Science and Google Scholar[J]. In Proceeding of Digital Libraries: Implementing Strategies and Sharing Experiences, Lecture Notes in Computer Science, 2005,3815:360－369.

[8] Meng Weiyi, Yu Clement, Liu Kinglup. Building Efficient and Effective Metasearch Engine[J]. ACM Computing Surveys, 2002,34(1):48－84.

[9] 阳小华,刘振宇.元搜索引擎系统合成算法的约束条件[J].软件学报,2002(13):1264－1270.

[10] Leydesdorff L. Clusters and Maps of Science Journals Based on Bi－connected Graphs in the Journal Citation Reports[J]. Journal of Documentation, 2004,60(4):317－427.

[11] Kleinberg J M. Authoritative Sources in a Hyperlinked Environment[J]. Journal of the ACM,1999,46(5):604－632.

(上接第 76 页)

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 Jiawei, Kamber M. 数据挖掘概念与技术[M].北京:机械工业出版社,2001.

[2] Cheung D W, Han J, Ng V T, et al. Maintenance of discovered association rules in Large database : An incremental updating technique[C]//In Proc 12th Int Conf on data engineering. New Orleans, Louisiana: IEEE Computer Society, 1996: 106－114.

[3] Cheung D W, Lee S D, Benjamin K. A general incremental technique for maintaining discovered association rules[C]//In Proceedings of the Fifth International Conference on Database Systems for Advanced Applications. Melbourne, Australia: [s. n.], 1997: 185－194.

[4] 闫　炜,崔杜武,付长龙.基于幂集的关联规则挖掘算法研究[J].计算机工程与应用,2004(1):192－193.

[5] 徐章艳,刘美玲.Apriori 算法的三种优化方法[J].计算机工程与应用,2004(36):190－192.

[6] 杨　明,孙志挥.频繁项目集的快速增量式更新算法[J].应用科学学报,2003(4):368－372.

(上接第 79 页)

参考文献:

[1] 都志辉, 陈　渝, 刘　鹏. 网格计算[M].北京:清华大学出版社,2002.

[2] Buyya R, Murshed M, Abramson D. A Deadline and Budget Constrained Cost－Time Optimization Algorithm for Scheduling Task Farming Applications on Global Grids[C]// The 2002 International Conference on Parallel and Distributed Processing Techniques and Applications(PDPTA'02). Las Vegas: IEEE Press,2002:540－552.

[3] Foster I, Kesselman C. Globus: A Metacomputing Infrastructure Toolkit[J]. Intl J. Supercomputer Applications, 1997, 11(2):115－128.

[4] Fudenberg D, Tirole J. Game Theory[M]. Cambridge: MIT Press,1991.

[5] Sun X H, Wu M. GHS: A Performance Prediction and Task Scheduling System for Grid Computing[C]// Proc. of 2003 IEEE International Parallel and Distributed Processing Symposium(IPDPS 2003). Nice: [s. n.], 2003: 123－135.

[6] Varianl H R. Microeconomic Analysis[M]. 3rd ed. New York: W W Norton & Company, 1992: 398－401.

[7] 傅晓明,张尧学,马洪军,等. 一种基于市场模型的网络带宽分配方法[J]. 电子学报, 1999(9):127－129.

[8] Foster I, Kesselman C, Tsudik G, et al. A Security Architecture for Computational Grids[C]// Proc. 5th ACM Conference on Computer and Communications Security Conference. Chicago: IEEE Press, 1998: 83－92.

[9] 曹鸿强,肖　侬,卢锡城,等.一种基于市场机制的计算网格资源分配方法[J].计算机研究与发展,2002,39(8):913－916.

[10] Basney, Livny M. Deploying a High Throughput Computing Cluster[M]// Buyya R. High Performance Cluster Computing. Vol. 1. Chpter 5. [s. l.]: Prentice Hall PTR, 1999.