计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
44(7):1177~1183, 2007

频繁闭项目集挖掘算法研究

朱玉全　　宋余庆
(江苏大学计算机科学与通信工程学院　镇江　212013)
(yqzhu@ujs.edu.cn)

Research on an Algorithm for Mining Frequent Closed Itemsets

Zhu Yuquan and Song Yuqing
(*School of Computer Science & Communication Engineering, Jiangsu University, Zhenjiang* 212013)

Abstract　Mining frequent itemsets is a fundamental and essential problem in data mining application. Most of the proposed mining algorithms are a variant of Apriori. These algorithms show good performance with spare datasets. However, with dense datasets such as telecommunications and medical image data, where there are many long frequent itemsets, the performance of these algorithms degrades incredibly. In order to solve this problem, an efficient algorithm MFCIA and its updating algorithm UMFCIA for mining frequent closed itemsets are proposed. The set of frequent closed itemsets uniquely determines the exact frequency of all frequent itemsets, yet it can be orders of magnitude smaller than the set of all frequent itemsets, thus lowering the algorithm computation cost. The algorithm UMFCIA makes use of the previous mining results to cut down the cost of finding new frequent closed itemsets. The experiments show that the algorithm MFCIA is efficient.

Key words　frequent itemsets; frequent closed itemsets; minimum frequent closed itemsets; maximal frequent closed itemsets; incremental updating

摘　要　目前已提出了许多基于 Apriori 算法思想的频繁项目集挖掘算法,这些算法可以有效地挖掘出事务数据库中的短频繁项目集,但对于长频繁项目集的挖掘而言,其性能将明显下降. 为此,提出了一种频繁闭项目集挖掘算法 MFCIA,该算法可以有效地挖掘出事务数据库中所有的频繁项目集,并对其更新问题进行了研究,提出了一种相应的频繁闭项目集增量式更新算法 UMFCIA,该算法将充分利用先前的挖掘结果来节省发现新的频繁闭项目集的时间开销. 实验结果表明算法 MFCIA 是有效可行的.

关键词　频繁项目集;频繁闭项目集;最小频繁闭项目集;最大频繁闭项目集;增量式更新

中图法分类号　TP311.13

发现频繁项目集是关联规则和序列模式等数据挖掘应用中的关键技术和步骤. 近年来,在频繁项目集的算法研究中先后出现了 Apriori[1], AprioriFREQ[2], NFUP[3], MPL[4], FUFIA[5] 等数据挖掘算法,在众多算法中,以 Agrawal 等人提出的 Apriori 算法最为著名,其后的数据挖掘算法大多数建立在 Apriori 算法基础之上,或进行改进,或衍生变种. 在巨大的候选项目集产生时,Apriori 类算法可以取得较好性能,但当挖掘任务有大量强模式、长模式或阈值较低时,该类算法有如下不足:1)算法必须耗费大量的时间处理规模巨大的候选项目集;2)算法必须多次重复扫描数据库,对候选项目集进行模式匹配.

收稿日期:2006-11-24;修回日期:2007-04-03
基金项目:国家自然科学基金项目(60572112)

为此,许多学者提出了最大频繁项目集挖掘算法,如:Pincer-Search[6],Top_Down_Miner[7],DMM[8]等,它们可以有效地发现事务数据库中的最大频繁项目集,由于最大频繁项目集中已经隐含了所有频繁项目集,因此在发现最大频繁项目集的同时,也发现了所有频繁项目集.

我们在图像数据挖掘技术研究中发现,Pincer-Search 等算法并不能完全解决以上两个问题,如某事务数据库中有 M 个频繁项目,最大频繁项目集的最大长度为 $L(L\leqslant M)$,最大频繁项目集的最小长度为 1,则该算法至少需要扫描数据库 M 次,而 Apriori 类算法仅需要扫描 L 次,扫描事务数据库的次数并没有减少. 另外,Pincer-Search 等算法还有一个明显的缺点,最大频繁项目集并没有保存其子集的支持度,为了寻找非最大频繁项目集的支持度,需要执行一次额外的数据库扫描.

频繁闭项目集提供了事务数据库的一个最小描述,其数量介于最大频繁项目集和频繁项目集之间,同时又记录了所有频繁项目集的支持度,因而发现频繁闭项目集对数据挖掘具有十分重要的意义,并引起了国内外许多学者的极大关注. 目前已经提出的可用于发现频繁闭项目集算法有:A-close[9],CHARM[10],CLOSE +[11],DCI-CLOSED[12]等. 这些算法可以归纳为两类:一类是基于 Apriori 算法思想的频繁闭项目集挖掘算法,它们具有与 Apriori 算法同样的不足;另一类是不产生候选集而直接生成频繁闭项目集的挖掘算法,它们首先构造一棵树,然后在树上进行相关操作来发现事务数据库中的频繁闭项目集. 从本质上来讲,这类算法有两个明显不足:一是当项目比较平衡或支持度阈值较低时,无法将树形结构一次性全部装入内存,此时算法不再可行;二是很难实现频繁闭项目集的增量式更新,从而严重限制了该类算法的应用.

本文系统地给出了频繁闭项目集及其挖掘算法,提出了一种有效的频繁闭项目集增量式更新算法,并就频繁闭项目集挖掘中的一些关键技术进行了探讨.

1 频繁项目集和频繁闭项目集

设 $I=\{i_1,i_2,\cdots,i_m\}$ 表示 m 个不同项目的一个集合,事务数据库 D 包含 n 个事务,即 $D=\{T_1,T_2,\cdots,T_n\}$. 项目集 $X\subseteq I$,X 在 D 中的支持数是指 D 中包含 X 的事务数,记为 $count(X)$,X 在 D 中的支持度是指 D 中包含 X 事务的百分比,记为 $sup(X)$. 如果 X 的支持度不小于用户给定的最小支持度阈值 $minsup$,则称 X 为 D 中的频繁项目集,项目集中项目的个数称为项目集的维数或长度,频繁 1-项目集简称频繁项目.

定义 1. 项目集 $X\subseteq I$,事务集 $T\subseteq D$,定义映射关系:

$t:2^I\to 2^T$,$t(X)=\{t\in T\mid$事务 t 支持项目集 $X\}$.

$i:2^T\to 2^I$,$i(T)=\{c\in I\mid T$ 中的任何事务均支持项目 $c\}$.

例 1. 设 $I=\{1,2,3,4,5\}$,$T_1=\{1,2,4,5\}$,$T_2=\{2,3,5\}$,$T_3=\{1,2,4,5\}$,$T_4=\{1,2,3,5\}$,$T_5=\{1,2,3,4,5\}$,$T_6=\{2,3,4\}$. 如项目集 $X=\{2,3,5\}$,则 $t(X)=\{T_2,T_4,T_5\}$,因为 T_2,T_4,T_5 均支持项目集 X,且其他事务均不支持 X. 如事务集 $T=\{T_1,T_4\}$,则 $i(T)=\{1,2,5\}$,因为 T_1 和 T_4 同时支持项目集 $\{1,2,5\}$,且不同时支持其超集.

定义 2. 设项目集 $X\subseteq I$,如果满足 $i(t(X))=X$,且 X 为频繁项目集,则称 X 为最大频繁闭项目集,简称频繁闭项目集.

定义 3. 设频繁项目集 $X\subseteq I$. 对于 X 的任意真子集 Y,均有 $sup(X)<sup(Y)$ 成立,则称 X 为最小频繁闭项目集.

频繁闭项目集挖掘的任务是:在给定的事务数据库 D 中,发现 D 中所有频繁闭项目集.

性质 1. 设项目集 $X,X_1,X_2,X_3\subseteq I$,事务集 $T,T_1,T_2,T_3\subseteq D$,则:

1) 如果 $X_1\subseteq X_2$,则 $t(X_1)\supseteq t(X_2)$;

2) 如果 $T_1\subseteq T_2$,则 $i(T_1)\supseteq i(T_2)$;

3) $i(t(X))\supseteq X$,$t(i(T))\supseteq T$;

4) $i(t(i(t(X))))=i(t(X))$.

证明.

1) 设事务 $t\in t(X_2)$,则 t 支持 X_2. 由于 $X_2\supseteq X_1$,因此 t 支持 X_1,即 $t\in t(X_1)$,$t(X_1)\supseteq t(X_2)$ 成立.

2) 设项目 $c\in i(T_2)$,则 T_2 中的各事务均支持 c,由于 $T_2\supseteq T_1$,因此 T_1 中的各事务也支持 c,即 $c\in i(T_1)$,$i(T_1)\supseteq i(T_2)$ 成立.

3) 设项目 $c\in X$,由于 $t(X)$ 中的各事务均支持 X,因而 $t(X)$ 中的各事务均支持项目 c,即 $c\in i(t(X))$,$i(t(X))\supseteq X$ 成立. 同样可以证明 $t(i(Y))\supseteq Y$ 成立.

4) 令性质 3) 中的 X 为 $i(t(X))$,则有 $i(t(i$

$(t(X))))\supseteq i(t(X))$成立. 令性质3)中的 T 为 $t(X)$,则有 $t(i(t(X)))\supseteq t(X)$,根据性质2),有 $i(t(i(t(X))))\subseteq i(t(X))$成立. 因此,$i(t(i(t(X))))=i(t(X))$. 证毕.

设 FI 为 D 中所有频繁项目集的集合,即 $FI=\{X\subseteq I \mid sup(X)\geqslant minsup\}$,$MaxFCI$ 为 T 中所有最大频繁闭项目集的集合.

性质2. $FI\supseteq MaxFCI$.

显然,性质2是成立的.

性质3. 对于任意频繁项目集 $X\in FI$,则存在最大频繁闭项目集 $Y\in MaxFCI$,满足 $X\subseteq Y$,且 $sup(X)=sup(Y)$.

证明. 分两种情况证明:

1) 如果 X 的所有超集均为非频繁项目集,则 X 本身就是最大频繁闭项目集,令 $Y=X$,性质3成立.

2) 如果存在 X 的超频繁项目集,又可以分为以下两种情况:

① X 的超频繁项目集的支持度均小于 $sup(X)$. 可以证明 X 为最大频繁闭项目集,证明方法为:对于任意项目 $c\in i(t(X))$,如果 $c\notin X$,则 $sup(\{c\}\cup X)=sup(X)$,矛盾,因此 $c\in X$,从而 $i(t(X))\subseteq X$,由性质1中的3),$i(t(X))\supseteq X$,故 $i(t(X))=X$,即 X 为最大频繁闭项目集. 令 $Y=X$,性质3成立.

② 存在 X 的超频繁项目集 Y_1,$sup(Y_1)=sup(X)$,则 X 为非最大频繁闭项目集. 如果 Y_1 为最大频繁闭项目集,令 $Y=Y_1$,性质3成立. 否则,存在 Y_1 的超频繁项目集 Y_2,$sup(Y_2)=sup(X)$,…. 由于 $Y_i\subseteq I$,因此必存在 j,Y_j 为最大频繁闭项目集,且 $sup(X)=sup(Y_1)=\cdots=sup(Y_{j-1})=sup(Y_j)$,令 $Y=Y_j$,性质3成立.

证毕.

由性质2可知,频繁最大闭项目集是频繁项目集的子集;由性质3可知,我们可以把发现所有频繁项目集的问题转化为发现所有最大频繁闭项目集的问题,且最大频繁闭项目集包含了所有频繁项目集的支持度.

例2. 事务数据库如例1所示,假设 $minsup=50\%$,则项目集 $\{3,5\}$ 为 D 中的一个最小频繁闭项目集,因为项目集 $\{3,5\}$ 的支持度均小于其真子集 $\{3\}$ 和 $\{5\}$ 的支持度;项目集 $\{2,3,5\}$ 为 D 中的一个最大频繁闭项目集,因为 $t(\{2,3,5\})=\{T_2,T_4,T_5\}$,$i(\{T_2,T_4,T_5\})=\{2,3,5\}$,即 $i(t(\{2,3,5\}))=\{2,3,5\}$.

2 频繁闭项目集挖掘算法

我们可以将求频繁闭项目集分为两步:第1步是求最小频繁闭项目集;第2步是根据最小频繁闭项目集确定频繁闭项目集.

定理1. 设项目集 $X=\{x_1,x_2,\cdots,x_k\}$ 为最小频繁闭 k-项目集,则其任意 $(k-1)$-项目子集均为最小频繁闭 $(k-1)$-项目集.

证明. 反证法. 假设存在项目集 X 的 $(k-1)$-项目子集 $X_1=\{x_1,x_2,\cdots,x_{i-1},x_{i+1},\cdots,x_k\}$ 为非最小频繁闭 $(k-1)$-项目集,则存在项目集 X_1 的 $(k-2)$-项目子集 X_2,有 $sup(X_2)=sup(X_1)$成立,即 $t(X_2)=t(X_1)$. 令 $X_3=\{x_i\}\cup X_2$,则 $X_3\subset X$,由性质1可知 $t(X_3)\supseteq t(X)$. 对于任意事务 $t\in t(X_3)$,t 支持项目集 X_3 或 $(\{x_i\}\cup X_2)$,t 必支持 $\{x_i\}$ 及 X_2,由于 $t((X_2))=t(X_1)$,因此 t 支持项目集 X_1,又由于 $X=\{x_i\}\cup X_1$,因此 t 支持项目集 X,$t\in t(X)$,故 $t((X_3))\subseteq t(X)$,从而有 $t((X_3))=t(X)$成立,即 $sup(X)=sup(X_3)$. 与项目集 X 为最小频繁闭 k-项目集矛盾. 因此项目集 X 的所有 $(k-1)$-项目子集均为最小频繁闭 $(k-1)$-项目集.

证毕.

根据定理1,我们可以得到最小频繁闭项目集挖掘算法,具体描述见算法1.

算法1. 最小频繁闭项目集挖掘算法.

输入: 事务数据库 D;最小支持度阈值 $minsup$;最小候选项目集数阈值 mc.

输出: 事务数据库 D 中的最小频繁闭项目集.

方法:

① $L_{c1}=\{\text{frequent 1-itemsets}\}$; /* L_{c1} 为最小频繁闭1-项目集 */

② for ($k=2$; $L_{c(k-1)}\neq\varnothing$; k++) do begin

③ $C_k=apriori\text{-}gen(L_{c(k-1)})$; /* $L_{c(k-1)}$ 为最小频繁闭 $(k-1)$-项目集,C_k 为候选集,函数 $apriori\text{-}gen(L_{c(k-1)})$ 的功能参见频繁项目集的 Apriori 算法 */

④ if($|C_k|>mc$) then do begin/* 本语句功能参见算法说明部分 */

⑤ for all transaction $t\in D$ do begin

⑥ $C_t=subset(C_k,t)$;

⑦ for all candidates $c\in C_t$ do

⑧ $count(c)$++;

⑨ end

⑩ $L_{ck}=\{c\in C_k \mid sup(c)\geqslant minsup\}$;

⑪ for all candidates $c\in L_{ck}$ do begin

⑫ for all $(i-1)$-subset s of c do

⑬ if $(sup(s)=sup(c))$ then

⑭ $L_{ck}=L_{ck}-\{c\}$;/*删除 L_{ck} 中的非最小频繁闭项目集,定理 1*/

⑮ end

⑯ end

⑰ end

说明:$|C_k|$ 表示 C_k 中项目集的个数. 如果候选项目集的个数较少(本文设置为小于等于 mc),那么暂不计算 C_k 中各项目集的支持度,其支持度在下次扫描 D 时一起计算,从而不必为了求几个候选项目集的支持度而需扫描 D 一次. 其中. mc 的值视具体情况而定.

例 3. 事务数据库 D 如例 1 所示,假设 $minsup=50\%$. 该事务数据库 D 中的最小频繁闭项目集挖掘过程如下:

1) 扫描 D 一次,找到 D 中的最小频繁闭 1-项目集 $L_{c1}=\{\{1\},\{2\},\{3\},\{4\},\{5\}\}$;

2) 利用 L_{c1} 生成 D 中的候选最小频繁闭 2-项目集 $C_2=\{\{1,2\},\{1,3\},\cdots,\{4,5\}\}$;

3) 扫描 D 一次,计算 C_2 中各项目集的支持数(度);

4) 删除 C_2 中支持度小于 50% 的项目集,得 $C_2=\{\{1,2\},\{1,4\},\{1,5\},\{2,3\},\{2,4\},\{2,5\},\{3,5\},\{4,5\}\}$;

5) 由于 $sup(\{1\})=sup(\{1,5\})$,删除 C_2 中的项目集 $\{1,5\}$,同样删除 C_2 中的项目集 $\{2,3\},\{2,4\},\{2,5\}$,得最小频繁闭 2-项目集 $L_{c2}=\{\{1,2\},\{1,4\},\{3,5\},\{4,5\}\}$;

6) 利用 L_{c2} 生成 D 中的候选最小频繁闭 3-项目集 $C_3=\varnothing$;

7) 由于 $C_3=\varnothing$,本次过程结束. D 中的最小频繁闭项目集为:$\{\{1\},\{2\},\{3\},\{4\},\{5\},\{1,2\},\{1,4\},\{3,5\},\{4,5\}\}$.

由此例可见,确定最小频繁闭项目集远比确定频繁项目集简单、省时. 然而,最小频繁闭项目集中并不包含所有最大频繁闭项目集,因此如何根据最小频繁闭项目集来确定最大频繁闭项目集是下面所要讨论的关键问题.

定理 2. 设 X 为最大频繁闭项目集,则存在最小频繁闭项目集 Y,使 $t(X)=t(Y)$ 成立.

证明. 设项目集 X 的长度为 k,分两种情况证明:

1) 如果 X 的所有 $(k-1)$-项目子集的支持度均大于 X 的支持度,则 X 本身就是最小频繁闭项目集,令 $Y=X$,$t(X)=t(Y)$,定理 2 成立.

2) 如果存在 X 的 $(k-1)$-项目子集 Y_1,$sup(Y_1)=sup(X)$,有 $t(X)=t(Y_1)$ 成立. 如果 Y_1 的所有 $(k-2)$-项目子集的支持度均大于 X 的支持度,则令 $Y=Y_1$,有 $t(X)=t(Y)$ 成立,定理 2 成立;否则,存在 X 的 $(k-2)$-项目子集 Y_2,$sup(Y_2)=sup(X)$,有 $t(X)=t(Y_2)$ 成立,如此下去,必存在 j,$sup(Y_j)=sup(X)$,且 Y_j 为最小频繁闭项目集,令 $Y=Y_j$,有 $t(X)=t(Y)$ 成立,定理 2 成立. 证毕.

定理 3. 设 X 为频繁项目集,则 $i(t(X))$ 为最大频繁闭项目集.

推论 1. 设 Y 为事务数据库 D 中的最小频繁闭项目集,则与 Y 对应的最大频繁闭项目集为 $i(t(Y))$.

推论 2. 设 $MinFCI$ 为事务数据库 D 中最小频繁闭项目集的集合,则 $MaxFCI=\{i(t(Y))\mid Y\in MinFCI\}$.

定理 4. 设项目集 $Y\in MinFCI$,$t(Y)=\{T_{Y_1},T_{Y_2},\cdots,T_{Y_m}\}$,则 $i(t(Y))=\bigcap_{j=1}^{m}T_{Y_j}$.

证明. 根据映射关系 i 和 t 的定义,有 $T_{Y_j}\supseteq i(t(Y))(j=1,2,\cdots,m)$ 成立,因此 $i(t(Y))\subseteq\bigcap_{j=1}^{m}T_{Y_j}$. 另外,如果项目 $c\in\bigcap_{j=1}^{m}T_{Y_j}$,则 $T_{Y_j}(j=1,2,\cdots,m)$ 支持项目 c,因此 $c\in i(t(Y))$,因而,$\bigcap_{j=1}^{m}T_{Y_j}\subseteq i(t(Y))$. 故 $i(t(Y))=\bigcap_{j=1}^{m}T_{Y_j}$. 证毕.

根据推论 2 和定理 4,我们可以得到最大频繁闭项目集的挖掘算法,具体描述如算法 2 所示.

算法 2. 最大频繁闭项目集挖掘算法 MFCIA (maximal frequent closed itemsets algorithm).

输入:D 中所有的最小频繁闭项目集 $MinFCI$;事务数据库 D.

输出:D 中所有的最大频繁闭项目集 $MaxFCI$.

方法:

① for all $Y\in MinFCI$ do

② $M_Y=I$;/*M_Y 用来存放最小频繁闭项目集 Y 所对应的最大频繁闭项目集,初始为全集 I*/

③ for all transaction $t\in D$ do

④ for all $Y\in MinFCI$ do

⑤ if $t\supseteq Y$ then

⑥ $M_Y = M_Y \cap t$; /* 根据定理 4 */

⑦ $MaxFCI = \varnothing$; /* 赋初值 */

⑧ for all $Y \in MinFCI$ do

⑨ $MaxFCI = MaxFCI \cup M_Y$.

例 4. 事务数据库 D 如例 1 所示，$MinFCI = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{1,2\}, \{1,4\}, \{3,5\}, \{4,5\}\}$. 事务数据库 D 中最大频繁闭项目集的挖掘过程如下：

1) 令 $M_{\{1\}} = M_{\{2\}} = M_{\{3\}} = M_{\{4\}} = M_{\{5\}} = M_{\{1,2\}} = M_{\{1,4\}} = M_{\{3,5\}} = M_{\{4,5\}} = \{1,2,3,4,5\}$；

2) 扫描事务数据库 D 一次；

3) 对于 $T_1 = \{1,2,4,5\}$，由于 T_1 支持 $MinFCI$ 中的项目集 $\{1\}, \{2\}, \{4\}, \{5\}, \{1,2\}, \{1,4\}$ 和 $\{4,5\}$，因此修改 $M_{\{1\}}, M_{\{2\}}, M_{\{4\}}, M_{\{5\}}, M_{\{1,2\}}, M_{\{1,4\}}$ 和 $M_{\{4,5\}}$，修改指令为 $M_Y = M_Y \cap \{1,2,4,5\}$，如 $M_{\{1\}} = \{1,2,3,4,5\} \cap \{1,2,4,5\} = \{1,2,4,5\}$，$M_{\{3\}}$ 和 $M_{\{3,5\}}$ 不变；

4) 对于 $T_2 = \{2,3,5\}$，由于 T_2 支持 $MinFCI$ 中的项目集 $\{2\}, \{3\}, \{5\}$ 和 $\{3,5\}$，修改 $M_{\{2\}}, M_{\{3\}}, M_{\{5\}}$ 和 $M_{\{3,5\}}$，其他不变；

5) 对于 $T_3 = \{1,2,4,5\}$，由于 T_3 支持 $MinFCI$ 中的项目集 $\{1\}, \{2\}, \{4\}, \{5\}, \{1,2\}, \{1,4\}$ 和 $\{4,5\}$，修改 $M_{\{1\}}, M_{\{2\}}, M_{\{4\}}, M_{\{5\}}, M_{\{1,2\}}, M_{\{1,4\}}$ 和 $M_{\{4,5\}}$，其他不变，如 $M_{\{1\}} = \{1,2,4,5\} \cap \{1,2,4,5\} = \{1,2,4,5\}$；

6) 对于 $T_4 = \{1,2,3,5\}$，由于 T_4 支持 $MinFCI$ 中的项目集 $\{1\}, \{2\}, \{3\}, \{5\}, \{1,2\}$ 和 $\{3,5\}$，修改相应的 M_Y，如 $M_{\{1\}} = \{1,2,4,5\} \cap \{1,2,3,5\} = \{1,2,5\}$；

7) 对于 $T_5 = \{1,2,3,4,5\}$，由于 T_4 支持 $MinFCI$ 中的项目集 $\{1\}, \cdots, \{4,5\}$，修改相应的 M_Y，如 $M_{\{1\}} = \{1,2,5\} \cap \{1,2,3,4,5\} = \{1,2,5\}$；

8) 对于 $T_6 = \{2,3,4\}$，由于 T_6 支持 $MinFCI$ 中的项目集 $\{2\}, \{3\}$ 和 $\{4\}$，修改相应的 M_Y，如 T_6 不支持项目集 $\{1\}$，因此 $M_{\{1\}}$ 不变，其值仍为 $\{1,2,5\}$；

9) $MaxFCI = M_{\{1\}} \cup M_{\{2\}} \cup M_{\{3\}} \cup M_{\{4\}} \cup M_{\{5\}} \cup M_{\{1,2\}} \cup M_{\{1,4\}} \cup M_{\{3,5\}} \cup M_{\{4,5\}} = \{\{2\}, \{2,3\}, \{2,4\}, \{2,5\}, \{2,3,5\}, \{1,2,4,5\}\}$.

3 频繁闭项目集更新算法

事实上，为了发现事先未知的频繁闭项目集或关联规则，用户必然需要不断地调整最小支持度和最小置信度，这将是一个动态的交互过程. 因此，能否动态地挖掘出事务数据库中的频繁闭项目集是衡量一个算法好坏的关键因素，而目前有关这方面的研究工作却很少. 本节主要考虑事务数据库不变而最小支持度发生变化后频繁闭项目集的更新问题.

设旧的最小支持度为 s，新的最小支持度为 s'，$MinFCI'$，$MaxFCI'$ 分别为支持度 s' 下 D 中的最小频繁闭项目集和最大频繁闭项目集的集合.

定理 5. 如果 $s' < s$，则 $MinFCI \subseteq MinFCI'$.

证明. 对于任意项目集 $Y \in MinFCI$. 由于 Y 为支持度 s 下 D 中的最小频繁闭项目集，因此，$sup(Y) \geqslant s$，并且对于 Y 的任意子集 Y_1，均有 $sup(Y_1) > sup(Y)$. 因为事务数据库 D 的不变性以及 $s' < s$，因此 $sup(Y) \geqslant s'$，$sup(Y_1) > sup(Y)$，故 Y 为支持度为 s' 下 D 中的最小频繁闭项目集，即 $Y \in MinFCI'$. 证毕.

定理 6. 如果 $s' > s$，则 $MaxFCI' \subseteq MaxFCI$.

证明. 对于任意项目集 $X \in MaxFCI'$，则 $sup(X) \geqslant s' > s$，即 $sup(X) \geqslant s$，X 为支持度 s 下的频繁项目集. 另一方面，对于 X 的任意超集 X_1，有 $sup(X_1) < sup(X)$ 成立，因此 X 的任意超集在支持度 s 下的支持度均小于项目集 X 的支持度，故 X 为支持度 s 下的最大频繁闭项目集，即 $X \in MaxFCI$，$MaxFCI' \subseteq MaxFCI$. 证毕.

由定理 5 和定理 6 可知，如果 $s' > s$，删除 $MaxFCI$ 中支持度小于 s' 的项目集即可得到支持度 s' 下 D 中所有的最大闭频繁项目集 $MaxFCI'$；如果 $s' < s$，$MinFCI$ 中的最小闭频繁项目集仍然有效，因此其关键问题是如何寻找新的最小频繁闭项目集.

算法 3. 最大频繁闭项目集更新挖掘算法.

输入：事务数据库 D；新、旧最小支持度 s'，s；旧的最大频繁闭项目集 $MaxFCI$ 和最小频繁闭项目集 $MinFCI$；最小候选项目集数阈值 mc.

输出：支持度 s' 下 D 中所有的最大频繁闭项目集 $MaxFCI'$.

方法：

① if ($s' > s$) then

② $MaxFCI' = \{X \in MaxFCI \mid sup(X) \geqslant s'\}$；

③ else do begin

④ $L'_{c1} = \{\text{frequent 1-itemsets}\}$; /* L'_{c1} 为最小频繁闭 1-项目集，支持度为 s' */

⑤ $C'_k = apriori\text{-}gen(L'_{c(k-1)})$; /* $L'_{c(k-1)}$ 为最小频繁闭 $(k-1)$-项目集，支持度为 s' */

⑥ $C''_k = C'_k - \{X \mid X$ 为 $MaxFCI$ 中某项目集的子集$\}$;/* C''_k 为 C'_k 中支持度未知的项目集 */

⑦ if($|C''_k| > mc$) then do begin/* 本语句功能参见算法 1 */

⑧ for all transaction $t \in D$ do begin

⑨ $C_t = subset(C''_k, t)$;

⑩ for all candidates $c \in C_t$ do

⑪ $count(c)$ ++ ;

⑫ end

⑬ $L'_{ck} = \{c \in C'_k \mid sup(c) \geqslant minsup\}$;

⑭ for all candidates $c \in L'_{ck}$ do begin

⑮ for all $(i-1)$-subset s of c do

⑯ if $(sup(s) = sup(c))$ then

⑰ $L'_{ck} = L'_{ck} - \{c\}$;

⑱ end

⑲ end

⑳ 调用最大频繁闭项目集挖掘算法,得到最新的频繁闭项目集 $MaxFCI'$;

㉑ end

4 算法分析与比较

将算法 MFCIA 与算法 Apriori 相比,我们可以得出如下结论:

1) 在算法 Apriori 中,扫描 D 的候选项目集为 $C = apriori\text{-}gen(L_{(k-1)})$. 而在算法 MFCIA 中,扫描 D 的候选项目集为 $C = apriori\text{-}gen(L_{c(k-1)})$. 由于 $L_{(k-1)} \supseteq L_{c(k-1)}$,因此,在扫描 D 时,算法 MFCIA 将比算法 Apriori 有更小的候选项目集.

2) 在算法 MFCIA 中,一方面,D 中更小的候选项目集将导致扫描 D 的次数的减少;另一方面,最小候选项目集数阈值 mc 的设置可以进一步减少扫描 D 的次数. 因此,算法 MFCIA 在扫描 D 次数上占一定的优势.

为了进一步验证算法的有效性,我们用 VC++ 6.0 在内存为 512MB、操作系统为 Windows XP 的机上实现了算法 MFCIA 和算法 CLOSE+. 算法 CLOSE+ 是一种典型的基于树形结构的频繁闭项目集挖掘算法,它较 Apriori 算法有很大的优势. 在我们的实验中,使用了与文献[1]同样的生成程序来合成所需要的测试数据,其有关参数与文献[1]相同,在本实验中,$N = 1000$,$|L| = 1000$,$|D| = 10240$,$|T| = 25$,$|I| = 15$. 另外,取 $mc = 3$,实验结果如图 1 所示,图 1 表明算法 MFCIA 是有效的.

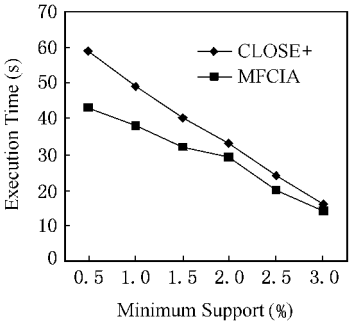

Fig. 1 Execution time of algorithms.

图 1 算法执行时间

5 结束语

频繁闭项目集提供了事务数据库的一个最小描述,其数量介于最大频繁项目集和频繁项目集之间,同时又记录了所有频繁项目集的支持度,因而发现频繁闭项目集对数据挖掘具有十分重要的意义. 本文系统地给出了频繁闭项目集及其挖掘算法,提出了一种有效的频繁闭项目集增量式更新算法,实验表明算法是有效可行的. 另外,本文仅对支持度发生变化后频繁闭项目集的更新问题进行了研究,至于其他更新问题,如事务数据库发生变化等等,我们将对此做进一步的研究.

参 考 文 献

[1] R Agrawal, R Srikant. Fast algorithm for mining association rules [C]. The 20th Int'l Conf on VLDB, Santiago, Chile, 1994

[2] Liu Pei-Qi, Li Zeng-Zhi, Zhao Yin-Liang. Effective algorithm of mining frequent itemsets for association rules [C]. In: Proc of the 3rd Int'l Conf on Machine Learning and Cybernetics. Piscataway, NJ: IEEE Press, 2004. 1447-1451

[3] Chang Chin-Chen, Li Yu-Chiang, Lee Jung-San. An efficient algorithm for incremental mining of association rules [C]. In: Proc of 15th Int'l Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications. Piscataway, NJ: IEEE Press, 2005. 3-10

[4] Xu Yong, Zhou Sen-Xin. Research on the distributed treatment of frequent itemsets extraction based on pruned concept lattices [C]. In: Proc of the 5th Int'l Conf on Machine Learning and Cybernetics. Piscataway, NJ: IEEE Press, 2006. 1332-1336

[5] Zhu Yuquan, Sun Zhihui, Zhao Zhuanshen. Fast updating frequent itemsets [J]. Journal of Computer Research and Development, 2003, 40(1): 94-99 (in Chinese)
(朱玉全,孙志挥,赵传申. 快速更新频繁项集[J]. 计算机研究与发展, 2003, 41(1): 94-99)

[6] Dao-I Lin, Z M Kedem. Pincer-Search: A new algorithm for discovering the maximum frequent set [C]. In: H J Schek, F Saltor, I Ramos, *et al*. eds. Proc of the 6th European Conf on Extending Database Technology. Heidelberg: Springer, 1998. 105-119

[7] Wang Xiaofeng, Wang Tianran, Zhao Yue. An effective top-down data mining method for long frequents [J]. Journal of Computer Research and Development, 2004, 41(1): 148-155 (in Chinese)
(王晓峰, 王天然, 赵越. 一种自顶向下挖掘长频繁项的有效方法[J]. 计算机研究与发展, 2004, 41(1): 148-155)

[8] S M Chung, C Luo. Distributed mining of maximal frequent itemsets from databases on a cluster of workstations [C]. In: Proc of 2004 IEEE Int'l Symp on Cluster Computing and Grid. Piscataway, NJ: IEEE Press, 2004. 499-507

[9] N Pasquier, Y Bastide, R Taouil, *et al*. Discovering frequent closed itemsets for association rules [C]. The 7th Int'l Conf on Database Theory, Jerusalem, Israel, 1999

[10] Z Zaki, C Hsiao. CHARM: An efficient algorithm for closed itemset mining [C]. The 2nd SIAM Int'l Conf on Data Mining, Washington DC, 2002

[11] J Y Wang, J Han, J Pei. CLOSET+: Searching for the best strategies for mining frequent closed itemsets [C]. The 9th ACM SIGKDD Int'l Conf on Knowledge Discovery and Data Mining, Washington, DC, 2003

[12] C Lucchese, S Orlando, R Perego. Fast and memory efficient mining of frequent closed itemsets [J]. IEEE Trans on Knowledge and Dada Engineering, 2006, 18(1): 21-36

Zhu Yuquan, born in 1966. Ph. D. and associate professor, member of China Computer Federation. His main research interests include knowledge discovery in database, complex information system integration, and intrusion detection.

朱玉全,1966 年生,博士,副教授,中国计算机学会会员,主要研究方向为知识发现、复杂信息系统集成和入侵检测等.

Song Yuqing, born in 1959. Ph. D. and professor. His main research interests include data mining, medical image database system, knowledge discovery in database, and pattern recognition.

宋余庆,1959 年生,博士,教授,主要研究方向为数据挖掘、知识发现、图像数据库系统、模式识别等.

Research Background

This paper is supported by the National Natural Science Foundation of China under grant No. 60572112.

Mining frequent itemsets is a fundamental and essential problem in data mining application. Most of the proposed frequent itemsets mining algorithms are a variant of Apriori. These algorithms show good performance with spare datasets such as market basket data, where the frequent itemsets are very short. However, with dense datasets such as telecommunications and medical image data, where there are many long frequent itemsets, the performance of these algorithms degrades incredibly. In order to solve this problem, two methods have been proposed. The first one is to mine only the maximal frequent itemsets, which are orders of magnitude fewer than all frequent itemsets. While mining maximal frequent itemsests help understand the long frequent itemsets in dense domains, maximal frequent itemsets are not suitable for generating rule, and lead to a loss of information. The second is to mine only the frequent closed itemsets. Using the set of frequent closed itemsets, we can also directly generate all frequent itemsets and their exact frequency. At the same time frequent closed itemsets can themselves be orders of magnitude smaller than all frequent itemsets, thus lowering the algorithm computation cost. In this paper, an efficient algorithm and its updating algorithm is proposed, which can mine all frequent closed itemsets in databases or incremental databases. The updating algorithm makes use of the previous mining results to cut down the cost of finding new frequent closed itemsets.