维普资讯 http://www.cqvip.com

基于索引数组和复合频繁模式树的频繁闭项集挖掘算法*)

宋 威[1] 杨炳儒[1] 徐章艳[1,2] 张桃红[1]

(北京科技大学信息工程学院 北京 100083)[1] (广西师范大学计算机系 桂林 541004)[2]

摘 要 频繁闭项集惟一确定频繁项集且规模小得多。CROP是一种基于复合频繁模式树的、频繁闭项集高效挖掘算法,但存在着候选结点过多的问题。这些非闭合结点的生成、检查和剪裁带来了大量不必要的操作。提出了一种改进的频繁闭项集挖掘算法CROP_Index。该算法用"索引数组"来组织数据,找到频繁共同出现的项集。基于二进制位图,给出了一个包含索引的计算方法,并利用索引启发信息合并,得到复合型频繁模式树的初始结点;同时给出一些新的性质,使得改进的算法只生成闭合结点,从而节省了大量不必要的操作,缩小了搜索空间。实验结果表明该算法效率较高。

关键词 数据挖掘,关联规则,频繁闭项集,索引数组,复合频繁模式树

Closed Itemset Mining Algorithm Based on Index Array and Compound Frequent Itemset Tree

SONG Wei[1] YANG Bing-Ru[2] XU Zhang-Yan[1,2] ZHANG Tao-Hong[1]

(School of Information Engineering, Beijing University of Science and Technology, Beijing 100083)[1]

(Department of Computer, Guangxi Normal University, Guilin 541004)[2]

Abstract The set of frequent closed itemsets determines exactly the complete set of all frequent itemsets and is usually much smaller than the latter. Based on compound frequent itemset tree (CFIST), CROP is an efficient algorithm for mining frequent closed itemset. However, there are a lot of redundant candidate nodes of CFIST. Correspondingly, the operations, composed of the generation, closure checking, and pruning of those non-closed nodes, lead to high computational cost. In this paper, CROP_Index, which is an improved algorithm for mining frequent closed itemset, is proposed. Firstly, the "index array" is proposed, which is used for discovering those itemsets that always appear together. Then, based on bitmap, an algorithm for computing index array is presented. Furthermore, frequent items are merged to initial nodes of CFIST according to heuristic information provided by index array. Finally, some new properties, which can avoid the generation of redundant nodes, are proposed. Thus the improved algorithm only generates closed nodes. Correspondingly, the unnecessary operations are avoided, and the search space is reduced to certain extents. The experimental results show that the proposed algorithm is efficient.

Keywords Data mining, Association rule, Frequent closed itemset, Index array, Compound frequent itemset tree

1 引言

频繁项集(或模式)是数据挖掘研究中的一个重要内容。频繁闭项集是所有频繁项集的无损表示。1999年,Pasquier等提出了频繁闭项集的概念,并设计了基于Apriori算法的频繁闭项集挖掘算法A-Close[1],但没有解决重复扫描数据库的问题。Zaki等提出的CHARM算法采用垂直格式的事务标识集来表示项集支持集[2],其困难在于存储开销大,投影操作效率不高。Lucchese等提出的DCI_Closed算法[3]引入了生成子保序的概念,提高了算法的效率。基于FP-Growth,Pei等,Wang等先后提出了CLOSET[4]和CLOSET+[5]。二者通过深度优先搜索来挖掘频繁闭合模式,其困难是,递归构造"条件FP-Tree"的CPU开销和存储开销很大。刘君强等提出了基于复合型频繁模式树的频繁闭合模式挖掘算法CROP[6]。缪裕青给出了共享存储器模型上基于频繁模式树的频繁闭项集并行挖掘算法[7]。

本文是CROP算法的改进。尽管CROP算法利用复合结点压缩了搜索空间,基于"局部剪裁"和"分支包容关系检查"进行剪枝,但还要剪裁掉许多候选结点。这些结点的生成、检查,以及剪裁带来了大量不必要的操作。为尽可能减少这种无用且费时的操作,本文提出了一种改进的频繁闭项集挖掘算法CROP_Index。该算法用"索引数组"来组织数据,通过为每个项目增加包含索引,找到频繁共同出现的项集。基于二进制位图,给出了一个包含索引的计算方法,并利用索引启发信息合并,得到复合频繁模式树的初始结点;同时,给出一些新的性质,使得改进的算法只生成闭合结点,从而节省了大量不必要的操作,缩小了搜索空间。实验结果表明该算法效率较高。

2 问题描述

定义1 数据源$D=(T,I,R)$,其中T是数据库D中事务的有限集合,I是项的有限集合,$R\subseteq T\times I$是事务与项间的二元关系。二元组$(t,i)\in R$表示事务$t\in T$具有项$i\in I$。

定义2 公共项集映射$f:2^T\rightarrow 2^I$,$f(T)=\{i\in I\mid \forall t\in$

*)基金项目:国家科技成果重点推广项目计划(2003EC000001)资助。宋 威 博士研究生,研究方向为数据挖掘;杨炳儒 教授,博士生导师,研究方向为知识发现与智能系统,柔性建模与集成技术;徐章艳 博士研究生,讲师,研究方向为粗糙集理论及其应用,数据挖掘;张桃红 博士,讲师,主要研究领域为数据可视化仿真与数据挖掘。

维普资讯 http://www.cqvip.com

$T, i \in t\}$，称 $f(T)$ 是 $T \subseteq T$ 的公共项集。

TID	Items
1	A B C D E F
2	A E G
3	B C E F
4	A B C D F
5	B C E F

图 1　示例数据集

项集 I 是 I 的子集。若项集 I 由 k 个项目组成，则称 I 为 k-项集，而公共项集则是事务子集 $T \subseteq T$ 中普遍存在的项的集合，如图 1 所示，事务 1 和 4 组成子集的公共项集是 $\{A, B, C, D, F\}$。

定义 3　投影映射 $g: 2^I \to 2^T$，$g(I)=\{t \in T \mid \forall i \in I, i \in t\}$，称 $g(I)$ 为项集 $I \subseteq I$ 的支持集。

定义 4　2^I 上闭包算子 $h=f \circ g$，$h(I)=f(g(I))$，$C \subseteq I$ 是闭项集，当且仅当 $h(C)=C$。

定义 4 也可写作，项集 X 是闭项集，如果不存在与 X 支持度相等的 X 的超集[2~5]。在图 1 所示的数据集中，因为 $h(\{B,C,F\})=\{B,C,F\}$，所以 $\{B,C,F\}$ 是闭项集。

定义 5　$I \subseteq I$ 的绝对支持度为 $\text{support}(I)=|g(I)|$，相对支持度为 $|g(I)|$ 除以 $|T|$ 的值。

定义 6　闭项集 $C \subseteq I$ 是频繁的，如果 $\text{support}(C) \geqslant \text{minsup}$。

3　复合型频繁模式树及其生成

定义 7[6]　给定数据源 $D=(T, I, R)$，项目排序，最小支持度 minsup，复合型频繁模式树 CFIST(compound frequent item set tree)由结点集 V 和有向边集 E 组成。每个结点 $v \in V$ 记作五元组 (i, w, A, O, I)，其中 $i \in I$ 是用于标识该结点的项目，记作 $v.item$；w 是根到 v 路径所表示模式的支持度，称为权重，记作 $v.weight$；$A \subset I$ 是附加项目集(*auxiliary item set*)，用于表征复合结点，记作 $v.AIS$，其中 $A=\{s \in I \mid i \prec s, g(\{i_0, i_1, \cdots, i\} \cup \{s\})=g(\{s\})\}$，$i_0, i_1, \cdots, i$ 分别是从根结点到该结点的标识项目；$O \subseteq T$ 是根到 v 路径所表示模式的支持集，称为投影事务子集(projected transaction subset)，记作 $v.PTS$；$I \subseteq I$ 是后代结点中出现的项目的集合，称为局部项目子集(item list)，记作 $v.IL$。结点 p 到 c 的有向边记为 $\langle p, c\rangle \in E$，$p$ 是父母，c 是子女。任意结点各子女标识项目排列和任意路径上标识项目排列符合 $\prec$。

推论 1[6]　给定 CFIST 结点 $p=(i_p, w_p, A_p, O_p, I_p)$ 是 $c=(i_c, w_c, A_c, O_c, I_c)$ 的父母，则 $i_c \in I_p$，$w_c=|O_c| \geqslant minsup$，$A_c \square I_p$，$O_c=g(\{i_c\}) \cap O_p=g(\{i_c\} \cup Ac) \cap O_p$，$I_c=\{i \in I_p \mid i_c \prec i \wedge i \square A_p\}$。

定义 8[6]　在从根结点到任意结点 v 的路径上，各结点标识项目的集合称为特征项目集，记作 $kis(v)$。kis 并上各结点附加项目集称为完全项目集，记作 $fis(v)$。

对于结点 v，$kis(v)$ 并上附加项目集的任意子集表示一个频繁模式，v 的权重是模式的绝对支持度。一条具有复合结点的路径压缩表达了多个频繁模式。CFIST 的生成实际上是确定各结点所表示的模式支持集的过程。任意结点 p 支持集 p。PTS 由所有支持 $fis(p)$ 的事务组成。根结点 PTS 就是 T，其他结点 PTS 由父亲 PTS 投影而来。

4　索引数组及其生成

定义 9　给定数据源 $D=(T, I, R)$，索引数组 $index[]$ 由一组三元组组成 $(item, subsume, flag)$，其中 $item$ 是项集，$flag$ 是用于标示 $item$ 是否为闭项集的布尔变量(初始值均为 0)，$subsume(i)=\{j \in I \mid j \neq i \wedge g(i) \subseteq g(j)\}$ 是 $item$ 的包含索引。

如在图 1 中，D 的包含索引为 $\{A, B, C, F\}$。

引理 1　若项集 $X \subseteq Y$ 且 $\sup(X)=\sup(Y)$，则 $g(X)=g(Y)$。

证明：由定义 3 可知，函数 g 是单调递减的，故当 $X \subseteq Y$ 时，$g(Y) \subseteq g(X)$。又因为 $\sup(X)=\sup(Y)$，故 $|g(X)|=|g(Y)|$，所以 $g(X)=g(Y)$。

定理 1　若 1-项集 X 的包含索引为 $\varnothing$，则 X 为闭项集。

证明：假设 X 不是闭项集，则根据定义，至少存在项集 Y，使得 $X \subset Y$，且 $\sup(X)=\sup(Y)$，由引理 1 可知：$g(X)=g(Y)$。令 $Z=Y \backslash X \neq \varnothing$，则至少存在一个 1-项集 $i \in Z$，使得 $g(X)=g(Y) \subseteq g(Z) \subseteq g(i)$，故根据定义 9 可知，$isubsume(X)$，这与假设矛盾，故定理 1 得证。

算法 1　计算索引数组

(1)计算全部 1-项集的支持度，删除非频繁项目；
(2)利用二进制位图表示数据集；
(3)for(j=1; j<=|FI|; j++)　//FI 为 1-频繁集的数量
(4)　$v(i_j)=\bigcap_{t \in g(i_j)} t$；
(5)　index[j]。subsume=$v(i_j)$中结果为 1 的位所对应的 1-项集(除 i_j 以外)；
(6)输出索引数组。

以图 1 所示的数据集为例，算法 1 首先扫描一遍数据集，得到所有 1-项集的支持度。设最小支持度为 2，删除非频繁项目 G。利用二进制位图表示数据集，如图 2 所示。

	A	B	C	D	E	F
1	1	1	1	1	1	1
2	1	0	0	0	1	0
3	0	1	1	0	1	1
4	1	1	1	1	0	1
5	0	1	1	0	1	1

图 2　示例数据集的位图表示

接下来逐个求每个频繁 1-项集的包含索引。以 D 为例，$v(D)=\bigcap_{t \in g(D)} t=1 \cap 4=111111 \cap 111101=111101$，其中 1 和 4 分别表示第 1 条和第 4 条事务；在结果中，共有 5 个 1，除了第 4 位对应 D 自身外，其余为 1 的位所对应的 1-项集分别为 A，B，C 和 F，它们便构成了 D 的包含索引，即 $\{A, B, C, F\}$。类似可求出其余 1-项集的包含索引。最终得到的频繁 1-项集的索引数组为 $((A, \varnothing, 0), (B, \{C, F\}, 0), (C, \{BF\}, 0), (D, \{A, B, C, F\}, 0), (E, \varnothing, 0), (F, \{B, C\}, 0))$。

5　CROP_Index 算法

5.1　初始结点的生成

复合型频繁模式树把每个 1-频繁集均作为树的第一层节点(初始结点)，若它们不闭合，则需要判断再剪裁，带来了大量不必要的操作，我们认为可直接把闭项集作为初始结点。

定义 10[3]　若 Y 是一个闭项集，1-项集 $i \notin Y$，则生成子 $X=Y \cup i$ 是保序的，当且仅当 $h(X)=X$ 或者 $i \prec (h(X) \backslash X)$。

定理 2[3]　对每个闭项集 $\overline{Y} \neq h(\varnothing)$ ($h(\varnothing)$ 是指包含在所

维普资讯 http://www.cqvip.com

有事务中的项集，若这样的项集不存在，则$=h(\emptyset)=\emptyset$)，存在一个由 n 个 1-项集组成的序列 $i_0 \prec i_1 \prec \cdots \prec i_{n-1}$，$n \geq 1$，使得

$\langle gen_0, gen_1, \cdots, gen_{n-1} \rangle = \langle Y_0 \cup i_0, Y_1 \cup i_1, \cdots, Y_{n-1} \cup i_{n-1} \rangle$

其中，gen_i 是保序生成子，且 $Y_0 = h(\emptyset)$，$\forall j \in [0, n-1]$，$Y_{j+1} = h(Y_j \cup i_j)$，$Y_n = \overline{Y}$。

推论 2[3] 对每个闭项集$\overline{Y} \neq h(\emptyset)$，由定理 2 所确定的保序生成子序列是唯一的。

定理 3 对任意 1-频繁集 i_k，若 $subsume(i_k) \neq \emptyset$，则 i_k 所对应的结点 $n_k = (i_k, w_k, A_k, O_k, I_k)$无需出现在 $CFIST$ 的第一层。

证明：由于 $subsume(i_k) \neq \emptyset$，由定理 1 可知，$i_k$ 不闭合。

若 n_k 无子女结点，由 CFIST 的构造可知，n_k 将被剪裁；若 n_k 有子女结点，由定理 2 可知，不存在 1-频繁集 $i_j \in I_k$，使得 $i_k \cup i_j$ 保序，故 n_k 的子女结点中不可能产生闭项集。定理 3 得证。

比如，在图 1 所示的数据中，由定理 1 可知，D 不是闭项集，故无需生成初始结点 D。

由定理 1 可找出 1-频繁闭项集。此外，还可依据如下的定理 4 将多个 1-频繁集合并得到频繁闭项集，作为 CFIST 的初始结点，从而减少了初始结点的数量。

定理 4 设 X 是一个项集，$i \in subsume(X)$，且 $\sup(X) = \sup(i)$，则 $i \in h(X)$。

证明：因为 $\sup(X) = \sup(i)$，由定义 9 和引理 1 可知 $g(X) = g(i)$，故 $g(X \cup i) = g(X) \cap g(i) = g(X)$。那么，$f(g(X \cup i)) = f(g(X))$，故 $h(X \cup i) = h(X)$，从而 $i \in h(X)$。

图 1 所示数据经过定理 4 合并后，可得 index [] = ((BCF,{C,F},1), (A,∅,1), (D,{A,B,C,F},0), (E,∅,1))。

5.2 CROP_Index 算法

在 CROP_Index 算法中，我们利用如下定理避免生成不必要的结点。

定理 5 给定项集 i_k，若存在 CFIST 结点 p，使得 $p.weight = support(i_k)$，且 $i_k \subseteq fis(p)$，则以 i_k 为标识项目的结点无需生成。

证明：定理 5 显然成立。

算法 2 CROP_Index 算法

```
CROP_Index(n)
//n 为 CFIST 的结点。对根结点，n 为 ∅，n. IL(即五元组中的 I)为索引数组中 flag 为 1 的项集
for all items ik∈n. IL do
  if is_gen(ik,n)then
    创建结点 nk=(ik,wk,Ak,Ok,Ik);
    输出 fis(n)∪fis(nk)以及该闭项集的支持度;
    if (sup(kis(nk)>minsup)and (nk. IL<>∅)then
      CROP_Index(nk);
    end if
  end if
end for
end CROP_Index
function is_gen (c,p) //c 为子结点，p 为父母结点
  if p=∅ then
    return true;
  else
    if 存在 CFIST 结点 p，使得 p. weight=support(ik)，且 ik⊆fis(p)then
      return false;
    end if
  end if
end function
```

5.3 CROP_Index 算法与 CROP 算法的比较

与文[6]提出的 CROP 算法相比，CROP_Index 算法利用定理 1 和定理 4，只把闭项集作为复合型频繁模式树的初始结点（第一层结点），而不是把每个 1-项集都作为初始结点，从而缩小了搜索空间，减少了不必要结点的生成、检查和剪裁操作，提高了算法的效率。在图 1 所示数据集上分别执行 CROP 和 CROP_Index，其搜索空间的比较如图 3 所示（虚线框代表非闭合结点）。

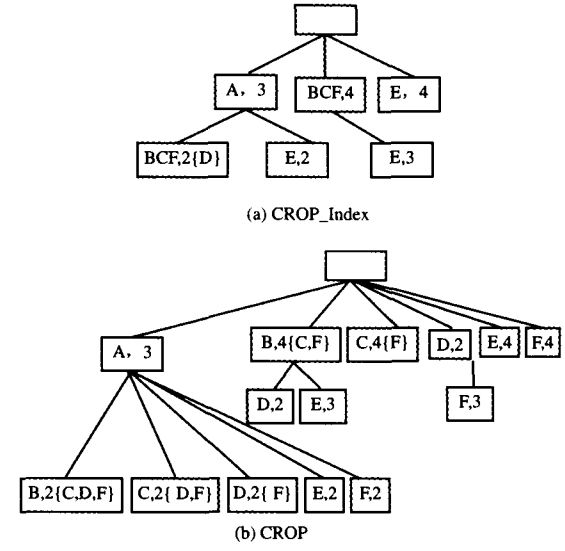

图 3 示例数据集上 CROP_Index 与 CROP 的搜索空间比较

6 性能评测

性能测评的硬件平台是 Pentium IV 2.8GHz CPU，512MB Memory，操作系统是 Windows 2003 Server，对比算法是 CROP，所使用的数据集及其特性如表 1 所示，其中前三个为真实的稠密数据集，而 T10I4D100K 为人工合成的稀疏数据集，这些数据集可由 FIMI（频繁项集挖掘实现）资源库下载[8]。

表 1 数据集的特性

Datasets	# Items	# Records	Avg. Length
Chess	75	3,196	37
Connect	129	65,557	43

性能对比指标是各算法在不同数据集、不同支持度下的运行时间。图 4 为 CROP 与 CROP_Index 运行时间的比较。

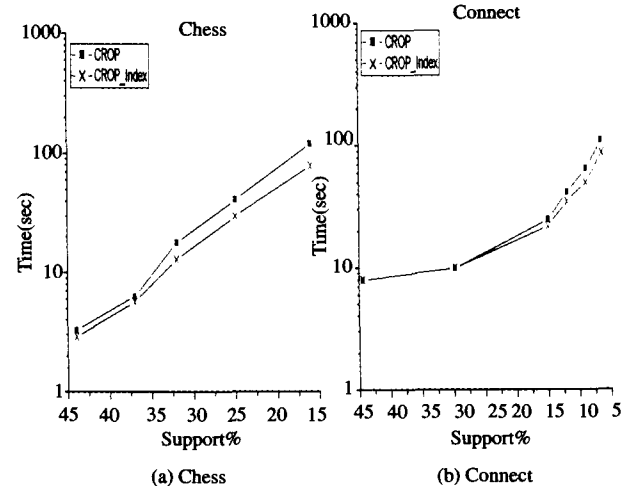

图 4 不同算法在 4 个数据集上执行时间的比较

（下转第 189 页）

维普资讯 http://www.cqvip.com

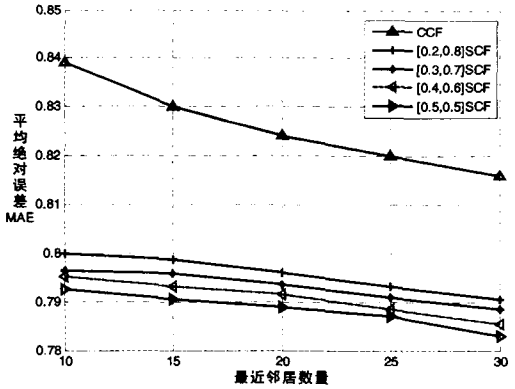

图 1 共同评分项目阈值固定为 20 时推荐精度比较

在进一步实验中，我们将目标用户的最近邻居数量 l 设定为 20。计算用户之间相似性时，共同评分项目数量阈值从 10 变化到 30，间隔为 5，再次进行对比实验，结果如图 2 所示。

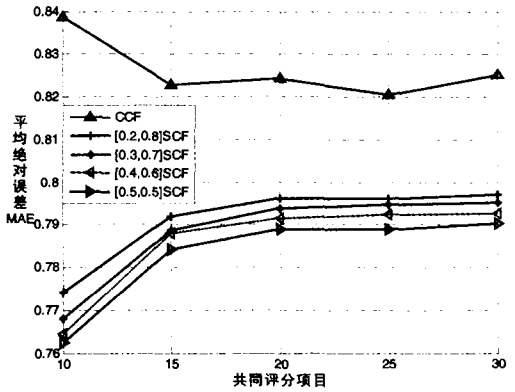

图 2 最近邻居阈值固定为 20 时推荐精度比较

由实验结果可以看出，在相同条件下，一般而言，基于多层相似性的协同过滤算法要的 MAE 均小于其他对比算法。同时，计算多层相似性时，我们特意变化了权重的 α 选取，实验结果也表明，推荐精度受其影响不大。也就是说，多层相似度对 α 的选取没有特别苛刻的要求。

传统的相似性计算方法仅仅考虑用户对各个项目的实际评分，并没有将项目之间的关系考虑进去，因而即使用户具有相似的喜好，由于没有对相同的项目进行评分，使用传统的相似性计算方法获得的结果并不将它们视为最近邻，这将影响推荐系统中实际最近邻的查找结果，进而影响推荐质量。多层综合相似性不仅考虑了用户对基本项目的评分情况，而且考虑了用户对项目类别的评分，即其对项目类别的喜欢程度，因而能够避免传统协同过滤系统中“相似不相同”的问题，同时能够获得更好的推荐质量。

结束语 针对协同过滤推荐系统的稀疏性，本文提出了一种新的用户相似性度量方法，在推荐系统项目集合中引入概念分层，并综合计算用户相似性。通过运用新的相似性度量方法，不仅缓解了系统数据稀疏性问题，而且避免了“相似不相同”的问题，同时实验表明新的相似性度量方法能够提高系统推荐质量。

参 考 文 献

1 Balabanovic M, Shoham Y. Fab: Content-Based, Collaborative Recommendation. Communications of ACM, 1997, 40(3): 66～72
2 Mooney R J, Roy L. Content-Based Book Recommending Using Learning for Text Categorization. In: Proceeding of ACM SIGIR'99 Workshop Recommender Systems: Algorithms and Evaluation, 1999
3 Pazzani M, Billsus D. Learning and Revising User Profiles: The Identification of Interesting Web Sites. Machine Learning, 1997, 27: 313～331
4 Goldberg D, Nichols D, O ki BM, et al. Using collaborative filtering to weave an information Tapestry. Communications of the ACM, 1992, 35(12): 61～70
5 Resnick P, Iacovou N, Suchak M, et al. GroupLens: An open architecture for collaborative filtering of Netnews. In: Proceeding of ACM 1994 Conference on Computer Supported Cooperative Work, 1994. 175～186
6 Breese J S, Heckerman D, Kadie C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In: Fourteen Conference on Uncertainty in Artificial Intelligence, 1998
7 Sarwar B, Karypis G, Konstan J, et al. Item-based collaborative filtering recommendation algorithm. In: Proceedings of the 10th International World Wide Web Conference, 2001. 285～295
8 Ha V, Haddawy P. Toward Case-Based Preference Elicitation: Similarity Measures on Preference Structures. In: Proceedings of 14th Conference on Uncertainty in Artificial Intelligence, 1998. 193～201
9 Shardanand U, Maes P. Social Information Filtering: Algorithms for Automating "Word of Mouth". In: Proceedings of ACM Conference on Human Factors in computing Systems, 1995. 210～217
10 Vozalis E V, Margaritis K G. Recommender systems: An experimental comparison of two filtering algorithms. http://macedonia.uom.gr/～mans/papiria/voz-epy9.pdf. (2006-07-06)
11 Han J, Kamber M. Data mining concepts and techniques [M]. Beijing: High Education Press, 2001. 87～93, 232～236
12 Leung C W, Chan S C, Chung F. A collaborative filtering framework based on fuzzy association rules and multiple-level similarity. Knowledge Information Systems, 2006, 9(4): 492～511
13 Kim C, Kim J. A recommendation algorithm using multi-level association rules. In: Proceedings of the IEEE/WIC International Conference on Web Intelligence, 2003

（上接第 167 页）

实验结果表明，算法 CROP_Index 在效率上优于 CROP。

结论 本文通过对 CROP 算法进行改进，提出了挖掘频繁闭项集的高性能算法 CROP_Index。该算法用“索引数组”来组织数据，通过为每个项目增加包含索引，找到频繁共同出现的项集。基于二进制位图，给出了一个包含索引的计算方法，并利用索引启发信息合并，得到复合型频繁模式树的初始结点；同时，给出一些新的性质，使得改进的算法只生成闭合结点，从而节省了大量不必要的操作，缩小了搜索空间。实验结果表明，CROP_Index 在效率上优于 CROP。

参 考 文 献

1 Pasquier N, Bastide Y, Taouil R, Lakhal L. Discovering Frequent Closed Itemsets for Association Rules [C]. In: Beeri C, Buneman P, eds. Proceedings of 7th International Conference Database Theory. Jerusalem: Springer-Verlag, 1999. 398～416
2 Zaki MJ, Hsiao CJ. CHARM: An Efficient Algorithm for Closed Itemset Mining [C]. In: Grossman R, et al, eds. Proceedings of the 2nd SIAM International. Conference on Data Mining. Arlington: SIAM, 2002. 12～28
3 Lucchese C, Orlando S, Perego R. Fast and Memory Efficient Mining of Frequent Closed Itemsets [J]. IEEE Transaction on Knowledge and Data Engineering, 2006, 18(1): 21～36
4 Pei J, Han J, Mao R. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets [C]. In: Gunopulos D, Rastogi R, eds. Proceedings of 2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. Dallas: ACM Press, 2000. 21～30
5 Wang J Y, Han J, Pei J. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets [C]. In: Getoor L, Senator T E, Domingos P, Faloutsos C, eds. Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2003. 236～245
6 刘君强，孙晓莹，庄越挺，潘云鹤. 挖掘闭合模式的高性能算法 [J]. 软件学报，2004，15(1)：94～102
7 缪裕青. 频繁闭合项目集的并行挖掘算法 [J]. 计算机科学，2004，31(5)：166～168
8 Frequent Itemset Mining Implementations Repository. http://fimi.cs.helsinki.fi/