第26卷 第3期
2011年6月

郑州轻工业学院学报(自然科学版)
JOURNAL OF ZHENGZHOU UNIVERSITY OF LIGHT INDUSTRY(Natural Science)

Vol.26 No.3
Jun. 2011

文章编号:1004-1478(2011)03-0012-06

一种基于后缀树的简洁关联规则挖掘有效剪枝方法

王秉政, 苏晓珂, 张素智

(郑州轻工业学院 计算机与通信工程学院, 河南 郑州 450002)

摘要:针对基于闭模式简洁关联规则挖掘中搜索空间和计算量过大、挖掘有效性降低的问题,提出一种新的剪枝和挖掘方法.该方法通过充分利用挖掘数据结构,保留尽可能少的挖掘后缀信息,直接挖掘简洁项集并枚举简洁关联规则;不需要保留大量中间结构和重复扫描数据库,可在较小代价下完成对项集搜索空间进行剪枝.实验表明,相比传统方法本方法更加有效,并对各种数据集具有较好的可规模性.

关键词:关联规则;闭模式;数据挖掘;有效剪枝方法;后缀树

中图分类号:TP391 **文献标志码**:A

An effective search space pruning method for concise associaiton rules mining based on suffix-tree

WANG Bing-zheng, SU Xiao-ke, ZHANG Su-zhi

(*College of Comp. and Com. Eng., Zhengzhou Univ. of Light Ind., Zhengzhou* 450002, *China*)

Abstract: For itemset search space is too huge which increases concise association rule based on closed itemset calculation and decreases mining efficiency, a new search space pruning method was proposed. It exploits suffix-tree like structure and just keeps as little as tail information. It can prune search space with little cost and mine concise itemsets and rules directly. Additionally, it avoids scanning original database recursively and achieves good performance. Experimental results show it is an effective method for concise association rule mining and has good scalibility.

Key words: association rule; generator; data mining; effective search space pruning method; suffix-tree

0 引言

数据挖掘提供了可以分析大规模数据的方法[1-2],然而随着数据量的逐渐变大,在某些应用中又会显得力不从心.选择更加简洁高效的数据表示方法和挖掘方法,是解决大数据量挖掘任务的需要.

频繁模式的挖掘是关联规则、序列模式、相关分析等许多重要数据挖掘任务的基础.在这方面已有大量的研究工作,并且出现了一些挖掘全部频繁模式的高效算法,如 Apriori, Tree-Projection, FP-Growth, OP, AFOPT, LIFPG 等[3-8].此类方法广泛采用高效的剪枝策略,其中有这样的启发式思想:若

收稿日期:2011-04-05

基金项目:河南省科技攻关项目(092102210108);郑州轻工业学院博士科研基金项目(2008BSJJ010)

作者简介:王秉政(1978—),男,河南省濮阳市人,郑州轻工业学院讲师,博士,主要研究方向为数据挖掘、智能信息处理.

一个项集是非频繁的,则它的任何超集也是非频繁的;或者说若一个项集是频繁的,则它的任何子集也是频繁的.尽管这些剪枝策略取得了比较好的效果,但对于1个包含长度为 m 模式的数据库来说,需要考察的子集有 2^m-2 个.更不用说其他基于频繁模式的数据挖掘任务的规模,基于此产生关联规则的个数将很庞大.当 m 逐渐变大时,此类挖掘任务的效率和有效性将大大降低.

闭模式表示方法是一种有趣的选择,它能确定所有频繁项集信息,但包含的模式的个数要比后者呈数量级减小.挖掘基于闭模式的关联规则和基于频繁模式的关联规则具有相同的功效.由于闭模式表示方法简洁,基于闭模式产生的关联规则数量大大减少,在数据库中维护规则的时空开销也大大降低,相关的挖掘任务效率也有较大提高.然而根据闭模式的性质,传统挖掘方法不可避免地采用通过保存频繁模式并不断扫描原始数据和过滤候选闭模式的方法,由于频繁模式数量庞大,且包含大量冗余操作,挖掘方法有效性降低.

本文重新考察关联规则挖掘的2个阶段,在采用类似于树结构的后缀树的基础上,在进行频繁模式搜索空间遍历时,对每个可能会产生候选闭模式的项进行判断,并对搜索空间提前剪枝,以避免较大候选的产生.把挖掘过的节点链接为后缀,故不用重新生成后缀链,且整个树结构在挖掘过程中由于不断被删除回收而变得越来越小.同时在搜索空间遍历时进行规则的枚举,使得关联规则挖掘的2个阶段在同一搜索空间中进行,可提高关联规则枚举的剪枝效率.

1　问题描述

设项的集合 $I=\{x_1,\cdots,x_M\}$,1个事务数据库表示为 $TDB=\{T_1,T_2,\cdots,T_N\}$,其中 $T_i\subseteq I$ 是1个事务.项的集合被称为1个项集或模式.事务 T 包含项集 X 当且仅当 $X\subseteq T$.在事务数据库 TDB 中,包含项集 X 的事务的个数称为 X 在 TDB 中的支持计数,记作 $count(X)$,X 的支持度表示为 $support(X)=count(X)/N$,其中 N 是 TDB 中事务的个数.给定的事务数据库 TDB 和最小支持度阈值 min_sup,项集 X 是频繁的当且仅当 $support(X)\geqslant min\_sup$.

通常,最小支持度阈值 min_sup 由用户或领域专家指定,而最小支持度计数 $min\_count=[min\_sup\times N]$.在事务数据库中挖掘频繁模式的问题可以描述为:给定事务数据库 TDB 和最小支持度阈值 min_sup,挖掘所有的频繁模式.

定义1　给定1个事务数据库 TDB 和最小支持度阈值 min_sup,项集 X 是闭模式,当且仅当以下2个条件成立:

1)项集 X 是频繁的;

2)$\forall Y\supset X$,满足 $support(Y)<support(X)$.

通过以上闭模式的定义,可以得到以下引理.

引理1　对于项集 X,任意 $\exists Y\supset X$,$support(Y)=support(X)$,则项集 X 不为闭模式.

证明　从闭模式定义可得,略.

引理2　对于项集 X,$Y(Y\supset X)$,$support(Y)=support(X)$,则对任意项集 $Z(Z\supset X,\ Y\not\subset Z)$,项集 Z 不为闭模式.

证明　从引理假设可得,任何包含项集 X 的事务均包含项集 Y.

又 $Z\supset X$,$Y\not\subset Z$,故任何包含项集 $Z(Z\supset X)$ 的事务必包含项集 $Z\cup Y$,即 $support(Z\cup Y)=support(Z)$,且 $Z\cup Y\supset Z$,从引理1可得结论.

定义2　给定一个事务数据库 TDB 和最小支持度阈值 min_sup,当 X 和 $X\cup Y$ 为闭模式时,规则 $X\Rightarrow Y$ 为基于闭模式表示的简洁关联规则.

2　支持度降序后缀树

不失一般性,假定项集 I 上存在1个偏序,记作 $<$.有许多方法定义 $<$,但通过经验发现当以项的关联性次序,特别以项的支持度降序挖掘时,效果较好.故定义 $<$ 为项的支持度降序.

设 $T=\{i_1,i_2,\cdots,i_m\}(i_1<i_2<\cdots<i_m)(1\leqslant m)$ 是一个事务.称项集 $\{j_1,j_2,\cdots,j_n\}\subseteq T(1\leqslant n\leqslant m,\ j_1<j_2<\cdots<j_n)$ 为事务 T 的约束.项集 $\{i_1,i_2,\cdots,i_k\}\subseteq T(1\leqslant k\leqslant m-n,\ i_k<j_1<i_{k+1})$ 称为 $\{j_1,j_2,\cdots,j_n\}$ 约束下的约束项集.

定义3　支持度降序后缀树(DSST)是一棵线索树,定义如下:

1)支持度降序后缀树包含1个根节点,用 *null* 标记.树的每个节点以一组子树作为根的子女.子树不包含根节点.具有较大支持度项的节点,比较接近树根.

2)子树的每个节点包含以下字段:*item*,*count*,*horizontal-link*,*suffix*.其中,*item* 记录该节点代表的项;*count* 是节点计数,如果项集 $\{i_1,\cdots,i_k\}$ 依次出现

在从根到该节点的路径上,则 *count* 等于所有以$\{i_1, \cdots, i_k\}$为前缀的约束项集的个数;*horizontal* 是一个指针,或者指向树中具有相同 *item* 值的下一个节点,或者为空(如果没有下一个节点).

3)*suffix* 字段是指向后缀链的指针,链中包含了出现在当前考察项集所出现的事务的项. 此字段初始设置为空.

4)支持度降序后缀树有序性:如果节点 M 是节点 N 的父母节点,则 $M.item < N.item$;如果节点 N_1 和 N_2 都是节点 M 的子女,并且 N_1 在 N_2 的左边,则 $N_1.item < N_2.item$.

5)支持度降序后缀树是一棵线索树. 具有相同 *item* 值的节点被 *horizontal* 指针链接在一起,形成节点链. 有 2 个数组 *link-head* 和 *totalcnt*: *link-head*[*item*]指向树中具有 *item* 值的第 1 个节点,而 *totalcnt*[*item*]记录对应节点链中节点的计数和.

6)支持度降序后缀树随着挖掘的深入,不断进行调整,形成新的树叶节点和后缀节点链. 同时更新和设置每个树叶节点的后缀.

在事务数据库 *TDB* 中,把项集$\{j_1, j_2, \cdots, j_n\}$ $(n \geq 1)$ 约束下的所有约束项集$\{i_1, i_2, \cdots, i_k\}$利用类似于创建 FP-树的方法创建并插入树中,设置新后缀,形成项集$\{j_1, j_2, \cdots, j_n\}$ $(j_1 < j_2 < \cdots < j_n)$约束下的支持度降序后缀树 $DSST(j_n, \cdots, j_2, j_1)$. 在树中出现的项成为树的候选项.

定义空集 $\emptyset$ 约束下的支持度降序后缀树为 $DSST()$,其类似于由初始原始数据创建的 FP-树,但其叶节点后缀指针已被设置为空链.

例 1 图 1 给出事务数据库 *TDB*,其中 TID 表示事务的标识,itemset 表示事务所包含的项. 设定最小支持度 20%. 由原始数据创建的支持度降序后缀树 $DSST()$ 如图 1 所示. 其中带箭头虚线链接具有相同项的节点,$DSST()$叶节点的后缀指针指向空链,而其他后缀指针指向为空. 后缀以实线链接.

TID	itemset
100	*b,d,a,e*
200	*d,c,f*
300	*f,a,c,b,d,e*
400	*a,c*
500	*a,f,e,d,b*
600	*c,f*
700	*c,g*
800	*f*

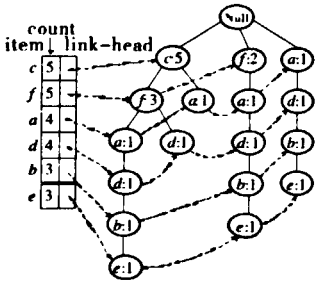

图 1 事务数据库 *TDB* 和产生的支持度降序后缀树 $DSST()$

在 *TDB* 中,项 a, b, c, d, e, f 和 g 的支持度计数分别为 4, 3, 5, 4, 3,5 和 1. 舍去非频繁项 g,按照支持度降序得到项的排列 $c < f < a < d < b < e$. 具体创建过程,从略.

性质 1 项集$\{j_1, j_2, \cdots, j_n\}$约束的支持度降序后缀树能正确产生所有包含$\{j_1, j_2, \cdots, j_n\}$及较小项组成的频繁模式及其支持度.

性质 2 设数据库中的项 $i_1 < i_2 < \cdots < i_n$ $(n > 1)$,则支持度降序后缀树 $DSST(i_n, i_{n-1}, \cdots, i_2)$ 可由 $DSST(i_n, \cdots, i_2, i_1)$ 得到.

在后缀树的搜索空间中,通过以上性质,后缀子树之间可通过递归得到.

引理 3 设数据库中的项 $i_1 < i_2 < i_3 < \cdots < i_n$ $(n > 1)$,若在 $DSST(i_n, i_{n-1}, \cdots, i_m)$ 中项 $i_1, i_2, \cdots, i_k$ 的计数等于$\{i_n, i_{n-1}, \cdots, i_m\}$的计数,则任何项集 $X \supseteq \{i_n, i_{n-1}, \cdots, i_m\}$ 且存在 $i_j (k \geq j \geq 1) \notin X$,$X$ 不是闭模式.

证明 根据支持度降序后缀树的定义,$DSST(i_n, i_{n-1}, \cdots, i_1)$包含了原始数据中项集信息,$DSST(i_n, i_{n-1}, \cdots, i_m)$中某个项 i_j 的计数等于包含项集$\{i_j, i_n, i_{n-1}, \cdots, i_m\}$的事务个数. 由题设可知 $support(\{i_n, i_{n-1}, \cdots, i_m\}) = support(\{i_j, i_n, i_{n-1}, \cdots, i_m\})$. 则由引理 2 可得结论.

引理 4 设数据库中的项 $i_1 < i_2 < i_3 < \cdots < i_n$ $(n > 1)$,若在 $DSST(i_n, i_{n-1}, \cdots, i_3)$ 中项 i_2 的计数等于项 i_2 下的项 i_1 的计数,则任何项集 $X \supseteq \{i_2, \cdots, i_n\}$,且 $i_1 \notin X$,X 不是闭模式.

证明 由支持度降序后缀树的定义,$DSST(i_n, i_{n-1}, \cdots, i_3)$中某个项 i_m 的计数等于包含项集$\{i_m, i_1, i_2, \cdots, i_n\}$的事务个数. 由题设可知 $support(\{i_1, i_2, i_3, \cdots, i_n\}) = support(\{i_2, i_3, \cdots, i_n\})$. 由引理 2 可得结论.

引理 5 设数据库中的项 $i_1 < i_2 < i_3 < \cdots < i_n$ $(n > 1)$,若在 $DSST(i_n, i_{n-1}, \cdots, i_3)$ 中项 i_1 的计数等于项 i_2 下的项 i_1 的计数,则任何项集 $X \supseteq \{i_2, \cdots, i_n\}$,且 $i_2 \notin X$,X 不是闭模式.

证明 根据支持度降序后缀树的定义,$DSST(i_n, i_{n-1}, \cdots, i_1)$包含了原始数据中项集信息,$DSST(i_n, i_{n-1}, \cdots, i_m)$中某个项 i_j 的计数等于包含项集$\{i_j, i_n, i_{n-1}, \cdots, i_m\}$的事务个数. 由题设可知 $support(\{i_n, i_{n-1}, \cdots, i_m\}) = support(\{i_j, i_n, i_{n-1}, \cdots, i_m\})$. 则由引理 2 可得结论.

3 简洁关联规则挖掘枚举与剪枝

在对整个频繁模式空间的搜索中,基于支持度降序后缀树性质和以上引理,对可能产生非闭候选项进行剪除,通过检测共同后缀项清除非闭模式.当挖掘完某个项时,把相应节点删除或链入后缀,无需对后缀做额外操作.同时创建后缀树时需要设置节点的后缀链.

3.1 有序枚举项集和挖掘过程

在挖掘过程中,设定 $CFCS(i_m,\cdots,i_2,i_1)(i_1\prec i_2\prec\cdots\prec i_m)$ 表示挖掘时,产生的以项集 $\{i_1,i_2,\cdots,i_m\}$ 为开头的候选频繁闭模式集.当 $CFCS(i_m,\cdots,i_2,i_1)$ 被设置为空时,则所有较小项 $i\prec i_1$,非空集 $CFCS(i_m,\cdots,i_2,i_1,i)$ 被递归清空.

例2 基于例1事务数据库 *TDB* 支持度为20%时产生的支持度降序后缀树 *DSST*( )进行频繁闭模式挖掘.挖掘过程中较大的项总被首先处理.

对 *DSST*( )中的项 e,产生候选集使 $CFCS(e)=\{\emptyset\}$.其横向链节点的后缀都为空链,仅需构建后缀树.由于 e 的计数等于在项 e 下项 b, d, a 的计数,由引理1,3可知项集 $\{e,b,d,a\}$ 的任何子集将不为闭模式.故需要对 (e,b,d,a) 的任意有序子排列 p,不需创建 *DSST*(p)而进行剪枝,同时对 $CFCS(e)$ 中任何项集并入 $\{b,d,a\}$.又由于项 b 的计数等于 e 下项 b 的计数,则根据引理4,包含项集 $\{b\}$ 而不包含项 e 的项集不为闭模式,故可剪枝相应的搜索空间和后缀树.并对 $CFCS(b)$ 中的每个项集并入 $\{e\}$.由于在按照项的固定顺序遍历闭模式的过程中,包含项 e 的闭模式先被创建.故在考察项 b 时仅需判断 *DSST*(b)需被剪枝即可.通过考察完项 e 后的后缀树节点,反向链接方向形成包含项 e 的节点后缀.发现每个 b 节点均包含相同项的后缀,则剪枝 *DSST*(b),如图2所示.

继续挖掘 *DSST*(e),考察项 f,使得 $CFCS(e,f)=\{\emptyset\}$.投影得到后缀树 *DSST*(e,f),继续挖掘项 c 为非频繁项,不可再进一步挖掘.然后进行回归,对 $CFCS(e,f)$ 中的每个项集加入项 f 并入 $CFCS(e)$,对 $CFCS(e)$ 中的每个项集加入项 e 并入 $CFCS(\ )$.

在 *DSST*(e)中考察项 c,由于项 c 不频繁则不继续挖掘,设置 $CFCS(e,f)=\{\}$.若项 c 频繁则需要进一步挖掘.

最终得到当前挖掘完项 e 后的条件闭模式集 $CFCS(\ )=\{\{e,b,d,a\}:3,\{e,f,b,d,a\}:2\}$.

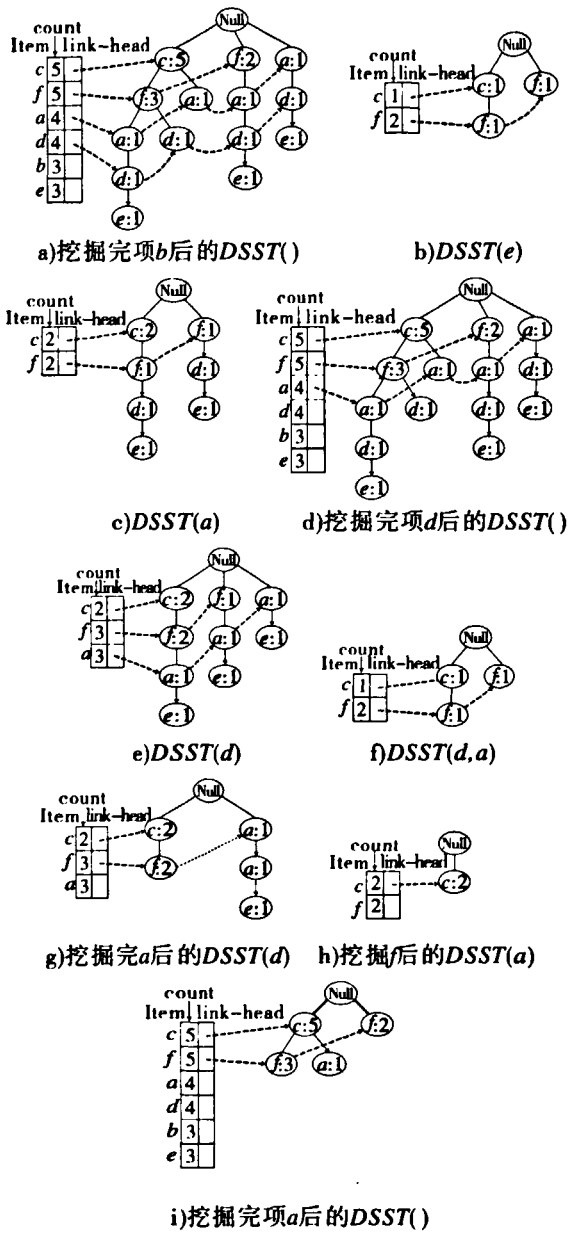

图2 挖掘过程

在 *DSST*( )中考察项 b,发现包含项 b 的节点后缀中均包含项 e,根据引理5,不可能产生包含项 b 而不包含项 e 的闭模式,则剪枝 *DSST*(b).设置 $CFCS(b)=\{\}$.

建立候选简洁关联规则集,对发现的闭模式查找已产生的子频繁闭模式和未产生的超频繁闭模式.计算置信度,形成简洁关联规则,同时输出.由于在挖掘过程中无需保存中间候选闭模式,且挖掘结果按照项的先后顺序产生,故只需保存少量候选简洁关联规则集.

随着挖掘的进一步深入,即可产生所有频繁闭模式集和简洁关联规则.

3.2 基于后缀进行剪枝

性质3 当一个项被考察时,若发现其节点的后缀链存在公共项,则对此项进行剪枝.

假设项 e 被考察,若其后缀包含公共项 f,则在当前挖掘中,包含项 f 的事物必包含项 e,由引理5,当前项 f 不可能组成更长的闭模式时,即可剪枝.

性质4 后缀树某节点 P 的后缀链仅包含以 P 为根的子树中出现的项,且这些项在子树中的总计数等于节点 P 的计数.

根据支持度后缀树的定义,节点 P 的项和其子树中的项同时出现在相同的事务中.后缀中的项与 P 的计数相同,意味着 P 的项与后缀项出现在相同事务中,则后缀中的项必然与当前考察项集出现在此事务中.

当在后缀树中挖掘完某个项时,只需把考察过的项的节点合并组成后缀链,而无需为父节点重新创建后缀链.若父节点后缀链为空,则把当前节点及其后缀都链入父节点后缀.如图2所示,在 $DSST(\ )$ 中挖掘完项 e 时,节点 d 的后缀链为空,则把 e 及其后缀链入 d 后缀.若父节点后缀不为空,则把当前后缀和父节点后缀链合并取交集,更新父节点后缀,并删除或回收当前节点.

随着挖掘的不断深入后缀树将变得越来越小.

性质5 当为某个候选项创建后缀树而设置后缀时,不需复制该项在原始后缀树中的共同后缀项.

3.3 简洁关联规则挖掘步骤

综合以上引理和分析,对给定事务数据库 TDB 和支持度 ξ,给出以下挖掘表示算法 CAM 的步骤:

1)创建 $DSST(\ )$.

2)调用过程 EnumerateAndPrune( ).

EnumerateAndPrune( )伪代码定义如下:

```
Procedure EnumerateAndPrune(α) (这里 α 表示 i1,i2,…,ik(ik≺…≺i2≺i1)) {
  for(在 DSST(α)中出现的未处理的最大项频繁 in) {
    if(in 节点后缀链不包含公共项) {
      设 in 的所有相同计数的条件项为 csi,不再处理,并插入 α 形成 β
      CFCS (β) = {∅};
      if(除 csi 中项外存在其他项 im) {
        创建项 im 的条件 DSST(α, im);
        EnumerateAndPrune(α, im); }
      对 CFCS (β)中的每个项集并入 csi;
      创建当前可能的简洁关联规则; }
    把项 in 的节点链入 DSST(α)中后缀链; }
  }
```

支持度降序后缀树具体创建过程,不做详细介绍.

定理1 CAM 算法能正确挖掘出所有频繁闭模式和简洁关联规则.

证明 由 CAM 算法基于后缀树进行挖掘,并通过相关引理实现了对搜索空间的剪枝.

基于后缀树的性质和相关引理,可知 CAM 算法不仅能够正确枚举所有频繁闭模式和简洁关联规则,且能够剪枝所有非简洁模式和规则.

4 实验结果

算法 FP-Growth 不产生条件模式,但挖掘所有的频繁模式,是已知的挖掘频繁模式的有效算法之一.算法 A-Close,CLOSET+挖掘结果为闭模式,本节在基于与 CLOSET+比较基础上说明本文算法要比基于数结构挖掘所有频繁模式方法和通过扫描数据集进行剪枝方法效率有明显优势.

实验环境:CPU Intel 3.0 GHz,内存 1 GB,Windows XP 操作系统.算法 CAM 采用 Microsoft Visual C++ 6.0 编写,CLOSET+由作者主页提供.使用具有不同特征的公共测试数据集[5] Kosarak,Accidents,Pumsb_star,数据集 T40I10D100K 由人工方法生成.结果如图3所示.

图3给出 CAM 与 CLOSET+在各数据集上的比较情况.从图中可以看出在所比较的数据机上 CAM 效率要比 CLOSET+快约1~7倍左右;且随着支持度降低,算法 ACM 具有良好的可规模性.

5 结语

本文提出了基于后缀树挖掘闭模式和简洁关联规则,通过采用顺序枚举和基于后缀进行条件项剪枝的机制,有效地实现了非闭项集的提前剪枝和直接产生,同时生成简洁关联规则,避免了通过重复扫描数据库确定项集支持度来进行的检测.实验表明了本方法的有效性.

参考文献:

[1] Pasquier N, Bastide Y, Taouil R, et al. Discovering frequent closed itemsets for association rules[C]//Proc 7th Int Conf Database Theory (ICDT'99), Jerusalem: Springer, 1999: 398-416.

[2] Han J, Pei I, Yin Y. Mining frequent paterns without candidate generation [C]//Proc of SIGMOD '00, Dallas: ACM, 2000: 1-12.

[3] Jiawei Han, Micheline Kamber. 数据挖掘:概念与技术

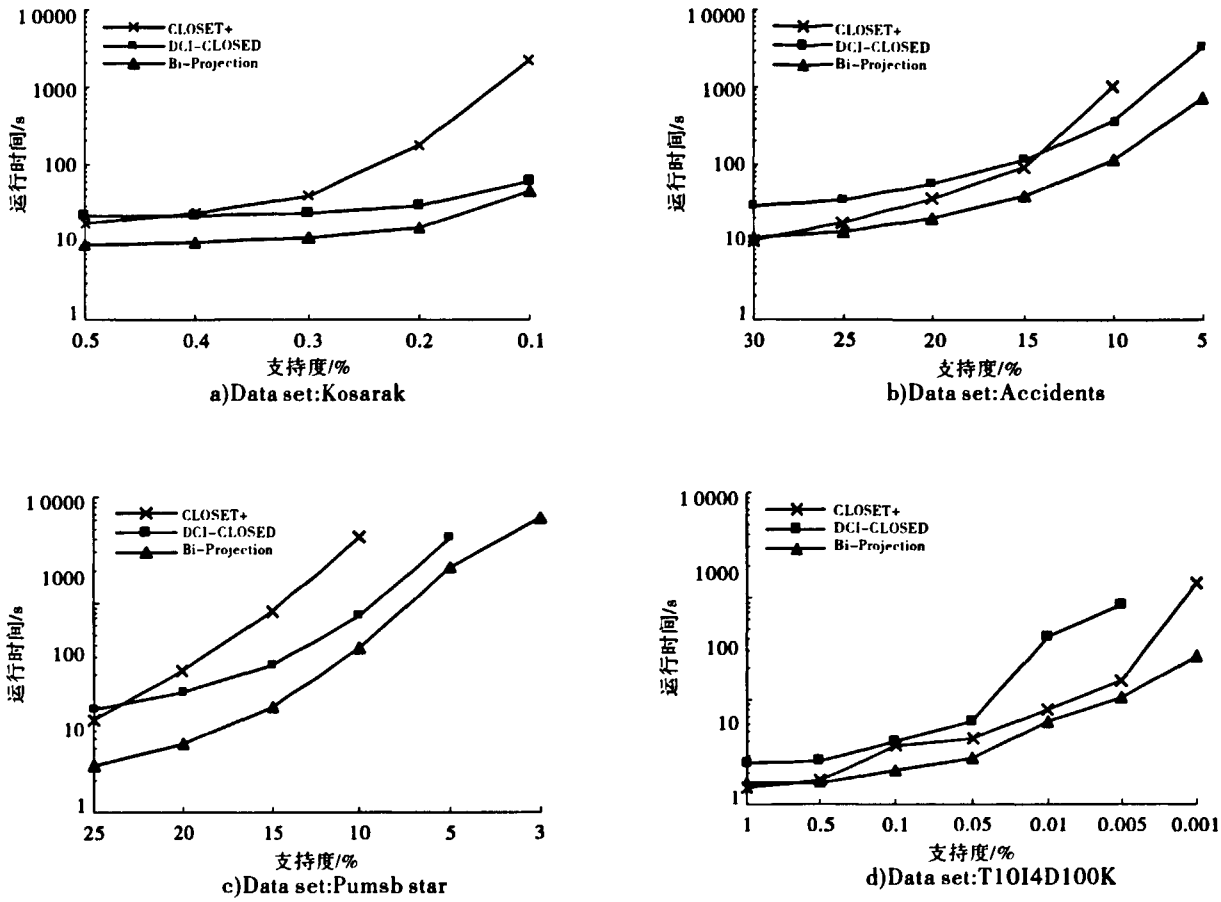

图3　实验结果

[M]. 苑明,孟小峰,译. 北京:机械工业出版社,2001:149-184.

[4] Zaki M J, Gouda K. Fast vertical mining using diffsets [R]. New York: Rensselaer Polytechnic Institute, 2001.

[5] Zaki M J, Hsiao C J. Charm: An efficient algorithm for closed itemsets mining[C]//Proc of 2nd SIAM Int Conf on Data Mining, Arlington: VA, 2002.

[6] Pei J, Han J, Wang J. Closet +: Searching for the best strategies for mining frequent closed itemsets [C]// SIGKDD '03, Washington: ACM, 2003.

[7] Maybin Muyeba, Sulaiman M Khan, Frans Coenen. Fuzzy weighted association rule mining with weighted support and confidence framework[C]//Proc of PAKDD 2008, Osaka: Springer, 2008: 49-61.

[8] Haifeng Feng, Marie-Jeanne Lesot, Marcin Detyniecki. Using association rules to discover color-emotion relationships based on social tagging[C]//Proc of the 14th Int Conf on Knowledge-based and Intelligent Infor and Eng Syst, Cardiff: Springer, 2010: 544-553.