基于约束的不确定数据频繁项集挖掘算法研究*

刘卫明, 杨 健, 毛伊敏
(江西理工大学 信息工程学院, 江西 赣州 341000)

摘 要: 针对基于约束的不确定数据频繁项的经典挖掘算法——U-FPS 算法的不足,提出了适用于基于约束的不确定数据的新算法——UC-Eclat 挖掘算法。该算法不需要构建频繁模式树,而采用了数据库垂直模式求交集的方式来计算支持度的方法,提高了挖掘效率。

关键词: 频繁项; 不确定数据; 项目约束; 反单调约束; 概念格
中图分类号: TP391 **文献标志码**: A **文章编号**: 1001-3695(2012)10-3669-03
doi:10.3969/j.issn.1001-3695.2012.10.017

Research of algorithm based on uncertain data for constrained frequent sets

LIU Wei-ming, YANG Jian, MAO yi-min
(*College of Information Engineering*, *Jiangxi University of Science & Technology*, *Ganzhou Jiangxi* 341000, *China*)

Abstract: For the shortage of the U-FPS(uncertain data mining algorithm based on constraint) algorithm, this paper proposed a new algorithm UC-Eclat, which didn't need build frequent pattern tree but adopt vertical mode database to find intersection, and then improved the mining efficiency.

Key words: frequent item; uncertain data; item constrain; anti-monotone constrain; lattices

在现实世界中,数据普遍存在着不确定性这一特征,特别是在实际工程中存在大量的不确定数据,如果挖掘模型不能准确地描述或没有充分考虑对象的不确定性,那么由挖掘模型得到的结果就是不完善的,甚至是错误的。只有充分考虑数据的不确定性,才能得到基于不确定数据并能准确反映客观事物知识的挖掘模型。

在没有约束的频繁项挖掘中,用户单单只设置了支持度,然后就等待系统交付挖掘结果。而这些频繁项中只有少数是用户所需要的。如果用户能对频繁项的挖掘过程进行监督指导,使得产生的频繁项是用户所需要的,这样挖掘效率就更高。通过引入适当的约束可以提高算法的效率,然而项目约束是一种应用十分广泛的约束,本文将其与不确定数据挖掘结合,提出了一种基于约束的不确定频繁项挖掘算法。

1 相关研究

不确定数据已经成为一个新兴的研究领域[1],目前国内外已提出了一些关于不确定数据频繁项挖掘的经典算法。Chui 等人[2]提出的 U-Apriori 算法,它基于 Apriori 算法挖掘频繁项集,并使用数据修整技术剪掉原始数据集中低概率出现的项提高效率;但鉴于修整阈值难以确定,Chui 等人[3]又提出 UCP-Apriori 算法,它是对 U-Apriori 的改进,提出对不确定数据频繁项集挖掘的交替递减剪枝技术;Leung 等人[4]提出的 UF-growth 算法,它拓展了 FP-tree,每个节点除了存储项名称外还存储项的期望支持度和同一项在不同事物中以相同概率出现的次数,算法分为构建 UF-tree 和在 UF-tree 中挖掘两个过程;高聪等人[5]提出的 UD-FP-tree 算法,是基于期望支持度模型研究特定领域项集的概率值计算方法,也通过拓展 FP-tree 挖掘频繁项集,但是基于约束的不确定频繁项挖掘的研究还很少;Leung 等人[6]提出了基于 UF-tree 的约束算法 U-FPS(SAM,基于反单调约束的算法),U-FPS 算法分为构建 UF-tree 和在 UF-tree 中挖掘两个过程,此算法需要构建频繁模式树,当数据量较大时会占用大量的内存资源,并且需要大量的递归调用导致挖掘效率降低。本文针对 U-FPS 算法存在的不足,提出了 UC-Eclat 算法,此算法在效率上较 U-FPS 有较大的提升。UC-Eclat 是在 Eclat 基础上改进后适用于不确定数据的一种算法,它继承了 Eclat 算法的特性,采用了一种事物数据库垂直模式来计算项集的支持度[7],项集的支持度等于事物列表的交集,这是一种全新的支持度计算方法。最后引入了约束条件,最终形成了 UC-Eclat 算法,并且在实验部分比较了 U-FPS(SAM)与 UC-Eclat 算法的算法效率。

2 概念描述

2.1 不确定数据频繁项定义

定义 1 (K,τ)不确定频繁项。不确定数据库为 D,W 为不确定数据库 D 的所有可能世界实例集合,若元素 t 为不确定频繁项,则满足下式:

$$\sum_{\omega \in W, m_t \geq K} P[\omega] > \tau \tag{1}$$

其中:K 为数量阈值;τ 为概率阈值;m_t 为元素 t 出现的次数。

收稿日期: 2012-02-21; **修回日期**: 2012-04-20 **基金项目**: 湖南省自然科学基金资助项目(11JJ4050);湖南省教育厅优秀青年资助项目(11B039);网络化软件运行时行为分析方法研究

作者简介: 刘卫明(1964-),男,江西新余人,教授,博士研究生,主要研究方向为计算机网络技术、管理信息系统及计算机在矿山中的应用(y-j2006@163.com);杨健(1984-),男,硕士研究生,主要研究方向为数据挖掘;毛伊敏(1970-),女,副教授,博士,主要研究方向为数据挖掘、网格计算.

要判断某元素是否为频繁项,需要列出所有的可能实例,在每个可能实例中元素 t 的个数均必须不小于 K 的概率和,如果大于则元素 t 是频繁项。

从定义 1 中可以看出,要确定不确定数据库中的频繁项,需要对所有的不同元素计算式(1)的概率,才能最终判定。由于被检测元素的数量巨大,以及可能世界实例空间的规模呈指数级增长,上述这种原始方法并不可行,因此,利用不确定数据的特殊性找到过滤方法,减少检测次数是十分必要的。

定义 2 不确定数据支持度计数。在不确定事物数据库中,定义支持度计数

$$S_t = \sum_{t=1}^{n} p_t \tag{2}$$

其中:p_t 表示项集分别在每一条事物中的概率,t 表示事务数据库中的 tid。该表达式的含义是:项集的支持度为每一条事务中的该项集的概率之和。

2.2 约束的定义

定义 3 对确定的数据集 D,约束 C 是在 2^I 上的断言。当且仅当 $C(X)$ 为真时,称项集 X 满足约束 C。设 $S \subseteq 2^I$,$\mathrm{SAT}_C(S) = \{X \in S \mid X$ 满足 $C\}$。当 $S = 2^I$ 时,$\mathrm{SAT}_C(S)$ 简写为 SAT_C。

定义 4 反单调约束。给定一个约束条件 C_m,如果一个项集 S 不满足 C_m,并且 S 的任何超集也不满足 C_m,则称约束条件 C_m 是反单调的。

2.3 概念格理论

本文提出的 U-Eclat 算法中使用到了概念格理论,以下给出关于概念格的定义[8]。

定义 5 令 P 是一个集合,P 上的一个二元关系 $\leqslant$ 称为 P 上的一个偏序,对于所有的 $X, Y, Z \in P$,满足

a) 自反性:$X \leqslant X$;

b) 反对称性:$X \leqslant Y$ and $Y \leqslant X$,则 $X \equiv Y$;

c) 传递性:$X \leqslant Y$ and $Y \leqslant Z$,则 $X \leqslant Z$;

定义 6 L 是一个有序集合。如果对于所有的 $X, Y \in L$,都有 $X \vee Y (X \wedge Y) \in L$,$L$ 称为一个 join(meet)半概念格。如果它既是一个 join 半概念格,也是一个 meet 半概念格,L 称为一个概念格。如果 L 的所有子集 $S \in L$ 都存在 $\vee S$、$\wedge S$ 则为完全概念格。一个有序集合的 $M \subseteq L$,如果 $X, Y \in M$,$X \vee Y \in M$,$X \wedge Y \in M$,则 M 称为 L 的子概念格。

3 算法

3.1 数据存储

所谓垂直的 tid-列表数据库指的是,数据库中的每一条记录由一个项目及其所出现过的所有事务记录的列表构成。这样使得任何一个频繁集的支持度都可以通过一些 tid-list 的交集运算求得。表 1 的事务数据库是文本的示例数据库,表 2 是它的 tid-列表形式的格式。

表 1 事物数据库示例

tid(transaction ID)	item
1	{A:0.9, B:0.8, C:0.7, D:0.6}
2	{B:0.9, C:0.7}
3	{A:0.9, B:0.7, D:0.1}
4	{A:0.9, B:0.6, C:0.5}
5	{A:0.9, B:0.7, C:0.6, D:0.3}
6	{B:0.9, C:0.4, D:0.2}

表 2 表 1 的垂直形式

item	tid-list
A	{1(0.9),3(0.9),4(0.9),5(0.9)}
B	{1(0.8),2(0.9),3(0.7),4(0.6),5(0.7),6(0.9)}
C	{2(0.7),4(0.6),5(0.6),6(0.4)}
D	{1(0.6),3(0.1),5(0.3),6(0.2)}

把不确定数据库中的概率信息存入矩阵,得到概率矩阵,如表 3 所示,将数据库垂直形式存入概率矩阵中。

表 3 概率矩阵

item	1	2	3	4	5	6
A	0.9	0	0.9	0.5	0.9	0
B	0.8	0.9	0.7	0.6	0.7	0.9
C	0	0.7	0	0.6	0.6	0.4
D	0.6	0	0.1	0	0.3	0.2

3.2 U-Eclat 算法描述

引理 对于任意 $X \in P$,令 $X = \cup_{Y \in J} Y$,则 $\sigma(x) = |\cap_{Y \in J} L(Y)|$。

引理的含义是:如果可以将 X 表示为 J 中的元素的并集,则可以通过计算这些元素的 tid 列表的交集来计算 X 的支持度。这里 $J = \{Y \in A(P(I)) \mid Y \leqslant X\}$,表示由 $P(I)$ 的原子项构成的集合,$\sigma(X)$ 表示 X 的支持度,$L(Y)$ 表示元素 Y 的 tid 列表。例如 $AB = A \cup B$,因此 $L(AB) = L(A) \cap L(B) = \{1,3,4,5\}$,于是 $|L(AB)| = S_t = \sum_{t=1}^{n} p_t = 0.9 \times 0.8 + 0.9 \times 0.7 + 0.9 \times 0.6 + 0.9 \times 0.7 = 2.52$。

在频集挖掘过程中,计算项集的支持度主要有计数和交集操作两种方法[9],U-Eclat 算法采用了数据库垂直模式对数据进行扫描[10],通过求交集的方式来计算支持度是该算法的核心。以下给出该算法的主要思想。

算法 1

输入:事物数据库 TDB,最小支持度阈值 minsupp。

输出:所有的频集 L。

第一次扫描数据库,得到频繁 1-项集 L_1,

$L = L \cup L_1$

U-Eclat(L_1):

for all $X_i \in L_1$ do

for all $X_j \in L_1, j > i$ do

产生新的候选项集 $R = X_i \cup X_j$,

tidsets(R) = tidsets(X_i) ∩ tidsets(X_j)

//求项集 X_i 与 X_j 的事物列表交集

若 $|\text{tidsets}(R)| = S_t = \sum_{t=1}^{n} p_t \geqslant$ minsupp

//计算交集的概率和

$L = L \cup R$,$T_i = T_i \cup R$,其中 T_i 初始为空

若 $T_i \neq \varnothing$,调用函数 U-Eclat(T_i)

在 U-Eclat 算法中求支持度需要计算交集:tidsets(R) = tidsets(X_i) ∩ tidsets(X_j),传统的计算交集的方法效率比较低,在此提出了一种新的计算交集的的算法。

初始化数组 $a[\ ]$、$b[\ ]$和 $c[\ ]$,设定它们的长度为 10,初始值为 0,取表 2 中前两条记录,查询出记录 A 中所包含的 tid 序号。将数组 $a[\ ]$中下标与 tid 相对应的元素置 1,如记录 A 中包含的 tid 序号有{1,3,4,5},相应的 $a[\ ]$ = {0,1,0,1,1,1,0,0,0,0},同理 $b[\ ]$ = {0,1,1,1,1,1,1,0,0,0},然后将两数组对应相加放在 $c[\ ]$数组中,$c[\ ]$中为数值为 2 的项即为 A、B 两条记录共有的项,即求出了交集。

定理 1　当执行 $c[i]=a[i]+b[i]$ 运算时,若 $a[i]\&\&b[i]=0$,则此时不必执行 $c[i]=a[i]+b[i]$ 运算,保持 $c[i]=0$ 即可。

证明　当执行 $c[i]=a[i]+b[i]$ 运算时,

a)若 $a[i]\&\&b[i]=0$,$c[i]=0$ 或 1,此时 $c[i]\neq 2$;

b)若 $a[i]\&\&b[i]=1$,$a[i]=1$,$b[i]=1$,此时 $c[i]=2$。

因为只有当 $c[i]=2$ 时,才说明 $a[i]=1$ 且 $b[i]=1$,也就是说明 A、B 两条记录中都存在相对应的 tid,此时 A、B 存在交集,故 $c[i]\neq 2$ 时,$a[i]+b[i]$ 并无意义,即可省去相加运算,保持 $c[\ ]$ 数组相应位置的原值即可。

算法 2

输入:tidsets(X_i),tidsets(X_j)。

输出:tidsets(R)。

a)初始化数组 $a[\ ]$、$b[\ ]$ 和 $c[\ ]$,其长度为事物数据库记录数 + 1,并置初始值为 0;

b)将数组 $a[\ ]$ 中对应 tidsets(X_i) 所包含的 tid 置 1,$b[\ ]$ 同理;

c)for ($i<$事物数据库记录数 +1) do

if($a[i]!=0\&\&b[i]!=0$)

//当 $a[i]$、$b[i]$ 都不为 0 时,执行相加操作

{$c[i]=a[i]+b[i]$;}

$c[\ ]$ 中为 2 的元素所对应的下标即为 tidsets(R)

3.3　UC-Eclat 算法描述

UC-Eclat 算法是在 U-Eclat 算法基础上引入了项目约束。本文来讨论项目约束中反单调约束的情况,如 $B=(\neg(A\wedge B))$,根据定义 4 可知,此约束条件的含义是:项集中不包含 $\{AB\}$。根据反单调约束的定义,如果一个项集包含了 $\{AB\}$ 的项集即不满足 B,其超集也会包含 $\{AB\}$,则其超集也不满足 B。

算法 3

输入:事物数据库 TDB,最小支持度阈值 minsupp,项目约束 B。

输出:所有的满足约束 B 的频集 L。

第一次扫描数据库,得到频繁 1-项集 L_1,$L=L\cup L_1$

UC-Eclat(L_1):

for all $X_i\in L_1$ do

for all $X_j\in L_1$,$j>i$ do

产生新的候选项集 $R=X_i\cup X_j$,

if (!(R 不满足 B))

/*判断候选集 R 是否满足约束 B,如果不满足,则不用计算其支持度,同时 R 的超集也不满足 B,故也不必计算超集的支持度*/

{tidsets(R) = tidsets(X_i) ∩ tidsets(X_j)

若 $|\text{tidsets}(R)|=S_i=\sum_{t=1}^{n}p_t\geqslant$ minsupp

$L=L\cup R$,$T_i=T_i\cup R$,其中 T_i 初始为空}

若 $T_i\neq\varnothing$,调用函数 UC-Eclat(T_i)。

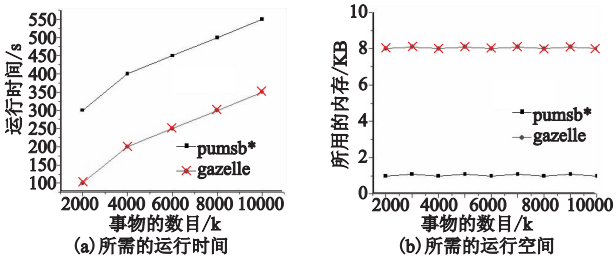
4　实验比较

实验的硬件环境为 Pentium 1.0 GHz CPU,512 MB 内存,操作系统为 Windows XP,并采用 C 语言编写模拟测试程序,在 VC++ 6.0 环境中运行。

从 IBM Almaden 研究中心得到的数据集 pumsb* 和 gazelle 被作为测试数据集[11]。通过随机复制,将原来的数据集转换为含有 100 000 事务的数据集。Pumsb* 是稠密数据集,在实验时取 ε 为支持度的 1/2;gazelle 是稀疏数据集,ε 为支持度的 1/10。

实验 1　UC-Eclat 收缩性的研究

在此实验中测试数据集长度对运行时间、空间的影响,理论上,当数据集长度增加时,运行时间是递增的。本文采用以上给出的两套数据集,最小支持度为 0.1,结果如图 1 所示。图 1(a)说明,当数据从 2000 k 增长到 10000 k 时,运行时间增长比较平缓;图 1(b)说明,当数据不断增长时,所消耗的存储容量趋于稳定。从图 1 可以看出,UC-Eclat 算法具有可收缩性和可行性。

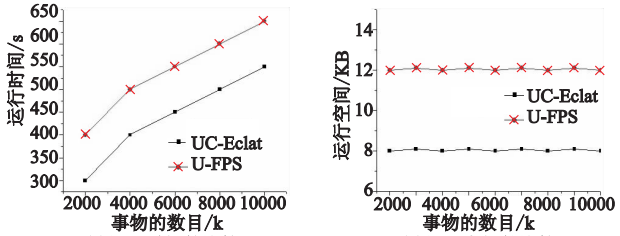


(a)所需的运行时间　　(b)所需的运行空间

图1　UC-Eclat算法收缩性的研究

实验 2　U-FPS 与 UC-Eclat 算法在相同数据集中挖掘频繁集的处理时间与空间的比较

在实验中,使用了数据集 pumsb*,最小支持度为 0.1,结果如图 2 所示。图 2(a)表明 UC-Eclat 算法的执行时间小于 U-FPS 算法;图 2(b)表明随着数据的增多,UC-Eclat 算法的存储容量趋于稳定,UC-Eclat 算法所需的存储容量小于 U-FPS 算法。

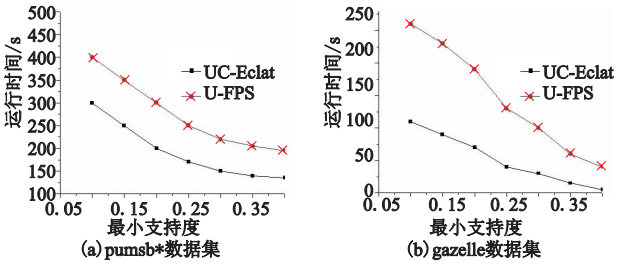


(a)运行时间的比较　　(b)运行空间的比较

图2　挖掘频繁集的处理时间与空间的比较

实验 3　U-FPS 与 UC-Eclat 算法在不同数据集中不同支持度下挖掘频繁集的处理时间的比较

在实验中,图 3(a)采用了 pumsb* 数据集,图 3(b)采用了 gazelle 数据集。从图 3 可以看出,U-FPS 算法较适合大支持度与稀疏数据集,UC-Eclat 算法对支持度及数据集的适应性比 U-FPS 算法好得多。



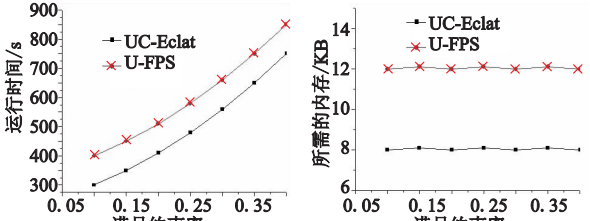
(a)pumsb*数据集　　(b)gazelle数据集

图3　挖掘频繁集的处理时间的比较

实验 4　U-FPS 与 UC-Eclat 算法在相同数据集中,不同满足约束项占事物数据库百分比的条件下,挖掘频繁项集处理时间和空间的比较

实验中,使用了数据集 pumsb*,由于算法 3 是一个反单调约束算法,不满足约束的项集不用参与支持度计算,并且其超集也不用参与计算,而满足约束条件的,其超集还要继续参与计算。理论上,当满足约束项占事物数据库百分比增加时,运行时间是递增的。图 4(a)表明,当在满足约　(下转第 3680 页)

(上接第 3671 页)束率较小时,两算法的运行时间都较短;随着满足约束率增加,两算法的运行时间都增加。这是因为满足约束率增加,符合条件的频繁项集越多,处理时间变长。图 4(b)表明随着满足约束率增加,UC-Eclat 算法的存储容量趋于稳定,UC-Eclat 算法所需的存储容量小于 U-FPS 算法。



(a)运行时间的比较　　(b)运行空间的比较

图4　挖掘频繁集的处理时间与空间的比较

5　结束语

本文首先提出了不确定数据挖掘和条件约束的基本概念,阐述了不确定数据与确定数据的本质区别。由于针对基于约束的不确定数据挖掘,目前已有的 U-FPS 算法需要构建频繁模式树,当数据量较大时,会占用大量内存,并且需要大量使用递归调用导致挖掘效率降低。为了提高挖掘效率,在 Eclat 算法的基础上,使之适用于不确定数据,并且加入反单调项目约束条件,提出了 UC-Eclat 算法。此算法在计算频繁项支持度时,采用了数据库垂直模式求交集的方法,并且在求交集的方式上进行了改进,大大提高了算法的效率。在实验部分,分别以支持度、数据集长度、满足约束项占事物数据库百分比为变量,将 UC-Eclat 算法与 U-FPS 算法作了时间复杂度与空间复杂度上的比较,结果显示,UC-Eclat 算法效率更高。

参考文献:

[1] 周傲英,金澈清,王国仁,等. 不确定性数据管理技术研究综述[J]. 计算机学报,2009,32(1):1-16.

[2] CHUI C K, KAO Ben, HUNG E. Mining frequent itemsets from uncertain data[C]//Proc of the 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining. Berlin: Springer-Verlay,2007:47-58.

[3] CHUI C K, KAO Ben. Decremental appriach for mining frequentitemsets from uncertain data[C]//Proc of the 12th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining. Berlin: Springer-Verlay. 2008:64-75.

[4] LEUNG C K S, CARMICHAEL C L, HAO Bo-yu. Efficient mining of frequent patterns from uncertain data[C]//Proc of the 7th IEEE International Conference on Data Mining. Washington DC: IEEE Computer Society,2007:489-494.

[5] 高聪,申德荣,于戈,等. 一种基于不确定数据的挖掘频繁集方法[J]. 计算机研究与发展,2008,45(suppl):71-76.

[6] LEUNG C K S, BRAJCZUK D A. Efficient algorithms for the mining of constrained frequent patterns from uncertain data[J]. ACM SIGKDD Explorations Newsletter,2009,11(2):123-130.

[7] ZAKI M J. Scalable algorithms for association mining[J]. IEEE Trans on Knowledge and Data Engineering,2000,12(3):372-390.

[8] DAVEY B A, PRIESTLEY H A. Introduction to lattices and order[M]. Cambridge: Cambridge University Press,1990.

[9] 宋长新,马克. 改进的 Eclat 数据挖掘算法的研究[J]. 微型计算机信息,2008,24(8):92-94.

[10] 梅锦锋. 改进的垂直数据表示的高效频繁项集挖掘算法研究[D]. 广州:中山大学,2007:18-19.

[11] IBM Almaden[EB/OL]. (2004-12-18). http://www.almaden.ibm.com/cs/quest/demos.html.