一种自适应阈值的简洁性约束频繁项目集挖掘算法

任永功 吕 朕 孙宇奇
(辽宁师范大学计算机与信息技术学院 大连 116029)

摘 要 基于约束关联挖掘,近几年在国际上受到较大关注。从许多约束的关联挖掘算法中发现,传统的约束阈值大多是通过专家给定或经过反复试验得出的,缺乏用户反馈与客观依据的支持。为了解决此问题,提出一种面向用户需求的阈值构造方法,该方法引用正态分布理论获得自适应约束阈值,并应用简洁性约束对FGC算法进行改进;同时提出一种快速、直观、有效的频繁项目集挖掘算法。实验证明,该算法在增强系统可用性的同时降低了算法运行时间。

关键词 频繁模式树,频繁项目集,简洁性约束,自适应阈值
中图法分类号 TP31 **文献标识码** A

Frequent Itemsets Mining Algorithm of Succinct Constraint with Adaptive Thresholds

REN Yong-gong LU Zhen SUN Yu-qi
(School of Computer and Information Technology, Liaoning Normal University, Dalian 116029, China)

Abstract In recent years, associate mining based on constraint is more and more focused. From the algorithm of existing associate mining, it is easy to observe that the traditional thresholds are given mostly by expert or found after repeated test. User's feedback and the support of objective evidence was lacked. According to this problem, the constructional method of thresholds catered to user requirement was proposed. To obtain the self-adaptation constraints thresholds, the theory of normal distribution is cited in this method. FGC algorithm is ameliorated by using succinct constraint. A speedy, effective and intuitive frequent items mining algorithm was proposed. Experiments show that the proposed methods can increase the system's availability and reduce the runtime of algorithm.

Keywords FP-Tree, Frequent item-sets, Succinct constraints, Adaptive threshold

1 引言

近年来,人们开始将注意力从模型和算法的研究转移到对系统工作的控制上,逐渐关注约束的重要性。这种关注一方面能提高挖掘效率和精度、控制系统使用规模,另一方面能增进用户与挖掘算法之间的交互,使挖掘工作按照预期方向发展。目前,对于约束与关联规则挖掘算法如何结合的问题,有两个经典的算法:ExAnte[1]和FP-bonsai[2]算法,它们分别在关联规则挖掘算法Apriori[3]和FP-growth[4]算法中嵌入了单调性与反单调性约束。

在传统的ExAnte和FP-bonsai算法中,都是直接运用单调性与反单调性约束来挖掘频繁项目集,因此在挖掘过程中会遇到两个问题:①单调性与反单调性的阈值选择问题;②对庞大数据的预处理问题。

针对以上两个问题,本文从用户主观意愿和全局性考虑,对FGC[5]算法进行了改进,提出一种基于自适应阈值的简洁性约束频繁项目集挖掘算法NSFGC。算法应用正态分布思想,获得单调性与反单调性阈值,并应用简洁性约束,简化数据集,尽量减少不必要的空间开销,直接对事务数据库做深层次挖掘。

2 基本定义

关联规则挖掘大量数据项集之间有趣的关系,而基于约束的挖掘则允许用户根据关注的目标,设定挖掘的约束,使得数据挖掘过程更有效率。设 D 为事务数据库,min_sup 为最小支持度,对于项集 $X\in I$,若 $\mathrm{Sup}(X)\geqslant \mathrm{min\_sup}$,则称 X 为 D 中的频繁项集。长度为 k 的频繁项目集称为频繁 k-项目集。设项目集 $I=\{i_1,i_2,\cdots,i_m\}$,事务数据库 $T=\langle TID, Item\rangle$,模式 S 和 S^* 都是项目集 I 的子集,如果 $S^*\in S$,则 S^* 是 S 的子模式,S 是 S^* 的超集。基于约束的挖掘,能够使挖掘在用户提供的各种约束的指导下进行,避免在挖掘过程中出现过多用户不感兴趣的规则。这些约束包括如下。

定义1(单调性与反单调性约束) 一个约束 C 是作用于项目集 I 的幂集上的谓词。

约束 C 对于一个模式 S 的结果用布尔变量来表示,即 $C(S)$=Ture/False。

$C(S)$=Ture 表示 S 满足约束条件。

$C(S)$=False 表示 S 不满足约束条件。

到稿日期:2010-10-29 返修日期:2010-12-30 本文受国家自然科学基金(60603047),教育部留学回国人员科研启动基金,辽宁省科技计划(2008216014),辽宁省教育厅高等学校科研基金(L2010229),大连市优秀青年科技人才基金(2008J23JH026)资助。
任永功(1972—),男,博士,教授,主要研究方向为数据挖掘技术,E-mail:renyg@dl.cn;吕 朕(1985—),女,硕士生,主要研究方向为数据挖掘;孙宇奇(1986—),男,硕士生。

一个约束 C_m 是单调性约束，是指满足 C_m 的任何项目集 S 的超集也能满足 C_m。

一个约束 C_{am} 是反单调性约束，是指对于任意给定的不满足 C_{am} 的项目集 S，不存在 S 的超集能够满足 C_{am}。

定义 2(简洁集) 指用特定规律挖掘的约束。一个项目子集 I_s 是一个简洁集(Succinct Set)，如果对于某些选择性谓词 P，该项目子集能够表示为 $\sigma_p(I)$ 的形式，其中 σ 是选择符。

定义 3(强简洁集) $SP\in 2^{item}$ 是一个强简洁集(Succinct Power Set)，如果有一个数目不变的简洁集 $item_1,\cdots,item_k\in item$，$SP$ 能够用 $item_1,\cdots,item_k$ 的并、差运算表示出来。

定义 4(简洁性约束) 如果一个约束 C 作用到项目集 $SAT_C(Item)$ 是一个强简洁集，则这个约束就是简洁性约束。

3 自适应阈值的简洁性约束关联挖掘算法(NSFGC)

用于关联规则挖掘的约束可分为单调性约束、反单调性约束、可转变的约束和简洁性约束。其中单调性约束有效剪除不符合约束的项，反单调性约束删除不满足约束的事务，简洁性约束列出所有满足该约束的集合。

3.1 FGC 算法

FGC 算法(FOLD-Growth Constraint) 通过一个预处理结构——SOTrieIT[6](Support Ordered Trie Itemset) 以最小代价直接获取频繁 1-项集和频繁 2-项集。

SOTrieIT 结构是一个按支持度与字序降序排列的结构。此结构只要通过简单的树遍历就能够得到频繁 1-项集和频繁 2-项集，即使支持度发生改变，也不需要重新建树。

FGC 算法将 SOTrieIT 结构与约束嵌入技术相结合，在扫描数据库的同时进行频繁 1-项集和频繁 2-项集约束检查，提高了检查效率。但是该算法并没有从用户的角度出发，去挖掘用户感兴趣项目之间的关联规则。并且，自定义约束没有一个明确标准，阈值设得过大或者过小，都不能真正表示出用户的需要，并且会造成时间和空间的浪费。

3.2 NSFGC 算法

NSFGC 算法应用简洁性约束过滤数据库，删除不符合要求的事务，通过正态分布找出最能提高系统可用性的动态阈值。最后运用动态阈值挖掘事务数据库，得到频繁项目集。

3.2.1 过滤数据库

在数据库挖掘过程中会出现大量频繁项集，也有不相关的项目进入挖掘过程，这会造成计算时间和空间的极大浪费。因此，要在此过程中应用简洁性约束[7]中的超集约束，使挖掘在用户提供的各种约束指导下进行。超集约束由两部分组成：强制性约束组(mandatory items group$_i$)和选择性约束组(optional items group)。满足简洁性的超集约束如式(1)所示。

$$v=\{x_1,\cdots,x_i\}\cup\gamma \tag{1}$$

式中，v 表示有效项目的集合，$x_i\in$ 强制性组$_i$，$\gamma\in((\bigcup_{i=1}^{l}$ 强制性组$_i)\cup$ 选择性组)。所有满足集合 v 的频繁项目集即满足简洁性约束。下面通过例子阐述简洁性约束的优越性。事务数据库如表 1 所列，设最小支持度为 2，$S.Type\in\{P_1,P_2\}$，$C_m>10$，$C_{am}<25$。

表 1 事物数据库及项目信息

项目	a	b	c	d	e	f	g
价格	40	10	25	30	20	35	15
类型	S_1	P_1	P_2	S_2	S_3	S_4	S_5

TID	项目
T_1	{a,b,c,d}
T_2	{b,d,f}
T_3	{a,b,d,e}
T_4	{a,b,c,e,g}
T_5	{c,e,g}

根据定义划分，分为强制性组 1：$\sigma_{type}=P_1$，强制性组 2：$\sigma_{type}=P_2$ 和选择性组：$\sigma_{type}\neq P_1\wedge\sigma_{type}\neq P_2$。根据式(1)，可得到既频繁又满足约束的项目集，即 $v=\{b,c\}\{b,c,a\}$。

3.2.2 计算动态阈值

通过简洁性约束过滤后，缩小了数据库规模，大大减少了扫描时间。为了快速产生用户关心的频繁项集，运用正态分布计算阈值。

从顾客购买商品的记录中可知，在购买人数少的情况下，人数与单次购买商品总价格之间的关系会出现离散或者跳跃型的数据，但在人数多的情况下则近似服从正态分布。因此通过确定正态分布公式的参数，可以找到单次价格与人数比例之间的关系。得到正态分布公式后，通过用户所提供的人数比例计算单调性约束和反单调性约束阈值，从而得出最精确的频繁项目集。根据顾客购买记录，人数与价格之间服从的正态分布如图 1 所示。

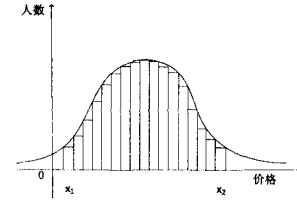

图 1 正态分布图 $X\sim N(\mu,\sigma^2)$

连续型随机变量 X 的分布函数为

$$F(X)=\frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{x}e^{\frac{-(t-\mu)2}{2\sigma^2}}dt(-\infty<x<+\infty) \tag{2}$$

式中，μ,σ 为常数。$\sigma>0$，则称 X 服从参数为 μ。σ 的正态分布记为 $X\sim N(\mu,\sigma^2)$。μ 表示 X 的平均数，σ^2 为方差。首先进行正态分布标准化，令 $Y=(X-\mu)/\delta$，此时 $Y\sim N(0,1)$，服从标准正态分布，记做 $\Phi(Y)$；其次，根据人数比例通过查表后计算得出阈值。

对于标准正态分布的分布函数 $\Phi(Y)$ 的值，当 $Y\geqslant 0$ 时，可查表得出。当 $Y<0$ 时，$\Phi(Y)$ 的值可通过性质 1 确定。

性质 1 设 $X\sim N(0,1)$，则 $\Phi(-x)=1-\Phi(x)$。

定理 1

$$F(X)=\Phi\left(\frac{x-\mu}{\sigma}\right) \tag{3}$$

由式(3)可得出，对 $\forall x_1,x_2\in R,(x_1<x_2)$，得

$$P\{x_1<X\leqslant x_2\}=F(x_2)-F(x_1)$$
$$=\Phi\left(\frac{x_2-\mu}{\sigma}\right)-\Phi\left(\frac{x_1-\mu}{\sigma}\right)$$

计算 X 落入任一区间内的概率归结于计算 $\Phi(X)$ 的值。在标准正态分布下，对于任意实数 x，可将 $P\{x_1<X\leqslant x_2\}$ 转化为 $P\{|X|\leqslant x\}=2\Phi(x)-1=2\Phi\left(\frac{x-\mu}{\sigma}\right)-1$。此时，需要

一个人数的比例 θ，θ 是由用户根据自己需要的客户群而选定的，可以根据市场的需要而变换。与此同时，单调性与反单调性的阈值也因此做出相应调整。下面是阈值变化的证明过程，其中 $\Phi(Y)$ 可通过标准正态分布表查得。

证明：$2\Phi\left(\frac{x-\mu}{\sigma}\right)-1<\theta$（$\theta$ 为用户选择的人数比例）

$$\Phi\left(\frac{x_1-\mu}{\sigma}\right)=\frac{\theta+1}{2}$$

$$x_1=\Phi^{-1}\left(\frac{\theta+1}{2}\right)\sigma+\mu$$

$$x_2=x_1+2(\mu-x_1)$$

3.2.3 挖掘频繁项目集

通过简洁性约束过滤，得到满足约束和支持度的项目集。此时应用 SOTrieIT 结构，能够快速准确获得频繁 1-项集和频繁 2-项集。SOTrieIT 结构的第一层包含了事物数据库中的每一个单独的项并按支持度降序排列，挖掘结果存储在 CSL_1 中。结构的第二层包含在事务数据库中与 CSL_1 中节点相关联的项并同样按支持度降序排列，结果存储在 CSL_2 中。挖掘 SOTrieIT 的过程中需要使用单调性与反单调性约束对事务数据库进一步筛选，动态获得 C_m 和 C_{am} 阈值就显得尤为重要，因为这直接影响频繁项目集的大小和运行时间。

算法 1 挖掘 SOTrieIT

输入：约束类型，最小支持度 min_supp

输出：频繁 1-项集 CSL_1 和频繁 2-项集 CSL_2

```
Begin
for every node x under ROOT
1. {if supp(x)>min_supp
2.   {if(multiple) / * 单调性与反单调性同时存在 * /
3.     for all x where Cam(x)
4.     {add x to csl1 whereCm(x);
5.       For all children y where Cam(xy) add xy to csl2 if Cm(xy);}
6.   else if(anti-monotone) / * 只满足反单调性 * /
7.   for all x where Cam(x)
8.       {add x to CSL1;
9.       for all children y add xy to csl2 where Cam(xy);}
10.  else/ * 只满足单调性 * /
11.       {add x to csl1 where Cm(x);
12.       for all children y add xy to csl2where Cm(xy);}
13.  else break;} / * 没有结点需要检查 * /
14.}
end
```

在上一步基础上，应用 SOTrieIT 挖掘的结果创建 FP-Tree，它的优势是除了频繁 1-项集参与剪枝外，频繁 2-项集也参与修剪事务数据库。在建树过程中，需再次应用单调性和反单调性约束对 FP-Tree 进行缩减，对于不满足约束的项目予以删除，同时将出现在 CSL_1 和 CSL_2 的该项目支持度减 1。删减后不满足 CSL_1 和 CSL_2 约束的项集从事物数据库中删除。本步骤中约束阈值同样影响 FP-Tree 的规模。

算法 2 创建 FP-Tree

输入：事务数据库，CSL_1，CSL_2，最小支持度 minsup

输出：FP-Tree

```
begin
1. if CSL1=∅;
2. break;
3. else
4.    {build a FP-Tree {
5.    for every transaction T∈D
6.        {remove transactions that do not satisfy Cm
7.        reduce support value of CSL1 and CSL2
8.        if(sup<minsup)
9.      {remove the item/itemset from CSL1 and CSL2;
10.       remove items from T not present in CSL1 and CSL2;
11.       remove items from T not present in CSL1 and CSL2;
12.       select and sort the items in T in the order CSL1;
13.       recursively insert all items in T into the tree;}
14.         }
15. to mine the constraints FP-tree call the function as CFP(D,CSL2,
    FCI,Cm,Cam)
end
```

在经过筛选的 FP-Tree 上挖掘符合约束的频繁项集，运用单调性与反单调性约束，得到最终频繁项目集。单调性约束能有效剪除候选数据库中的事务，而反单调性约束能有效降低搜索空间。

算法 3 挖掘频繁项目集

输入：事务数据库，已找到频繁项目集 FCI，CSL_2，C_m，C_{am}

输出：符合约束的频繁项目集

```
Begin
1. for every element ai in CSL2
2. { identify all transactions which contain ai as ai's conditional data-
     base;
3. remove transactions in ai's conditional database which do not satis-
   fy Cm;
4. flistai= set of local frequent items in ai's conditonal database;
5. E=flistai∪{ ai};
6. for all itemsets ai∈E such that supp(ai aj)>min supp and Cam(E)
   and j≠i
7.   { create a conditional database DBai aj
8.     flistai aj=set of all local frequent items in DBai aj;
9.     if Cm(ai aj) then add ai aj to FCI;
10.    call CFP(DB ai aj,flist ai aj,FCI,Cm,Cam);}
16.}
End
```

在算法 3 中，首先为 CSL_2 的每个元素建立条件数据库，删除不满足单调性约束的条件数据库及对应元素，把得到的条件数据库的频繁项目保存至集合 E；用最小支持度过滤 E 中元素与原来元素组成的 2 项集，获得频繁 2-项集；再为频繁 2-项集建立条件数据库，递归调用算法 3，最终得到频繁项目集。

假设数据库中有 t 条交易，每条交易有 n 个项目，则算法 2 需扫描 SOTrieIT 结构 t 遍，时间复杂度为 $O(nt)$。假设经过算法 2 数据库过滤后，在数据库中，平均每条交易的项目为 p 个。在算法 3 中，首先按照支持度大小将交易排序，此步骤的时间复杂度为 $O(p\log(p))$；然后，需要扫描 t 次数据库，平均每条有 p 个项目，时间复杂度为 $O(tp)$。算法 3 将满足条件的项目节点插入 FP-Tree 中，在最坏情况下，时间复杂度为 $\sum_{i=1}^{m}\sup(a_i)$。综合考虑，算法的时间复杂度为 $O(nt)+O(p\log(p))+O(tp)+\sum_{i=1}^{m}\sup(a_i)$。$\sum_{i=1}^{m}\sup(a_i)\leqslant t$ 并且 $p\leqslant t$，那么经简化后，时间复杂度为 $O(n)$。

（下转第 196 页）

结束语 本文针对 FOCUSS 算法迭代公式中容易出现计算病态性的问题，提出了应用增广 Lagrange 函数优化算法解决 l^p 模优化模型，实现信号的重构。该算法通过精确罚函数的方法改良迭代公式，同时应用共轭梯度法加快了算法的收敛速度，整体上提高了信号的恢复精度，使得研究成果更具有实际的意义。

参 考 文 献

[1] Gorodnitsky I F, Rao B D. SparseSignal Reconstruction from Limited Data Using FOCUSS: A reweighted minimum norm algorithm[J]. IEEE Trans, Signal Process, 1997, 45(3): 600-616

[2] Kim S-J, Koh K, Lustig M, et al. An Interior-Point Method for Large-Scale l1-Regularized Least Squares[J]. IEEE Journal on Selected Topics in Signal Processing, 2007, 1(4): 606-617

[3] Donoho D. Compressive sampling[J]. IEEE Trans on Information Theory, 2006, 52(4): 1289-1306

[4] Rao B D, Kreutz-Delgado K. An affine scaling methodology for best basis selection[J]. IEEE Trans. SignalProcess, 1999, 47(1): 187-200

[5] Nocedal J, Wright S J. Numerical Optimization[M]. New York: Springer-Verlag, 2006

[6] Magnusr R. Multiplier and gradientmethods[C]// the Second International Conference on Computing Methods in Optimization Problems. San Remo, Italy, 1968

[7] He Zhao-shui, Cichocki H A, Zdunek R, et al. Improved FOCUSS method with conjugate gradient iterations [J]. IEEE Transactions on Signal Processing, 2009, 57(1): 399-404

[8] Candès E J, Wakin M B, Boyd S P. Enhancing spasity by reweighted l1 minimization[J]. Journal of Fourier Analysis and Applications, 2008, 14(5/6)

[9] Gorodnitskya I F, Georgeb J S, Raoa B D. Neuromagnetic source imaging with FOCUSS: a recursive weighted minimum norm algorithm[J]. Electroencephalography and Clinical Neurophysiology, 1995, 95(4): 231-251

[10] 张贤达. 矩阵分析与应用[M]. 北京：清华大学出版社，2004

(上接第 157 页)

4 实验测试与结果分析

在支持度相同的情况下比较两个算法的运行时间。实验采用 FIMI Reposito 中的 BMS-POS 产品交易记录数据集[8]及由 IBM 提供的数据生成器所产生的合成数据集。FIMI Reposito 数据集包括 515,597 条数据，我们选取其中的10,000条作为实验数据；数据生成器所产生的数据集近似服从正态分布，共 12000 条，平均每条包含 25 个项目。所设计的实验方法如下：设定支持度阈值从 1%到 10%。对于两个数据集，分别比较 FGC 和 NSFGC 算法的运行时间。实验结果如图 2 所示。

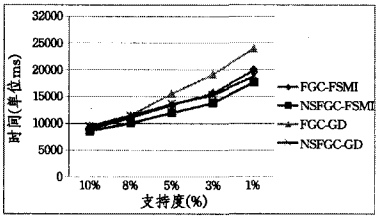

图 2 两种算法运行时间的比较

从上述实验结果可知，NSFGC 算法在支持度为 10%的时候，运行时间要少于 FGC 算法。并且，随着支持度的减小，NSFGC 算法的优势也越发明显。合成数据集近似服从正态分布，NSFGC 更易发挥优势。虽然合成数据集交易比 FSMI 数据集多 2000 条，但其速度能达到 FGC 处理后者的速度。实验证明 NSFGC 算法是快速有效的。

随着人数的变化，两个算法运行时间的变化如图 3 所示。其中，选定支持度为 5%。

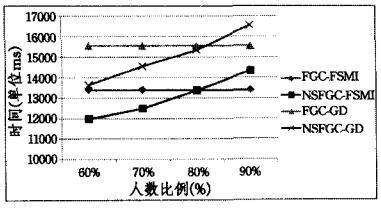

图 3 阈值变化示意

根据用户对购买商品的人数比例的需要，我们选取动态的阈值来达到时间上的优化。通过图 3 看出，随着人数比例减少，本算法运行时间与 FGC 相比明显减少。并且在大部分情况下，本程序都能快速找到频繁项目集。由于人数比例由用户决定，使用户掌握一定的主动性并积极参与挖掘过程，在增加系统可用性的同时提高了算法的执行效率。

结束语 本文提出了一种改进的 FGC 算法。相对于一般的约束关联规则挖掘算法，它更关注与用户交互方面的改进。通过简洁性约束和正态分布的运用，在算法的规模和时间上有了一定程度的改善。未来的工作可以从扩展算法的其他约束或结合频繁闭项目集上着手。

参 考 文 献

[1] Bonchi F, Giannotti F, et al. ExAnte: A preprocessing method for frequent-pattern mining[J]. 2005, 20(3): 25-31

[2] Bonchi F, Lucchese C. Extendingthe state-of-the-art of constraint- based pattern discovery[J]. Data & Knowledge Engineering, 2007, 60(2): 377-399

[3] Agrawal A, Mannila H, et al. Relational data mining [J]. Data Mining and Knowledge Discovery Handbook, Part 6, 2010: 887-911

[4] Erwin A, Gopalan R P, et al. Efficient mining of high utility itemsets from large datasets[J]. Advances in Knowledge Discovery and Data Mining Lecture Notes in Computer Science, 2008, 5012: 554-561

[5] Pears R, Kutty S. FGC: An Efficient Constraint Based Frequent Set Miner[C]// 2007 IEEE/ACS Inter national Conference on Computer Systems and Applications. 2007: 424-431

[6] Seeja K R, Alam M A, Jain S K. An association rule mining approach for co-regulated signature genes identification in cancer [J]. World Scientific Connecting Great Minds, 2009, 18(8): 1409-1423

[7] Leung C K, Lakshmanan L V S, Ng R T. Exploiting Succinct Constraints Using FP-Trees[J]. Column: Constraints in datamining, ublication, 2002: 40-49

[8] http://fimi.cs.helsinki.fi/