维普资讯 http://www.cqvip.com

基于充分挖掘增量事务的关联规则更新算法*)

蔡 进[1,2] 薛永生[1] 林 丽[1] 张东站[1]

(厦门大学计算机科学系 厦门 361005)[1] (三峡大学护理学院计算机中心 宜昌 443000)[2]

摘 要 目前已提出了许多快速的关联规则增量更新挖掘算法,但是它们在处理对新增事务敏感的问题时,往往会丢失一些重要规则。为此,文章提出了一种新的挖掘增量更新后的数据库中频繁项集的算法 EUFIA(Entirety Update Frequent Itemsets Algorithm),该算法先对新增事务数据分区,然后快速扫描各分区,能全面有效地挖掘出其中的频繁项集,且不丢失重要规则。同时,最多只扫描 1 次原数据库也能获得更新后事务数据库的全局频繁项集。研究表明,该算法具有很好的可测量性。

关键词 关联规则,增量式更新,强频繁项集,次频繁项集,弱频繁项集

Updating Algorithm for Association Rules Based on Fully Mining Incremental Transactions

CAI Jin[1,2] XUE Yong-Sheng[1] LIN Li[1] ZHANG Dong-Zhan[1]

(Department of Computer Science, Xiamen University, Xiamen 361005)[1]

(Computer Center of College of Nursing, Three Gorges University, Yichang 443000)[2]

Abstract Incremental Association rules Mining is an important content of data mining technology. This study proposes a new algorithm, called the Entirety Update Frequent Itemsets Algorithm (EUFIA) for efficiently incrementally mining association rules from large transaction database. Rather than rescanning the original database for some new generated frequent itemsets, EUFIA partitions the incremental database logically according to unit time interval, then accumulates the occurrence counts of new generated frequent itemsets and deletes infrequent itemsets obviously by backward method. Thus, EUFIA can discover newly generated frequent itemsets more efficiently and need rescan the original database only once to get overall frequent itemsets in the final database if necessary. EUFIA has good scalability in our simulation.

Keywords Association rules, Incremental updating, Powerful frequent itemsets, Inferior frequent itemsets, Weak frequent itemsets

1 引言

关联规则是由 Agrawal 等人在文[1]中首先提出的一个重要的研究课题。目前已经研究出多种关联规则的快速挖掘方法[1~7],这些算法能有效地挖掘出静态事务数据库中的关联规则。然而,在现实世界事务数据库中,数据是随时间的变化而变化的,使得当前已发现的关联规则可能不再有效,而可能存在新的有效规则有待于进一步去发现。因此设计高效的算法来更新和管理已挖掘出来的关联规则同样是很重要的研究课题。

目前,这方面也有一些研究成果,文[8]中给出的 FUP 算法是最典型的算法之一。它通过对整个数据库重复利用 Apriori 算法来解决更新问题,在应用中可利用已有的频繁项集信息提高算法效率。FUP 算法必须耗费大量时间多次重复扫描原数据库,为提高效率,一些基于 FUP 算法的改进方法也先后出现[9~11],同时也提出了一些不基于 FUP 的算法[12~15]。它们有效减少了对原数据库的扫描次数。但它们往往将新旧事务混合在一起根据最小值支持度来获取全局频繁项集,这在处理对新增事务敏感的问题(例如,销售情况,股票交易,出版物等)时是不利的,往往会丢失一些重要规则。为此,本文提出了一种新的挖掘增量更新后的数据库中频繁项集的算法 EUFIA,它将增量后的事务分为若干个区间进行扫描获得局部频繁项集,并将扫描结果归入不同的 3 个类别,即强频繁项集集合,次频繁项集集合,弱频繁项集集合。这样不需要扫描原数据库便可快速获得一些有用的局部规则。同时根据需要,最多只扫描 1 次原数据库也能获得更新后的全局频繁项集。

2 相关概念与描述

2.1 频繁项集

设 $I=\{i_1,i_2,\cdots,i_n\}$ 是 n 个不同项目的集合。如果对于项集 X 有,$X\subseteq I$,且 $k=|X|$,则 X 称为 k 项集。对给定的事务数据库,记 D 为事务 T 的集合,$T\subseteq I$。对应每一个事务有唯一的标识,如事务号,记作 TID。设 X 是 I 中的一个项集,如果 $X\subseteq T$,则称事务 T 包含 X。定义 X 的支持度计数为 D 中包含 X 的事务个数,记为 $X.CountD$,X 的支持度为 $X.SupD$。用户可自定义一个最小支持度,记为 min_sup。

定义 1 给定事务数据库 D 和最小支持度 min_sup,对于项集 $X\subseteq I$,若 $X.SupD\geqslant min\_sup$,则称 X 为 D 中的频繁项集。

*)基金项目:国家自然科学基金项目(50474033),福建省自然科学基金项目(A0310008),福建省高新技术研究开放计划重点项目(2003H043)。

蔡 进 硕士研究生,讲师;薛永生 教授;林 丽 硕士研究生;张东站 讲师,博士。

维普资讯 http://www.cqvip.com

性质 1 给定 D,若 X 为 D 中的频繁项集,则对任意 $Y\subseteq X$,都有 Y 是 D 中的频繁项集。

推论 给定 D,若 X 为 D 中的非频繁项集,则对任意 $Y\supseteq X$,都有 Y 是 D 中的非频繁项集。

这一性质是由 Agrawal 和 Srikant[2] 提出并证明的,这里就不再赘述。

2.2 Apriori 算法

Apriori 算法主要集中考虑根据用户提出的最小支持度域值(计为 min_sup)发现频繁项集[2]。该算法采用的是迭代方法,多遍扫描事务数据库,并利用性质 1 及推论来减少项目搜索空间。根据这一性质,进行第 k 遍扫描之前,可先产生候选集 C_k。C_k 可以分两步来产生,设前一步(第 $k-1$ 步)已生成频繁 $k-1$ 项集集合 F_{k-1},则首先可以通过对 F_{k-1} 中的成员进行联接来产生候选,F_{k-1} 中的两个成员必需满足如下条件方可联接,

$F_{k-1}\Theta F_{k-1}=\{A\Theta B\mid A,B\subset F_{k-1},|A\cap B|=k-2\}$。

接着,从 C_k 中删除所有包含不是频繁的($k-1$)子集的成员。然后,计算 C_k 中每个成员 X 的支持度计数 $X.CountD$,若 $X.CountD\geqslant |D|\cdot min\_sup$,则 X 计入 F_k,从而产生 F_k。

2.3 FUP 算法

设新增事务集为 d,则其更新后的事务数据库 $D^+=D\cup d$。算法 FUP 的基本思想为:对任意一个 $k(k\geqslant 1)$项集,若其在 D 和 d 中都是频繁的,则其在 D^+ 中同样也是频繁项集;若它仅是 D(或 d)中的频繁项集,则比较 $X.CountD+X.Countd$ 与 $|D^+|\cdot min\_sup$ 的值,若前者较大,则 X 是频繁的,否则 X 是非频繁的[8]。FUP 算法假设在 D 中发现的频繁项集集合 $L=\bigcup_{i=1}^{n}L_i$ 已被保存下来。它基于 Apriori 算法,需要对 D 和 d 进行多次扫描。通常 $|D|\square|d|$,因此需尽量减少对 D 的扫描,但该算法最坏情况下不能减少对原数据库的扫描次数。

3 全面更新频繁项目集算法 EUFIA

为从更新的事务数据库中挖掘新的感兴趣的规则,将新增事务集 d 按时间间隔(如月、季度或年)划分为 n 个部分,即 $d=\bigcup_{i=1}^{n}Q_i$(Q_i 为 d 的第 i 个划分)。为方便起见,可令 0 号区代表原数据库,即 $Q_0=D$,并定义 F_i 为 Q_i 中的频繁项集。定义 $d^{m,n}$ 为 Q_m 到 Q_n 的所有分区的并集,即 $d^{m,n}=\bigcup_{i=m}^{n}Q_i$,$m,n\in N$ 且 $n\geqslant m$,并定义 $F_{m,n}$ 为 $d^{m,n}$ 中频繁项集的集合。显然,$D^+=d^{0,n}$。

定义 2 对项集 X,如果有 $X.SupD\geqslant min\_sup$,且 $X.Supd\geqslant min\_sup$ 成立(设支持度不变),则称 X 为 D(或 d)中的强频繁项集,d 中所有强频繁项集的集合为 Fa。

定理 1 设 F^+ 为 D^+ 的频繁项集集合,Fa 为 D(或 d)中所有强频繁项集集合,则有 $Fa\subseteq F^+$。

证明:由强频繁项集定义,有

$\forall((Fa(X).SupD\geqslant min\_sup)\wedge(Fa(X).Supd\geqslant min\_sup))$

$\Rightarrow\forall X((Fa(X).CountD\geqslant|D|\cdot min\_sup)\wedge(Fa(X).Countd\geqslant|d|\cdot min\_sup))$

$\Rightarrow\forall X(Fa(X).CountD+Fa(X).Countd\geqslant(|D|+|d|)\cdot min\_sup)$

$\Rightarrow\forall X(Fa(X).SupD^+\geqslant min\_sup)$,(其中 $Fa(X).SupD^+$ 为 X 在 D^+ 中的支持度)。

故定理 1 成立。

定义 3 对项集 X,设 X 在 $d^{m,n}$ 中的支持度为 $Supd^{m,n}$,在 $d^{m-1,n}$ 中的支持度为 $Supd^{m-1,n}$ $(1\leqslant m\leqslant n)$,如果有 $X.Supd^{m,n}\geqslant min\_sup$,且 $X.Supd^{m-1,n}<min\_sup$,则称 X 为 $d^{m,n}$ 中的次频繁项集,d 中所有次频繁项集的集合为 Fb。

定义 4 对项集 X,设 X 在 $d^{m,m}$ 中的支持度为 $Supd^{m,m}$,在 $d^{m+1,n}$ 中的支持度为 $Supd^{m+1,n}$ $(0\leqslant m\leqslant n)$,如果有 $X.Supd^{m,m}\geqslant min\_sup$,且 $X.Supd^{m+1,n}<min\_sup$,则称 X 为 $d^{m,n}$ 中的弱频繁项集,D 与 d 中所有弱频繁项集的集合为 Fr。

定理 2 对项集 X,若 $X.SupD^+\geqslant min\_sup$,则 $\exists F_j\in\{F_i\mid i\in N,\text{且 } i\leqslant n\}$($F_i$ 为 Q_i 中的频繁项集),使得 $X\in F_j$。

证明:假设对 $\forall F_i\in\{F_i\mid i\in N,\text{且 } i\leqslant n\}$,都有 $X\notin F_j$,由频繁项集定义,令 $X.SupQ_j$ 为 X 在 Q_j 中的支持度,则对 $\forall Q_j\in\{Q_i\mid i\in N,\text{且 } i\leqslant n\}$ 都有 $X.SupQ_j<min\_sup\Rightarrow X.CountQ_j<|Q_j|\cdot min\_sup$,可得 $\sum_{j=0}^{n}X.CountQ_j<\sum_{j=0}^{n}|Q_j|\square min\_sup\Rightarrow X.CountD^+<|D^+|\cdot min\_sup$,这与题设矛盾。故定理 2 成立。

定理 3 设 F^+ 为 D^+ 的所有频繁项集集合,则有 $F^+\subseteq Fa\cup Fb\cup Fr$。

证明:对任意 $X\in F^+$,由定理 2,$\exists F_j\in\{F_i\mid i\in N,\text{且 } i\leqslant n\}$使得 $X\in F_j$。不妨设 $X\in F_m$ $(0\leqslant m\leqslant n)$,分两种情况讨论,若 $X\notin F_{m+1}$,$n(F_{m+1,n}$ 为 $d^{m+1,n}$ 中频繁项集的集合),由 Fr 的定义知,$X\in Fr$;否则有 $X\in F_{m,n}$。此时又分两种情况,若 $X\notin F_{m-1,n}$,由 Fb 的定义知,$X\in Fb$;否则有 $X\in F_{m-1,n}$。继续使用前面的方法讨论,最终若有 $X\in F_{0,n}$,则由 Fa 的定义知,$X\in Fa$。由上述证明知 $X\in Fa\cup Fb\cup Fr$,故定理 3 成立。

为获得 Fa,Fb,Fr,EUFIA 采用类似定理 3 的构造证明,从 Q_n 向前扫描至 Q_1。扫描过程可利用 Apriori 算法,先生成候选,再过滤。扫描 Q_n 时,其中的频繁项集先放入 Fa 中,在扫描完 Q_1 后,由于 Q_0(即原数据库 D)中的频繁项集计数已知,故只需将 Q_0 中的频繁项集与前面的结果累积计算,便可得 Fa,Fb,Fr。虽然有些应用中,不需要求所有全局频繁项集,但考虑一般情况 EUFIA 仍给出了求所有全局频繁项集的方法。由定理 3 及定理 1 知,要获得 F^+ 只需对 Fb,Fr 中的项集扫描原数据库 D 一次。

算法 1 全面更新频繁项目集算法 EUFIA

输入:(1)原数据库 D 它具有 $|D|$ 的事务数,增量数据库 $d^{1,n}$,其事务数 $|d^{1,n}|=|\sum_{m=1}^{n}Q_m|$;(2)$F_k$:$D$ 中所有频繁 k 项集的集合,$k=1,2,\cdots,g$;(3)s:最小支持度 min_sup。

输出:Fa,Fb,Fr,FDd(即 F^+)。

方法:

/* C_m^k 为分区 m 的候选 k 项集集合,Ft^k 为当前分区频繁 k 项集集合;Fa^k,Fb^k,Fr^k 分别为 Fa,Fb,Fr 中频繁 k 项集集合,F 为 D 中频繁项集集合,Ftb 表示在 Fb 中 start=1 且在 D 中非频繁的项集集合,每个候选或频繁项集有三个属性 count,start,sort */

```
01 Fa=Fb=Fr=Φ,Ftb=Φ;
02 for m=n to 1{
03    k=1,C_m^1=Q_m 中所有项的集合;
04    SelectFabr();
05    for k=2 to g{
06       C_m^k=Apriori-gen(Ft^{k-1});
07          SelectFabr ();}}
```

维普资讯 http://www.cqvip.com

```
08 for X∈Fa∪F{
09   if X∈Fa && X∈F{Fa(X).start=0;Fa(X).count
     +=F(X).count;}
10   else if X∈Fa && X∉F {X.start=1;X.sort=b;Fb
     +={X};Fa-={X};Ftb+={X};}
11   else if X∈F && X∉Fa∪Fb∪Fr {X.sort=r; X.
     start=0; Fr+={X};Fa-={X};}}
12 SelectFDd();
13 return Fa,Fb,Fr,FDd;
SelectFabr()
//扫描分区 m 的 k 项候选,获得当前 Fa^k,Fb^k,Fr^k.
01 if C_m^k=Φ{brea k;}
02 for X∈C_m^k{X.count=0;}
03 Ft^k=Φ
04 for T∈Q_m{
05   for X∈(k-subset of T){
06     if X∈C_m^k{
07       X.count++;
08       if X.count≥s*|Q_m| Ft^k+={X};}}}
09 for X∈C_m^k{
10   if X∈Fa^k{
11     if X.count≥s*|Q_m|{Fa^k(X).start=m;Fa^k(X).
     count+=X.count;}
12     else{
13       if X.count+Fa^k(X).count≥s*|d^{m,n}|
14         {Fa^k(X).start=m;Fa^k(X).count+=X.
     count;}
15       else{X.start=m+1;X.sort=b;
16         X.count=Fa^k(X).count;
17         Fa^k-={X};Fb^k+={X};}}}
18   else if X∈C_m^k-Fa^k-Fb^k-Fr^k{
19     if X.count≥s*|P^m|{
20       if m==n{X.sort=a; Fa^k+={X};}
21       else{X.sort=r;Fr^k+={X};
22         Fr^k(X).start=m;}}}
23 }
SelectFDd()
//扫描 D,d 获得 FDd,这一步骤有些应用中可以省去。CDd 表示候选集(CDd⊆Fb∪Fr)
01 FDd=Fa;
02 CDd=Fb-Ftb-{Fb 中 start=2 的项}+Fr;
03 for X∈CDd{X.count=0;}
04 for T∈d{
05   for X∈CDd && |X|≤|T|{ if X⊆T{X.count+
     +;}}}//扫描 d
06 for X∈CDd{
07   if X.count≥s*|d|{if X.sort==r && X.start==
     0 {CDd-={X}; FDd+={X};}
08   else if X.sort==r&&X.start==0{
09     if F(X).count+X.count ≥s*(|d|+|D|){CDd
     -={X};FDd+={X};}
10     else CDd-={X};}}
11 CDd+=Ftb;
12 for T∈D{ for X∈CDd && |X|≤|T|{ if X⊆T{X.
     count++;}}}//扫描 D
13 for X∈CDd{ if X.count≥s*|d|{FDd+={X};}}
```

表 1 数据库 D 及增量数据集 d

Date	Part	TID	Transaction
1998—2001	D	001	A B C D E
		002	A C D
		003	B E
		004	A B C E
		005	A B E
		006	A B C
		007	C D E
2002	Q1	008	A B C D
		009	B C E
		010	A B E
2003	Q2	011	B C D E
		012	E F
		013	B C F

例 1:设事务数据库 D 及增量事务数据集 d 如表 1 所示,最小支持度为 50%。先列出各分区中的频繁项集,如表 2 所示,由算法可得如表 3 的处理过程。根据表 3 的最终结果,从 1998～2003 年,有{B},{C},{E},{B C}四个项集是持续出现的。{F}在 2003 年(Q_2 中)变得有价值了。{A C}在 1999～2001 年是感兴趣的,但在 2002～2003 年不再是感兴趣的。{A},{A B},{B E}四个项集在 2002 年还是有趣的,但在 2003 年不再受到关注。从全局来看,1998～2003 年,F^+ ={{B},{C},{E},{B C},{A},{B E}}。若按传统方法,{A},{B E}只能被列入 F^+ 中,但实际在 2003 年它们已是非频繁的了。

表 2 各分区中的中频繁项集

D		Q1		Q2	
Itemset	Count	Itemxet	Count	Itemset	Count
{A}	5	{B}	2	{A}	2
{B}	5	{C}	2	{B}	3
{C}	5	{E}	2	{C}	2
{E}	5	{F}	2	{E}	2
{AB}	4	{BC}	2	{AB}	2
{AC}	4			{BC}	2
{BE}	4			{BE}	2

表 3 EUFIA 生成 Fa,Fb,Fr 中频繁项集的过程

Q2 is scanned			Q1 is scammed			Final frequent sets		
Fa	Start	Count	Fa	Start	Count	Fa	Start	Count
{B}	2	2	{B}	1	5	{B}	0	10
{C}	2	2	{C}	1	4	{C}	0	9
{E}	2	2	{E}	1	4	{E}	0	9
{F}	2	2	{BC}	1	4	{BC}	0	7
{BC}	2	2	Fb	Start	Count	Fb	Start	Count
			{F}	2	2	{F}	2	2
			Fr	Start	Count	Fr	Start	Count
			{A}	1	2	{A}	1	2
			{AB}	1	2	{AB}	1	2
			{BE}	1	2	{BE}	1	2
						{AC}	0	4

4 算法性能分析

我们用 VC++ 6.0 在 Intel P4 2.4G/512MB 80GB,操作系统为 WindowsXP 的计算机上实现了 EUFIA 算法,使用了与文[2]同样的生成程序来合成所需要的测试数据。测试数据库的有关参数包括:$|D|$ 表示事务数据记录的数目;$|T|$ 表示事务数据记录的平均长度;$|d|$ 表示新增事务数据记录的数目;Q 表示增量 d 的分区数;$|I|$ 表示最大频繁项集的平均长度;$|L|$ 表示最大频繁项集的数目;N 表示事务项集的个数,而一个测试数据库实例(D^+),记为 $Tx.Iy.Dr.dm.Q_n$。实验中 N=1000,$|L|$=2000,$|T|$=10,$|I|$=4,图 1 表示在 $|D|$=50k,$|d|$=25k,Q=1,2,4 最小支持度变化时算法所用时间。

从实验结果看,EUFIA 算法在 d 只有 1 个分区时,执行时间最少。考虑算法的应用,取 Q=2。图 2 表示在 $|d|$ 变化时算法所用时间,其中支持度分别为 0.2% 及 0.4%,其它条件不变。从实验结果看,算法运行时间随 $|d|$ 增大线性增长。图 3 表示在 $|D|$ 变化时算法所用时间,其中支持度分别为

(下转第 233 页)

维普资讯 http://www.cqvip.com

明暗光照。这种方法明显地减少了传统的基于三维纹理映射的体绘制方法的巨大内存占用，并且加快了整个绘制过程。由实验结果可见，利用目前通用图形硬件灵活的纹理操作和强大的光栅化能力，可以得到理想的绘制结果，使得体绘制更具实用和推广价值。

参 考 文 献

1 Van Gelder A, Kim K. Direct Volume Rendering with Shading via Three-Dimensional Textures [C]. In: Proc. of Volume Visualization Symposium, San Francisco, CA, October 1996. 23～30
2 Cabral B, Cam N, Foran I. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware [C]. In: Workshop on Volume Visualization, Washington, DC, USA, October 1994. 91～98
3 Rezk-Salama C, Engel K, Bauer M, et al. Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-texturing and Multi-stage Rasterization [C]. In: Proc. of Eurographics/SIGGRAPH Workshop on Graphics Hardware, Interlaken, Switzerland, August 2000. 109～118
4 Dachille F, Kreeger K, Chen B, et al. High-Quality Volume Rendering Using Texture Mapping Hardware [C]. In: Proc. of Eurographics/SIGGRAPH Workshop on Graphics Hardware, Lisbon, Portugal, August 1998. 69～76
5 Kniss J, Kindlmann G, Hansen C. Multidimensional Transfer Functions for Interactive Volume Rendering [J]. IEEE Transactions on Visualization and Computer Graphics, 2002, 8(3): 270～285
6 Meißner M, Hoffmann U, Straßer W. Enabling Classification and Shading for 3D Texture-based Volume Rendering Using OpenGL and Extensions [C]. In: Proc. of IEEE Visualization, San Francisco, CA, USA, October 1999. 207～214
7 Westermann R, Ertl T. Efficiently Using Graphics Hardware in Volume Rendering Applications [C]. In: Proc. of ACM SIGGRAPH, Orlando, FL, USA, August 1998. 169～177
8 ATI Corporation. ATI Developer Relation Site. Available at: http://www.ati.com/developer

(上接第 222 页)

0.2%及 0.4%。这是由于 D 变化，而考虑了求全局频繁集合 F^+。从实验结果看，算法运行时间随 $|D|$ 变大而基本呈线性增长。图 1、图 2、图 3 的实验结果也表明了 EUFIA 算法的有效性和较好的可测量性。

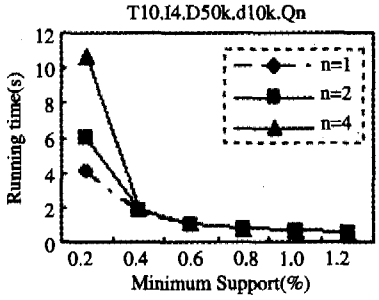

图 1 最小支持度变化时 EUFIA 运行时间

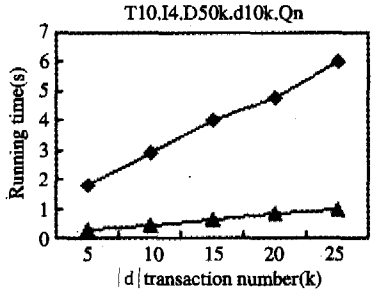

图 2 $|d|$ 增大时 EUFIA 运行时间

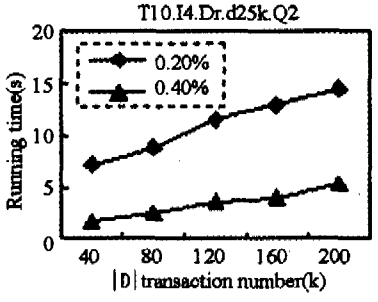

图 3 $|D|$ 增大时 EUFIA 运行时间

结束语 在实际应用中，事务数据库是不断变化的，并且许多时候新增事务比旧的事务更加有价值，更值得去充分挖掘。在更新中如何保证既快速又不丢失有效的规则是一个很重要的问题。为此，本文提出了一个全面更新频繁项集的算法 EUFIA，它将新增事务划分成若干分区进行扫描，并将结果归入不同的 3 个类别，从而达到了在不扫描原数据库的情况下，充分、快速地挖掘增量事务，获取频繁项集，且又不丢失有效的规则。另外，该算法的设计思想可以方便地应用到多层关联规则和多层序列模式的发现中，我们将对此作进一步的研究。

参 考 文 献

1 Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. In: Proceedings of ACM SIGMOD International Conference on Management of Date, Washington D C, 1993. 207～216
2 Agrawal R, Srikant R. Fast algorithm for mining association rules. In: Proceedings of the 20th International Conference on VLDB, Santiago, Chile, 1994. 487～499
3 Han J, Kamber M. Data Mining: Concepts and Techniques. Beijing: Higher Education Press, 2001
4 Han J, Jian P, et al. Mining frequent patterns without candidate generation. In: Proceedings of ACM SIGMOD International Conference on Management of Data, Dallas, TX, May 2000. 1～12
5 Shoemaker C, Ruiz C. Association Rule Mining Algorithms for Set-valued Data. In: Proc. 4th Intl. Conf. on Intelligent Data Engineering and Automated Learning. LNCS Vol. 2690, Hong Kong, china, 2003. 669～676
6 Wang Jianyong, Han J, Lu Y, Tzvetkov P. TFP: an efficient algorithm for mining top-k frequent closed itemsets. Knowledge and Data Engineering, IEEE Transactions, 2005, 17(5): 652～663
7 颜跃进，李舟军，陈火旺．频繁项集挖掘算法．计算机科学，2004, 31(3): 112～114
8 Cheung D W, et al. Maintenance of discovered association rules in large databases: an incremental updating technique. In: Proceedings of the 12th International Conference on Data Engineering, New Orleans, Louisana, 1996. 106～114
9 Cheung DW, LEE SD, Kao B. A general incremental technique for maintaining discovered association rules. In: Topor RW, Tanaka K, eds. Proc. of the 5th Int'l Conf. on Database Systems for Advanced Applications. World Scientific, 1997. 185～194
10 Hong T P, Wang C Y, Tao Y H. A new incremental data mining algorithm using pre-large itemsets. Intelligent Data Analysis, 2001, 5(2): 111～129
11 Ayan N F, Tansel A U, Arkun E. An efficient algorithm to update large itemsets with early pruning. In: Proc. 5th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, San Diego, Aug. 1999. 287～291
12 Otey M E, Parthasarathy S, Wang C, Veloso A, Meira W Jr. Parallel and distributed methods for incremental frequent itemset mining. Systems, Man and Cybernetics, Part B, IEEE Transactions, 2004, 34(6): 2439～2450
13 Veloso A A, Meira W Jr, de Carvalho M B, Pösas B, Parthasarathy S, Zaki M J. Mining frequent itemsets in evolving databases. In: Proc. 2nd SIAM Intl. Conf. on Data Mining, Arlington, TX, May 2002. 494～510
14 Lee C H, Lin C R, Chen M S. Sliding-window filtering: An efficient algorithm for incremental mining. In: Proc. 10th Intl Conf. on Information and Knowledge Management, Atlanta, Nov. 2001. 263～270
15 朱玉全，孙志挥，季小俊．基于频繁模式树的关联规则增量式更新递法．计算机学报，2003, 26(1): 91～96