数据流上近似非可导项集的挖掘算法

黄崇争[1),2)] 李海峰[3)] 陈 红[1)]

[1)](中国人民大学信息学院 北京 100872)

[2)](广西建设职业技术学院 计算机与信息技术系 南宁 530003)

[3)](中央财经大学信息学院 北京 100081)

摘 要 频繁项集是通过对大规模数据进行挖掘获取的代表数据模式的知识结构.非可导频繁项集作为频繁项集的有效压缩方式,能够高效深入地挖掘海量数据、稠密数据与数据流当中的规律.针对项集在计算界限值时代价昂贵的缺点,提出了近似可导项集的概念,并基于纵向数据格式实现了挖掘算法 MANDI,能够提高支持度计算和项集间操作的速度.另外,为了满足数据流实时、快速的特点,讨论并证明了近似可导项集的增量性质,提出了可动态更新的算法 UANDI.通过实验验证了两种算法的可行性和有效性.

关键词 近似非可导频繁项集;纵向数据格式;数据流;数据流挖掘

中图法分类号 TP18 **DOI 号**:10.3724/SP.J.1016.2010.01427

An Approximate Non-Derivable Itemset Mining Algorithm over Data Streams

HUANG Chong-Zheng[1),2)] LI Hai-Feng[3)] CHEN Hong[1)]

[1)](*School of Information, Renmin University of China, Beijing* 100872)

[2)](*Technological Institute of Guangxi Construction, Nanning* 530003)

[3)](*The Central University of Finance and Economics, Beijing* 100081)

Abstract Frequent itemset mining is one of the traditional and important problems in data mining. Non-derivable frequent itemsets are the condensed reprentation of frequent itemsets, and they can not only reduce the memory cost, but also make association rules more understandable for user. Because the bound computations of non-derivable frequent itemsets are high, the authors propose the conception of approximate non-derivable itemsets, and present an approximate non-derivable frequent itemset mining algorithm MANDI based on itemset idlist. In addition, the authors present the stream mining algorithm UNADI, which maintains the negative borders of approximate non-derivable frequent itemsets to conduct efficient incremental mining. The experimental results show that both algorithms are effective and efficient.

Keywords approximate non-derivable frequent itemsets; vertical dataset; data stream; data stream mining

1 引 言

频繁项集是 Agrawal 在 1994 年提出来的[1],其动机是为了寻找超市事务数据的频繁集以分析客户的购买行为.近年来随着在众多行业中广泛的应用,频繁项集挖掘受到了广泛关注,逐渐发展成为数据挖掘中重要的分支,其目标是在数据集中寻找用户感兴趣的模式,例如关联规则[2]、数据相互关系[3]和序列模式[4]等.随着网络及大规模商业集成等应用

收稿日期:2010-06-11.本课题得到国家"八六三"高技术研究发展计划项目基金(2008AA01Z120)资助.**黄崇争**,男,1978 年生,博士研究生,主要研究方向为数据流处理、数据挖掘.E-mail:52708332@qq.com.**李海峰**,男,1979 年生,博士,讲师,主要研究方向为数据流处理、数据挖掘.**陈 红**,女,1965 年生,教授,博士生导师,主要研究领域为数据仓库、数据挖掘、数据流处理和传感器数据管理.

的出现,海量数据、稠密数据和数据流的产生,频繁项集挖掘逐渐不能满足用户的需要,这主要体现在两个方面:一方面,在数据高度相关或者最小支持度设置过低的情况下,会生成大量的频繁项集,这些频繁项集的数量甚至超过了原有数据集中数据的数量,浪费了大量的系统资源;另一方面,频繁项集本身包含了大量的冗余信息,影响了用户的判断.因此,简单的频繁项集挖掘对于用户来说已经意义不大.

研究人员开始寻找频繁项集的压缩表示方法,希望去除频繁项集中的冗余数据,实现有效的数据精简.迄今为止,已经找到了包含闭合频繁项集[5-7]、最大频繁项集[8-9]、无关频繁项集[10]和非可导频繁项集[11-15]在内的多种压缩表示.在这些压缩表示中,非可导频繁项集被证明是最小的压缩表示方式[14],不但可以节省大量系统资源,而且能够使用户更好地理解关联规则[15],因此具有重要的意义.

非可导频繁项集是在 2002 年根据演绎规则[11]提出来的.目前已经提出的挖掘非可导项集的算法有 NDI[11]、QIE[12]以及 dfNDI[13]等.NDI 是最简单的基于 Apriori 的宽度遍历算法,通过候选集的生成,扫描数据库得到频繁项集,并进一步计算以判断项集是否可导;dfNDI 利用一种新颖的方法,通过项集的逆序分析实现了深度遍历;另外,QIE 针对非可导项集计算边界时的巨大计算量进行了改进,用数组来保存需要边界计算的项集,有效提高了性能;文献[15]将非可导项集应用到关联规则挖掘上,取得了良好的效果.文献[14]综述了非可导项集挖掘的方法,并与其它频繁项集的压缩表示方式进行了比较.但是,这些算法都是针对静态数据集来挖掘精确的结果,具有以下的缺点:首先,一旦数据增加,算法需要重新运行来获取新的结果;其次,由于需要多遍扫描数据库来计算支持度,算法的计算代价偏高;最后,在项集变长的情况下,决定项集是否可导的计算量迅速增加,计算代价昂贵.综上所述,以上算法都不能有效处理数据动态变化的情况.

本文在这些研究的基础上,从挖掘结果的近似性,改变数据格式和增量式挖掘 3 个方面来考虑数据流上非可导项集挖掘,主要贡献有

(1)提出了近似非可导项集的概念,可以有效地实现搜索空间的剪枝和计算代价的降低,通过设置近似参数,可以实现算法性能与结果精确度的折衷.

(2)提出了基于纵向数据格式的近似非可导项集挖掘算法,只需要扫描数据集一次就可以完成支持度计算,降低了算法的运行时间.

(3)目前国内外还没有关于近似非可导项集增量式挖掘的工作.本文对近似非可导项集的性质进行了研究,证明了近似非可导项集在新数据库中的不变性,通过维护近似非可导项集的临界项集集合实现增量式更新算法,以处理动态变化的数据流数据.

(4)在两种模拟数据集和两种真实数据集上进行了算法的实验验证,并与迄今为止性能最好的非可导项集挖掘算法 dfNDI 进行了比较,实验结果表明了本文两种算法的可行性和有效性.

本文第 2 节介绍非可导项集的相关知识;第 3 节提出近似非可导项集的概念与基于 *id* 列表的纵向数据格式;第 4 节提出近似非可导项集的挖掘算法 MANDI;第 5 节提出近似非可导项集的增量更新算法 UANDI;第 6 节是算法的时间复杂度分析;第 7 节是实验;第 8 节是本文的总结.

2 预备知识

2.1 频繁项集

频繁项集是数据集中发生次数不小于用户定义阈值的项集.用 $\Gamma=\{i_1,i_2,\cdots,i_n\}$ 来表示所有不同项的集合,其中 $|\Gamma|=n$ 表示 Γ 的数量,称长度为 k 的 Γ 的子项集 X 为 k-项集.为了简单起见,可以把项集 $X=\{x_1,x_2,\cdots,x_m\}$ 表示为 $x_1x_2\cdots x_m$.给定数据集 $D=\{T_1,T_2,\cdots,T_v\}$,每一个 $T_i(i=1\cdots v)$ 表示一个基于 Γ 的事务,包含事务 *id* 和对应的项集 X.对于事务 $T=(tid,X)$ 来说,如果存在项集 Y 使得 $Y\subseteq X$,则称事务 T 支持项集 Y,所有 D 中支持项集 Y 的事务集合称为 Y 的覆盖,表示为 $cover(Y,D)=\{T|T\subseteq D\wedge Y\subseteq X\wedge X\in T\}$. Y 的绝对支持度是其覆盖集合的大小,表示为 $\Lambda(X,D)=|cover(X,D)|$,其相对支持度是 Y 在 D 中发生的概率,即 Y 的绝对支持度与数据集 D 大小的比值,表示为 $\Lambda_r(Y,D)=\frac{\Lambda(Y,D)}{|D|}$.给定最小支持度阈值 γ,如果 $\Lambda(Y,D)\geq\gamma$,则称项集 Y 为频繁项集.

2.2 纵向数据格式

数据集的纵向表达方式是指采用二元组(*item*,*tidlist*)形式表示的数据库,即每个项目 i 对应覆盖该项集的事务 *id* 列表 $TS(i)$.给定数据库 D,对于项集 $I=\{i_1,i_2,\cdots,i_m\}$ 来说,I 的事务 *id* 列表可以通过每个项目的事务 *id* 列表得到,即 $TS(I)=TS(i_1)\cap TS(i_2)\cap\cdots\cap TS(i_m)$,项集 I 的支持度 $\Lambda(I)$ 即 $TS(I)$ 中含 *id* 的数量.

例 1. 图 1 中的数据集可以表示为(*itemset*,*tidlist*)的形式.可以看到,项目 a 被全部事务覆盖,

对应的 id 列表为 $TS(a)=\{1,2,3,4,5\}$，因此 a 的支持度为 5. 对于项集 ab，其对应的 id 列表 $TS(ab)=TS(a)\cap TS(b)=\{2,3,5\}$，因此 ab 的支持度为 3.

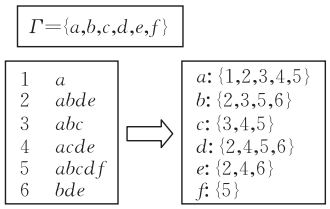


图 1　纵向数据集

2.3　演绎规则

非可导项集的概念来源于演绎规则[11].

给定项目 a，a 的逆表示为 $\bar{a}$. 一个事务 T 如果覆盖 $\bar{a}$ 即 T 不覆盖 a. 一个泛化项集就是项目与项目的逆组成的集合，给定项集 $I=X\cup Y$，基于 I 的泛化项集表示为 $G=X\bar{Y}$. 如果一个事务 $T=\{tid,J\}$ 满足 $X\subseteq J$ 且 $T\cap J=\varnothing$，则称 T 支持 G. 例如，在 $\Gamma=\{a,b,c,d\}$ 的情况下，给定泛化项集 $G=ab\overline{cd}$，那么对于数据集 $\{ab,abc\}$ 来说，因为 ab 覆盖 a 和 b，但不覆盖 c 和 d，因此 ab 支持 G；而 abc 虽然覆盖 a 和 b，但同样覆盖 c，因此 abc 不支持 G.

演绎规则是通过包含排除(inclusion-exclusion)原则[16]推导出来的，给定基于项集 $I=X\cup Y$ 的泛化项集 $X\bar{Y}$，演绎规则可以表示为 $\Lambda(X\bar{Y})=\sum_{X\subseteq J\subseteq I}(-1)^{|J/X|}\Lambda(J)$，从而得到以下的定理.

定理 1[11]. 令 Γ 表示不同项目的集合，对于项集 I，如果 $X\subseteq I\subseteq\Gamma$ 且 $Y=I/X$，则有以下不等式成立.

$$\Lambda(I)\leq\sum_{X\subseteq J\subset I}(-1)^{|I\setminus J|+1}\Lambda(J),\ |I\setminus X|\text{是奇数}\quad(1)$$

$$\Lambda(I)\geq\sum_{X\subseteq J\subset I}(-1)^{|I\setminus J|+1}\Lambda(J),\ |I\setminus X|\text{是偶数}\quad(2)$$

例 2.　表 1 是项集 $I=abc$ 针对每个 $X\subseteq I$ 的演绎规则 $R_X(I)$.

表 1　abc 的演绎规则

I	X	$R_X(I)$
abc	ϕ	$\Lambda(abc)\leq\Lambda(ab)+\Lambda(ac)+\Lambda(bc)-\Lambda(a)-\Lambda(b)-\Lambda(c)+\Lambda(\phi)$
abc	a	$\Lambda(abc)\geq\Lambda(ab)+\Lambda(ac)-\Lambda(a)$
abc	b	$\Lambda(abc)\geq\Lambda(ab)+\Lambda(bc)-\Lambda(b)$
abc	c	$\Lambda(abc)\geq\Lambda(ac)+\Lambda(bc)-\Lambda(c)$
abc	ab	$\Lambda(abc)\leq\Lambda(ab)$
abc	ac	$\Lambda(abc)\leq\Lambda(ac)$
abc	bc	$\Lambda(abc)\leq\Lambda(bc)$
abc	abc	$\Lambda(abc)\geq0$

2.4　非可导项集

从项集 I 的演绎规则可以得到 I 的支持度 $\Lambda(I)$ 基于每个 $X\subseteq I$ 的界限 $ul_X(I)$. 当 $|I\setminus X|$ 是奇数的时候，$ul_X(I)$ 为 $\Lambda(I)$ 的上限，表示为 $u_X(I)$，$u_X(I)$ 的最小值称为 $\Lambda(I)$ 的最小上限，表示为 $mu(I)$；当 $|I\setminus X|$ 为偶数的时候，$ul_X(I)$ 为 $\Lambda(I)$ 的下限，表示为 $l_X(I)$，$l_X(I)$ 的最大值称为 $\Lambda(I)$ 的最大下限，表示为 $ml(I)$. 因此，$\Lambda(I)$ 的最大下限和最小上限组成了 $\Lambda(I)$ 的范围区间，即 $\Lambda(I)\in[ml(I),mu(I)]$. 如果 $ml(I)=mu(I)$，那么 $\Lambda(I)=ml(I)=mu(I)$，称 I 为可导项集，可导项集是单调的，即对于两个项集 S 和 T 且 $S\subseteq T$，如果 S 是可导的，那么 T 也是可导的；反之，如果 $ml(I)\neq mu(I)$，则称 I 为非可导项集. 非可导项集有以下性质.

性质 1[14].　给定项集 $I\subset\Gamma$ 和项目 $a\in\Gamma\setminus I$，有

$$mu(I\cup a)-ml(I\cup a)\leq\frac{mu(I)-ml(I)}{2}.$$

性质 2[14].　项集 I 的支持度 $\Lambda(I)$ 的范围区间大小 $mu(I)-ml(I)$ 随着项集长度增大呈指数趋势降低. 给定事务数据库 D 和项集 $I\subset\Gamma$，如果 $|I|>\log_2(|D|)+1$，那么 I 一定是可导的.

例 3.　给定事务数据集 $\{abc,ab,ac,bc\}$，对于 abc 来说，有

$$\Lambda(abc)\leq\begin{cases}u_{ab}(abc)=2\\u_{ac}(abc)=2\\u_{bc}(abc)=2\\u_{\phi}(abc)=1\end{cases}\Rightarrow\Lambda(abc)\leq1$$

和

$$\Lambda(abc)\geq\begin{cases}l_a(abc)=1\\l_b(abc)=1\\l_c(abc)=1\\l_{abc}(abc)=0\end{cases}\Rightarrow\Lambda(abc)\geq1,$$

即 $\Lambda(abc)\in[1,1]$，所以 abc 是可导项集；对于 ab 来说，有

$$\Lambda(ab)\leq\begin{cases}u_a(ab)=3\\u_b(ab)=3\end{cases}\Rightarrow\Lambda(ab)\leq3$$

和

$$\Lambda(ab)\geq\begin{cases}l_{\phi}(ab)=2\\l_{ab}(ab)=0\end{cases}\Rightarrow\Lambda(ab)\geq2,$$

即 $\Lambda(ab)\in[2,3]$，所以 ab 是非可导项集. 另外，根据性质 1，因为 $ab\subset abc$，所以 $mu(abc)-ml(abc)=0\leq(mu(ab)-ml(ab))/2=1/2$ 成立.

3　近似非可导项集

3.1　定义与性质

根据定理 1 可知，对于一个项集 I 来说，如果 I 的长度为 m，计算 I 范围区间的代价最多达到了 2^m 次，已经成为非可导项集挖掘中耗时最多的计算之

一,为了尽快实现剪枝,防止非可导项集的长度太大,提出了近似的非可导项集的概念,来降低运行时间的代价.

定义 1(近似非可导项集). 给定整数 $\theta \geq 0$,对于项集 I 来说,如果存在 $X,Y \subseteq I$ 使得 $u_X(I) - l_Y(I) \leq \theta$,则称项集 I 为近似可导项集,反之称 I 为近似非可导项集.θ 称为近似参数.

近似非可导项集是非可导项集的扩展,当 $\theta=0$ 的时候,近似非可导项集挖掘变为精确的非可导项集挖掘.通过近似参数的调整,可以提高非可导项集挖掘的灵活性和健壮性,保证在不同的计算环境下用尽可能少的时间得到合理的结果.

性质 3. 近似可导项集是单调的.

证明. 令近似参数为 θ,给定项集 S,T 和项目 a,$T=S \cup a$,如果 S 是近似可导项集,则存在 $X,Y \subseteq S$ 使得 $u_X(S) - l_Y(S) \leq \theta$,根据性质 1 可知,至少存在 $X',Y' \subseteq T$ 使得 $u_{X'}(T) = mu(T)$ 和 $l_{Y'}(T) = ml(T)$,即存在 $X',Y' \subseteq T$,使得 $u_{X'}(T) - l_{Y'}(T) = mu(T) - ml(T) \leq \frac{mu(S) - ml(S)}{2} \leq \frac{u_X(S) - l_Y(S)}{2}$

$\Rightarrow u_{X'}(T) - l_{Y'}(T) \leq \theta$,所以 T 也是近似可导项集.

证毕.

近似非可导项集具有 3 方面的优势:首先,近似可导项集的单调性允许算法进行搜索空间的剪枝,可以减少项集的存储代价;其次,与可导项集必须基于每个项集的子项集计算界限相比,近似可导项集的界限计算有了更容易获取的目标,只需要根据项集的部分子项集来进行计算就可以得到期望的结果;最后,由于近似可导项集提供了更宽松的范围区间,因此可以使得项集的范围区间能够更加快速地收敛,从近似非可导项集变为近似可导项集,不仅可以提前剪枝,而且进一步减少了项集界限计算的代价.

例 4. 对于图 1 的数据集来说,令最小支持度 $\gamma=2$,近似参数 $\theta=1$.对于项集 ab 来说,$\Lambda(ab) \in [3,4]$,当采用精确的挖掘方法时,ab 是非可导项集,因此其超集仍然可能是非可导项集,因此需要进一步生成 ab 的超集 $\{abc,abd,abe\}$,并计算相应的支持度和界限;当采用近似的挖掘方法时,ab 是近似可导项集,因此其超集都是近似可导项集,可以全部被裁剪.对于项集 abc 来说,假定需要计算其界限,当采用精确的挖掘方法时,需要根据 abc 的子集 $\{ab,ac,bc,a,b,c\}$ 分别计算界限,计算的数量为 6 次;当采用近似的方法时,当计算出 $u_{bc}(abc)=2$ 和 $l_a(abc)=1$ 时,由于 $u_{bc}(abc) - l_a(abc) = 1$,即可以停止计算,计算的数量为 4 次.

3.2 数据结构

为了描述分析各种搜索空间的剪枝策略,避免冗余项集的生成和检测,引入项集之间的词典序.给定项目 a、b 和 c,定义三者的词典序 $a \prec b \prec c$,通过这种方法,可以建立数据库所有项集的唯一顺序,在图 1 的数据库中,对于项集 ab,ac,abc 来说,其顺序为 $ab \prec abc \prec ac$.

由于采用纵向数据格式不需要重新扫描数据库即可以得到支持度,因此在以往采用纵向数据格式的算法中,无论是频繁项集挖掘算法 Eclat[17]、闭合频繁项集挖掘算法 CHARM[7],还是最大频繁项集挖掘算法 MAFIA[9],都具有良好的性能.因此本文借鉴了以往基于纵向数据格式的算法,应用于非可导项集的挖掘.

采用了三元组 $\langle it,sup,TS \rangle$ 来保存每个项集.其中 it 用来保存项集,每个项集根据字符的词典序排序,可以提高项集的交并操作的速度,降低冗余项集的检测和生成代价;sup 用来保存项集在数据集中的支持度;TS 用来保存数据集中覆盖项集 it 的事务的 id 列表.

4 近似非可导项集挖掘算法 MANDI

定义 2(临界近似可导项集). 给定项集 I,如果 I 是近似可导的,并且任给 $J \subset I$,J 都是近似非可导的,则称项集 I 为临界近似可导项集.

定义 3(临界非频繁项集). 给定项集 I,如果 I 是非频繁的,并且任给 $J \subset I$,J 都是频繁的,则称项集 I 为临界非频繁项集.

MANDI 算法是基于 Apriori 的宽度遍历算法,采用基于非频繁项集的单调性来裁剪非频繁项集,采用基于近似可导项集的单调性来裁剪近似非可导项集.为了实现剪枝,MANDI 算法保存了近似非可导频繁项集的临界项集集合:一部分是最有希望成为频繁项集的临界非频繁项集的集合 IFS_D,另一部分是最有希望成为近似非可导项集的临界近似可导项集的集合 DIS_D.给定图 1 的前 4 个事务,最小支持度 $\gamma=2$,近似参数 $\theta=1$,如图 2 所示,可以得到近似非可导项集集合 $\{a,b,c,d,e,de\}$、临界近似可导项集集合 $\{ab,ac,ad,ae\}$ 和临界非频繁项集集合 $\{f,bc,bd,be,cd,ce\}$.可以看到,尽管 $abc,abd,abe,acd,ace,bde,cde$ 都是非频繁项集,但由于它们有非频繁的子集,因此他们不是临界非频繁项集,可以被剪裁掉;而尽管 ade 是近似可导项集,但它有可导子

集$\{ad,ae\}$，因此它不是临界近似可导项集，可以被剪裁掉.

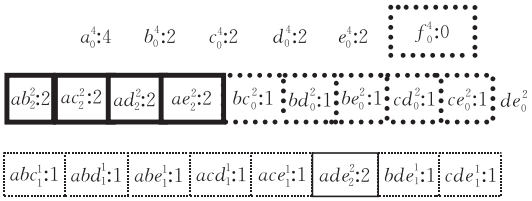


图 2　$NDIS$、DIS 和 IFS

对于每个项集 I，MANDI 算法通过 I 的子集的事务 id 列表来计算支持度，如果 I 是非频繁的，将 I 存入临界非频繁项集集合 IFS_D，并且停止生成 I 的超集；如果 I 是频繁的，则计算 I 的界限，如果 I 是近似可导的，将 I 存入临界近似可导项集集合 DIS_D，否则将 I 存入近似非可导项集集合 $NDIS_D$，并生成 I 的超集.

算法 1.　MANDI$(D,\gamma,\theta,NDIS_D,DIS_D,IFS_D)$.

输入：数据集 D，相对最小支持度 γ，近似参数 θ

输出：近似非可导项集集合 $NDIS_D$，临界近似可导项集集合 DIS_D，临界非频繁项集集合 IFS_D

```
begin
1. 扫描数据集 D,生成 D 的纵向表达方式 D_v;
2. l=1; NDIS_D={}; DIS_D={}; IFS_D={};
3. C_l=D_v;  //C_l表示长度为 l 的项集的集合;
4. while C_l非空 do
5.    Gen={};
6.    foreach C_l中的项集 I do
7.       if Λ(I)≥γ then
8.          根据每个 X⊆I 计算 I 的上限 u_X(I)和下限
            l_X(I);
9.          if 存在 X,Y⊆I 使得
               u_X(I)−l_Y(I)≤θ
            then
10.            DIS_D=DIS_D∪I;
11.         else
12.            NDIS_D=DNIS_D∪I;
13.            Gen=Gen∪I;
14.         endif
15.      else
16.         IFS_D=IFS_D∪I;
17.      endif
18. end foreach
19. 生成 Gen 中项集的超集集合 C_{l+1};
20.    根据 Gen 中项集的 id 列表计算 C_{l+1}中项集的 id
       列表和支持度;
21.    l=l+1;
22. end while
end
```

例 5.　假设数据集 D 为图 1 中的前 4 个事务，令最小支持度 $\gamma=2$，近似参数 $\theta=1$. MANDI 算法的执行过程如下所示：

1. 扫描数据集 D，有 $C_1=\{\langle a,4,(1,2,3,4)\rangle,\langle b,2,(2,3)\rangle,\langle c,2,(3,4)\rangle,\langle d,2,(2,4)\rangle,\langle e,2,(2,4)\rangle,\langle f,0,()\rangle\}$，得到 $NDIS_D=\{a,b,c,d,e\}$，$DIS_D=\{\}$，$IFS_D=\{f\}$ 和 $Gen=\{a,b,c,d,e\}$.

2. 生成 C_2 为 $\{\langle ab,2,(2,3)\rangle,\langle ac,2,(3,4)\rangle,\langle ad,2,(2,4)\rangle,\langle ae,2,(1,2)\rangle,\langle bc,1,(3)\rangle,\langle bd,1,(2)\rangle,\langle be,1,(2)\rangle,\langle cd,1,(4)\rangle,\langle ce,1,(4)\rangle,\langle de,2,(2,4)\rangle\}$，得到 $NDIS_D=\{a,b,c,d,e,de\}$，$DIS_D=\{ab,ac,ad,ae\}$，$IFS_D=\{f,bc,bd,be,cd,ce\}$ 和 $Gen=\{de\}$.

3. 生成 C_3 为空，算法结束.

5　近似非可导项集更新算法 UANDI

定理 2[18].　给定一个增加的事务 $S=\{tid,T\}$ 和项集 I，令 $N=T\cap I$，对于每个项集 $X\subseteq I$，用 $l_X(I)$ 和 $u_X(I)$ 表示 I 在原数据集 D 中的下限和上限，用 $l'_X(I)$ 和 $u'_X(I)$ 表示 I 在新数据集 $D\cup S$ 中的下限和上限，有以下公式成立.

如果 $N\subset I$ 并 $N=X$，有

$$\begin{cases} l'_X(I)=l_X(I)-1, & |I\backslash X| \text{是偶数} \\ u_X'(I)=u_X(I)+1, & |I\backslash X| \text{是奇数} \end{cases} \tag{3}$$

如果 $N\subset I$ 并且 $N\neq X$，有

$$\begin{cases} l'_X(I)=l_X(I), & |I\backslash X| \text{是偶数} \\ u'_X(I)=u_X(I), & |I\backslash X| \text{是奇数} \end{cases} \tag{4}$$

如果 $N=I$，有

$$\begin{cases} l'_X(I)=l_X(I)+1, & |I\backslash X| \text{是偶数} \\ u'_X(I)=u_X(I)+1, & |I\backslash X| \text{是奇数} \end{cases} \tag{5}$$

推论 1[18].　给定项集 I，增加一个新事务 $S=\{tid,T\}$ 只会使得 $mu(I)-ml(I)$ 的值增大，且最多增大 1.

推论 2.　增加事务不会将近似非可导项集变为近似可导项集，但可以将近似可导项集变为近似非可导项集.

证明.　给定近似参数 θ，对于近似非可导项集 I 来说，$mu(I)-ml(I)>\theta$，根据推论 1，增加新事务会提高 $mu(I)-ml(I)$ 的值，所以 $mu(I)-ml(I)>\theta$ 仍然成立，所以 I 仍然是非可导项集；对于近似可导项集 I 来说，$mu(I)-ml(I)\leq\theta$，增加新事务可以提高 $mu(I)-ml(I)$ 的值，所以 $mu(I)-ml(I)$ 可能变为大于 θ 的正整数，所以 I 会变为近似非可导项集.　证毕.

根据推论 2 可以知道，原数据集中的近似非可导项集在增加数据后中仍然是近似非可导的. 这种情况下，只要检测原数据集的临界项集是否改变，并

进行相应的扩展,就可以得到新数据集的近似非可导项集集合.

令原数据集为 D,新增数据集为 d,则合并后的新数据集为 $D\cup d$. 在增加 d 后,采用 UANDI 算法对 d 进行扫描,更新原数据集的近似非可导项集的支持度,并重新计算临界项集集合的支持度和界限,就可以生成更新后的近似非可导项集集合 $NDIS_{D\cup d}$,临界近似可导项集集合 $DIS_{D\cup d}$ 和临界非频繁项集集合 $IFS_{D\cup d}$.

算法 2. UANDI($D,d,\gamma,\theta,NDIS_{D\cup d},DIS_{D\cup d},IFS_{D\cup d}$).

输入:原数据集 D,新增数据集 d,相对最小支持度 γ,近似参数 θ

输出:近似非可导项集集合 $NDIS_{D\cup d}$,临界近似可导项集集合 $DIS_{D\cup d}$,临界非频繁项集集合 $IFS_{D\cup d}$

```
begin
1. 扫描数据集 d,更新 NDIS_D,DIS_D 和 IFS_D 中项集的 id 列表和支持度;
2. foreach DIS_D 中的项集 I do
3.    根据每个 X⊆I 计算 I 的上限 u_X(I)和下限 l_X(I);
4.    if 任给 X,Y⊆I 都有 u_X(I)-l_Y(I)>θ then
5.       将 I 从 DIS_D 移到 NDIS_D 中;
6.          调用过程 Explore(NDIS_D,DIS_D,IFS_D,I,γ,θ);
7.    end if
8. end foreach
9. foreach IFS_D 中的项集 I do
10.   if Λ(I)≥γ then
11.      根据每个 X⊆I 计算 I 的上限 u_X(I)和下限 l_X(I);
12.      if 存在 X,Y⊆I 使得 u_X(I)-l_Y(I)≤θ then
13.         将 I 从 IFS_D 移到 NDIS_D 中;
14.      调用过程 Explore(NDIS_D,DIS_D,IFS_D,I,γ,θ);
15.      else
16.         将 I 从 IFS_D 移到 DIS_D 中;
17.      endif
18.   endif
19. end foreach
end
Explore(NDIS_D,DIS_D,IFS_D,I,γ,θ)
全局变量:队列 Gen={};
begin
1. foreach NDIS_D 中与 I 长度相同的项集 J do
2.    if |I∩J|=|I|-1 then
3.       K=I∪J;
4.       根据 I 与 J 的 id 列表计算 K 的 id 列表和支持度;
5.       if Λ(K)≥γ then
6.          根据每个 X⊆K 计算 I 的上限 u_X(K) 和下限 l_X(K);
7.          if 存在 X,Y⊆K,使得 u_X(K)-l_Y(K)≤θ then
8.             DIS_D=DIS_D∪K;
9.          else
10.            NDIS_D=NDIS_D∪K;
11.            push(Gen,K);
12.         endif
13.      else
14.         IFS_D=IFS_D∪K;
15.   endif
16. end foreach
17. while Gen 非空 do
18.      K=pop(Gen);
19.      调用过程 Explore(NDIS_D,DIS_D,IFS_D,K,γ,θ);
20. end while
end
```

例 6. 假设图 1 中的前 4 个事务为原数据集 D,后 2 个事务为新增数据集 d,令最小支持度 $\gamma=2$,近似参数 $\theta=1$. UANDI 算法的执行过程如下所示.

1. 扫描数据集 d,得到更新后的近似非可导项集集合 $NDIS_D=\{\langle a,5,(1,2,3,4,5)\rangle,\langle b,4,(2,3,5,6)\rangle,\langle c,3,(3,4,5)\rangle,\langle d,4,(2,4,5,6)\rangle,\langle e,3,(2,4,6)\rangle,\langle de,3,(2,4,6)\rangle\}$,近似可导项集集合 $DIS_D=\{\langle ab,3,(2,3,5)\rangle,\langle ac,3,(3,4,5)\rangle,\langle ad,3,(2,4,5)\rangle,\langle ae,2,(2,4)\rangle\}$ 和非频繁项集集合 $IFS_D=\{\langle f,0,()\rangle,\langle bc,2,(3,5)\rangle,\langle bd,2,(2,5)\rangle,\langle be,2,(2,6)\rangle,\langle cd,2,(4,5)\rangle,\langle ce,1,(4)\rangle\}$.

2. 对于 DIS_D 中的每个项集分别计算上下界,ab,ac,ad,ae 仍然可导,因此 $NDIS_D$ 和 DIS_D 没有变化.

3. 对于 IFS_D 中的新频繁项集计算上下界,可知 bc,bd,be,cd 变为非可导项集,因此得到 $NDIS_D=\{a,b,c,d,e,bc,bd,be,cd,de\}$,$DIS_D=\{ab,ac,ad,ae\}$ 和 $IFS_D=\{f,ce\}$.

4. 对 $NDIS_D$ 中新的近似非可导项集 bc,bd,be,cd 分别调用过程 Explore,得到 $NDIS_D=\{a,b,c,d,e,bc,bd,be,cd,de\}$,$DIS_D=\{ab,ac,ad,ae,bde\}$ 和 $IFS_D=\{f,ce,bcd,bce\}$.

5. 结束.

6 算法时间复杂性分析

如算法 1 所示,MANDI 算法的实际运行时间取决于非可导项集的数量和长度,并且与近似参数 θ 相关:非可导项集的数量越多,且平均长度越大,则运行时间越长;θ 越小,精确度越高,需要完成的界限计算量越大,则运行的时间越长. 假设长度为 $i(1\leq i\leq n)$ 的非可导项集的数量为 m_i,且计算非可导项集上下界限的平均时间为 t,则 MANDI 算法的时间复杂度为 $O\left(t\sum_{i=1}^{n}m_i\right)$. 需要注意的是,尽管 t 的

大小随着项集长度的增大而呈指数级增长，但实际性能却要好得多. 首先，给定数据集 D，对于一个项集 I 来说，其界限计算代价不是随着项集 I 的长度增长无限变大的，从性质 2 可知，如果 $|I|>\log_2(|D|)+1$，则 I 一定是可导的，即使在最坏情况下最小支持度为 1 时，其界限计算的时间复杂度为 $O(2^{\log_2(|D|)+2}|I|)=O(|D||I|)$；其次，根据定义 1 可知，项集的界限不需要进行完全计算即可获得；最后，由于只需要考虑挖掘非可导项集，因此真正需要计算的结点并不多.

如算法 2 所示，UANDI 算法的实际运行时间与临界项集的数量有关，但主要取决于新增数据集中的非可导项集的数量和长度. 这是因为每次生成一个非可导项集的时候，需要调用 Explore 函数来生成新的项集. 其时间复杂度与 MANDI 算法相同.

7　实　验

用 Visual C++ 在内存 2GB，CPU 为 Xeon 2.0GHz，操作系统为 Windows Server 2003 的机器上实现了 MANDI 和 UANDI 算法，并与目前性能最优的非可导项集挖掘算法 dfNDI 进行比较. 另外，采用了 4 种标准的数据集作为实验的测试数据集：其中包括两种由 IBM 的数据生成器生成的模拟数据集 T10I4D100K 和 T40I10D100K 以及两种 KDDCUP 2000 采用的真实数据集 BMS-POS 和 BMS-Webview，其中 BMS-POS 是商品零售数据，BMS-Webview 是电子商务网站的点击流数据. 表 2 列举了 4 种数据集的主要数据特征. 为了便于比较，给定数据集 D，以下实验均采用绝对支持度完成挖掘，绝对支持度的值通过给定的相对支持度与 $|D|$ 的乘积得到.

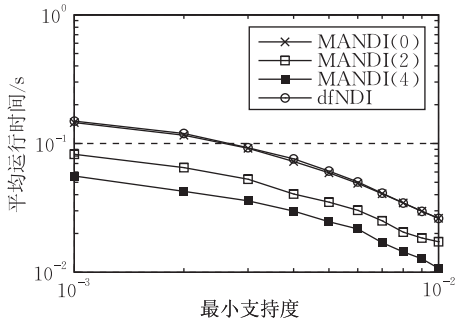
表 2　数据集的主要特征

数据集	事务数量	平均事务长度	最小事务长度	最大事务长度	项目数量
T10I4D100K	100000	10.1	1	29	870
T40I10D100K	100000	39.6	4	77	942
BMS-POS	515597	6.5	1	164	1657
BMS-Webview	59601	2.5	1	267	497

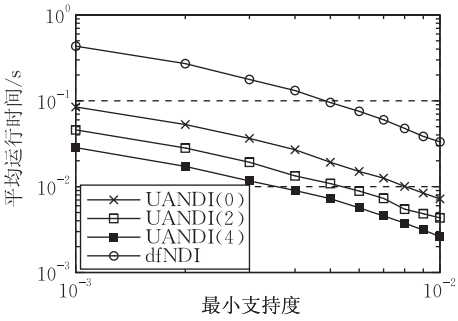
7.1　静态数据中的性能比较

选取了每个数据集的前 80% 作为静态数据集. 图 3～图 6 的(a)部分显示了算法 MANDI 和 dfNDI 在不同支持度下的平均运行时间代价(每事务)的比较，为了体现近似性，MANDI 分别采用近似参数 $\theta=0,2,4$ 完成挖掘. 之所以这样取值，是因为根据性质 1，项集支持度的范围区间随着项集长度的增大是呈 2 的指数级递减，因此 θ 以 2 的指数作为取值可以实现最大程度的优化.

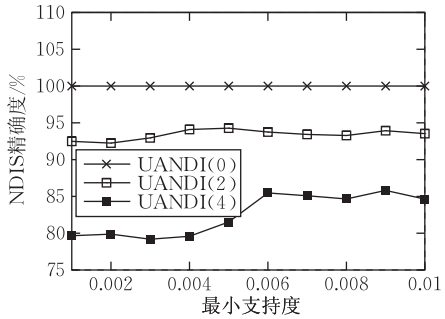
MANDI 采用了纵向数据格式，能够迅速完成支持度计算，而 dfNDI 由于采用了深度遍历数据的方式，在一定程度上能够提高数据查找的速度，并减少冗余数据的产生. 可以看到，在 $\theta=0$ 的时候，MANDI 实际上是完成精确的挖掘，除了在数据集 T10I4D100K 上的性能与 dfNDI 相似外，MANDI



(a) MANDI与dfNDI的性能比较

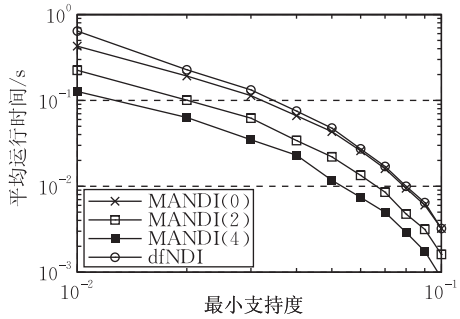


(b) UANDI与dfNDI的性能比较

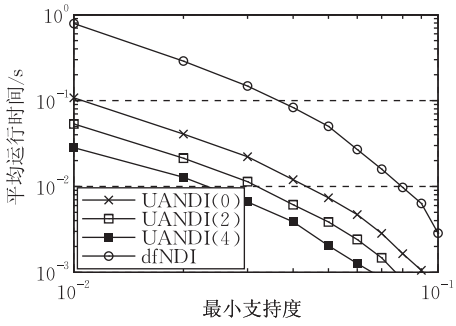


(c) UANDI的精确度

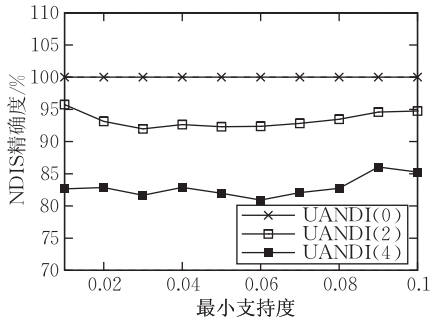
图 3　T10I4D100K 数据集



(a) MANDI与dfNDI的性能比较

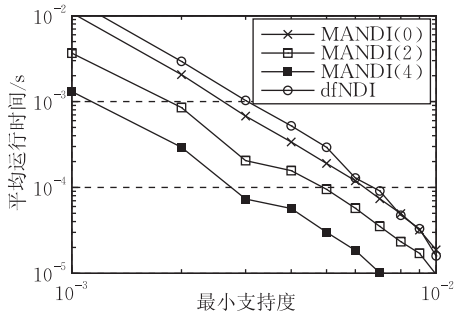


(b) UANDI与dfNDI的性能比较

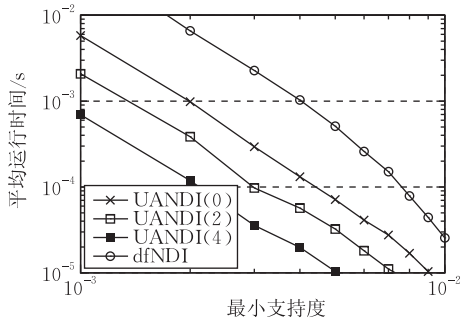


(c) UANDI的精确度

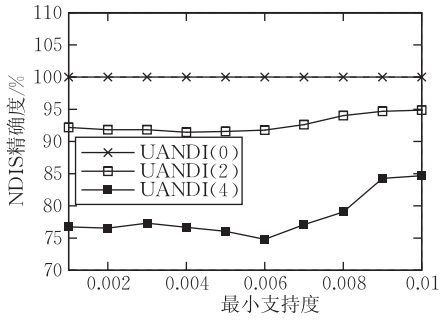
图 4 T40I10D100K 数据集



(a) MANDI与dfNDI的性能比较



(b) UANDI与dfNDI的性能比较



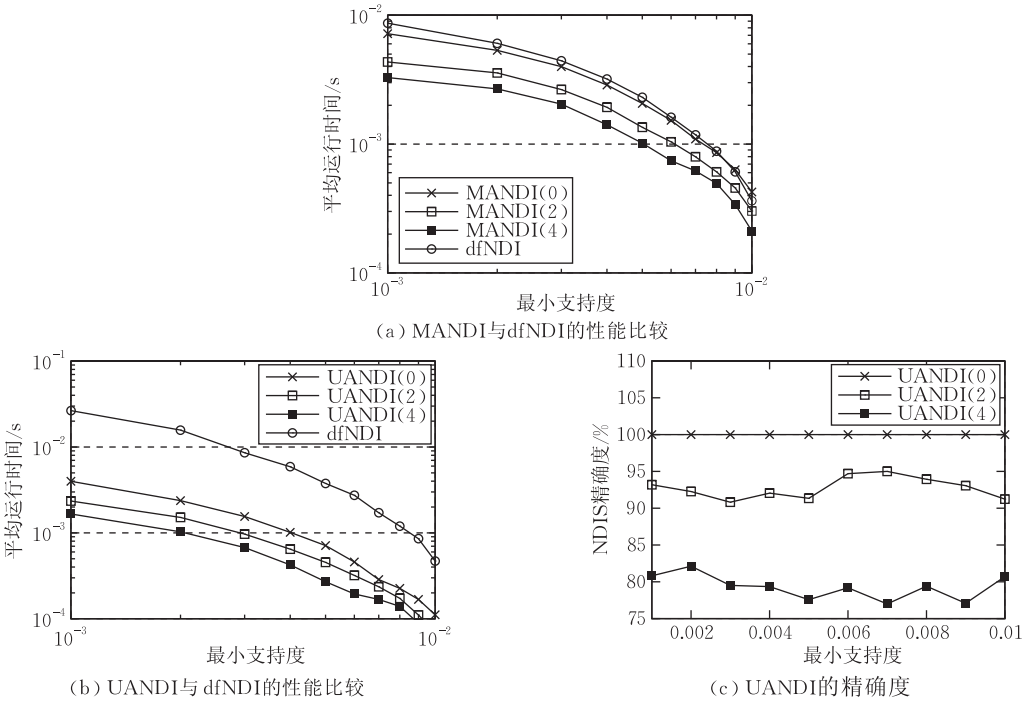
(c) UANDI的精确度

图 5 BMS-POS 数据集

在其它数据集上的性能均优于 dfNDI. 当近似参数 θ 增大的时候,减少了非可导项集的数量和项集界限的计算量,MANDI 算法的性能有了进一步的提高. 从图中可以看到,θ 的值越大,其计算时间越少.

7.2 动态数据中的性能比较

选取了每个数据集的前 80%作为原数据集,剩余的 20%作为新增数据集. 图 3～图 6 的(b)部分显示了算法 UANDI 和 dfNDI 在不同支持度下的平均运行时间代价(每事务)的比较. 可以看到,即使在 $\theta=0$ 的时候,UANDI 的性能仍然要远高于 dfNDI,这是因为与 dfNDI 需要重新对整个数据库进行挖掘相比,UANDI 只需要考虑处理原数据集中的临界项集集合即可. 这样不需要重复计算原有数据集中项集的界限,节省了大量的计算开销. 从图中可以看到,UANDI 提高了至少 2 倍以上的计算性能,最多能够提高 10 倍左右. 同样,在 θ 增大的情况下,



(a) MANDI与dfNDI的性能比较

(b) UANDI与dfNDI的性能比较

(c) UANDI的精确度

图 6　BMS-Webview 数据集

UANDI 算法的性能有了进一步的提高.

随着 θ 的变大,尽管 UANDI 算法中非可导项集数量变小,但这是因为最深层的非可导项集变成了临界项集,所以 UANDI 算法不会产生错误的非可导项集,即 UANDI 算法产生的结果一定是精确结果的子集或真子集. 因此,我们只比较了 UANDI 算法的精确度. 图 3～图 6 的(c)部分显示了在不同的 θ 取值情况下 UANDI 算法挖掘结果的精确度,即 UANDI 算法与 dfNDI 算法的的非可导项集数量的比值. 可以看到,当 $\theta=0$ 的时候,UANDI 算法挖掘的结果精确度为 100%,随着 θ 的增大,精确度有所降低,当 $\theta=2$ 的时候,精确度能够保证在 90%以上,当 $\theta=4$ 的时候,精确度有了比较明显的下降,但仍然保持在最低 75%的水平之上.

综合以上的实验,可以看到 MANDI 算法尽管只是采用广度遍历的方式,但在支持度计算方面进行了优化,仍然能够达到提高性能的目的;而 UANDI 算法则通过增量式的更新来处理新增数据,减少了数据的重复计算,使得性能有了一个数量级的飞跃;另外,由于引入了近似参数,使得算法的实现更加灵活,能够在保证用户要求精确度的同时进一步降低计算代价,从图中可以看到,算法能够在保证精确度 90%以上的情况下将性能提高 3～5 倍.

8　总　结

非可导项集挖掘是频繁项集挖掘的重要分支之一. 本文提出了一种快速灵活的基于纵向数据格式的近似非可导项集挖掘方法 MANDI,通过近似参数的调整,来实现性能与精确度的有效折衷. 另外,本文针对数据流的特点对数据的增量更新进行了研究,证明了非可导项集在新增数据的情况下的不变性,提出了近似非可导项集的增量更新算法 UANDI,通过维护近似非可导项集的临界项集集合实现增量式的挖掘. 实验证明这两种算法是可行的、有效的.

参　考　文　献

[1] Agrawal R, Imielinski T, Swami A N. Mining association rules between sets of items in large databases//Proceedings of ACM SIGMOD the International Conference on Management. Washington, D.C., USA, 1993: 207-216

[2] Agrawal R, Srikant R. Fast algorithms for mining association rules//Proceedings of VLDB the Very Large Databases. Santiago de Chile, Chile, 1994: 487-499

[3] Xiong H, Tan P-N, Kumar V. Hyperclique pattern discovery. DMKD the Data Mining and Knowledge Discovery, 2006, 13(2): 219-242

[4] Agrawal R, Srikant R. Mining sequential patterns//Proceedings of the 11th International Conference on Data Engineering (ICDE). Taipei, China, 1995: 3-14

[5] Pei J, Han J, Mao R. CLOSET: An efficient alogrithm for mining frequent closed itemsets//Proceedings of the ACM/SIGMOD International Workshop on Research Issues on Data Mining and Knowledge Discovery. Dallas, TX, USA, 2000: 21-30

[6] Wang J, Han J, Pei J. CLOSET+: Searching for the best strategies for mining frequent closed itemsets//Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington, D.C.,

USA, 2003: 236-245
[7] Zaki M J, Hsiao C. CHARM: An efficient algorithm for closed itemset mining//Proceedings of SDM the SIAM International Conference on Data Mining. Arlington, VA, USA, 2002: 457-473
[8] Gouda K, Zaki M J. Efficiently mining maximal frequent itemsets//Proceedings of ICDM the IEEE International Conference on Data Mining. San Jose, California, USA, 2001: 163-170
[9] Burdick D, Calimlim M, Gehrke J. MAFIA: A maximal frequent itemsets algorithm for transactional databases//Proceedings of International Conference on Data Engineering (ICDE). Heidelberg, Germany, 2001: 443-452
[10] Calders T, Goethals B. Minimal k-free representations of frequent sets//Proceedings of ECML PKDD the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. Cavtat, Dubrovnik, Croatia, 2003: 71-82
[11] Calders T, Goethals B. Mining all non-derivable frequent itemsets//Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery. Helsinki, Finland, 2002: 74-85
[12] Calders T, Goethals B. Depth-first non-derivable itemset mining//Proceedings of SDM the SIAM International Conference on Data Mining. Newport Beach, California, USA, 2005: 207-208
[13] Hamrouni T, Ben Yahia S. Sweeping the disjunctive search space towards mining new exact concise representations of frequent itemsets. Data & Knowledge Engineering, 2009, 68 (10): 1091-1111
[14] Calders T, Raïssi Chedy. Mining conjunctive sequential patterns. Data Mining and Knowledge Discovery, 2008, 17(1): 77-93
[15] Calders T, Goethals B. Non-derivable itemset mining. DMKD the Data Mining and Knowledge Discovery, 2007, 14 (1): 171-206
[16] Galambos J, Simonelli I. Bonferroni-Type Inequalities with Applications. New York, USA: Springer, 1996
[17] Zaki M J, Parthasarathy S, Ogihara M, Li W. New algorithms for fast discovery of association rules//Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD). Newport Beach, California, USA, 1997: 283-286
[18] Li H, Chen H. Mining non-derivable frequent itemsets over stream. DKE the Data & Knowledge Engineering, 2009, 68 (5): 481-498

HUANG Chong-Zheng, born in 1978, Ph. D. candidate. His research interests include data stream and data mining.

LI Hai-Feng, born in 1979, Ph. D., lecturer. His research interests include data stream and data mining.

CHEN Hong, born in 1965, professor, Ph. D. supervisor. Her research interests include data warehouse, data mining and data management in wireless sensor network.

Background

The concept of frequent itemset mining was firstly proposed in 1994 by Agrawal. In the recent years, with the wide applications in industry, frequent itemset mining has been developed into an important branch of data mining. An itemset is frequent if its support is more than the threshold specified by users. The goal of frequent itemset mining is looking for the data model which users may be interested at, such as association rules, sequential patterns and so on.

With the growing number of large-scale business applications and the emergence of new data types, such as massive data, density data and stream data, traditional frequent itemset mining can not meet the needs of users, mainly in two aspects: the first, a large number of frequent itemsets will be generated if minimum support is set lowly; the second, frequent itemsets themself include a lot of redundant information, consuming a lot of unnecessary store resource and affecting the user's judgments. Therefore, researchers began to look for the method to compress frequent itemsets. So far, several method have been proposed, such as closed itemset, maximal itemset, non-derivable itemset and so on. In the above method, non-derivable itemset is considered as the smallest and lossless condensed representation.

Recently, many non-derivable itemset mining algorithms have been proposed. Calders and Goethals used the a priori-based method NDI to mine non-derivable itemsets; Calders and Goethals presented the depth-first non-derivable itemset mining method dfNDI to improve performance; Quick inclusion-exclusion principles were proposed based on arrays to accelerate the non-derivable itemset mining speed; Calders and Goethals surveyed non-derivable itemset mining in comparison to mining of the other condensed representations; However, all of these algorithms are over static data sets and unsuitable for data stream mining. There are some reasons as follows. First of all, once the data is increased, these algorithms need to re-run to obtain new results; Secondly, these algorithms need to scan the database many times to calculate the support, which consumes expensive computational resource; Finally, in the case of variable-length item sets, to determine whether a itemset is derivable, we need to perform costly computation also. So, the above algorithm can not effectively process the data stream.

The paper aims to test the feasibility of online incremental mining of non-derivable itemsets. This ppaper proposes the conception of approximate non-derivable itemsets, and presents an approximate non-derivable frequent itemset mining algorithm MANDI based on itemset idlist. In addition, it presents the stream mining algorithm UNADI, which maintains the negative borders of approximate non-derivable frequent itemsets to conduct efficient incremental mining. The experimental results show that both algorithms are effective and efficient.