一种为保密挖掘预处理数据的新方法

刘 亮 谢舒婷 李顺东
(陕西师范大学计算机科学学院 西安 710062)

摘 要 Apriori算法是数据挖掘中一个里程碑式的经典算法,在该算法的思想上衍生出了许多通过产生频繁项集来导出关联规则的算法。提出了一种基于数据项闭包的、为保密数据挖掘进行数据预处理的全新方法。该方法针对类Apriori算法的特点和过程对不同特点的数据项进行不同的变换,使得挖掘请求方既能够正确地得到自己想要得到的关于己方产品的信息,又无法正确地得到关于潜在的竞争对手的信息。因此数据提供方在采用该方法预处理数据后,能够提供有利于双方的数据,以达到双赢的目的。

关键词 Apriori算法,隐私保护,数据项闭包,数据挖掘
中图法分类号 TP391 **文献标识码** A

New Data Preprocessing Method for Privacy-preserving Data Mining

LIU Liang XIE Shu-ting LI Shun-dong
(School of Computer Science, Shaanxi Normal University, Xi'an 710062, China)

Abstract The Apriori algorithm is a milestone in the development of data mining. A number of other algorithms, which generate association rules by producing frequent itemsets, are derived from it. This paper proposed a data preprocessing algorithm based on item closure, which is an absolutely new method for privacy-preserving data mining. According to the characteristics and the processes of Apriori-like algorithms, this method transforms different items in various ways. So the data mining applicant can only obtain the information of its own products correctly, but cannot obtain useful information regarding its potential competitors' products. Hence, through the application of this method, the cooperated data mining between the data provider and data miner will benefit both sides, and obtains a win-win result.

Keywords Apriori algorithm, Privacy-preserving, Item closure, Data mining

1 引言

关联规则挖掘是由R Agrawal等人于1993年在文献[1]中首先提出的一个重要的数据挖掘(Data Mining)研究方向。不久,R Agrawal和R Srikant又提出了关联规则挖掘中的经典算法:Apriori算法[2]。该算法的核心思想是通过逐层搜索和连接生成频繁项集,进而产生关联规则。其后许多类似的算法也是基于这个思想。而只要是基于产生频繁项集的算法,都必然会受到两种非平凡开销的影响[3]:在面对大量的候选项集或挖掘长频繁模式时会产生大量频繁项集,带来巨大的甚至是无法承受的开销。而这是本文所要利用的性质之一。

随着计算机网络和数据库以及数据仓库技术的不断发展,数据挖掘研究和应用的范围越来越广,新的问题也在不断产生。在因特网上,安全问题是一个永恒的话题。在过去单一的应用领域里,数据挖掘的安全问题似乎并没有那么重要,然而当数据挖掘的领域越来越广,无论是用户类型还是用户数量都在飞速增长时,安全问题变得前所未有的重要起来[4]。数据挖掘的安全问题所涉及的情况相当多。例如当数据来源于不同的用户或者组织,而两方需要联合挖掘才能得到结果时,如何同时保护双方的数据[5];当出现了多个数据库之间共享信息以应对用户查询、检索等请求时,如何更少地泄露己方的信息[6];某些大型的通过大量的调查来收集数据然后进行挖掘的活动中,如何保护个人隐私等。

保密的数据挖掘,也称隐私保护的数据挖掘(PPDM, privacy-preserving data mining),在未来将会是一个无论在研究领域还是在应用领域都有着光明前景的课题,近年来已有不少学者在这方面做出了卓有成效的工作[5-13]。其中就目前的研究现状来看,主要沿着两个大方向来设计相关的隐私保护的数据挖掘方法:一种是基于密码学的方法[5],其借鉴了多方安全计算(secure multiparty computation)和不经意传输(oblivious transfer)的思想;另一种是基于数据变换的方法,或称为数据扰动(data perturbation)[9],其通过随机响应技术来达到隐私保护的目的。本文立足于数据挖掘的数据预处理部分,采用对数据进行变换的方法,同时利用数据项闭包的性质来达到对数据项进行变换,然后进行保密挖掘的目的。

2 问题引入和相关描述

2.1 问题的引入

考虑实际生活中的如下情况,某大型连锁超市同时销售

到稿日期:2010-08-16 返修日期:2010-11-12 本文受国家自然科学基金(60673065,61070189),陕西省自然科学基金(2008K01-58)资助。
刘 亮(1985-),男,硕士生,主要研究方向为密码学、保密数据挖掘等,E-mail:biaoxyz@foxmail.com;谢舒婷(1985-),女,硕士生,主要研究方向为密码学与网络信息安全;李顺东(1963-),男,教授,博士生导师,主要研究方向为密码学、多方保密计算。

多个品牌的纸巾。A 公司提出挖掘该超市销售记录的事务数据库以获得在销售环节中的一些有用信息，如顾客的购买喜好等。这样有利于 A 公司减少其存储、运输或者其他过程的花费，并且能够提供更加及时和高效的服务。当然，作为回报，A 公司会对该超市进货时提供一个更优惠的价格。表面看这是一个双赢的请求，但是当 A 公司在挖掘中得到了竞争对手 B 的信息后，可能会针对 B 公司产品的销售特点采取一些针对性措施。如当 A 公司发现顾客经常会同时购买某种品牌的牛奶和 B 公司的纸巾这个特点时，可以推出策略，如果同时购买该品牌的牛奶和 A 公司的纸巾，那么纸巾的价格可以便宜 50 美分[14]。

表面上看，此时 A 公司仅仅打击了竞争对手 B，跟超市似乎还没有多大关系。但是，文献[14]为我们描述了接下来的境况，当 B 的经营越来越惨淡时，超市的纸巾供应就越来越依赖 A 公司，此时 A 公司就可以趁火打劫，在谈判时开出种种有利于自己的条件，不但刚开始用作交换的优惠条件可以收回，甚至可以向超市提出更多的要求。

然而，如果超市考虑到了这点，拒绝 A 公司挖掘自己数据库的请求又将会如何呢？很显然，这样做白白浪费了一个对两方都有好处的机会。因此，现在的问题就是，如何能够既让 A 通过挖掘正确地得到关于自己商品的隐含的销售信息，又能保护 B 公司，乃至 C 公司、D 公司等等那些 A 的竞争对手的信息不被其在挖掘过程中发现。当然，仅仅删除含有代表 B 公司产品的数据项 I_B 的事务，或者仅仅在这些事务中删除 I_B 是不行的。前者有可能误删含有 A 的事务，从而影响 A 对自己信息的挖掘。而后者 A 公司很明显地就可以发现超市的手法，并提出反对意见和重新提供数据库的要求。

2.2 相关内容的表示和分析

在这部分我们主要进行一些预备工作，给出一些将要用到的概念、定义等内容，并对下文做出一些铺垫性的分析。

概念 1　设 $I=\{I_1,I_2,\cdots,I_n\}$ 是项的集合。设任务相关的事务数据库 D 是事务 T 的集合，其中每个事务 T 是项的集合，使得 $T\subseteq I$。每个事务有一个唯一的标识符，称作 TID。设 A 是一个项集，事务 T 包含 A 当且仅当 $A\subseteq T$。关联规则是形如 $A\Rightarrow B$ 的蕴涵式，其中 $A\subset I, B\subset I$，并且 $A\cap B=\varnothing$。规则 $A\Rightarrow B$ 在事务集 D 中成立，具有支持度 s，其中 s 是 D 中包含 $A\cup B$ 的百分比，它是概率 $P(A\cup B)$。规则 $A\Rightarrow B$ 在事务集 D 中具有置信度 c，则是指 D 中包含 A 事务的同时也包含 B 的百分比，这是条件概率 $P(B|A)$。即

$$support(A\Rightarrow B)=P(A\cup B)$$

$$confidence(A\Rightarrow B)=P(B|A)$$

挖掘关联规则就是产生那些支持度和置信度分别大于用户给定的最小支持度阈值(min_sup)和最小置信度阈值(min_conf)的规则，阈值可以由用户或领域专家根据经验设定[3]。

定义 1(数据项闭包)　事务数据库 D 中某一个数据项 I_j 的闭包是该数据库中所有含有项 I_j 的事务 T 的并集。引入一个新的符号 I_j^* 来表示关于项 I_j 的闭包，即 $I_j^*=\bigcup_{i=1}^{|D|}T_i(I_j\in T_i)$。其中 $|D|$ 表示待变换数据库 D 的事务总数。

数据项的闭包是本文的核心定义，其思想借鉴了近世代数中关系的闭包。从非形式化的角度讲，一个数据项 I_j 的闭包 I_j^* 内的数据项都是有可能与 I_j 结合进而产生关联规则的。而闭包以外的数据项绝不可能与 I_j 同时在一条关联规则内出现。因为从定义 1 可知，若 $I_k\notin I_j^*$，则不存在事务 T_i 同时满足 $I_j\in T_i$ 且 $I_k\in T_i$，因此 $P(I_j\cup I_k)=0$。所以我们从所要保护的项的闭包出发来构造算法，就既不会遗漏可能与待保护项有关的数据项，也不会多做无用功。

定义 2(多个数据项闭包)　多个数据项的闭包是在定义 1 的基础上进行的扩展。事务数据库 D 中多个项 $\{I_{j_1},I_{j_2},\cdots,I_{j_k}\}$ 的闭包在表示时只需对下标做出相应的改变即可，其含义为该数据库中含有 $I_{j_1},I_{j_2},\cdots,I_{j_k}$ 中至少一项的事务的并集。如 $\{I_1,I_2,I_3\}$ 的闭包表示为 I_{123}^*，且从定义可知，$I_{123}^*=I_1^*\cup I_2^*\cup I_3^*$。

定义 3(闭包的补)　某一条包含于 $I_{j_1j_2\cdots j_k}^*(k\in N^+)$ 的事务记录 T_i 关于 $I_{j_1j_2\cdots j_k}^*$ 的补采用符号 $\overline{T_{i(j_1j_2\cdots j_k)}}$ 表示。它表示所有符合这样条件的项所组成的集合：这些项都包含在 $I_{j_1j_2\cdots j_k}^*$ 中而没有任何一项包含在 T_i 中。亦即 $\overline{T_{i(j_1j_2\cdots j_k)}}=I_{j_1j_2\cdots j_k}^*-T_i$。

例 1　一个数据库片段中数据项的闭包以及闭包的补的示例。

在表 1 中，I_1 的闭包

$I_1^*=T_1\cup T_4\cup T_6=\{I_1,I_2,I_3,I_4,I_5,I_8\}$

I_2 的闭包

$I_2^*=T_2\cup T_5\cup T_6=\{I_1,I_2,I_5,I_6,I_{10}\}$

$\{I_1,I_2\}$ 的闭包

$I_{12}^*=I_1^*\cup I_2^*=\{I_1,I_2,I_3,I_4,I_5,I_6,I_8,I_{10}\}$

事务 T_6 关于 I_1^* 的补

$\overline{T_{6(1)}}=I_1^*-T_6=\{I_3,I_4,I_5,I_8\}$

表 1　一个数据库片段中定义的示例

TID 集	项集
T_1	I_1,I_3,I_4,I_5
T_2	I_2,I_6,I_{10}
T_3	I_4,I_6,I_7
T_4	I_1,I_5,I_8
T_5	I_2,I_5
T_6	I_1,I_2
T_7	I_3,I_4,I_8
T_8	I_4,I_5,I_9
T_9	I_5,I_{10}
T_{10}	I_7,I_9

引入数据项的闭包概念，就能够把数据库中出现的所有与该项相关的数据项统一起来。一个数据项的闭包一定把所有与该项有关联的数据项都包含在内了。该项的闭包一定是所有数据项集合 I 的一个子集。当然也可能和 I 相等。但是，在实际生活中，大型的数据库或者数据仓库中包含着海量的各种各样的数据项，这种情形下某数据项的甚至某些相似项的闭包，都可以认为是 I 的真子集。例如大型连锁超市里销售着各种商品，某个品牌甚至所有的计算机品牌所代表的项的闭包也不可能与这个超市的数据库中的 I 相等，因为 I 中包含着各种各样的商品。下文中将会看到，结合闭包的特点以及数据挖掘中两个基本的事实，我们能够做到或者让 A 在计算上无法做到挖掘其竞争对手的信息，或者即使它做到了，得出的结果亦是无用的。

事实 1　所有类 Apriori 算法的共同缺点是当挖掘过程中出现大量候选项集(可能的原因有支持度-置信度阈值设置过低，或数据项本身较特殊等)或者挖掘长频繁模式时会产生

大量的频繁项集，这会产生非常巨大的开销。即使使用先进的设备能做到这一点，相对于挖掘隐含模式所能得到的收益，付出可能是得不偿失的。

事实 2　强关联规则不一定有趣。例 2 通过举出一种完全可能在实际中发生的情况来说明这点。为简单起见，将数据设置得小一点，由于是用概率来分析，因此即使是一个数据库片段也能够准确地说明该事实。

例 2　考虑一个含有 100 条事务记录的示例数据库，其中含有 A 产品的事务数等于含有 B 产品的事务数，且均设为 60 条，即 $P(I_A)=P(I_B)=60\%$。而同时含有 A、B 的事务数为 30 条，即 $P(I_A \cup I_B)=30\%$。设定最小支持度阈值为 min_sup=25%，最小置信度阈值为 min_conf=50%。于是通过挖掘将发现下列规则：

$$buys(I_A,"Production\ A") \Rightarrow buys(I_B,"Production\ B") \quad [sup=30\%, conf=50\%] \tag{1}$$

但是事实上该规则并不可信，因为在没有购买 A 产品的 40 条事务中，有 30 条事务是含有 B 产品的。此时，如下规则才是真正的有趣的规则：

$$buys(\overline{I_A},"No\ Production\ A") \Rightarrow buys(I_B,"Production\ B") \quad [sup=30\%, conf=75\%] \tag{2}$$

理由是显然的。在相同的支持度下，式(2)比式(1)的置信度更高，也就是说式(2)比式(1)具有更高的可信程度。其直观的意义是，购买 A 产品反而降低了购买 B 产品的可能性。由此，我们引出提升度[3]的概念。

概念 2　规则的提升度(lift)是一种相关度量[15](correlation)，其定义如下：两数据项 I_A，I_B 的提升度为：

$$lift(I_A,I_B)=\frac{P(I_A \cup I_B)}{P(I_A)P(I_B)}$$

提升度具有如下的性质：当提升度 $lift(I_A,I_B)<1$ 时，I_A 的出现和 I_B 的出现是负相关的，此时得出的关联规则即使满足支持度和置信度，也是无效的；当 $lift(I_A,I_B)>1$ 时，I_A 和 I_B 是正相关的，意味着一个的出现蕴涵着另一个的出现；当 $lift(I_A,I_B)=1$ 时，I_A 和 I_B 相互独立。由于我们的算法是利用数据项的兴趣度小于 1 来构造无用规则，因此我们仅证明当提升度小于 1 时的情形。

证明：

$$lift(I_A,I_B)=\frac{P(I_A \cup I_B)}{P(I_A)P(I_B)}<1 \Rightarrow \frac{P(I_A \cup I_B)}{P(I_A)}<P(I_B) \Rightarrow P(I_B|I_A)<P(I_B)$$

由全概率公式：

$P(I_B)=P(I_B|I_A)P(I_A)+P(I_B|\overline{I_A})P(\overline{I_A})$ 代入上式继续推导：

$$\Rightarrow P(I_B|I_A)<P(I_B|I_A)P(I_A)+P(I_B|\overline{I_A})P(\overline{I_A})$$

$$\Rightarrow P(I_B|I_A)-P(I_B|I_A)P(I_A)<P(I_B|\overline{I_A})P(\overline{I_A})$$

$$\Rightarrow P(I_B|I_A)P(\overline{I_A})<P(I_B|\overline{I_A})P(\overline{I_A})$$

$$\Rightarrow P(I_B|I_A)<P(I_B|\overline{I_A})$$

该式蕴涵了这样的意思：在 I_A 出现的项中，I_B 出现的概率反而比那些没有 I_A 出现的项的概率要低。那么显然，这条关联规则是无用的。所以，可以通过规则的提升度来构造一些“虚假”的关联规则。这些规则看似是满足支持度-置信度框架的，实际上是无用的。

有了以上概念、定义、事实和性质，我们结合问题来分析：作为数据提供方，怎样变换自己的数据库，才能既让 A 公司通过挖掘得到关于自己产品的信息，而又不能得到关于竞争对手 B，C 等的信息。简单起见，我们先从最基本的、竞争对手只有一个的情况入手。

此时 A 在开销允许的前提下，会选择挖掘整个变换后的数据库 D'，因为可以获得最多的信息。如果挖掘 D' 开销太大，A 会退而求其次，只挖掘含有代表本公司产品的数据项 I_A 和代表竞争对手产品 I_B 的数据项的部分数据库。亦即，在频繁项集生成过程中加入约束条件[16]，对不含 I_A 和 I_B 的项在挖掘过程中就进行剪枝。下面给出关于隐私保护要求的形式化描述，然后在第 3 节给出基于闭包的变换算法。

我们针对类 Apriori 算法的特点，对隐私保护要求定义如下：设挖掘请求方为 A，其对应产品在数据库中的代表数据项为 I_A；设 A 的竞争对手依次为 B，C，$D\cdots$，同理代表其产品的数据项依次为 I_B，I_C，I_D，$\cdots$。用同一个 Apriori-like 算法分别挖掘 D 和 D' 产生的两组规则的集合，记为 R 和 R'(值得注意的是这两个集合的元素 r 和 r' 的实质是概念 1 中定义的蕴涵式)。那么，当同时满足如下条件时，我们认为达到了隐私保护的要求。

$$\forall r \in R, I_A \in r, I_B, I_C, I_D, \cdots \notin r$$
$$\exists r' \in R', I_A \in r', I_B, I_C, I_D, \cdots \notin r'$$
$$r=r' \qquad \text{(Condition1)}$$

Condition 1 的含义是对于任何一条在原数据库中挖掘到的规则 r，若 r 仅和挖掘请求方 A 相关而和待保护方数据项无关，那么一定存在一条规则 r' 与其对应。也就是说，A 可以正确挖掘自己的信息。

$$\forall r \in R, 1=(I_B \in r) \cup (I_C \in r) \cup (I_D \in r), \cdots$$
$$\forall r' \in R', 1=(I_B \in r') \cup (I_C \in r') \cup (I_D \in r'), \cdots$$
$$r \neq r' \qquad \text{(Condition2)}$$

Condition 2 则表示凡是与待保护项相关的规则，r 和 r' 都不可能相同。那么也就是说 A 不可能正确挖掘到自己竞争对手的信息。

3　算法的描述

为了叙述方便，我们按自然数递增的顺序为所涉及的主要数据项的下标排序。即假定挖掘请求方的数据项为 I_1，而所有我们认为需要保护的各个数据项依次为 I_2，I_3，$\cdots$，I_k，并记 $I_{sec}=\{I_2,I_3,\cdots,I_k\}$，即 I_{sec} 为所有需要保护的数据项的集合。

算法描述如下：

输入：事物数据库 D，待挖掘其关联关系的项 I_1，待保护的数据项 I_2，I_3，$\cdots$，I_k。

输出：改变后的数据库 D'。

过程：

(1) calculate $I_{sec}=\{I_2,I_3,\cdots,I_k\}$. //确定待保护的数据项 I_2，I_3，$\cdots$，I_k 的并集 I_{sec}。

(2) initialize $I_1^*=I_2^*=\cdots=I_k^*=\emptyset$;

(3) scan database D, do

(4) for ($i=1$; $i\leqslant|D|$; i++)

{

(5)　if(($T_i \cap I_{sec}=\{I_{t_1},I_{t_2},\cdots,I_{t_m}\}\neq\emptyset$)

{

(6) $I^*_{t_1t_2\cdots t_m}=I^*_{t_1t_2\cdots t_m}\cup T_i$;//求含有最多的待保护项的闭包。
(7) for ($j=1;j\leqslant m;j++$)
(8) $I^*_{t_j}=I^*_{t_j}\cup T_i$;//求单个项的闭包。
}
(9) else All the item closure remain the same;
(10) return(4);
}
(11) for ($i=1;i\leqslant|D|;i++$)
call Procedure_Transformation(i);
(12) return D';
(13) Procedure_Transformation(i)
{
(14) if ($T_i\cap I_{sec}=\{I_{t_1},I_{t_2},\cdots,I_{t_m}\}\neq\emptyset$)
{
(15) if ($I_1\in T_i$)
{
(16) find the nearest j that satisfies $i<j$ and $T_j\cap\{I_1,I_2,\cdots,I_k\}=\emptyset$;//找到最近的一个相对于 I_1 和 I_{sec} 的零事务。
(17) for ($p=1;p\leqslant m;p++$)
{
(18) if ($lift(I_1,I_{t_p})>1$)
{$T_i=T_i-I_{t_p}$;$T_j=T_j+I_{t_p}$}//支持度大于 1 的待保护项外移。
}
}
(19) else $T_i=T_i+\overline{T_{i(t_1t_2\cdots t_m)}}-I_1$;//该步骤的 else 对应于步骤(15)中的 if。
}
(20) else T_i remains the same;//此时 $T_i\cap I_{sec}=\emptyset$,无需变换。该步骤的 else 对应的是(14)中的 if。
(21) return(11);
}

4 算法的分析和证明

4.1 算法分析

整个算法需要扫描两次数据库,分别为步骤(3)和 Procedure_Transformation 子过程。由于所有的变换计算都在 Procedure 子函数中完成,因此算法的主要开销在于 Procedure 子过程。对于每一个事务记录 T_i,如果它和 I_{sec} 无交集,也就是 T_i 中不含有待保护数据项,那么对这个事务就无须做出变换,这种情况表现在步骤(9)。当事务 T_i 中含有待保护的项时,我们对 T_i 的变换遵循下面的原则:

1) 不对 T_i 中待挖掘的项 I_1 做变化,这样做的目的是不改变 I_1 的支持度计数。由于我们并没有增减数据库 D 的总事务数,因此这些变换过程不会改变 I_1 的支持度。

2) 保护 I_{sec} 的两个主要手段是:(1)截断任何一个含有 I_1 的事务记录中 I_1 和 I_{sec} 里某些项的联系;(2)针对类 Apriori 算法的特性复杂化那些含有 I_{sec} 里的某些项却不含 I_1 的事务记录。

4.2 证明过程

算法的(1)-(10)步完成了一些准备工作,尤其是计算出了 Procedure 子过程需要的数据项闭包。关于这些步骤中需要的定义、性质等已在第 2 节给出,所以证明的重点仍然放在 Procedure 子过程。证明的思路很直观,只要能够证明对数据库中所有的数据项进行这样的变换后,一定能满足我们提出的隐私保护要求即可。

证明:

$\forall T_i\in D$,T_i 按照其与 I_{sec} 的交集的势来划分,只有两种情况:交集为 $\emptyset$ 或者交集为非空集。

在步骤(20)中,我们处理交集为 $\emptyset$ 的情况。实际上,这种数据项是事物数据库中以 I_{sec} 为考察项集的零事务(null-transaction)。由于零事务不会影响考察项集的计数,而且即使这种类型的数据项包含代表挖掘方的数据项 I_1,挖掘方在挖掘过程中也只能得到自己的一些信息,无法收集到任何与 I_{sec} 有关的信息,因此对它们无需处理。

当交集非空时,根据 I_1 是否属于 T_i 又可以分为两种情况。

第一种情况 算法当前正在处理的事务 T_i 既含有待挖掘项 I_1,又含有某些待保护项。我们的做法是对 T_i 中的 I_1 保持不变,只变换属于 T_i 的那个 I_{sec} 的子集。这样做可以达到一举两得的目的:

〈1〉不会改变 I_1 的所有不含待保护项的规则。因为我们的变换步骤中没有对 I_1 和其他不需要保护的项做变化,所以从整个数据库 D 的视角来看,在使用类 Apriori 算法挖掘的过程中,所有不涉及 I_{sec} 的频繁项都没有变动,因此可以保证与 I_1 相关的非保护项的规则正常挖掘。

〈2〉所有含有 I_{sec} 的某个非空子集的项都做出了相应的变化。我们在这一步中不断地将待保护的项 I_{t_p} 向下移动到最近的一个零事务中,直到这一项和 I_1 的提升度 $lift(I_1,I_{t_p})$ 小于 1 后停止。这种变化可行的原因是,由于我们不断地将 I_{t_p} 外移,但与此同时 I_1 又保持不变,因此很显然 $P(I_1)$ 和 $P(I_{t_p})$ 保持不变,而 $P(I_1\cup I_{t_p})$ 在不断变小,所以 $lift(I_1,I_{t_p})$ 在不断变小并趋近于 0。但是,根据算法的步骤,当 $lift(I_1,I_{t_p})$ 已经小于 1 时,就不需要再外移了,因为此时对于仍留在 T_i 中的 I_{t_p},已经足以保证提升度小于 1。即使请求挖掘方强行挖掘所有有关 I_1 和 I_{t_p} 的关联关系,得到的亦是一个无用的结果。所有这些过程在步骤(14)-(18)完成。

第二种情况 算法当前正在处理的事务不含 I_1。这时我们变化所有含有 I_{sec} 的某个子集的项,将其变为对应的闭包,并同时去掉 I_1。这样做可以大大增加挖掘请求方的挖掘难度,同时因为在变成闭包的过程中会增加数据项,这也保证了最后的挖掘结果仍然是不准确的。只是这种不准确性因数据库的不同而无法给出定量的估计,因为不同的数据库会有不同的情况。去掉闭包中的 I_1 的原因是为了保证 I_1 的计数不变,否则挖掘方连自己的信息都可能挖掘错误。第二种情况的处理表现在步骤(19)。

综上所述,证明过程中第一种情况的〈1〉部分证明算法能满足 Condition 1,第一种情况的〈2〉部分和第二种情况则证明了算法满足 Condition 2。所以,算法能够达到我们所要求的双赢的目的。

5 算法的两种特殊情形

该算法针对不同的硬件情况,可以在空间需求上有较大的灵活性。不妨考虑算法两次扫描数据库的过程。在步骤(3)中,第一次扫描必须完整地扫描整个数据库,因为这是计算闭包的步骤。但是第二次在 Procedure_Transformation 子过程中的扫描,事实上可以不用扫描整个数据库了,因为我们

的算法要变换的只是那些含有某个 I_{sc} 的非空子集的数据项，所以完全可以在第(3)步中第一次扫描时用一些存储空间来存储需要做变换的项的 TID 即可。但是这样做虽然节省了时间，却占用了更多空间。对于海量的数据仓库，在变换时可能空间需求会更难满足。但是当设备十分先进时，可能更看重的是时间因素。所以，可以针对实际情况灵活运用该方法。

此外，该算法并非十全十美。注意到在 Procedure_Transformation 子过程中，步骤(16)的表述为"find the nearest j that satisfies $i<j$ and $T_j \cap \{I_1, I_2, \cdots, I_k\} = \emptyset$;"，事实上，这句话所假设的前提有可能为假，也就是说事物数据库 D 中已经不存在相对于 $I_1 \cup I_{sc}$ 的零事务了。这种情况当然有可能发生，但是我们在第 2 节关于数据项闭包的定义已经指出，由于是大型数据库，因此存在着海量的数据，相对于某一类型的商品来说，不存在相对于它们的零事务的概率是微乎其微的。但是一旦这种情况真的发生了，可能要进行更加复杂的变换，而这也是进一步要研究的课题。

结束语 本文提出的基于数据项闭包的保密数据挖掘方法，能够有效解决这样的问题：(1)数据提供方不完全信任挖掘请求方，但是又希望通过提供变换过的数据库给对方挖掘来获得一些利益；(2)明确对方在挖掘过程中使用的是类 Apriori 算法，且已方为水平格式(horizontal data format)的事务数据库。与此同时，基于数据项闭包这一思想是否可以应用在更广阔的数据挖掘领域，仍未可知。比如，能否对垂直数据格式的数据库进行闭包变换，能否对序列模式数据库进行闭包变换[17,18]等，都是有待解决的问题。另一方面，从该算法能够针对的挖掘方法来说，是否也能通过该算法针对其他数据挖掘算法(如 FP 增长算法)来进行保护呢？同样没有答案，而新的发现有待进一步研究。

参 考 文 献

[1] Agrawal R, Imieliński T, et al. Mining asso-ciation rules between sets of items in large databases[C]// Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. Washion D. C., USA, May 1993: 207-216

[2] Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules[C]// Proceedings of the 20th International Conference on Very Large Databases. Santiago, Chile, Sept. 1994: 487-499

[3] Han Jia-wei, Micheline K. 数据挖掘：概念与技术(第二版)[M]. 范明，孟小峰，译. 北京：机械工业出版社，2007

[4] Tavani H T. Information privacy, data mining, and the internet [J]. Ethics and Information Technology, 1999, 1(2): 137-145

[5] Lindell Y, Pinkas B. Privacy preserving data mining[J]. Journal of Cryptology, 2002, 15(3): 177-206

[6] Agrawal R, Evfimievski A, Srikant R. Information sharing across private databases[C]// Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2003: 86-97

[7] Agrawal R, Srikant R. Privacy-preserving data mining [C]// Proceedings of the 2000 ACM SIGMOD Conference on Management of Data. 2000: 439-450

[8] Kantarcioglu M, Clifton C. Privacy-preserving distributed mining of association rules on horizontally partitioned data[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(9): 1026-1037

[9] Du Wen-liang, Zhan Zhi-jun. Using randomized response techniques for privacy-preserving data mining[C]// The 9th ACM SIGKDD Int'l Conf Knowledge Discovery in Databases and Data Mining. Washington D. C., USA, August 2003: 24-27

[10] 葛伟平，汪卫，周皓峰，等. 基于隐私保护的分类挖掘[J]. 计算机研究与发展，2006，43(01)：39-45

[11] 罗永龙，黄刘生，等. 一个保护私有信息的布尔关联规则挖掘算法[J]. 电子学报，2005，33(5)：900-903

[12] 张锋，常会友. 基于分布式数据的隐私保持协同过滤推荐研究[J]. 计算机学报，2006，29(8)：1487-1495

[13] 周水庚，李丰，陶宇飞，等. 面向数据库应用的隐私保护研究综述[J]. 计算机学报，2009，32(5)：847-861

[14] Chris C, Donald M. Security and privacy implications of data mining[C]// Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery. 1996: 15-19

[15] Brin S, Motwani R, Silverstein C. Beyond Market Baskets: Generalizing Association Rules to Correlations[C]// Proceedings of SIGMOD'97. AZ, June 1997: 265-276

[16] Srikant R, Vu Q, Agrawal R. Mining association rules with item constraints[C]// KDD-97. 1997: 67-73

[17] Aggarwal C C, Yu P S. Privacy-preserving Data Mining: Models and Algorithms[M]. New York: Springer-Verlag, 2008

[18] 王虎，丁世飞. 序列模式挖掘研究与发展[J]. 计算机科学，2009，36(12)：14-17

(上接第 143 页)

[2] Jeffrey D, Gupta N. Improving Fault Detection Capability by Selectively Retaining Test Cases During Test Suite Reduction[J]. IEEE Transactions on Software Engineering, 2007, 33(2): 108-123

[3] Beydeda S, Gruhn V. BINTEST-Binary Search-based Test Case Generation[C]// Proceedings of the 27th Annual International Conference on Computer Software and Applications. IEEE Computer Society, 2003

[4] Kiczales G, et al. Aspect-oriented Programming [C]// Proceedings of the European Conference on Object-oriented Programming. Finland: Springer-Verlag, 1997

[5] vim: http://www.vim.org/

[6] emacs: http://www.gnu.org/software/emacs/

[7] Lee H B, Zorn B G. BIT: A Tool for Instrumenting Java Bytecodes

[8] BCEL: The Byte Code Engineering Library[EB/OL]. http://jakarta.apache.org/bcel/

[9] Java Instrumentation API(JIAPI) [EB/OL]. http://jiapi.sourceforge.net/

[10] Eclipse Instrumentation Framework [EB/OL]. http://dev.eclipse.org/viewcvs/index.cgi/~checkout~/platform-ui-home/instrumentation/index.html

[11] Java Source Code Instrumentation[EB/OL]. http://www.glenmccl.com/instr/instr.htm

[12] Seesing A, Orso A. InsECTJ: a generic instrumentation framework for collecting dynamic information within Eclipse[C]// ETX. 2005: 45-49

[13] Log4J project[EB/OL]. http://logging.apache.org/log4j/1.2/index.html

[14] Kiczales G, Hilsdale E, Hugunin J, et al. An Overview of AspectJ[C]// Proceedings of the 15th European Conference on Object-oriented Programming. London, UK: Springer-Verlag, 2001: 327-353