计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20160612
54(12):2785-2796, 2017

基于簇和阈值区间的高效关联规则隐藏算法

牛新征[1] 王崇屹[1] 叶志佳[1] 佘 堃[2]
[1](电子科技大学计算机科学与工程学院 成都 611731)
[2](电子科技大学信息与软件工程学院 成都 610054)
(xinzhengniu@uestc.edu.cn)

An Efficient Association Rule Hiding Algorithm Based on Cluster and Threshold Interval

Niu Xinzheng[1], Wang Chongyi[1], Ye Zhijia[1], and She Kun[2]
[1](*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu* 611731)
[2](*School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu* 610054)

Abstract Association rules hiding is a very important method of privacy-preserving data mining (PPDM). Because the current association rules hiding algorithm operates the transaction database directly, it leads to a lot of I/O overhead. To solve this problem, we put forward a quick association rules hiding algorithm based on FT-tree, called FP-DSRRC. Firstly, the algorithm improves the structure of FP-tree by adding an index to the transaction number and establishing the bidirectional traverse structure. Then FP-DSRRC uses the improved FP-tree to quickly handle transaction data set, avoiding a large number of I/O overhead caused by traversal the raw transaction data set. Furthermore, FP-DSRRC finds the sensitive items quickly by building and maintaining a transaction index table, and then handles the association rules based on the clustering strategy. We eliminate the sensitive rules by clusters, and reduce the negative influence caused by association rules hiding progress to the original data set by adopting the idea of rule support and confidence degree interval at the same time. Finally, the experiment shows that compared with traditional association rules hiding algorithm, the executive time of FP-DSRRC has been decreased by 50%~70% while guaranteeing the quality of general data, moreover, FP-DSRRC has better availability on a large-scale real data set.

Key words privacy preservation; association rule hiding; FP-tree; sensitive rule; data sanitization

摘 要 关联规则隐藏是隐私保护数据挖掘(privacy-preserving data mining, PPDM)的一种重要方法.针对当前的关联规则隐藏算法直接操作事务数据、I/O开销较大的缺陷,提出一种基于FP-tree快速关联规则隐藏的算法FP-DSRRC.算法首先对FP-tree的结构进行改进,增设事务编号索引并建立双向遍历

收稿日期:2016-08-16;**修回日期**:2017-01-06

基金项目:国家自然科学基金项目(61300192);国家科技支撑计划基金项目(2013BAH33F02);中央高校基本科研业务费专项资金项目(ZYGX2014J052);四川省科技支撑计划基金项目(2015GZ0096);成都市科学技术局软科学研究项目(2015-RK00-00046-ZF);四川省公安厅科研项目(2015SCYYCX06);四川省自贡市公安局项目

This work was supported by the National Natural Science Foundation of China (61300192), the National Key Technology Research and Development Program of China (2013BAH33F02), the Fundamental Research Funds for the Central Universities (ZYGX2014J052), the Key Technology Research and Development Program of Sichuan Province (2015GZ0096), the Soft Science Research Project of Chengdu Science and Technology Bureau (2015-RK00-00046-ZF), Scientific Research Project of Sichuan Provincial Public Security Department (2015SCYYCX06), and the Project of Public Security Bureau of Zigong City, Sichuan Province.

结构,进而利用改进的 FP-tree 对事务信息进行快速处理,避免了遍历原始数据集产生的大量 I/O 时间;然后通过建立和维护事务索引表实现对敏感项的快速查找,并基于分簇策略对关联规则处理,以簇为单位进行敏感规则消除,同时采用规则支持度和置信度阈值区间的思想,减少了关联规则隐藏处理对原始数据集的影响;最后通过实验测试证明:相较于传统关联规则隐藏算法,FP-DSRRC 算法在保证生成的数据集质量的同时,减少了 50%~70%的算法执行时间,并在大规模真实数据集上有较好的可用性.

关键词 隐私保护;关联规则隐藏;频繁模式树; 敏感规则; 数据清洗

中图法分类号 TP301.6

随着关联规则技术的逐渐成熟以及大数据云计算的不断发展,通过关联规则能挖掘出的信息越来越多,其中可能包含了用户的敏感数据和隐私信息[1].因此,Agrawal 等人[2]在 2000 年提出的隐私保护数据挖掘(privacy-preserving data mining, PPDM)成为数据挖掘领域的研究重点,而关联规则隐藏作为解决数据挖掘中隐私问题的一项关键技术也引起了国内外学者的重视.

关联规则挖掘最初由 Agrawal[3]在 1993 年提出,其目的是为了挖掘事务数据库中各项集存在的价值关联,为相关决策提供依据.对于一些包含敏感关联规则的数据,在发布数据前修改原始数据集,从而达到隐藏敏感规则的目的,这个过程就是关联规则隐藏.

1 相关工作

当前关联规则隐藏技术大致可分为 6 种[4]:基于启发式的方法(heuristic approach)、基于边界的方法(border approach)、基于重构的方法(reconstruction approach)、基于加密的方法(cryptographic approach)、精确方法(exact approach)和基于混合的方法(hybrid techniques approach).

启发式方法主要采用数据失真技术和数据阻塞技术,通过向原始事务数据库中添加噪声数据或将原敏感项集的某些属性删除,以达到降低敏感规则支持度或置信度的目的.在基于启发式的关联规则隐藏算法中,文献[5]提出基于频繁项集相交点阵(intersection lattice)的关联规则隐藏算法(heuristic for confidence and support reduction based on intersection lattice, HCSRIL),算法首先确定对原始数据库影响最小的敏感规则,并求解最小的待修改事务数,在消除敏感规则的同时保留频繁项集,通过以上步骤尽量降低修改敏感规则对原始数据库产生的影响;文献[6]提出一种基于数据失真技术的关联规则隐藏算法(cuckoo optimization algorithm for the sensitive association rules hiding, COA4ARH),该算法分别构造了 3 个适应度函数用于优化隐藏方案的选择,并利用迁移函数避免陷入局部最优.

在利用减少敏感规则右项支持度的启发式算法中,文献[7]提出了 DSRRC(decrease support of R. H. S item of rule clusters)算法,算法首次利用分簇和划分敏感度的思想进行关联规则隐藏,尽可能地减少算法执行对原事务数据集产生的影响,但是 DSRRC 算法需要对数据集进行多次排序且不能消除有多个右项的关联规则,为了解决这个问题,文献[8]提出了 2 个新算法:ADSRRC(advanced DSRRC)算法和 RRLR(remove and reinsert L. H. S. of rule)算法,克服了这些问题. ADSRRC 也是基于相同右项对敏感规则分簇,通过计算事务数据的敏感度并将其降序排序,消除对算法执行结果的影响,而 RRLR 算法能处理具有多个右项的规则的能力;文献[9]提出的 MDSRRC(modified DSRRC)算法通过每次删除敏感度最高的敏感项保证修改后的事务数据集的质量,启发式算法通常需要向原始数据库中添加噪声或删除某些属性,当敏感规则较多时对原始数据改动较大,同时在基于数据阻塞的方法中,由于项集的某些属性被删除,后续可能无法计算其支持度和置信度.

基于边界的方法以贪心方式选择对边界影响最小的数据清洗策略,文献[10]提出一种基于阻塞的关联规则隐藏算法(border rule based blocking algorithm, BRBA).该算法基于边界规则筛选合适的事务数据进行清洗,通过安全边界策略(safety margin)降低所有敏感规则的支持度或置信度,最小化敏感规则隐藏对其他非敏感规则的影响.

基于精确方法的思想是把数据清洗当成一种原子操作,这类算法可以使消除敏感规则后的数据库没有任何影响,但是需要多次选择、比较,相比启发式算法时间开销巨大.文献[11]提出的关联规则隐藏

算法无需修改频繁项集支持度,而是提出一个代表规则(representative rule, RR)的概念.通过运用代表规则,算法无需访问主数据库就能推断出敏感规则,其基本思路是改变数据项所在的位置,而不是将其从事务数据库中删除.这种策略的优点在于算法不会对频繁项集的支持度和数据库规模产生影响,然而该算法无法保证敏感规则的置信度小于阈值.文献[12]将敏感规则消除问题抽象为多目标优化问题,并采用遗传进化理论进行优化求解.该算法在快速隐藏敏感规则的同时,对原事务数据库的修改幅度较小,然而该算法会对非敏感规则产生较大的丢失率,并且查找待删除项的过程复杂度较高.

基于重构的方法首先执行数据失真处理,然后重构数据分布.文献[13]提出一种基于重构的算法,首先通过频繁项集挖掘算法找到所有频繁项集及其对应的支持度,再将敏感规则从频繁项集中删除到支持度之下,然后利用这些不包含敏感规则的频繁项集去构建一棵 FP-tree[14],这样关联规则隐藏的问题就转化成了反向频繁项集挖掘问题(inverse frequent set mining, IFM)[15],最后通过反向频繁项集挖掘算法生成不包含敏感规则的数据库.

基于数据加密的方法通过对敏感数据进行加密编码,文献[16]提出结合 RSA 公钥加密和伪随机数生成器技术,对隐私数据进行加密保护.但是对于规模较大的数据集,该算法计算过程较复杂,实用性较差.

基于混合的方法采用多种技术的融合进数据保护.文献[17]结合 2 种启发式算法,分别从降低敏感规则的支持度和置信度入手,兼顾 2 种算法的优势.但该算法仅仅是 2 种算法的简单叠加,未能针对不同数据特点自适应选择处理算法,并且该算法同样具有启发式算法对原始数据影响较大的缺陷.

通过对现有工作的分析可以看出,现有的关联规则隐藏技术主要存在 3 个问题:

1) 现有的关联规则隐藏技术往往直接对原事务数据集进行操作,当事务数据集规模较大时,对数据的遍历处理会导致严重的 I/O 开销,时间效率低下.

2) 现有的算法仅仅着眼于最大限度消除敏感规则,但缺乏对非敏感规则的保护,从而导致经过关联规则消除的数据集规则丢失率和数据损失度较高.

3) 通过人为向原事务数据集中增添项集或规则来隐藏敏感规则的方案,将破坏非敏感规则从而导致错误的挖掘结果.总之,当前的关联规则隐藏算法未能在敏感消除和数据质量之间做到很好的权衡.

关联规则隐藏的前提条件是进行频繁项集挖掘,找到所有关联规则,但是现有算法都割裂了关联规则隐藏和频繁项集挖掘之间的联系,多次遍历和修改原始数据集导致了大量的 I/O 时间.针对传统关联规则隐藏算法效率较低以及对原始数据集影响较大的缺陷,本文提出一种 FP-DSRRC(FP-tree DSRRC)算法,利用一棵改进的 FP-tree,使得频繁项集挖掘和关联规则隐藏 2 个过程都在 FP-tree 上进行操作,并结合基于启发式和基于边界的关联规则隐藏方法,使算法既快速又高效.

本文主要贡献有 3 个方面:

1) 改进经典的 FP-tree 结构,添加事务索引表将算法对 FP-tree 的修改映射到数据集中.同时算法直接在 FP-tree 上操作避免了遍历原始数据集产生的时间代价,通过 FP-tree 的头表信息、节点间域信息可以快速确定敏感事务编号,减少了查找开销.

2) 采用启发式方法,通过将敏感规则分簇,使得单次敏感项的删除影响到多个敏感规则,减少了对原数据集的修改.

3) 基于规则支持度和置信度阈值区间的思想,通过将敏感项按影响度大小排序,优先删除对原数据集影响较小的项,保证了数据集质量的同时显著提升算法效率.

2 问题描述

2.1 关联规则挖掘

关联规则挖掘的目的是从事务数据库 DB 中找出支持度和置信度阈值区间内的强关联规则[18].具体而言,支持度大于最小支持度阈值 MST,并且置信度大于最小置信度阈值 MCT 的规则才被认为是强关联规则.

设 $I=\{i_1,i_2,\cdots,i_n\}$ 是全部项的集合,其中 i_k $(k=1,2,\cdots,n)$ 为不同的项,令事务数据库 $DB=(T_1,T_2,\cdots,T_n)$,则 T_i 称为 1 条事务,其是由若干个项组成的项集.

定义 1. 支持度.是项集 X 支持的事务 T 在事务数据库 DB 中出现的百分率,即:

$$support(X)=\frac{|\{t\in DB|X\in t\}|}{|t\in DB|}. \tag{1}$$

定义 2. 关联规则.是指形如 $X\rightarrow Y$ 的蕴含式,X 和 Y 分别是关联规则的先导(left-hand-side, L. H. S.)和后继(right-hand-side, R. H. S.),也叫左项集和

右项集,且 $X \cap Y \neq \varnothing$. 关联规则 $X \rightarrow Y$ 的支持度为项集 $X \cup Y$ 在 DB 出现的百分率,即:

$$support(X \rightarrow Y) = \frac{|\{t \in DB | X \cup Y \in t\}|}{|t \in DB|}, \quad (2)$$

关联规则 $X \rightarrow Y$ 的置信度为项集 $X \cup Y$ 在项集 X 出现的百分率,即:

$$confidence(X \rightarrow Y) = \frac{support(X \rightarrow Y)}{support(X)}. \quad (3)$$

2.2 关联规则隐藏

定义 3. 敏感规则 SR. 是不符合用户安全策略的关联规则集合,即需要隐藏的关联规则.

定义 4. 关联规则隐藏. 是指通过关联规则隐藏算法将原始数据库 DB 转换成 1 个新的事务数据库 DB',DB' 中不包含给定的敏感规则 SR.

关联规则隐藏算法通常应该满足 3 个条件:

1) DB' 中不包含任何给定敏感规则 SR;

2) DB' 中应该包含所有 DB 原来的非敏感规则;

3) DB' 中不包含任何人为产生的新规则.

寻找一种满足关联规则隐藏的全部条件的算法已经被证明是一种 NP-hard 问题. 一般采用 HF (hiding failure),MS (missing cost),AP (artificial patterns)这 3 个指标来衡量一个关联规则隐藏算法和关联规则算法对原数据集的影响.

1) HF 表示敏感规则的隐藏失败率:

$$HF = \frac{DB' \text{中的敏感规则数}}{DB \text{ 中的敏感规则数}}, \quad (4)$$

HF 越低表明关联规则隐藏效果越好.

2) MC 表示规则丢失率:

$$MC = \frac{DB' \text{中丢失的非敏感规则数}}{DB \text{ 中的非敏感规则数}}, \quad (5)$$

MC 越低表明 DB' 的质量越高.

3) AP 表示人为规则产生率:

$$AP = \frac{DB' \text{中新增规则数}}{DB \text{ 中的规则数}}, \quad (6)$$

AP 越低表明 DB' 中人为产生的规则越少.

3 FP-DSRRC 算法

FP-DSRRC 算法的流程图如图 1 所示. 算法首先将事务数据库 DB 的信息压缩到一棵改进的 FP-tree 上,并通过 FP-Growth[14] 算法挖掘出所有强关联规则. 将强关联的敏感规则按敏感项分簇,以簇为单位消除所有敏感规则,当所有簇都消除完毕后,将 FP-tree 还原成一个不包含敏感规则的数据库 DB'.

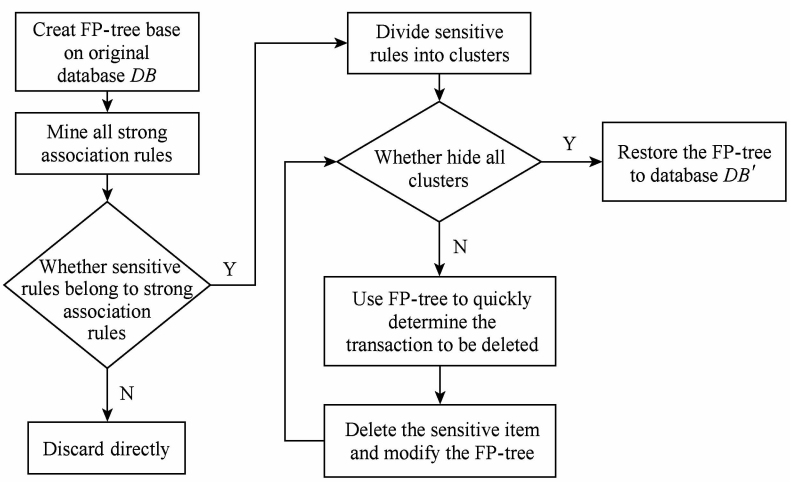

Fig. 1 FP-DSRRC algorithm process

图 1 FP-DSRRC 算法流程

3.1 改进的 FP-tree 及其性质

相对于经典的 FP-tree 树[14],本文的 FP-tree 树添加了 1 个事务索引表用于标识每一条事务的编号,用于关联规则隐藏过程中事务的索引和新数据库的重构. 这样通过 FP-tree 每个节点到根节点有唯一路径. 并且相对于经典的 FP-tree 是从节点到根单向遍历的,本文的 FP-tree 建树时保留了节点的 Child 域,使得该 FP-tree 是双向可遍历的. 改进的 FP-tree 具体构成如下:

1) 由 1 个树根 Root 及其频繁项子树、1 个频繁项头表和 1 个事务索引表组成;

2) 频繁项子树的每一个项节点用 Order,Count,Child,Parent,Sibling,Link 六个域组成;

3) 频繁项头表由 Count 域和 Link 域构成;

4) 事务索引表中每个表项由 Index 域和 Length 域构成.

改进的 FP-tree 如图 2 所示,构成的事务数据为 ABC,A,B 三条数据.

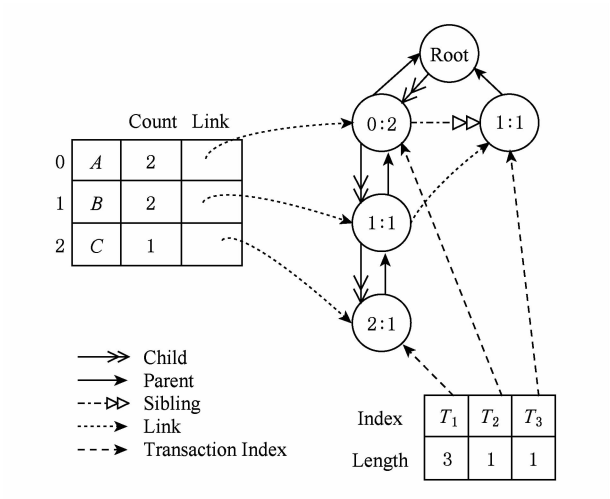

Fig. 2 Improved FP-tree

图 2 改进的 FP-tree

构建 FP-tree 先从原始数据集 DB 中读取所有事务,按照频繁项的支持度递减排序并编号,生成 1 个项序转换表;然后再次读取 DB,逐一将事务按照项序转换表排序后插入 FP-tree 中,并更新事务索引表. 通过 FP-Growth[14] 算法可以挖掘出所有频繁项集并生成所有强关联规则,建树算法和挖掘频繁项集算法可以参考文献[19],本文的建树方式相对于文献[19]保留了 Child 域和 Sibling 域,并添加了 1 个事务索引表.

为了直观展示建树的过程,给出 1 组精简的数据集 TestDB,设定 $MST=1/3$,$MCT=0.6$. 根据频繁项集进行项序转换,A,B,C,D,E 的序号分别为 3,0,1,2,4,其中序号表示项的支持度,序号越小,支持度越高. 筛选后的事务项集表如表 1 所示:

Table 1 Test Data and the Index after Transforming

表 1 测试数据集 TestDB 以及项序转换后的索引

TransactionID	Original Itemsets	Itemsets After Transformation
T_1	A,B,C,D	0,1,2,3
T_2	B,D,E,F	0,2,4
T_3	B,C,D	0,1,2
T_4	A,B,C,D,E,F	0,1,2,3,4
T_5	A,C,E	1,3,4
T_6	B,C,E	0,1,4
T_7	A,C,D	1,2,3
T_8	A,B,C,D,G	0,1,2,3
T_9	A,B,D,G	0,2,3

建立 FP-tree,为了保持图的简洁,未画出 Child 域和 Sibling 域,FP-tree 建立后如图 3 所示.

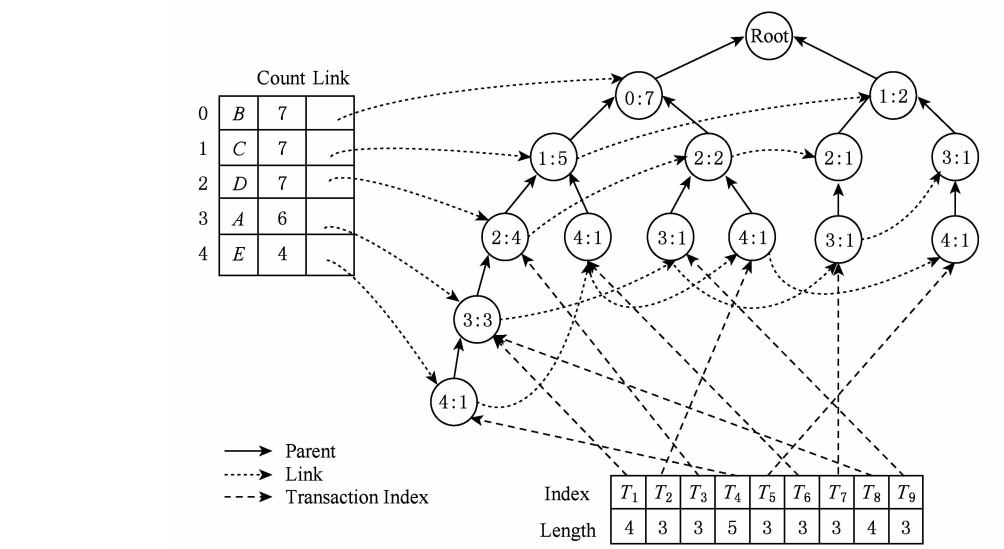

Fig. 3 Construct FP-tree

图 3 建立 FP-tree

综上所述,改进的 FP-tree 有 3 个性质:

性质 1. 对于形如 $X\rightarrow Y$ 的规则,根据项序转换表找出$\{X,Y\}$中项序最大的项 a,通过对所有的节点 a 向上遍历可知当前路径是否支持该规则,然后根据事务索引表快速确定哪些事务支持该规则.

性质 2. 由于 FP-tree 保留了 Child 和 Sibling 域,可双向遍历,FP-tree 可以根据事务索引表取出某条事务,对该事务进行修改后插回 FP-tree 中,而不影响其他事务.

性质 3. 通过遍历事务索引表并统计每个节点

到根节点的唯一路径可以快速将 FP-tree 还原成事务数据库.

采用 FP-Growth 算法对图 3 中的 FP-tree 进行频繁项集挖掘,可以得到表 2 的结果,表现形式为(频繁项集:支持数).

Table 2 Frequents Itemsets of TestDB

表 2 TestDB 的频繁项集表

Suffix	Frequent Itemsets
B	$(B:7)$
C	$(C:7),(C,B:5)$
D	$(D:7),(D,C:5),(D,B:6),(D,C,B:4)$
A	$(A:6),(A,D:5),(A,D,C:4),(A,D,B:4),(A,D,C,B:3),(A,C:5),(A,B:4),(A,C,B:3)$
E	$(E:4),(E,C:3),(E,B:3)$

3.2 敏感规则消除

算法通过删除敏感规则的右项集中的某个项,以实现其支持度和置信度满足阈值区间,并根据敏感度和最小删除数来选择待删除的事务,降低消除敏感规则带来的影响.对于敏感规则 $X\rightarrow Y$,将其支持度减小到 MST 之下至少删除 $\lfloor N\times MST-support(X\rightarrow Y)\rfloor+1$ 项,将其置信度减小到 MCT 之下至少需要删除项的个数为 $\lfloor support(X)\times MCT-support(X\rightarrow Y)\rfloor+1$,所以只需删除敏感项个数为两者之间的较小数即可将该敏感规则变成非强关联规则.

为了尽可能减少对原事务数据库的修改,需要尽量减少敏感项的删除,因此本算法的核心思想是通过 FP-tree 快速定位待删除的事务,并尽量使单次删除可以同时作用于多个敏感规则.算法用敏感度来确定删除的顺序,设敏感规则 SR 包含敏感项 $si_1,si_2,\cdots,si_n$,多个敏感规则可划分为簇 $SC=\{SR_1,SR_2,\cdots,SR_m\}$.则项敏感度 is_k 定义为 si_k 在敏感规则簇 SC 中出现的总次数:

$$is_k=\sum_{i=1}^{m}Num(i,k),\tag{7}$$

其中,$Num(i,k)$表示敏感规则 SR_i 中敏感项 si_k 的数量.规则敏感度定义为规则中项敏感度之和:

$$rs_j=\sum_{si_k\in SR_j}is_k.\tag{8}$$

类似地,簇敏感度定义为簇中规则的敏感度之和:

$$cs=\sum_{SR_k\in SC}rs_k.\tag{9}$$

算法根据待隐藏的敏感规则的待删除项分簇.形如 $X\rightarrow Y$ 的敏感规则,当有多个右项时,选择具有最大项敏感度 $is_{\max}$ 的作为待删除项;若敏感度相同,则待删除项选取支持度最大的项.将具有相同待删除项的规则归为同一个簇,然后以簇为单位消除所有的敏感规则,最后重构 1 个新的事务数据库 DB'.

对于表 1 中的 TestDB 数据集,假设敏感规则为 $SR=\{E\rightarrow C,A\rightarrow B,A\rightarrow BD\}$,则可以得到敏感项的敏感度为 $A:2,B:2,C:1,D:1,E:1$,对于敏感规则 $A\rightarrow BD$,有 2 个后项 B 和 D,选择敏感度高的项即项 B 作为删除项,因此敏感规则集可以分为 2 个簇,然后计算当前敏感度最高的簇中敏感规则的较小删除数,并通过 FP-tree 快速确定支持敏感规则的事务索引.

以敏感规则 $A\rightarrow B$ 为例,将其支持度降低到 MST 之下需要删除 2 个后项,将其置信度降低到 MCT 之下需要删除 1 个后项,所以敏感规则 $A\rightarrow B$ 的最小删除数为 1.根据项序转换表,项 A 的序号比 B 大,所以通过频繁项头表的项 A 出发,经过的每一个节点 A 同时向根节点查找,如果路径经过节点 B 说明当前路径支持该敏感规则,即通过该节点 A 的事务都支持敏感规则 $A\rightarrow B$,得到事务 T_1,T_4,T_8,T_9 支持 $A\rightarrow B$.具体过程如图 4 所示.图 4 中带灰度的节点为路径支持敏感规则的敏感节点,而带灰度的事务索引为支持该敏感规则的敏感事务索引,敏感规则分簇后如表 3 所示:

Table 3 Divide Sensitive Rules into Clusters

表 3 敏感规则分簇

Cluster1	Cluster2
Delete Item:B Cluster Sensitivity:9	Delete Item:C Cluster Sensitivity:2
$A\rightarrow B$(Delete Count:1,Support Index:1,4,8,9) $A\rightarrow BD$(Delete Count:1,Support Index:1,8,9)	$E\rightarrow C$

对表 3 中选择敏感度最大的簇即 Cluster1 进行敏感规则隐藏,对敏感事务的支持规则数进行计数,得到 $T_1:2,T_8:2,T_9:2,T_4:1$,选择长度最小的事务 T_9 进行删除,删除节点后的 FP-tree 如图 5 所示,此时敏感规则 $A\rightarrow B$ 和 $A\rightarrow BD$ 的 delete count 都减 1 变成 0,将 $A\rightarrow B$ 和 $A\rightarrow BD$ 从 Cluster1 删除,此时 Cluster1 所有敏感规则已经隐藏完毕,重新挖掘 FP-tree 中的频繁项,得到新的频繁项集后则处理其他的簇.

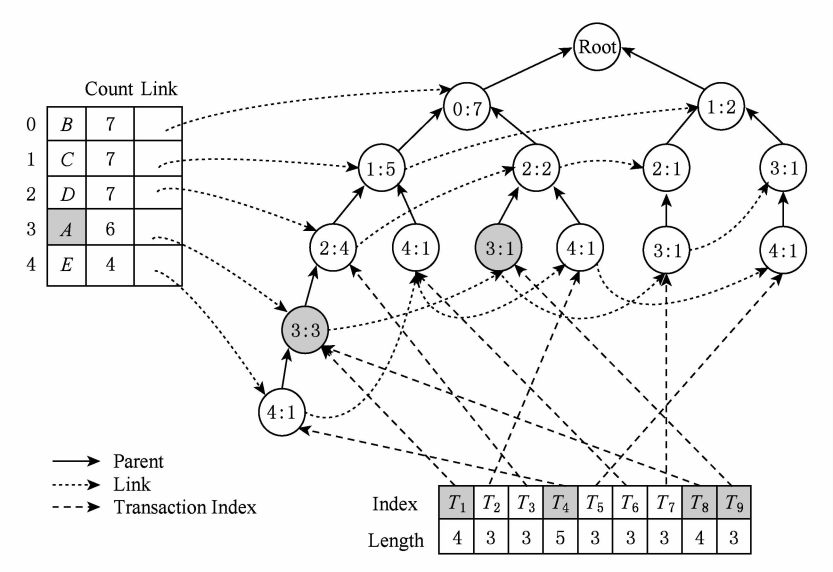

Fig. 4 Find transaction index which supports $A \rightarrow B$

图 4 查找支持 $A \rightarrow B$ 的事务索引

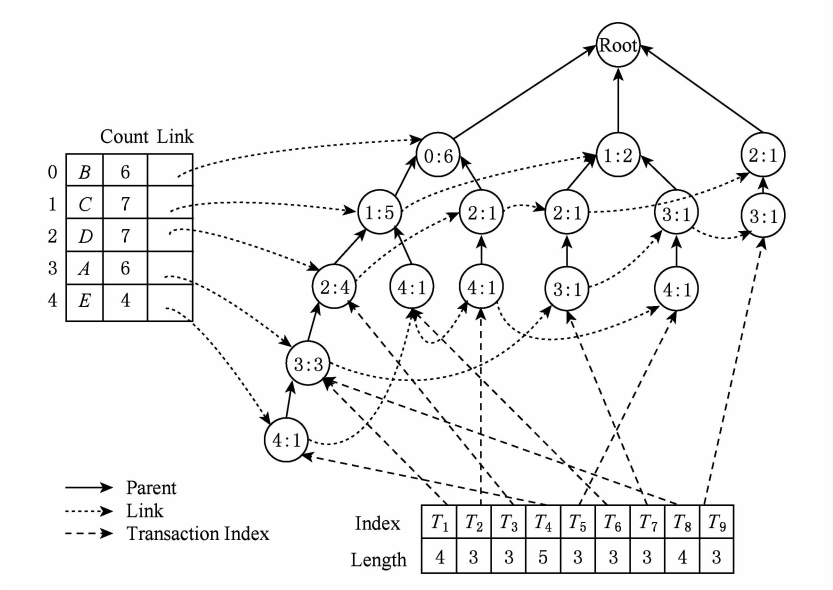

Fig. 5 FP-tree after deleting node B

图 5 删除节点 B 后的 FP-tree

3.3 FP-DSRRC 算法描述

FP-DSRRC 算法的伪代码如算法 1 所示.

算法 1. FP-DSRRC 算法.

输入:原事务数据库 DB、最小支持度阈值 MST、最小置信度阈值 MCT、敏感规则集 SR;

输出:消除敏感规则的事务数据库 DB'.

① 根据 DB 和 MST 建立 FP-tree;

② 利用 FP-Growth 挖掘频繁项集,根据 MCT 计算强关联规则 AR;

③ 根据定义 5 计算敏感度;

④ For each $sr \in SR$

⑤ 根据定义 6 确定待删除项;

⑥ 将 SR 根据待删除项分簇得到 $Clusters$;

⑦ End For

⑧ For each $c \in Clusters$

⑨ 根据定义 5 计算簇敏感度;

⑩ End For

⑪ 将 $Clusters$ 按簇敏感度降序排列;

⑫ While $Clusters \neq \varnothing$ /* 以簇为单位按簇敏感度进行关联规则消除 */

⑬ 选择当前簇敏感度最高的簇 c;

⑭ For each $sr \in c$

⑮ 计算最小删除数 *deleteCount*;
⑯ 维护 1 张表 *T* 记录敏感事务支持的敏感规则数量;
⑰ End For
⑱ While exist *sr*.*deleteCount*≠0 / * 敏感规则未完全隐藏 * /
⑲ 删除 *T* 中支持数最大的事务的敏感项;
⑳ 更新 *deleteCount*;
㉑ End While
㉒ 从 *Cluster* 删除簇 *c*;
㉓ 重新对 FP-tree 进行关联规则挖掘;
㉔ End While
㉕ 根据事务索引表生成 DB';
㉖ Return DB'.

算法 1 中行①～③首先建立 FP-tree,通过挖掘频繁项集得出强关联规则集合,进而计算其敏感度;随后,行④～⑪确定各敏感规则 1 的待删除项,并根据待删除项分簇,同时将簇进行降序排序;算法行⑫～㉔,对所有划分出的簇按照簇敏感度进行关联规则消除,并维护事务索引表;最后,行㉕㉖当所有敏感簇处理完毕后,根据事务索引表生成消除敏感规则的事务数据库 DB'.

4 实验及结果分析

4.1 TestDB 数据集

对于 2.1 节表 1 中给出的精简数据集 TestDB,利用 FP-DSRRC 算法处理后的结果如表 4 所示:

Table 4 New Dataset DB' after Hiding Sensitive Rules

表 4 消除敏感规则后的 DB'

TransactionID	Itemsets After Sanitization
T_1	A,B,C,D
T_2	B,D,E
T_3	B,C,D
T_4	A,B,C,D,E
T_5	A,C,E
T_6	B,E
T_7	A,C,D
T_8	A,B,C,D
T_9	A,D

由表 1 和表 4 中的数据,采用式(3)可分别求得敏感规则的置信度,则对于原始事务数据库 *DB* 中,$E\rightarrow C$ 为 0.75,$A\rightarrow B$ 为 0.67,$A\rightarrow BD$ 为 0.67;而经过 FP-DSRRC 算法处理后生成的事务数据库 DB' 中,$E\rightarrow C$ 为 0.50,$A\rightarrow B$ 为 0.50,$A\rightarrow BD$ 为 0.50,置信度均得以减少到 *MCT* 之下,消除了敏感规则的强关联性.

4.2 真实数据集

为了验证算法的高效性,实验将 FP-DSRRC,DSRRC[7],ADSRRC[8],MDSRRC[9] 在真实数据集下进行测试对比.实验环境在 AMD A6-3420 1.5 GHz CPU 和 8 GB 内存的计算机上,操作系统为 Windows 10,开发语言 Java.本实验采用的数据集是数据挖掘中的 2 个经典数据集 Chess 和 Mushroom[20].其中,Chess 数据集包含 74 个项,共 3 196 条事务记录,其平均长度为 37,比较稠密;Mushroom 数据集含有 119 个项,共 8 124 条记录,其平均长度为 23,比较稀疏,数据集信息如表 5 所示.Chess 数据集的 *MST* 和 *MCT* 分别设置为 90% 和 80%,Mushroom 数据集的 *MST* 和 *MCT* 分别设置为 30% 和 60%,随机选择一定数量的规则作为敏感规则,对比不同算法之间的 *HF*,*MC*,*AP* 运行时间 Runtime 指标.

Table 5 Details of Chess and Mushroom

表 5 Chess 和 Mushroom 数据集信息

Database	Number of Transactions	Number of Items	Average Transaction Length
Chess	3 196	74	37
Mushroom	8 124	119	23

4 种算法都采用删除敏感项的方式来减小敏感规则的支持度或置信度,所以理论上敏感规则都可以被隐藏,在 2 个数据集上的实验也证明了这 4 种算法的隐藏失败率 *HF*=0.

图 6 展示了 4 种算法在 Chess 和 Mushroom 数据集上消除敏感规则后的规则丢失率 *MC*.由于 Chess 数据集较稠密而 Mushroom 数据集较稀疏,Chess 数据集中频繁项的支持度也较大,所以当删除部分敏感项后,4 种算法在 Chess 数据集比 Mushroom 数据集有更低的规则丢失率.

由于 FP-DSRRC 算法将敏感规则分簇后,按簇为单位消除敏感规则,每次选择删除的事务都尽可能支持多个敏感规则,从而使事务删除过程对原始数据集修改较小.从图 6 可以看出随着敏感规则数的增加,所删除的敏感项也越多,规则丢失率也随之增加,但 FP-DSRRC 在较稠密和较稀疏的数据集上的规则丢失率都低于其他算法.

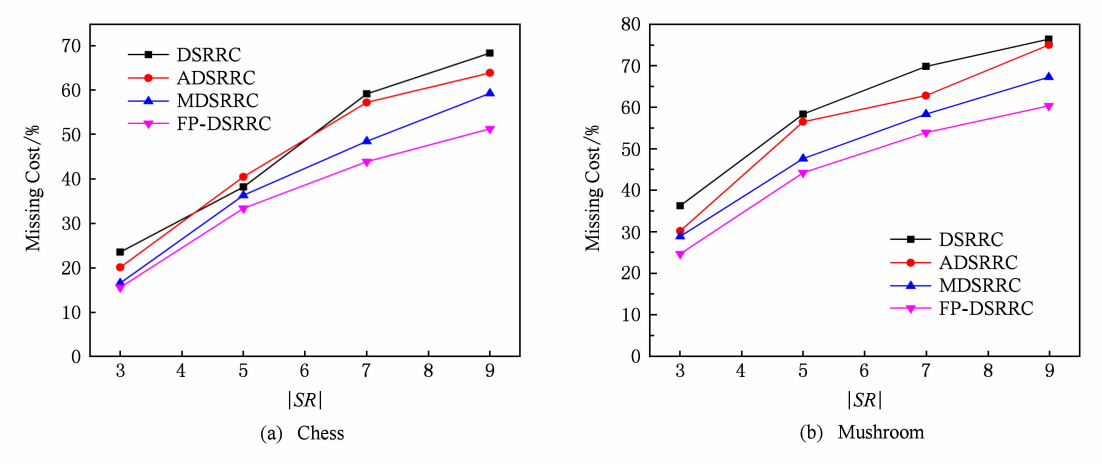

Fig. 6 Missing cost

图 6 规则丢失率

图 7 展示了 4 种算法在 Chess 和 Mushroom 数据集上消除敏感规则后的人为规则产生率 AF,4 种算法均通过删除敏感项改变敏感项的支持度和与其关联的频繁项集支持度,所以都不产生太多的人为规则. 从图 7 可以看出,随着越多的敏感项被删除,4 种算法产生的人为规则都在增加,但总体都处于一个较低的水平. 在 AP 这项指标上,FP-DSRRC 算法与 MDSRRC 算法接近,优于 DSRRC 和 ADSRRC 算法.

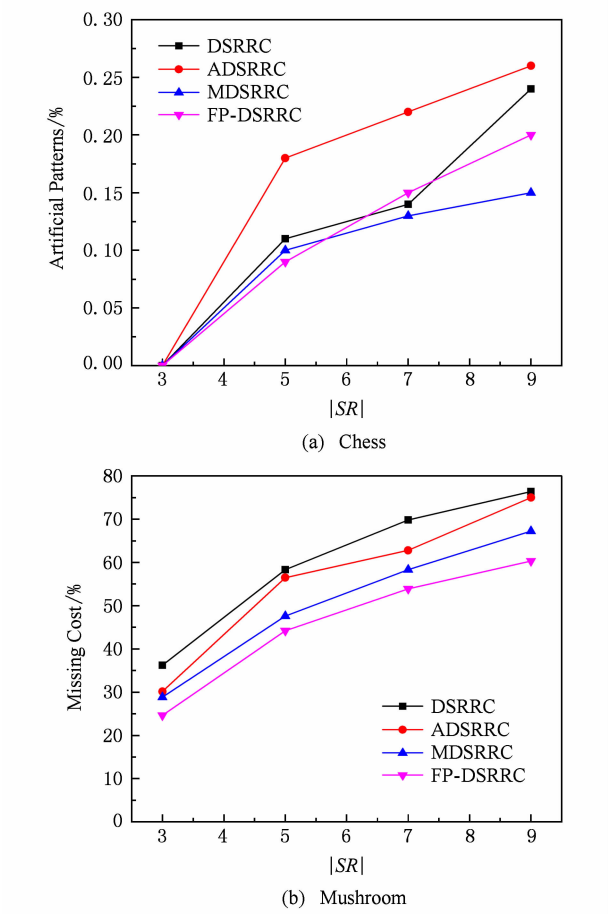

Fig. 7 Artificial patterns

图 7 规则产生率

图 8 展示了 4 种算法在 Chess 和 Mushroom 数据集上的运行时间 Runtime,这也是 FP-DSRRC 算

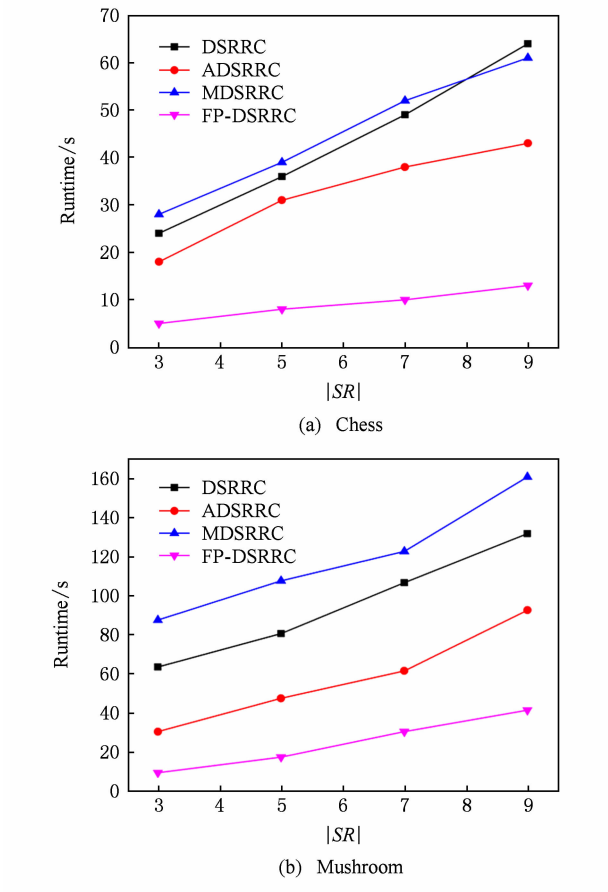

Fig. 8 Runtime of hiding sensitive rules on Chess and Mushroom

图 8 Chess 和 Mushoom 数据集上消除敏感规则的运行时间

法相对于其他 3 种算法提升最显著的指标，由于 FP-DSRRC 算法自始至终都通过维护一棵 FP-tree 而避免了对原事务数据库的重复遍历，并且在每次消除簇后直接利用 FP-Growth 重新挖掘关联规则，减少了大量的 I/O 时间. 当数据集较大时，FP-DSRRC 算法的执行效率优势就越明显，因为数据集越大，遍历数据集所需要的 I/O 时间就越多，而 FP-DSRRC 算法只需要通过 FP-tree 的节点信息就能快速确定事务的索引及敏感度. 从图 8 可以看出在 2 个数据集上 FP-DSRRC 的运行效率都高于其他 3 种算法，并且在更大的 Mushroom 数据集上优势越明显，而且随着敏感规则数的增加，FP-DSRRC 的优势会越来越大.

4.3 大型数据集

为了验证算法的可用性，实验在 2 个较大的数据集 Retail 和 BMS-Webview-2[20] 上进行测试. Retail 数据集由一个比利时零售超市匿名捐赠用于科研分析，数据集包含 5 133 个顾客对 16 469 种商品的 88 163 条购买记录. 由于人们购买那些必备商品、常用商品的频率远远高于那些冷门商品，所以整个数据集呈现出稀疏、不均匀的特点. BMS-Webview-2 数据集由是一个电子商务网站对点击流量的统计，包含 77 512 条记录，数据集的总项数较少，平均事务长度较短. Retail 和 BMS-Webview-2 数据集的详细信息如表 6 所示：

Table 6 Detail of Retail and BMS-Webview-2

表 6 Retail 和 BMS-Webview-2 数据集挖掘结果

Database	Number of Transactions	Number of Items	Average Transaction Length
Retail	88 162	16 470	10.3
BMS-Webview-2	77 512	3 340	5.0

我们设置 Retail 数据集的 $MST=1\%$，$MCT=20\%$；BMS-Webview-2 数据集的 $MST=0.5\%$，$MCT=10\%$，由于数据集较大，DSRRC 算法、ADSRRC 算法和 MDSRRC 算法需要频繁对原事务数据库进行 I/O，时间上已经变的不可控，即这些算法已经不适用于这个大数据集，所以我们给出 HCSRIL 算法[5]、COA4ARH 算法[6]和 FP-DSRRC 算法在 2 个数据集上的运行时间对比，实验依旧随机选取一定数量的强关联规则作为敏感规则.

图 9 展示了 3 种算法在 2 个数据集上的运行时间，随着敏感规则数的增加，算法的执行时间也随之增加. 由于 COA4ARH 算法通过迭代选择最优敏感项消除策略，且一般需要迭代 10 次以上才能获得较好的效果，当数据集较大时单次迭代时间较长，故该算法运行时间最长；HCSRIL 算法先评估所有敏感项对频繁项集的影响，然后逐条消除敏感规则，而 FP-DSRRC 算法通过分簇方式可同时对多条敏感规则进行消除，故其执行时间比 HCSRIL 算法短. 由于 BMS-Webview-2 数据集的密度较低，数据项也较少，所以 3 种算法在 BMS-Webview-2 数据集下的运行时间比 Retail 数据集要长.

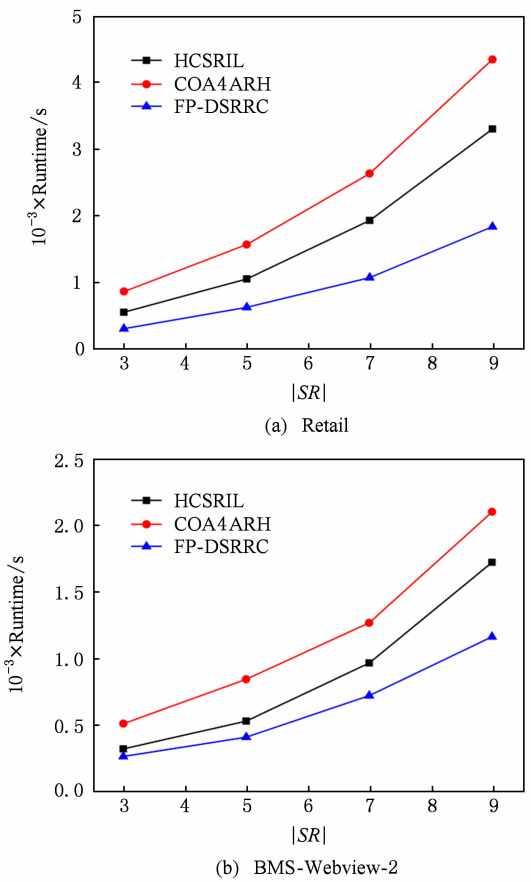

Fig. 9 Runtime of hiding sensitive rules on Retail and BMS-Webview-2

图 9 Retail 和 BMS-Webview-2 数据集上消除敏感规则的运行时间

同时，我们通过 Retail 数据集和 BMS-Webview-2 数据集测试算法的内存开销，实验结果如图 10 所示. BMS-Webview-2 数据集由于事务平均长度较小，项的总个数也较少，所以在相同事务数量的时候，算法执行的内存消耗较在 Retail 数据集时低. 随着事务数量的增加，算法所需内存的增长率在不断变小，这是因为算法很好地利用了 FP-tree 本质上是一棵前缀树的性质，事务在插入 FP-tree 前进行了项序转换并按序号排序，所以越频繁的项总是在

前面.随着 FP-tree 节点的不断增加,FP-tree 会包含更多的频繁的项序,这时有相同前缀的事务就可以只修改节点的计数值而不是耗费内存去创建 1 个新的节点.总体来说,FP-DSRRC 算法在面对这些较大的数据集上仍然有较好的可用性.

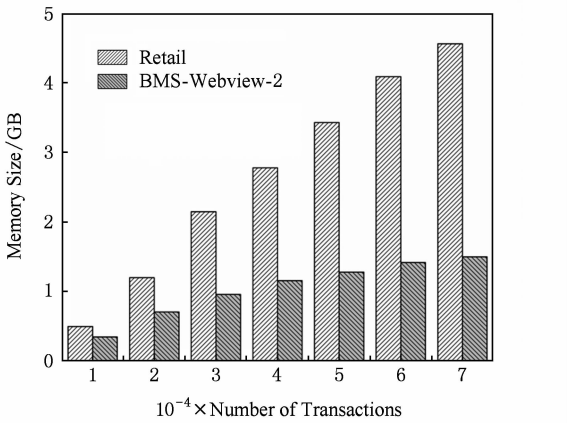

Fig. 10 Memory consumption in Retail

图 10 Retail 数据集上的内存开销

5 结束语

通过 FP-tree,将关联规则隐藏的 2 个步骤频繁项集挖掘和消除关联规则结合起来,提出了一种兼数据集质量和运行效率的算法 FP-DSRRC,通过对敏感规则进行簇划分,以簇为单位进行敏感规则隐藏,并通过计算敏感规则的最小删除数和统计事务的敏感规则支持数保证了清理后数据集的质量,降低了关联规则隐藏对原事务数据库的影响,显著提升了算法运行效率.

参 考 文 献

[1] Chong Zhihong, Ni Weiwei, Liu Tengteng, et al. A privacy-preserving data publishing algorithm for clustering application [J]. Journal of Computer Research and Development, 2010, 47(12): 2083-2089 (in Chinese)
(崇志宏, 倪巍伟, 刘腾腾, 等. 一种面向聚类的隐私保护数据发布方法[J]. 计算机研究与发展, 2010, 47(12): 2083-2089)

[2] Agrawal R, Srikant R. Privacy-preserving data mining [C] // Proc of the 2000 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2000: 439-450

[3] Agrawal R. Mining association rules between sets of items in large databases [C] //Proc of the 1993 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 1993: 207-216

[4] Refaat M, Aboelseoud H, Shafee K, et al. Privacy preserving association rule hiding techniques: Current research challenges [J]. International Journal of Computer Applications, 2016, 136(6): 11-17

[5] Le H Q, Arch-Int S, Nguyen H X, et al. Association rule hiding in risk management for retail supply chain collaboration [J]. Computers in Industry, 2013, 64(7): 776-784

[6] Afshari M H, Dehkordi M N, Akbari M. Association rule hiding using cuckoo optimization algorithm [J]. Expert Systems with Applications, 2016, 64: 340-351

[7] Modi C N, Rao U P, Patel D R. Maintaining privacy and data quality in privacy preserving association rule mining [C] //Proc of the 2nd Int Conf on Computing Communication and Networking Technologies. Piscataway, NJ: IEEE, 2010: 1-6

[8] Shah K, Thakkar A, Ganatra A. Association rule hiding by heuristic approach to reduce side effects & hide multiple RHS items [J]. International Journal of Computer Applications, 2012, 45(1): 1-7

[9] Domadiya N H, Rao U P. Hiding sensitive association rules to maintain privacy and data quality in database [C] //Proc of the 3rd Int Advance Computing Conf (IACC). Piscataway, NJ: IEEE, 2013: 1306-1310

[10] Peng Cheng, Ivan L, Li Li, et al. BRBA: A blocking-based association rule hiding method [C] //Proc of the 13th AAAI Conf on Artificial Intelligence (AAAI'16). Berlin: Springer, 2016: 4200-4201

[11] Jain D, Khatri P, Soni R, et al. Hiding sensitive association rules without altering the support of sensitive items [C] // Proc of the 2nd Int Conf on Computer Science and Information Technology. Berlin: Springer, 2012: 500-509

[12] Cheng Peng, Pan Jeng-Shyang, Lin Chunwei. Use EMO to protect sensitive knowledge in association rule mining by removing items [C] // Proc of the 2014 IEEE Congress on Evolutionary Computation. Piscataway, NJ: IEEE, 2014: 65-66

[13] Guo Yuhong. Reconstruction-based association rule hiding [C] //Proc of ACM SIGMOD 2007. Berlin: Springer, 2007: 51-56

[14] Han Jiawei, Pei Jian, Yin Yiwen. Mining frequent patterns without candidate generation [C] //Proc of the 2000 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2000: 1-12

[15] Ainen T M. On inverse frequent set mining [C] //Proc of Workshop on Privacy Preserving Data Mining. Piscataway, NJ: IEEE, 2003: 18-23

[16] Gui Qiong, Cheng Xiaohui, Rao Jianhui. Privacy preservation association rules mining algorithm based on RSA [J]. Computer Engineering, 2009, 35(17): 138-140 (in Chinese)

(桂琼, 程小辉, 饶建辉. 基于 RSA 的隐私保护关联规则挖掘算法[J]. 计算机工程, 2009, 35(17): 138-140)

[17] Domadiya N H, Rao U P. A hybrid technique for hiding sensitive association rules and maintaining database quality [C] //Proc of the 1st Int Conf on Information and Communication Technology for Intelligent Systems. Berlin: Springer, 2016: 359-367

[18] Shen Yan, Song Shunlin, Zhu Yuquan. Mining algorithm of association rules based on disk table resident FP-TREE [J]. Journal of Computer Research and Development, 2012, 49(6): 1313-1322 (in Chinese)
(申彦, 宋顺林, 朱玉全. 基于磁盘表存储 FP-TREE 的关联规则挖掘算法[J]. 计算机研究与发展, 2012, 49(6): 1313-1322)

[19] Niu Xinzheng, Yang Jian, She Kun. Algorithm of frequent itemsets mining based on array prefix-trees [J]. Journal of Chinese Computer Systems, 2014, 35(8): 1693-1698 (in Chinese)
(牛新征, 杨健, 佘堃. 基于数组前缀树的频繁项集挖掘算法[J]. 小型微型计算机系统, 2014, 35(8): 1693-1698)

[20] Philippe F. SPMF: A Java open-source data mining library [DB/OL]. [2016-07-16]. http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php

Niu Xinzheng, born in 1978. PhD, associate professor. Senior member of CCF. His main research interests include data mining and information security.

Wang Chongyi, born in 1995. Master candidate. His main research interests include social network and data mining.

Ye Zhijia, born in 1993. Master candidate. His main research interest include big data and social network.

She Kun, born in 1969. Professor and PhD supervisor of University of Electronic Science and Technology of China. His main research interests include information security and big data.