软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(3):648−666 [doi: 10.13328/j.cnki.jos.005686]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

差分隐私的数据流关键模式挖掘方法*

王金艳[1,2], 刘 陈[2], 傅星珏[2], 罗旭东[1,2], 李先贤[1,2]

[1](广西多源信息挖掘与安全重点实验室(广西师范大学),广西 桂林 541004)

[2](广西师范大学 计算机科学与信息工程学院,广西 桂林 541004)

通讯作者: 李先贤, E-mail: lixx@gxnu.edu.cn

摘 要: 频繁模式挖掘是数据挖掘的重要任务之一,在数据流上挖掘简洁的关键模式比频繁模式更有优势,因为关键模式既可以避免频繁模式里包含的冗余信息以减少内存存储空间,又可以高效无损地提取频繁模式.但是由于相邻时间戳的统计信息可以作为背景知识增强攻击者的推理能力,所以从包含个人信息的数据流中挖掘关键模式比静态场景下更容易泄露隐私.分析指出了数据流关键模式挖掘的隐私泄露问题及原理,并提出了一种满足差分隐私的数据流关键模式挖掘算法 DP-CPM,该算法在每个时间戳设计一种两阶段机制:差异计算阶段和噪音挖掘阶段.该机制既考虑了隐私和数据效用之间的权衡,又考虑了挖掘时间和维护开销之间的权衡.为了提高数据流中连续发布时的数据效用性,在第 1 阶段通过计算差异来决定当前时间戳是返回低噪音统计值还是精确的近似统计值.如果是返回低噪音统计值,算法进入噪音挖掘阶段.在噪音挖掘阶段,首先通过判断查询集筛选出关键模式候选集,然后通过给筛选出的候选集里的模式支持度加入服从拉普拉斯分布的随机噪音,得到最终的噪音支持度.最后,给出了严格的理论分析和大量的实验,表明 DP-CPM 算法的有效性和执行效率.

关键词: 关键模式;数据流;差分隐私;数据挖掘;隐私泄露

中图法分类号: TP18

中文引用格式: 王金艳,刘陈,傅星珏,罗旭东,李先贤.差分隐私的数据流关键模式挖掘方法.软件学报,2019,30(3):648−666. http://www.jos.org.cn/1000-9825/5686.htm

英文引用格式: Wang JY, Liu C, Fu XC, Luo XD, Li XX. Crucial patterns mining with differential privacy over data streams. Ruan Jian Xue Bao/Journal of Software, 2019,30(3):648−666 (in Chinese). http://www.jos.org.cn/1000-9825/5686.htm

Crucial Patterns Mining with Differential Privacy over Data Streams

WANG Jin-Yan[1,2], LIU Chen[2], FU Xing-Cheng[2], LUO Xu-Dong[1,2], LI Xian-Xian[1,2]

[1](Guangxi Key Laboratory of Multi-Source Information Mining & Security (Guangxi Normal University), Guilin 541004, China)

[2](School of Computer Science and Information Engineering, Guangxi Normal University, Guilin 541004, China)

Abstract: Frequent patterns mining is an important task for data mining. Nevertheless, mining concise crucial patterns is more promising than frequent patterns over data streams, since crucial patterns can avoid redundancy to reduce storage space and extract

* 基金项目: 国家自然科学基金(61502111, 61763003, 61672176, 61762016, 61562007); 广西自然科学基金(2016GXNSFAA380192); 广西科技基地与人才专项(AD16380008); 广西高等学校千名中青年骨干教师培育计划; “八桂学者”工程专项经费资助项目; 广西区域多源信息集成与智能处理协同创新中心

Foundation item: National Natural Science Foundation of China (61502111, 61763003, 61672176, 61762016, 61562007); Guangxi Natural Science Foundation (2016GXNSFAA380192); Guangxi Special Project of Science and Technology Base and Talents (AD16380008); Guangxi 1000-Plan of Training Middle-aged/Young Teachers in Higher Education Institutions; Guangxi “Bagui Scholar” Teams for Innovation and Research Project; Guangxi Collaborative Innovation Center of Multisource Information Integration and Intelligent Processing

本文由智能数据管理与分析技术专刊特约编辑樊文飞教授、王国仁教授、王朝坤副教授推荐.

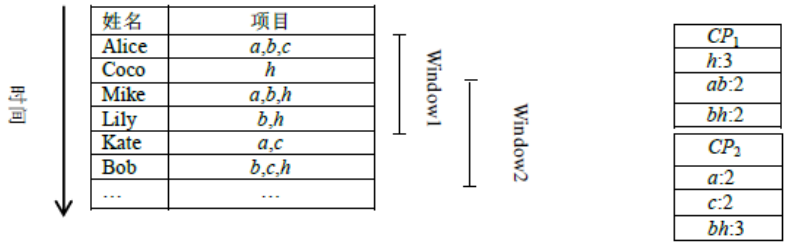
收稿时间: 2018-07-18; 修改时间: 2018-09-20; 采用时间: 2018-11-01

lossless information from frequent patterns. Nevertheless, mining crucial patterns from data streams which aggregate information from individuals is more likely to reveal privacy than static scenarios, because the background knowledge of the release at adjacent time instances can enhance the adversary's inferential ability. This study points out the problems and principles of privacy leakage over mining crucial patterns in data streams, and proposes a differentially private crucial patterns mining algorithm which designs a two-phase mechanism at every timestamp. Specifically, the two-phase mechanism includes the dissimilarity calculation phase and the noise-mining phase, which considers not only the tradeoff between privacy and utility but also the tradeoff between mining time and maintenance cost. To improve data utility over successive releases in streams, the dissimilarity is computed to decide to return either low noisy statistic or accurately approximated statistic in the first phase. When the low noisy statistic needs to be turned, the algorithm goes into the noise-mining phase. In the noise-mining phase, crucial pattern candidate set with a judgment query set is firstly identified, and then random noise drawn from the Laplace distribution to their supports are added to obtain the noisy supports. Finally, strict theoretical analysis and extensive experiments are provided to confirm the effectiveness and efficiency of our algorithm.

Key words: crucial pattern; data stream; differential privacy; data mining; privacy leakage

随着大数据分析技术的发展,从数据流中挖掘频繁模式有着广泛的应用,如生物信息学数据分析、网络流量分析和 Web 使用分析等多种在线应用[1,2].然而,许多频繁超集可以推导出与其子集重复的关联规则.此外,当数据集是密集型或者最小支持度阈值较低时,挖掘出的频繁模式会导致组合爆炸问题[3].因此,研究者们提出了闭合频繁模式和最大频繁模式等简洁模式[4−6],其中使用最广泛的是闭合频繁模式,它是频繁模式的一个有限子集,但包含了频繁模式所有的非冗余信息[5].Das 和 Zaniolo[7]提出一种被称为关键模式的简洁模式,并证明了它是闭合频繁模式的有效子集.关键模式需要更少的存储空间,并且可以高效无损地提取频繁模式.因此,事务数据流环境下更适合挖掘最简洁的关键模式.然而,当事务数据流含有个人的敏感信息时(如顾客的购买记录、用户的行为记录等),直接发布挖掘的统计信息会给数据流中的个人隐私带来很大的威胁[8].显然,数据流连续窗口定期发布统计信息比静态情况下泄漏隐私更为严重:一方面,如果我们将每个滑动窗口视为静态数据集,那每个窗口的隐私泄漏程度则与静态数据相同;另一方面,由于相邻时间戳中的数据具有相关性,所以连续发布相邻窗口的统计信息会增强攻击者的推理能力.因此,攻击者需要更少的背景知识就可以推断出目标的敏感信息.

图 1 的实例表明,连续发布数据流上挖掘出的关键模式将导致隐私泄露问题.假设图 1(a)数据流中的事务代表病人在某医院的购药记录,最小支持度阈值为 2;图 1(b)中的两个关键模式统计信息分别表示了从图 1(a)的第 1 个窗口和第 2 个窗口挖掘出的关键模式集.假设攻击者的背景知识为知道 Mike 是数据流中的第 3 条记录并且患有艾滋病,那么他/她想知道 Alice 是否患有艾滋病.显然,攻击者可以从图 1(b)中所示的统计信息得出,用于治疗艾滋的药物 *a* 和 *b* 的计数在第 1 个窗口中为 2,而在第 2 个窗口中小于 2.这样,他/她只需要知道 Coco 的信息就可以推断出 Alice 是否患有艾滋病.



(a) 数据流 (b) 统计信息

Fig.1 A sliding window example with window size=2 and pane size=2

图 1 窗口大小为 2 窗格大小为 2 的数据流滑动窗口模型

目前,差分隐私[9]已经成为一种广泛应用的隐私保护模型,它能提供严格的理论证明,而且不需要假设攻击者的背景知识.迄今为止,关于差分隐私频繁模式挖掘的研究主要集中在静态场景[10−16],而现有的关于数据流下差分隐私的研究仅限于构成数据流的元组是数值或者分类值的情况[17−23].据掌握的资料来看,尚未有差分隐私

的工作研究数据流上关键模式的挖掘这类更复杂的挖掘任务.此外,上述静态场景中的差分隐私频繁模式挖掘方法不适用于数据流上保护用户隐私的关键模式挖掘:一方面,静态场景下的这些方法需要对整个数据集进行多次扫描,而数据流的动态性要求算法只能对传入的数据扫描一次并提供实时响应;另一方面,现有的静态场景下的研究大多倾向于挖掘少量的 top-k 个频繁模式,这样并不能得到所有频繁模式的完整信息.针对上述问题,本文提出了一种差分隐私关键模式挖掘算法 DP-CPM,实现从事务数据流中挖掘满足差分隐私的关键模式.

为了考虑隐私和数据效用之间以及挖掘时间与维护成本之间的权衡,DP-CPM 算法在每个时间戳设计一种两阶段机制:差异计算阶段和噪音挖掘阶段.在差异计算阶段,为了防止一个窗口中包含的 w 个时间戳分配的隐私预算之和超过总预算,我们首先检查当前的时间戳是否需要强制近似.如果当前时间戳隐私预算足够不需要强制近似,则改进 Das 和 Zaniolo[7]提出的关键模式挖掘算法,先构造当前滑动窗口的前缀树 T_i,并调整 T_i 使其更加紧凑,然后通过调用关键模式计算算法 CPC[7]从当前滑动窗口中挖掘出准确的关键模式集 CP_i.最后计算 CP_i 与最近的隐私发布 $O_t(O_t\in(O_1,\ldots,O_{i-1}))$之间的差异 dis_i,然后根据差异 dis_i 决定当前时间戳是进入噪音挖掘阶段返回低噪音统计值还是直接返回精确的近似统计值.在噪音挖掘阶段,我们设计了两步顺序加噪方法来获得隐私的关键模式及其噪音支持度.通过隐私分析,证明了 DP-CPM 算法满足ε_i-差分隐私,并且预算吸收策略满足 w-event 隐私.在密集和稀疏数据集上的大量实验结果表明:DP-CPM 算法不仅提高了隐私和数据效用之间的权衡,而且还提高了挖掘时间和维护成本之间的权衡.

本文的主要贡献如下:

1) 首次讨论在数据流上挖掘关键模式存在的隐私问题,并指出由于连续时间戳的发布可作为背景知识增强了攻击者的推断能力,所以动态数据流上挖掘的隐私泄漏比静态场景更严重;
2) 为了解决隐私问题,根据关键模式的性质设计了两步顺序加噪方法来发布隐私的关键模式及其噪音支持度;
3) 为了对一个窗口内所包含的 w 个时间戳进行合理的隐私预算分配,我们在每个时间戳设计了一种两阶段机制;
4) 通过隐私分析证明了 DP-CPM 算法满足ε_i-差分隐私;并且在密集和稀疏数据集上的大量实验也表明了 DP-CPM 算法的效用性和执行效率.

本文第 1 节对相关工作进行讨论,并说明我们的工作与现有工作的不同.第 2 节介绍全文所涉及的符号以及差分隐私和关键模式挖掘算法.第 3 节首先详细描述 DP-CPM 算法以及两个关键技术,然后证明 DP-CPM 算法满足ε_i-差分隐私,最后证明预算吸收策略满足 w-event 隐私.第 4 节给出实验结果分析.最后,第 5 节对本文进行总结,并给出下一步研究工作.

1 相关工作

本节从静态环境下满足差分隐私的频繁模式挖掘方法、数据流中精确频繁模式挖掘方法和数据流下满足差分隐私的数据发布方法这 3 个与本文直接相关的方面阐述现有的研究工作,并指出我们方法与其不同之处.

1.1 静态环境下满足差分隐私的频繁模式挖掘方法

从静态数据集中挖掘满足差分隐私的频繁模式的方法有很多,这些方法都是基于经典的 Apriori[24]和 FP-growth[15]算法设计的.例如,Bhaskar 等人[10]采用指数机制[25]和拉普拉斯机制[26]提出了两种不同的差分隐私频繁模式挖掘算法.这两种方法通过截断频率来挖掘隐私的 top-k 个频繁模式,尽管截断频率有效地减少了候选集的大小,但当用户定义的最终输出模式的数目 k 或事务的最大长度限制 l 很大时,这些方法的效率和性能就会降低.为了解决事务数据库维度较高的困难,Li 等人[11]结合θ-基和投影技术提出了 Privbasis 方法,该方法首先挖掘出所有的频繁模式,然后从中识别出最频繁的模式对,根据这些模式对构造θ-基集合 B,最后为θ-基集合 B 产生的频繁模式候选集 $C(B)$里的每个模式的支持度添加随机拉普拉斯噪音.为了应对 Privbasis 造成的数据效用损失过高的问题,Lee 等人[12]表明:阈值查询集相比于计数查询集而言,可以结合前缀树的效用性和紧凑性来修剪候选集,进而提高数据效用性.然而,该方法不适用于 k 很大的情况.Zhang 等人[13]提出了 DP-top-kP 方法,该方法

处理噪音支持度时,通过后处理步骤来保持一致性.同样,该方法无法处理 k 或 l 很大的情况.由于直接截断事务会导致信息损失量太大,Sen 等人[14,15]指出:在处理事务长度约束时,拆分比截断能够保留更多的有用信息.因此,他们分别基于 Apriori[24]和 FP-growth[15]设计了两种分割算法.这两个方法的主要思想是:如果事务长度大于最大长度限制 l,则采用加权分裂方法将其划分为多个子集,最终每个子集的长度都在限制范围内并且更好地保留了原始事务包含频繁模式的结构.但是,事务拆分技术只适用于包含大量短事务的数据集.Ning 等人[16]提出一种新的算法 Privsuper,该算法与以往的差分隐私频繁模式挖掘算法从频繁单个模式开始相反,它采用超集优先的方法,仅在最大频繁模式的支持度中加入噪音来获得所有的隐私频繁模式,从而大大减少了噪音的加入.

为了解决全局敏感性高的问题,DP-CPM 算法首先通过一个判断查询集在不考虑候选集大小的前提下从所有模式中筛选出关键模式候选集,然后对候选集中的每个关键模式的支持度加入随机拉普拉斯噪音.同时,我们挖掘的是关键模式,既能解决只挖 top-k 带来的模式损失的问题,又能解决挖掘全部频繁模式带来的组合爆炸和冗余问题.

1.2 数据流中精确频繁模式挖掘方法

现有的研究大都集中在不考虑隐私的情况下从事务数据流中挖掘精确的频繁模式.Leung 等人[27,28]基于 FP-tree 提出了 CanTree 和 DSTree 两种方法.为了满足数据流一次扫描的限制,他们把扫描进的新事务按照字典顺序插入到前缀树中.然而,这样构造的前缀树不具有像 FP-tree 一样的紧凑性,从而导致挖掘效率较低.Mozafari 等人[29]提出了 SWIM 算法,该算法采用快速计数技术减少挖掘时间,但是当前缀树的尺寸较大时,该算法的性能无法令人满意.Tanbil 等人[1]提出了一种 CPS-Tree,该方法同样按字典顺序将扫描的新事务插入到前缀树,然后再按照降序对前缀树进行重构.然而在这种情况下,因为需要在每个滑动窗口都完全按降序重构前缀树,这就会带来相当大的维护开销.由于在事务数据流上挖掘频繁模式会产生能推出冗余关联规则的频繁子集和频繁超集,所以一些研究更倾向于在事务数据流上挖掘频繁模式的有限子集[7,30,31],这些简洁的模式包括闭合频繁模式、最大频繁模式和关键模式等.一些研究[30,31]强调:当新的事务到达时,它们一方面只需要对已有的节点的支持度进行相应的更新,另一方面需要从新的分支中挖掘新的模式.其中,Chi 等人[30]提出一种内存闭合枚举树方法,该方法可以有效地监视数据流上的闭合频繁模式.最近,Das 和 Zaniolo[7]提出在数据流中挖掘关键模式,并且提高了之前研究中挖掘时间和维护开销的权衡.该算法的主要思想是:采用两种排序方案对新扫描的事务进行排序,即:对“一次性频繁”的项采取已有的顺序($\prec_{est}$)排序,而对“仍然不频繁”的项采取降序($\prec_{desc}$)排序.因为他们认为,随着窗口的滑动,“仍然不频繁”的项的相对顺序会发生改变,而“一次性频繁”的项不用考虑.换句话说,对于窗口滑动导致频繁项变得不频繁时,他们会保留这样的垃圾节点,直到这些垃圾节点在频繁节点中占一定比例才完全重建树.

本文也采用了两种排序方案对新扫描进的事务进行排序.我们的方法与文献[7]的不同之处在于:随着窗口的滑动,我们不仅将新的频繁节点冒泡到已有的顺序($\prec_{est}$)中,而且还从已有的顺序($\prec_{est}$)中去掉了由频繁变得不频繁的垃圾节点,因为差分隐私关键模式算法的性能会受前缀树的紧凑性影响,并且垃圾节点在加噪后也有可能成为关键节点,干扰了关键模式的候选集的筛选.

1.3 数据流下满足差分隐私的数据发布方法

近年来,数据流下满足差分隐私的数据发布研究大多局限于输入记录是数值或分类值的情况[32].Dwork[17]在计数中添加 $Lap(1/\varepsilon)$的拉普拉斯噪音,但是这种方法在稀疏数据中是无效的.Chan 等人[18]考虑在分布式场景下使用不可信聚合器持续监测多条流,他们希望保证每个流的隐私,同时允许不可信的聚合器准确地检测事件及其近似频率.但该方法只检测是否超过阈值并不发布具体的统计值.Bolot 等人[19]考虑最近的数据比过去的数据更重要,使用衰减窗口来处理数据流.Fan 等人[20]利用滑动窗口模型提出了 FAST 方法,该方法包含若干个子机制,且每个子机制都处理一个包含 w 个时间戳的不相交的窗口.因此,他们给每个子机制分配$\varepsilon/2$ 的隐私预算,并将每个子机制包含的子序列当成长度为 w 的有限流来处理.此外,该方法根据预先指定的数目进行抽样,对抽样的时间戳进行发布,对跳过的时间戳直接根据抽样的结果近似.然而,该方法的预算分配取决于样本的数

量和流的长度,所以它不适合无限流.Cao 等人[21]先在时域内指定一组范围查询,其中每个查询请求更新的和,这样做的目的是,通过使用较小子区间的噪音回答来响应较大范围的查询.为了合理地分配隐私预算,进一步提高数据发布的效用性,Kellaris 等人[22]在二进制流上提出了两种隐私分配策略来保护任意 w 个连续时间戳的事件序列的隐私.为了实现滑动窗口中 w 个连续时间戳的隐私发布,该方法计算当前时间戳的精确统计值与最近一个隐私时间戳的噪音发布值之间的差异,然后根据这个差异决定当前时间戳是返回低噪音统计值还是精确的近似统计值.Zhang 等人[23]提出一种自适应采样技术来实现连续时间戳的隐私直方图发布,该方法的主要思想是:当预测值和准确值的差值小于一定阈值时,发布预测值;否则,需要添加随机噪音来发布隐私值.

本文研究复杂事务数据流上满足差分隐私的关键模式挖掘方法.为了合理分配隐私预算,我们也采用了差异的计算来决定当前时间戳是返回低噪音统计值还是精确的近似统计值,与 Kellaris 等人[22]在二进制流中使用的方法不同之处在于噪音规模.在二进制流中,只需要对计数添加 $Lap(1/\varepsilon)$的噪音,因为添加或删除一个用户最多只影响计数为 1.而在我们考虑的事务数据流中,添加或删除一条事务会影响查询集里多个模式的支持度.具体的加噪细节将在第 3 节给出.

综上所述,与现有的静态下频繁模式隐私保护方法不同,本文考虑先使用判断查询集扰动关键模式再对候选集中模式的支持度进行加噪来提高数据效用与隐私的权衡.我们挖掘的关键模式是频繁模式的最佳无损子集,更加适合数据流实时动态的特性.另一方面,对于已有的数据流下快速挖掘频繁模式的相关研究,通常会牺牲一定的紧凑性来提高连续挖掘的效率,而本文首次考虑数据流下连续挖掘关键模式的隐私问题,所以为了提高数据效用性,我们提出了前缀树调整方法,保证在挖掘满足差分隐私的关键模式之前尽可能提高树的紧凑性.最后,本文研究的是复杂事务数据流,与已有的简单的二进制流下加噪不同,我们需要考虑不同的噪音规模.

2 预备知识

2.1 基本概念

给定字母表 $I=\{i_1,...,i_n\}$,S 是由事务 t_i 构成的事务数据流,其中,$t_i\subseteq I$.本文使用图 1 所示的滑动窗口模型[33]来处理事务数据流.滑动窗口模型只对当前数据感兴趣.在滑动窗口模型中,挖掘算法仅维护并挖掘固定宽度为 w 的当前窗口中的关键模式,每个滑动窗口包含$[i-w+1,i]$个时间戳的连续子窗口序列,我们称这些子窗口为窗格,且每个窗格固定宽度为 p,即一个滑动窗口包含 w 个窗格,每个窗格包含 p 条事务.当时间戳过期,超出了当前滑动窗口时,需要删除过期的窗格彻底消除它对当前挖掘结果的影响.当时间戳前进,我们需要把新到达的窗格加入,挖掘新滑动窗口的关键模式.图 1 给出了一个事务数据流和两个连续的滑动窗口,每个窗口包含两个窗格,每个窗格包含两条事物.

设 $\sup_p(W)$表示一个滑动窗口 W 中模式 P 的支持度,即 W 中包含 P 的事务条数.给定最小支持度阈值τ,当一个模式 P 满足 $\sup_p(W)\geqslant\tau$时,则称 P 是滑动窗口 W 中的频繁模式.本文使用的符号描述见表 1.

Table 1 Notations

表 1 标记

名称	描述
$S=\{t_1,t_2,\dots\}$	一个连续的事务数据流
$I=\{i_1,\dots,i_n\}$	组成事务的项的集合
τ	用户定义的最小支持度阈值
CP_i	时间戳 i 挖掘的精确关键模式集
O_i	时间戳 i 发布的关键模式集
C_i	时间戳 i 从所有模式中筛选出的关键模式候选集
D_i	当前滑动窗口包含的从第$(i-w+1)$个时间戳到第 i 个时间戳的事务数据集
β_i	用来与 dis_i 比较的拉普拉斯噪音分布标准差
O_t	属于$\{O_1,\dots,O_{i-1}\}$,离当前时间戳 i 最近的噪音时间戳的隐私发布

接下来介绍关键模式的相关概念.分支号[7]是分配给前缀树中包含频繁项的节点的唯一标签,分配条件是:(i) 如果一个节点是叶子节点;或者(ii) 如果一个非叶子节点的支持度大于其所有孩子节点的支持度之和.

有效分支号组合[7]表示一个模式分支号的集合,例如,集合{1,2}是图 2 中模式 *ab* 的有效分支号组合.如果一个频繁模式的有效分支号组合里至少有一个分支号没有出现在该模式的任何频繁超集的有效分支号组合中,就称该频繁模式为关键模式[7],例如,图 2 中的 *h*,*ab*,*bh* 是第 1 个窗口挖掘出的所有关键模式.

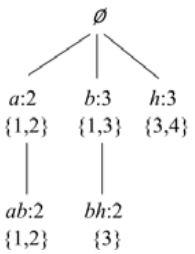


Fig.2 A set enumeration tree of the first window when τ=2

图 2 最小支持度阈值为 2 时第 1 个窗口的集合枚举树

2.2 差分隐私保护模型

Dwork 在 2006 年提出了差分隐私模型[9],该模型有两大优势:(1) 定义了相当严格的攻击模型,不需要假设攻击者的背景知识;(2) 对隐私保护水平给出了严格的理论证明和量化方法.

给定最多相差一条记录的邻居数据集 D 和 D',差分隐私确保更改输入数据库中的单个记录不会影响任何查询的输出结果,从而达到隐私保护的目的.差分隐私的形式化定义如下:

定义 1(**ε-差分隐私**[9]). 对于所有相差一条记录的邻居数据集 D 和 D',给定隐私算法 M,若对于所有输出 $O\in Range(M)$都满足下列不等式,则算法 M 满足ε-差分隐私:

$$\Pr(M(D)\in O)\leqslant \exp(\varepsilon)\Pr(M(D')\in O) \tag{1}$$

其中,参数ε是隐私保护预算,用来决定隐私保护的程度.

定义 2(***w*-邻居流前缀**[22]). 给定一个正整数 w,如果两个流前缀 S_t 和 S_t' 满足:(i) 对于每个 $i\in[t]$且 $S_t[i]\neq S_t'[i]$都有 $S_t[i]$和 $S_t'[i]$ 是相邻的;(ii) 对于每个 $i_1<i_2$ 且 $S_t[i_1]\neq S_t'[i_1], S_t[i_2]\neq S_t'[i_2]$都有 $i_2-i_1+1\leqslant w$,则称流前缀 S_t 和 S_t' 是 w-邻居流前缀.

定义 3(***w*-event 隐私**[22]). 对于所有 w-邻居流前缀 S_t 和 S_t',给定一个机制 M,若对于所有输出 $O\in Range(M)$都满足下列不等式,则 M 满足 w-event 隐私:

$$\Pr(M(S_t)\in O)\leqslant \exp(\varepsilon)\Pr(M(S_t')\in O) \tag{2}$$

噪音机制是用来实现差分隐私的主要技术,其中,基于机制且满足差分隐私算法所需要的噪音大小与全局敏感性[9]密切相关.

定义 4(**全局敏感性**[9]). 对于任意一个函数 $f:D\to R^d$,函数 f 的全局敏感性为:

$$\Delta f=\max_{D,D'}\|f(D)-f(D')\|_1 \tag{3}$$

其中,D 和 D'为相差一条记录的邻居数据集,d 表示函数 f 的查询维度,R 表示所映射的实数空间.全局敏感性只与查询函数本身有关.

最常用的噪音机制是拉普拉斯机制[26],该机制通过拉普拉斯分布产生的噪音去扰动真实的结果值,以达到隐私保护的目的.

定理 1(**拉普拉斯机制**[26]). 对于任意一个函数 $f:D\to R^d$,若算法 M 的输出结果满足下列等式,则算法 M 满足 ε-差分隐私:

$$M(D)=f(D)+\langle Lap_1(\Delta f/\varepsilon),\ldots,Lap_d(\Delta f/\varepsilon)\rangle \tag{4}$$

其中,$Lap_i(\Delta f/\varepsilon)(1\leqslant i\leqslant d)$是相互独立的拉普拉斯变量,噪音规模与$\Delta f$ 成正比,与ε成反比,即:函数 f 的全局敏感性越大,所需的噪音越多.

通常,一个实际的问题都需要多个隐私算法组合实现,这就需要用到差分隐私的序列组合性[34].

定理 2(序列组合性[34]). 给定数据库 D,设 M_i 为任意一个随机算法($1\leqslant i\leqslant n$)满足 ε_i-差分隐私,则算法 M_i 在 D 上的顺序操作满足 $\sum\varepsilon_i$ -差分隐私.

2.3 关键模式挖掘算法

下面描述基于 FP-growth[15]的数据流关键模式挖掘[7]的过程.如图 3 中的前缀树所示,Das 和 Zaniolo[7]在每个节点上都用一个数组分别存储单个窗格项的支持度,这样有利于窗口滑动的更新.在第 1 个窗口,他们先按照字典顺序将扫描到的事务插入到树中构造当前窗口的前缀树,然后按照降序来调整前缀树更加紧凑,一旦得到调整好的前缀树,就根据分支号分配条件给树中的节点分配分支号.接下来根据条件模式基得到如图 2 所示的集合枚举树,最后调用关键模式计算算法 CPC[7]从集合枚举树中识别出关键模式.

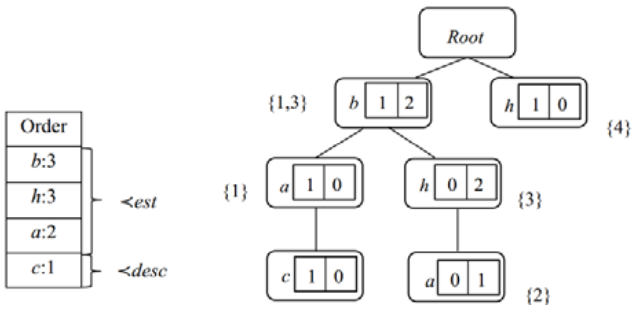


Fig.3 Tree with branch IDs at the first window

图 3 第 1 个窗口带有分支号的前缀树

对于后续的窗口滑动,采用前缀树的增量维护来减少每个窗口完全重建树的操作.一旦窗口滑动,一方面只需要相应地更新已有节点存储项的支持度的数组;另一方面,对于新来的分支,则按照前一时间戳项头表的顺序插入到前缀树.例如,图 4 就是由图 3 窗口滑动得到的.因为挖掘算法的性能与包含频繁项的子树的紧凑性密切相关,因此他们采用两种排序方式来提高挖掘时间和维护成本的权衡.对于新扫描进的事务,他们将“一次性频繁”的项按照已有的顺序($\prec est$)排序,而把“一直不频繁”的项按照降序($\prec desc$)进行排序.窗口滑动一般会面临两种情况:(i) 频繁的项变得不频繁;(ii) 不频繁的项变得频繁.对于情况(i),不采取任何行动;而对于情况(ii),一旦 $\prec desc$ 中有不频繁的项变得频繁,则使用冒泡法将这些项冒泡到 $\prec est$ 里,并从这些新的频繁节点所在分支挖掘新的关键模式.

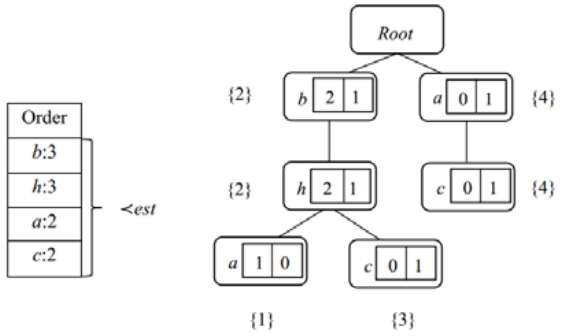


Fig.4 Tree with branch IDs at the second window

图 4 第 2 个窗口带有分支号的前缀树

3 DP-CPM 算法

本节介绍基于差分隐私的数据流关键模式挖掘算法 DP-CPM,用来实现事务数据流上关键模式的隐私发

布.在第 3.1 节,受 Kellaris[22]在二进制流上提出的预算吸收的启发,我们提出了作用在每个时间戳的两阶段机制 M_i.第 3.2 节提出实现数据流上满足差分隐私的关键模式挖掘算法 DP-CPM 的两个子算法.一方面,为了提高挖掘算法的性能和数据效用性,我们提出了维护紧凑的前缀树算法来获得更紧凑的前缀树;另一方面,为了提高隐私和数据效用之间的权衡,我们提出了扰动关键模式候选集算法,该算法受 NoisyCut 算法[12]中的 l-阈值查询集的启发,利用判断查询集从所有模式中筛选出关键模式候选集.第 3.3 节先证明了扰动关键模式候选集算法满足 $\varepsilon_{i,1}$-差分隐私,再根据差分隐私的序列组合性证明 DP-CPM 算法满足 ε_i-差分隐私,最后,证明预算吸收策略满足 w-event 隐私.

3.1 两阶段机制

两阶段机制由差异计算阶段和噪音挖掘阶段组成,不仅考虑了隐私和数据效用之间的权衡,而且考虑了挖掘时间和维护成本之间的权衡.在第 1 阶段,我们计算当前时间戳的准确关键模式集 CP_i 和最近的噪音时间戳的隐私发布 O_t 之间的差异.在时间戳 i 我们需要区分两种情况:(i) 如果之前的时间戳已经吸收了隐私预算,则为了满足差分隐私的序列组合性,ε_i 必须强制为 0,在这种情况下,当前时间戳直接用 O_t 来近似发布 O_i;(ii) 如果之前的时间戳隐私预算没使用,则 ε_i 吸收之前跳过的隐私预算,在这种情况下,我们根据计算差异进一步决定当前时间戳是返回低噪音统计值还是返回准确的近似值.

第 i 个时间戳的两阶段机制 M_i 如图 5 所示,可以看出 M_i 又分成两个子机制 $M_{i,1}$ 和 $M_{i,2}$,这两个子机制独立且按顺序执行.其中,$M_{i,1}$ 先检查时间戳 i 是否必须强制近似,如果不需要强制近似,就调用 CPC 算法[7]挖掘精确的关键模式集 CP_i,然后根据 CP_i 和 O_t 计算当前时间戳的差异 dis_i.这个差异用来判断当前时间戳是使用隐私预算加噪合适还是使用最近的隐私发布来近似合适.接下来,$M_{i,2}$ 首先使用判断查询集从所有模式中筛选出关键模式候选集,然后调用 CPC 算法[7]从扰动过的前缀树中挖掘出关键模式,最后给筛选出的关键模式候选集里的每个模式的支持度添加随机的拉普拉斯噪音.

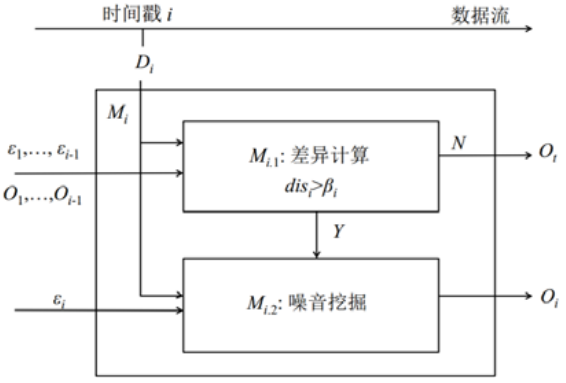


Fig.5 Internal mechanics of M_i

图 5 两阶段机制 M_i 的内部构造

- 差异计算阶段

算法 1 的第 1 部分给出了差异计算阶段的伪代码.一开始,DP-CPM 算法先检查之前的隐私预算是否足够被最近的噪音时间戳吸收,根据最近的噪音时间戳的隐私发布 O_t 和它所使用的隐私预算 ε_t,用 ε_t 除以一个窗口内 w 个时间戳平均分得的隐私预算再减去 1,就可以得到被 ε_t 吸收了的时间戳的个数(即 $to_{nullify}$)(第 2 行),因此,如果 $i-t<to_{nullify}$,则说明隐私预算不够(第 3 行),所以 DP-CPM 算法需要强制 ε_i 为 0,并用 O_t 来近似 O_i(第 4 行).相反,如果隐私预算足够的话,ε_i 则吸收之前跳过的隐私预算(第 5 行~第 7 行).β_i 是用来与差异 dis_i 进行比较的拉普拉斯噪音分布的标准差[23](第 8 行).因为前缀树的紧凑性决定了关键模式挖掘算法的性能,所以 DP-CPM 算法调用维护紧凑的前缀树算法得到紧凑的前缀树 T_i'(第 9 行).接下来,DP-CPM 算法调用 CPC 算法[7]挖掘 CP_i(第 10 行).根据 CP_i,我们计算出 CP_i 和 O_t 之间的差异 dis_i(第 11 行).一方面,如果 $dis_i>\beta_i$,则需要进入噪音挖掘阶段使用

隐私预算去挖掘当前时间戳的隐私关键模式(第 12 行~第 14 行);否则,DP-CPM 算法直接用 O_t 来近似 O_i(第 15 行、第 16 行).

差异计算阶段主要设计了一个公式来计算当前时间戳的精确关键模式集 CP_i 和最近的噪音时间戳的隐私发布 O_t 之间的差异(即 dis_i),因为需要计算两个集合里所包含模式支持度上的差异,所以要递归检查$|O_t \cup CP_i|$个模式支持度的变化,最终计算出模式支持度总差异值的均值来与计数查询的拉普拉斯标准差进行比较.

- 噪音挖掘阶段

算法 1 中的噪音挖掘函数对应子机制 $M_{i.2}$ 的伪代码(第 17 行).首先,DP-CPM 算法调用扰动关键模式候选集算法得到扰乱的关键模式候选集 C_i(第 19 行);然后,DP-CPM 算法给 C_i 里的每个模式的支持度添加:

$$Lap\left(\frac{\max_{Y\in T'}(c(Y)-\sum_{Z\in Y.children}c(Z))}{\varepsilon_{i.2}}\right) \tag{5}$$

噪音得到最终的噪音支持度(第 20 行、第 21 行).

算法 1. DP-CPM:关键模式隐私挖掘算法.

输入:前缀树 T_i,排好序的项头表 L_i',当前时间戳的隐私预算前缀$(\varepsilon_1,\ldots,\varepsilon_{i-1})$,之前的统计发布$(O_1,\ldots,O_{i-1})$;

输出:当前时间戳的统计发布 O_i.

1: **function *DP-CPM***$(T_i, L_i', (\varepsilon_1,\ldots,\varepsilon_{i-1}),(O_1,\ldots,O_{i-1}))$

2: $to_{nullify}=\dfrac{\varepsilon_t}{\varepsilon/w}-1$;

3: **if** $i-t<to_{nullify}$ **then**

4: **return** $O_i=O_t$;

5: **else**

6: $to_{absorb}=i-(t+to_{nullify})$;

7: $\varepsilon_t=\dfrac{\varepsilon}{w}\times\min(to_{absorb},w)$;

8: $\beta_i=\dfrac{2\sqrt{2}}{\varepsilon_i}$;

9: $T_i'=$ ***MaintionCompactPrefixTree***(T_i,L_i');

10: $CP_i=$ ***CPC***(T_i');

11: $dis_i=\dfrac{1}{|O_t\cup CP_i|}\sum_{k=1}^{|O_t\cup CP_i|}|CP_i[k]-O_t[k]|$;

12: **if** $dis_i>\beta_i$ **then**

13: $O_i=$ ***NoiseMining***$(T_i',L_i',\varepsilon_i)$;

14: **return** O_i;

15: **else**

16: **return** $O_i=O_t$;

17: ***function NoiseMining***$(T_i',L_i',\varepsilon_i)$

18: $C_i=\varnothing$;

19: $C_i=$ ***PerturbCrucialPatternCandidateSet***$(T_i',L_i',\varepsilon_{i.1})$;

20: **for** 候选集 C_i 里的每一个模式 R **do**

21: $\tilde{c}(R)=c(R)+Lap\left(\dfrac{\max_{Y\in T'}(c(Y)-\sum_{Z\in Y.children}c(Z))}{\varepsilon_{i.2}}\right)$;

22: $O_i=O_i\cup R$;

23: **return** O_i;

算法 1(DP-CPM 算法)作为每个时间戳的总算法,调用了维护紧凑的前缀树和扰动关键模式候选集两个子算法,经过分析可知:DP-CPM 算法的时间复杂度就是两个子算法时间复杂度的较大值,也是 $O(n^2)$;而空间复杂度为 $O(\max(|O_i|,|O_t|)+6+|CP_i|+(1+|V|)n+|I|)$,其中,$n$ 为前缀树的规模,$|O_i|$,$|O_t|$和$|CP_i|$分别为时间戳 i 发布的关键模式集大小、离当前时间戳 i 最近的噪音时间戳的隐私发布集大小以及时间戳 i 挖掘的精确关键模式集的大小.$|V|$为每个节点有效分支号组合的大小,$|I|$为项头表 L_i' 的大小.因为 DP-CPM 算法在第 2 行、第 6 行~第 8 行、第 11 行以及第 9 行调用的维护紧凑的前缀树算法里的排序操作都需要消耗一个内存空间,所以空间复杂度为 $O(6)$;并且算法 1 分两种情况输出,所以输出的空间复杂度为 $O(\max(|O_i|,|O_t|))$.第 9 行调用维护紧凑的前缀树算法不仅需要一个内存空间来存储节点信息和一个数组来存储该节点的分支号组合 V,还需要$|I|$个内存空间存储项头表 L_i',所以前缀树的空间复杂度为 $O((1+|V|)n+|I|)$;第 10 行消耗$|CP_i|$个内存空间.

3.2 子算法描述

3.2.1 维护紧凑的前缀树算法

因为挖掘算法的性能取决于前缀树的紧凑性,所以我们需要对前缀树进行调整使其更紧凑.给定一个原始前缀树 T_i 和一个按两种排序方式排好序的项头表 L_i',算法 2 首先递归检查项头表中的每个元素来获得可能挖掘出关键模式的所有路径集 $PathArray_i$(第 2 行~第 7 行).对于 $PathArray_i$ 中的每条路径,维护紧凑的前缀树算法递归检查路径中的每个节点 N_k(第 9 行、第 10 行).如果节点 N_k 中包含的项在树 T_i' 中已经存在节点包含相同的项,则直接将 N_k 的支持度添加到已有的那个节点支持度上(第 11 行、第 12 行);否则,需要将 N_k 插入到前缀树 T_i' 中(第 9 行、第 10 行).一旦一条新路径插入完成,维护紧凑的前缀树算法根据分支号分配条件给路径中的节点分配新的分支号(第 13 行、第 14 行).我们提出了一个时间复杂度为 $O(n^2)$、空间复杂度为 $O((1+|V|)n+|I|+1)$的维护紧凑的前缀树算法,以牺牲一定的空间来降低查询时间的开销,以达到提高效率的目的.其中,n 表示前缀树的规模.因为算法 2 先遍历原始树中每条路径的每个节点并需要查找该节点对应的项是否已经存在于排好序的前缀树中,所以时间复杂度为 $O(n^2)$.而算法 2 使用 C#编程实现,对于前缀树里的每个节点,不仅需要一个内存空间来存储节点信息,还需要分配一个数组来存储该节点的分支号组合 V,所以前缀树的空间复杂度为 $O((1+|V|)n)$;而且前缀树所对应的项头表 L_i' 也需要一个大小为$|I|$的数组来存储,空间复杂度为 $O(|I|)$;最后还需要消耗一个内存空间来存储当前节点对应的项用于后面步骤去树中进行查找匹配操作.综上所述,算法 2 的空间复杂度为 $O((1+|V|)n+|I|+1)$.

算法 2. 维护紧凑的前缀树算法.

输入:原始前缀树 T_i,排好序的项头表 L_i';

输出:紧凑的前缀树 T_i'.

```
1:  function MaintainCompactPrefixTree(T_i, L_i')
2:      PathArray_i=∅;
3:      for L_i' 里每个项 E_j do
4:          for E_j 在前缀树 T_i 里的每个连接节点 Node_k do
5:              if (Node_k 是一个支持度大于 0 的叶子节点)||节点 Node_k 是一个关键节点 then
6:              Path_k=从 Node_k 自底向上遍历得到一条路径;
7:                  添加 Path_k 到路径集 PathArray_i;
8:      T_i'=∅;
9:      for PathArray_i 里的每条路径 P_j do
10:         for P_j 里的每个节点 N_k do
11:             if T_i' 里已经出现和节点 N_k 里包含相同项的节点 then
12:                 累加 N_k 的支持度到已有的节点支持度;
13:             else
14:                 插入 N_k 到前缀树 T_i';
```

15: **if** N_k 是路径 P_j 的尾节点&&(N_k 是叶子节点||N_k 是个关键节点) **then**

16: 给路径 P_j 重新分配新的分支号;

17: **return** T_i';

正如第 1.2 节所述,现有的数据流精确频繁模式挖掘方法中,通常采用 PathAdjustingMethod[7]来调整前缀树的紧凑性.与本文提出的维护紧凑的前缀树算法不同:在不考虑隐私泄露的数据流精确频繁模式挖掘时,为了提高挖掘效率,通常通过保留垃圾节点(支持度不满足最小支持度阈值)以牺牲紧凑性为前提来减少树的调整工作;而本文的维护紧凑的前缀树算法第 1 次考虑到数据流关键模式挖掘中的隐私泄露问题,所以需要通过维护紧凑的前缀树算法,在保证一定效率的前提下,尽可能维护最紧凑的前缀树.与之前的调整方法不同,本文提出的维护紧凑的前缀树算法不仅去掉了原始前缀树中的垃圾节点以提高数据的效用性,而且在检查每条路径的时候考虑了分支号条件,高效地获取可能挖掘出关键模式的所有路径集.

3.2.2 扰动关键模式候选集算法

算法 3,我们详细介绍噪音挖掘阶段的第 1 步加噪,即:通过判断查询集,从所有模式中得到扰动的关键模式候选集.从所有模式中扰动关键模式候选集不仅要考虑一个模式的支持度是否大于最小支持度阈值,而且还要考虑一个模式的支持度是否满足分支号分配条件.扰动关键模式候选集算法首先给最小支持度阈值加上 $Lap(4/\varepsilon_{i.1})$的噪音,这个噪音阈值用来筛选模式是否频繁(第 3 行).然后,扰动关键模式候选集算法递归地给前缀树中代表相应模式的节点的支持度添加 $Lap(8/3\varepsilon_{i.1})$的噪音(第 4 行、第 5 行).需要注意的是:这一步筛选操作我们只考虑一个模式是不是关键模式,还没有得到满足差分隐私的支持度.一旦得到一棵扰动的前缀树,扰动关键模式候选集算法需要更新加噪后的项头表,并按照两种排序方案重新调整好顺序(第 6 行),再调用维护紧凑的前缀树算法来调整树 T_i' 的紧凑性(第 7 行).最后,对调整完的前缀树 T_i'' 使用 CPC 算法[7]挖掘出关键模式候选集(第 8 行).因为算法 3 调用了算法 2,且经过分析可得出,该算法的时间复杂度就等于算法 2 维护紧凑的前缀树算法的时间复杂度,所以算法 3 的时间复杂度为 $O(n^2)$.用 C#实现算法 3,所消耗的空间复杂度为 $O(1+(1+|V|)n+1+|I|+|C_i|)$.其中,$n$ 为前缀树的规模,$|V|$为每个节点有效分支号组合的大小,$|I|$为项头表 L_i' 的大小,$|C_i|$为最终从前缀树中挖掘出的关键模式的个数.因为第 3 行需要一个内存空间存储最小支持度阈值,第 4 行、第 5 行需要$(1+|V|)n$ 个内存空间存储树中节点信息以及每个节点的分支号信息,第 6 行排序的空间复杂度为 $O(1)$,第 7 行调用算法 2,需要考虑给项头表 L_i' 分配一个数组空间,最后,第 8 行需要消耗$|C_i|$个内存,所以,算法 3 的空间复杂度为 $O(1+(1+|V|)n+1+|I|+|C_i|)$.

算法 3. 扰动关键模式候选集算法.

输入:紧凑的前缀树 T_i',排好序的项头表 L_i',当前时间戳筛选步骤的隐私预算$\varepsilon_{i.1}$;

输出:关键模式候选集 C_i.

1: ***function PerturbCrucialPatternCandidateSet***$(T_i', L_i', \varepsilon_{i.1})$

2: $C_i=\varnothing$;

3: $\tilde{\tau} = \tau + Lap\left(\dfrac{4}{\varepsilon_{i.1}}\right)$;

4: **for** 前缀树 T_i' 中每个节点 Y **do**

5: $\tilde{c}(Y) = c(Y) + Lap\left(\dfrac{8}{3\varepsilon_{i.1}}\right)$;

6: $L_i'' = L_i'$ 聚合前缀树中连接节点的噪音支持度,并按照 2 种排序方式重新调整头表里项的顺序;

7: T_i''= ***MaintionCompactPrefixTree***(T_i', L_i'');

8: C_i=***CPC***(T_i'');

9: **return** C_i;

3.3 隐私分析

我们对 DP-CPM 算法进行了严格的理论分析:首先证明了噪音挖掘阶段的第 1 步加噪扰动关键模式候选集

算法满足$\varepsilon_{i,1}$-差分隐私,接下来证明了每个时间戳的 M_i(即 DP-CPM)满足ε_i-差分隐私,最后证明了预算吸收满足 w-event 隐私.

定理 3. 算法 3 扰动关键模式候选集算法满足$\varepsilon_{i,1}$-差分隐私.

证明:分 2 步证明从所有模式中扰动关键模式候选集满足$\varepsilon_{i,1}$-差分隐私,其中,分配$\varepsilon_{i,1}$/4 隐私预算用来计算噪音阈值,分配 3$\varepsilon_{i,1}$/4 隐私预算用来识别关键模式候选集.值得注意的是,此阶段不返回所有模式的噪音支持度,而是返回所有满足关键模式判断条件的查询集的二进制结果,所以只需要把隐私预算使用在满足条件的查询集上,直接对不满足条件的模式添加其孩子节点的噪音,不消耗任何隐私预算.根据关键模式的定义可知:判断一个模式是否关键不仅需要考虑模式的支持度是否满足最小支持度阈值,还需要考虑模式的支持度是否满足分支号分配条件,并且这两个步骤是独立的.因此,给定 D 和 D'是邻居数据集,相差一条事务最多影响最小支持度阈值为 1,所以给τ加上拉普拉斯噪音 $Lap(4/\varepsilon_{i,1})$满足$\varepsilon_{i,1}$/4-差分隐私.另外,在与噪音阈值比较判断每个模式是否是关键模式时,需要分两个条件判断,其中每个判断查询集返回一个二进制的向量 $\boldsymbol{v}$,满足判断条件返回 1,否则返回 0.一方面,相差一条事务最多会影响每个模式的支持度为 1,即影响满足最小支持度阈值的候选集的二进制结果为 1;另一方面,相差一条事务最多也会影响支持度满足分支号分配条件的模式候选集的二进制结果为 1.再根据 NoisyCut[12]算法中定理 2 阈值查询集的隐私证明,同样可证无论满足判断条件的查询有多少个,判断查询集都满足差分隐私.所以,给前缀树中的每个节点的支持度添加拉普拉斯噪音 $Lap(8/3\varepsilon_{i,1})$满足 $3\varepsilon_{i,1}$/4-差分隐私.最后,根据差分隐私的序列组合性,算法 3 扰动关键模式候选集满足$\varepsilon_{i,1}$-差分隐私. □

定理 4. 算法 1(即 M_i)满足ε_i-差分隐私.

证明:值得注意的是,算法 1 中只有 $M_{i,2}$ 是隐私的,所以我们将所有的隐私预算都投资在它上面.在噪音挖掘阶段,我们采用两步加噪来得到最终扰动的关键模式及其噪音支持度,其中每个步骤各分配一半的隐私预算(即ε_i=2$\varepsilon_{i,1}$=2$\varepsilon_{i,2}$),其中,第 1 步加噪就是算法 3 扰动关键模式候选集算法,我们在定理 3 已经证明它满足$\varepsilon_{i,1}$-差分隐私.因此,这里证明第 2 步得到噪音支持度满足$\varepsilon_{i,2}$-差分隐私.给定邻居数据集 D 和 D',它们最多相差一个关键模式,则敏感性为

$$\Delta f = \max_{Y\in T'}(c(Y) - \sum_{Z\in Y.children} c(Z)) \tag{6}$$

其中,Y 表示前缀树中的所有节点,Z 表示 Y 的所有孩子节点.因此,向关键模式的支持度中添加 $Lap(\Delta f/\varepsilon_{i,2})$满足$\varepsilon_{i,2}$-差分隐私.根据定理 2 差分隐私的序列组合性,可证明算法 1 满足ε_i-差分隐私,其中,ε_i=$\varepsilon_{i,1}$+$\varepsilon_{i,2}$. □

定理 5. 预算吸收满足 w-event 隐私.

证明:当前时间戳 i 的隐私预算ε_i依赖于之前的时间戳,有如下 3 种情况: (1) ε_i必须强制为 0;(2) ε_i可以吸收之前跳过的隐私预算;(3) ε_i可能被吸收.假设当前时间戳 i 可以从之前的α个时间戳中吸收跳过的隐私预算.根据预算吸收策略[22],要满足:

(i) ε_i=(α+1)×ε/w;

(ii) 对于所有满足(i−α≤t≤i−1)∧(i+1≤t≤i+α)有ε_t=0,

(iii) 0≤α≤w−1.

因此,对于任何包含时间戳 i 并且覆盖 n>α个隐私预算为 0 的滑动窗口,这 n 个隐私预算要么被强制为 0 要么被ε_i吸收了,所以时间戳 i 拥有的总隐私预算最多为(α+1)×ε/w,即,最多的情况是这 n+1 个时间戳都使用均匀分配的隐私预算 $\varepsilon_t = \dfrac{(\alpha+1)\times\varepsilon/w}{n+1} \leqslant \varepsilon/w$. 上面的分析对于任何决定吸收以前跳过的隐私预算的时间戳都是独立的,所以 $\sum_{t=i-w+1}^{i}\varepsilon_t \leqslant \sum_{t=i-w+1}^{i}\varepsilon/w = \varepsilon$. □

4 实验结果与分析

本节通过大量实验表明 DP-CPM 算法的效用性和运行时间,具体分析了隐私预算ε、滑动窗长度 W=窗口大小 w×窗格大小 p,以及相邻窗口的重叠系数 r 这 3 个参数对算法性能的影响.实验环境为 Intel Core i7 CPU 3.60 GHz,8GB 内存,Windows 7 操作系统.使用 C#实现了 DP-CPM 算法,且每个实验数据是 50 次运行的平均值.

4.1 数据集

本实验使用 3 个公开的真实数据集,其中,

- Chess 和 Accidents 是密集型数据集:Chess 由 UCI 提供;而 Accidents 是由 Geurts 提供的匿名的交通事故数据集,该数据集中事务的最大长度为 63,平均长度为 50.5;
- Retail 是由 Brijs 提供的稀疏型数据集,包含匿名的比利时零售市场购物篮数据,该数据集中的每笔交易都是一个收据中的一组项目,其中,最大的事务长度为 76,平均事务长度为 10.3(http://fimi.ua.ac.be/data).

数据集的特点见表 2.

Table 2 Data characteristics

表 2 数据特点

数据集	事务	项
Accidents	340 183	468
Retail	88 162	16 470
Chess	3 196	75

4.2 实验结果

我们从两个方面来评估算法的效用性:(1) 算法发布的结果集中模式的准确性;(2) 算法发布的关键模式的支持度的准确性.为此,我们在实验中使用 F-score[15]和相对误差 RE[11]作为效用性的度量标准,这两个指标可以涵盖以上两个方面的效用性评价指标.

定义 5(F-score). 设 C 和 C'分别表示精确的和已发布的关键模式的集合,则 F-score 的定义如下:

$$F\text{-}score = 2 \times \frac{precision \times recall}{precision + recall} \tag{7}$$

其中, $precision = \frac{|C \cap C'|}{C'}, recall = \frac{|C \cap C'|}{C}$.

定义 6(相对误差 RE). 已发布的关键模式集合 C'的相对误差定义为

$$RE = median_{Y \in C'} \frac{|\tilde{c}(Y) - c(Y)|}{c(Y)} \tag{8}$$

其中,Y 是 C'中的一个关键模式,$c(Y)$表示 Y 的真实支持度,$\tilde{c}(Y)$ 表示发布的 Y 的支持度.

4.2.1 参数 W 和ε变化对效用性和运行时间的影响

为了表明参数 W 和ε对 DP-CPM 算法性能的影响,本组实验固定 r=75%,W 从 50 变化到 250,隐私预算从 0.5 变化到 2.5.根据不同数据集的大小和密集程度,我们给数据集设置了不同的最小支持度阈值.对于数据集 Chess,Accidents 和 Retail,τ分别为 40,45 和 6.

图 6 显示了在 W 和ε变化时,F-score 的变化趋势.

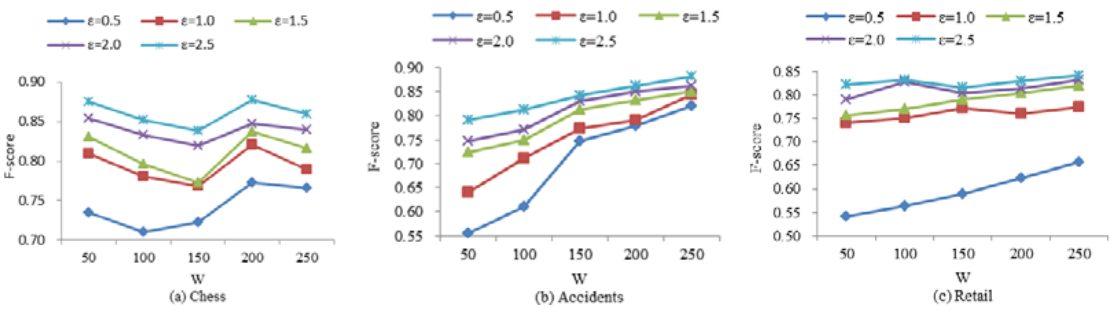


Fig.6 Effect of parameters W and ε on F-score

图 6 参数 W 和ε对 F-score 的影响

从图 6(a)和图 6(c)我们可以看出:当隐私预算相同时,F-score 不会随着 W 的增加而改变.这是因为我们的扰动关键模式候选集算法与前缀树的大小是不相关的,只依赖于前缀树的紧凑性,因此在数据集 Chess 和 Retail 中,关键模式的扰动不会随着 W 的增加而改变.相反,我们可以看到,图 6(b)中的 F-score 随着 W 的增加而变得更好.这是因为数据集 Accidents 比之前的两个数据集更密集,所以尺寸越大的前缀树会变得更加紧凑.因此,我们可以挖掘出更多准确的关键模式.为了理解参数ε如何影响 F-score,从图 6 我们观察到:ε越大,F-score 越高,性能越好.这是因为更大的ε计算出的噪音更小;并且当$\varepsilon \geqslant 1$ 时,性能就变得趋于稳定.因此,我们在其他组实验中设置 ε=1.

从图 7 可以看出:随着 W 增大,RE 变小,算法性能更好;尤其是在密集型数据集 Chess 和 Accidents 中,效果更明显.这是因为我们从更大的窗口中挖掘出来的模式支持度更大,而噪音的大小和窗口长度无关,所以 W 越大 RE 越小.但随着窗口变大,图 7(c)中的 RE 减少的比在其他数据集中更少.这是因为稀疏数据集 Retail 随着 W 增大,模式的支持度增加的相对较少.从图 7 中我们还注意到:当ε变大时,RE 变得更小.因为ε越大,噪音越小.

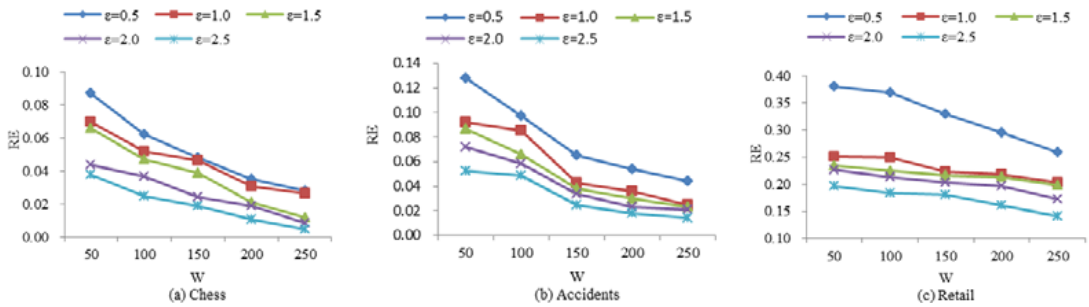


Fig.7 Effect of parameters W and ε on RE

图 7 参数 W 和ε对相对误差的影响

图 8 通过改变 W 和ε来显示我们 DP-CPM 算法的运行时间.可以看到:在每个数据集中,随着ε的变化,运行时间并没有任何规律.其原因是参数ε只影响算法效用性,而不影响算法的的运行时间.从图 8 我们还可以看到:对于所有数据集,运行时间都会随着 W 的增加而不断增加.这是因为窗口越大,前缀树越大,扫描的时间更长.

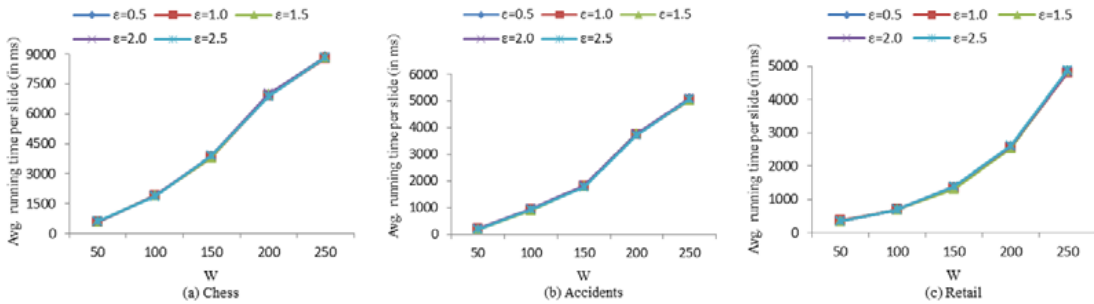


Fig.8 Effect of parameters W and ε on runtime

图 8 参数 W 和ε对运行时间的影响

4.2.2 参数 r 和ε变化对效用性和运行时间的影响

为了度量参数 r 和ε对 DP-CPM 算法的影响,本组实验固定 W=100,r 从 50%变化到 90%,隐私预算从 0.5 变化到 2.5.根据不同数据集的大小和密集程度我们给数据集设置了不同的最小支持度阈值.对于数据集 Chess, Accidents 和 Retail, τ分别为 40,45 和 6.

图 9 显示了在 r 和ε变化时,3 个数据集上 F-score 的变化趋势.首先,我们可以看到,随着ε变大,F-score 越大,算法性能会变得更好,原因是较大的ε会产生更少的噪音.同时,我们可以看到:当ε等于或大于 1 之后,性能趋于稳

定,所以我们在其他组实验中设置ε=1.此外,从图 9 中还可以观察到:r 越大,F-score 越大,性能会变得更好.这是因为 r 越高,意味着相邻窗口的重叠越多,所以两阶段机制计算出的差异就越小,我们就可以吸收更多跳过的隐私预算来挖掘更准确的隐私关键模式.

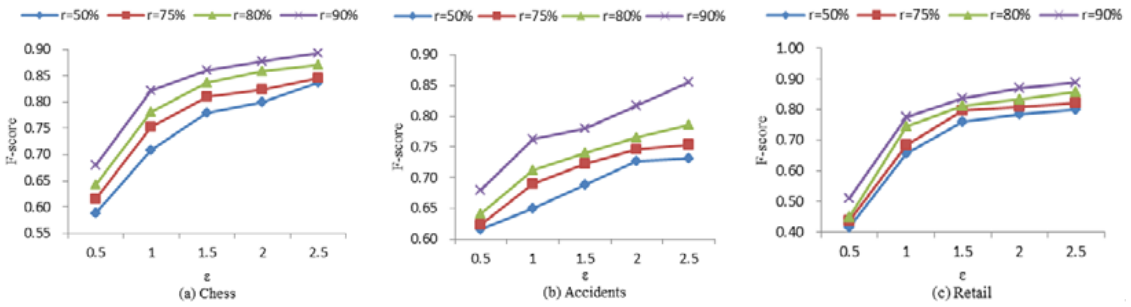


Fig.9 Effect of parameters r and ε on F-score

图 9 参数 r 和ε对 F-score 的影响

图 10 显示了在 3 个数据集上,通过改变ε和 r 对 RE 的影响.同样地,当ε变得更大时,RE 越小,算法的性能更好.这是因为越大的ε意味着越小的噪音.图 10(a)和图 10(b)分别显示了在密集型数据集 Chess 和 Accidents 里的 RE,我们可以看到,密集数据集里的 RE 比图 10(c)中稀疏数据集里的 RE 值更小.这是因为当 W 相同时,从密集数据集挖掘出的模式的支持度比稀疏数据集的大.因此,在密集数据集中 RE 更小.另外,从图 10 我们还可以注意到:r 越大,RE 越小,性能越好.原因是较高的 r 意味着相邻窗口的重叠更多,因此两阶段机制计算出的差异更小,可以吸收更多跳过的隐私预算来提高数据效用性.

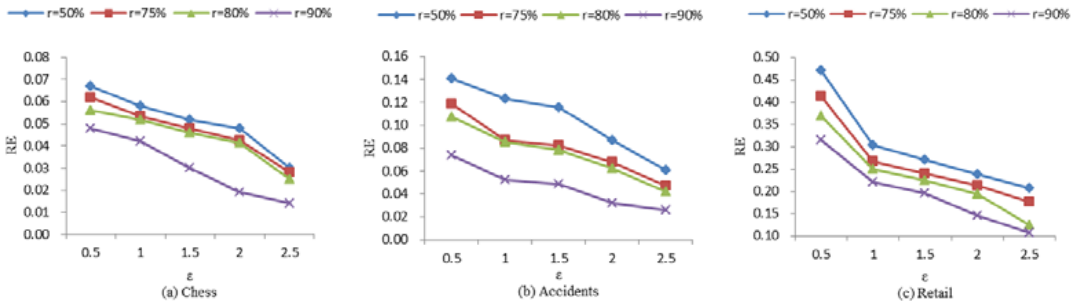


Fig.10 Effect of parameters r and ε on RE

图 10 参数 r 和ε对相对误差的影响

图 11 给出了ε和 r 不同时算法的运行时间变化趋势.

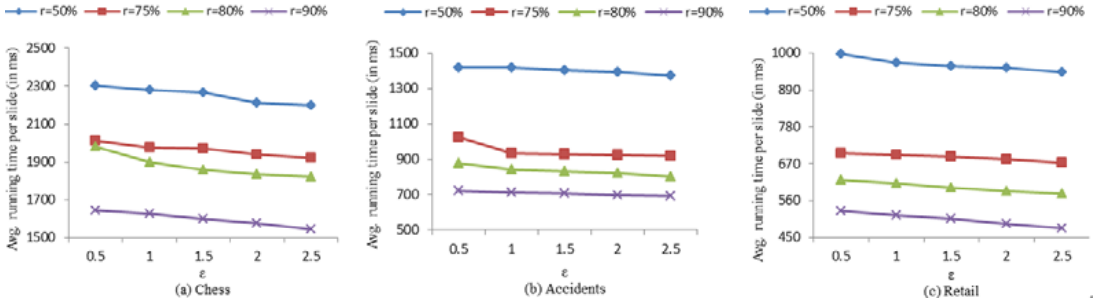


Fig.11 Effect of parameters r and ε on runtime

图 11 参数 r 和ε对运行时间的影响

很明显,r 越大,运行时间越少.原因如下:一方面,较高的重叠系数意味着窗口滑动时需要扫描更少的新事务;另一方面,较高的 r 则意味着相邻窗口的重叠更多,因此,我们根据计算出的更小的差异而直接返回近似统计值.此外,在每个数据集中,当 r 和 W 相同时,运行时间不随着 ε 的变化而变化.这是因为参数 ε 只影响数据效用而不影响运行时间.

4.2.3 参数 r 和 W 变化对效用性和运行时间的影响

为了表明参数 W 和 r 如何影响 DP-CPM 算法的性能,本组实验我们固定 ε=1,W 从 50 变化到 250,r 从 50% 变化到 90%.根据不同数据集的大小和密集程度,我们给数据集设置了不同的最小支持度阈值.对于数据集 Chess,Accidents 和 Retail,τ 分别为 40,45 和 6.

图 12 显示了在 W 和 r 变化时,F-score 的变化趋势.从图 12(a)和图 12(c)我们可以看出:当 r 相同时,F-score 不会随着 W 的增加而改变.这是因为我们扰动关键模式候选集算法与前缀树的大小是不相关的,只依赖于前缀树的紧凑性,因此在数据集 Chess 和 Retail 中,关键模式的扰动不会随着 W 的增加而改变.相反,我们可以看到,图 12(b)中的 F-score 随着 W 的增加而变得更好.这是因为数据集 Accidents 比之前 2 个数据集更密集,所以尺寸越大的前缀树会变得更加紧凑,因此我们可以挖掘出更多准确的关键模式.另外,从图 12 也可以看出,r 越大,F-score 越大,性能会变得更好.这是因为 r 越高,意味着相邻窗口的重叠越多,所以两阶段机制计算出的差异就越小,我们就可以吸收更多跳过的隐私预算来挖掘更准确的隐私关键模式.

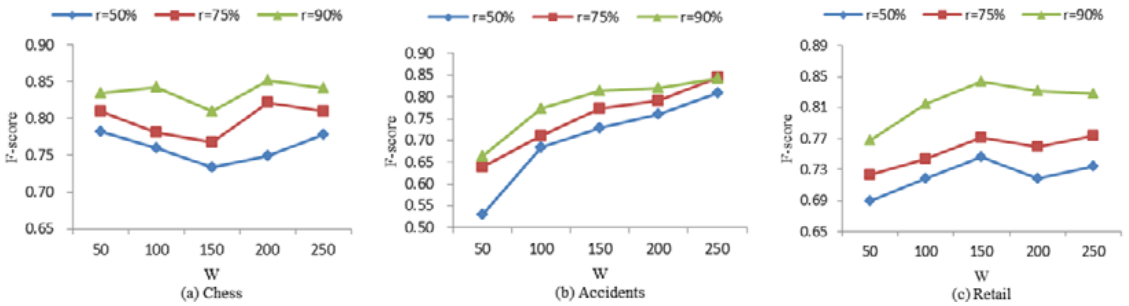


Fig.12 Effect of parameters W and r on F-score

图 12 参数 W 和 r 对 F-score 的影响

从图 13 我们可以看出:随着 W 增大,RE 变得更小,算法性能更好,尤其是在密集型数据集 Chess 和 Accidents 中效果更明显.这是因为我们从更大的窗口中挖掘出来的模式的支持度更大,而噪音的大小和窗口长度无关.综上所述,W 越大,RE 越小.但随着窗口变大,图 13(c)中的 RE 减少的比在其他数据集中更少.这是因为稀疏数据集 Retail 随着 W 增大,相同模式的支持度增加的相对较少.另外,从图 13 我们还可以注意到:r 越大,RE 越小,性能越好.原因是较高的 r 意味着相邻窗口的重叠更多,因此两阶段机制计算出的差异更小,可以吸收更多跳过的隐私预算来提高数据效用性.

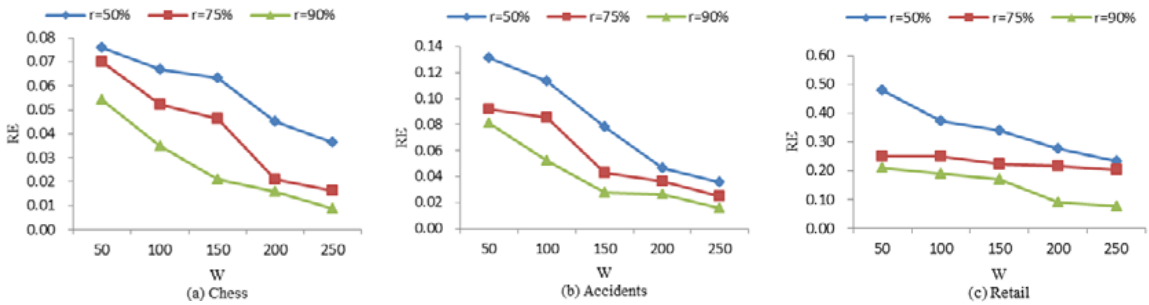


Fig.13 Effect of parameters W and r on RE

图 13 参数 W 和 r 对相对误差的影响

图 14 通过改变 W 和 r 来显示我们 DP-CPM 算法的运行时间.我们可以看到:对于所有数据集,运行时都会随着 W 的增加而不断增加.这是因为窗口越大,前缀树越大,扫描的时间更长.另外,r 越大,运行时间越少.原因如下:一方面,较高的重叠系数意味着窗口滑动时需要扫描更少的新事务;另一方面,较高的 r 意味着相邻窗口的重叠更多,因此我们根据计算出小的差异而直接返回近似统计值.

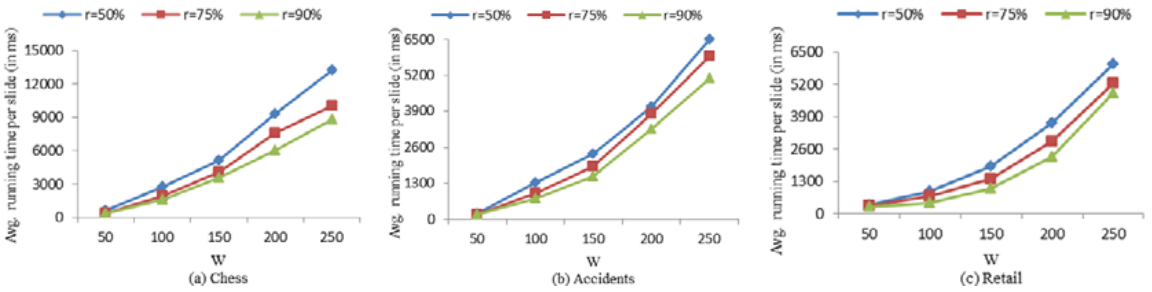


Fig.14 Effect of parameters W and r on runtime

图 14 参数 r 和 W 对运行时间的影响

综上所述:F-score 与参数 W 的变化无关;而相对误差 RE 会随着参数 W 的增加而减小,尤其是在密集型数据集里效果更明显,并且运行时间会随着参数 W 的增加而增加.这说明窗口越大,挖掘出的关键模式支持度上的误差越小,而挖掘时间越长.另一方面,可以看出,随着参数 r 的增加,F-score 变大,相对误差 RE 变小,且运行时间变小.这说明窗口滑动重合的部分越多,连续发布的数据效用性越高,连续挖掘的时间越少.运行时间与参数ε的变化无关,而随着ε的增加,F-score 变大,相对误差 RE 变小.这说明隐私预算越高,数据效用性越好.

5 总结语

随着大数据的发展,挖掘频繁模式不再局限于静态的集值数据,更多的是从动态的事务数据流中实时挖掘频繁模式.目前,已有相关的工作分析并解决了静态下频繁模式挖掘的隐私问题;而随着一些数据流下实时高效的频繁模式挖掘工作的出现,我们发现数据流下连续发布精确的频繁模式比静态更容易泄露隐私,且还没有相关工作涉及这个亟待解决的问题.本文首次讨论并指出了在事务数据流上挖掘关键模式存在的隐私问题,为了解决这个问题,我们利用差分隐私设计了一种包含两阶段机制的 DP-CPM 算法,该算法不仅提高了隐私和数据效用之间的权衡,而且还提高了挖掘时间和维护成本之间的权衡.在加噪阶段,我们根据关键模式的特点,利用稀疏向量技术提出了一种新的加噪方法,可以在满足隐私的前提下,最大限度地提高数据效用性.最后,我们在密集型数据集和稀疏型数据集上进行了大量的实验来表明 DP-CPM 算法的效用性和效率性.目前,数据流上频繁序列以及频繁子图的挖掘有着广泛的应用,但是这些挖掘方法也存在隐私泄露问题,因此在今后的工作中,可以改进我们的方法来实现数据流上频繁序列和频繁子图挖掘的隐私保护.

References:

[1] Tanbeer SK, Ahmed CF, Jeong B, Lee Y. Sliding window-based frequent pattern mining over data streams. Information Sciences, 2009,179(22):3843−3865.

[2] Zheng YY, Tian Y, Shi C. Method of entity set expansion based on frequent pattern under meta path. Ruan Jian Xue Bao/Journal of Software, 2018,29(10):2915−2930 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5549.htm [doi: 10.13328/j.cnki.jos.005549]

[3] Taouil R, Pasquier N, Bastide Y, Lakhal L. Mining bases for association rules using closed sets. In: Proc. of the 16th Int'l Conf. on Data Engineering. 2000. 307.

[4] Calders T, Rigotti C, Boulicaut JF. A survey on condensed representations for frequent sets. Constraint-based Mining and Inductive Databases, 2006,48(38):64−80.

[5] Giacometti A, Li DH, Marcel P, Soulet A. 20 years of pattern mining: A bibliometric survey. ACM SIGKDD Explorations Newsletter, 2014,15(1):41−50.

[6] Hellal A, Romdhane LB. Minimal contrast frequent pattern mining for malware detection. Computers & Security, 2016,62:19−32.

[7] Das A, Zaniolo C. Fast lossless frequent itemset mining in data streams using crucial patterns. In: Proc. of the 2016 SIAM Int'l Conf. on Data Mining. 2016. 576−584.

[8] Meng XF, Lin DD. Introduction to the topic of data opening and privacy management. Ruan Jian Xue Bao/Journal of Software, 2016,27(8):1889−1890 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5097.htm [doi: 10.13328/j.cnki.jos.005097]

[9] Dwork C. Differential privacy. In: Proc. of the 33rd Int'l Colloquium on Automata. Languages and Programming, 2006. 1−12.

[10] Bhaskar R, Laxman S, Simth A, Thakurta A. Discovering frequent patterns in sensitive data. In: Proc. of the 16th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. 2010. 503−512.

[11] Li N, Qardaji W, Su D, Cao J. Privbasis: Frequent itemset mining with differential privacy. VLDB Endowment, 2012,5(11):1340−1351.

[12] Lee J, Clifton C. Top-*k* frequent itemsets via differentially private FP-trees. In: Proc. of the 20th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. 2014. 931−940.

[13] Zhang XJ, Wang M, Meng XF. An accurate method for mining top-*k* frequent pattern under differential privacy. Journul of Computer Research and Development, 2014,51(1):104−114 (in Chinese with English abstract).

[14] Cheng X, Su S, Xu S, Li Z. DP-apriori: A differentially private frequent itemset mining algorithm based on transaction splitting. Computers & Security, 2015,50:74−90.

[15] Su S, Xu S, Cheng X. Differentially private frequent itemset mining via transaction splitting. IEEE Trans. on Knowledge and Data Engineering, 2015,27(7):1875−1891.

[16] Wang N, Xiao X, Yang Y, Zhang Z, Gu Y, Yu G. PrivSuper: A superset-first approach to frequent itemset mining under differential privacy. In: Proc. of the 33rd Int'l Conf. on Data Engineering. 2017. 809−820.

[17] Dwork C. Differential privacy in new settings. In: Proc. of the 21th ACM-SIAM Symp. on Discrete Algorithms. 2010. 174−183.

[18] Chan THH, Shi E, Song D. Private and continual release of statistics. ACM Trans. on Information and System Security, 2011,14(3):1−24.

[19] Bolot J, Fawaz N, Muthukrishnan S, Nikolov A, Taft N. Private decayed predicate sums on streams. In: Proc. of the 16th Int'l Conf. on Database Theory. 2013. 284−295.

[20] Fan L, Xiong L, Sunderam V. FAST: Differenyially private real-time aggregate monitor with filtering and adaptive sampling. In: Proc. of the 2013 ACM Conf. on Management of Data. 2013. 1065−1068.

[21] Cao J, Xiao Q, Ghinita G, Li N, Bertino E, Tan KL. Efficient and accurate strategies for differentially-private sliding window queries. In: Proc. of the 16th Int'l Conf. on Extending Database Technology. 2013. 191−202.

[22] Kellaris G, Papadopoulos S, Xiao X, Papadias D. Differentially private event sequences over infinite streams. Int'l Conf. on Very Large Data Bases, 2014,7(12):1155−1166.

[23] Zhang XJ, Meng XF. Streaming histogram publication method with differential privacy. Ruan Jian Xue Bao/Journal of Software, 2016,27(2):381−393 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4863.htm [doi: 10.13328/j.cnki.jos.004863]

[24] Agrawal R, Srikant R. Fast algorithms for mining association rules in large databases. In: Proc. of the 20th VLDB. 1994. 487−499.

[25] Mcsherry F, Talwar K. Mechanism design via differential privacy. In: Proc. of the 48th IEEE Symp. on Foundations of Computer Science. 2007. 94−103.

[26] Dwork C, Mcsherry F, Nissim K, Smith A. Calibrating noise to sensitivity in private data analysis. Theory of Cryptography Conf., 2006,76(38):265−284.

[27] Leung CK, Khan QI. DSTree: A tree structure for the mining of frequent sets from data stream. In: Proc. of the 6th Int'l Conf. on Data Mining. 2006. 928−932.

[28] Leung CKS, Khan QI, Li Z, Hoque T. CanTree: A tree structure for efficient incremental mining of frequent patterns. In: Proc. of the 5th Int'l Conf. on Data Mining. 2005. 274−281.

[29] Mozafari B, Thakkar H, Zaniolo C. Verifying and mining frequent patterns from large windows over data streams. In: Proc. of the 24th Int'l Conf. on Data Mining. 2008. 179−188.

[30] Chi Y, Wang H, Yu PS, Muntz RR. Moment: Maintaining closed frequent itemsets over a stream sliding window. In: Proc. of the 4th Int'l Conf. on Data Mining. 2004. 59−66.

[31] Jiang N, Gruenwald L. CFI-stream: Mining closed frequent itemsets in data streams. In: Proc. of the 12th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. 2006. 592−597.

[32] Qin Z, Yang Y, Yu T, Khalil I, Xiao X, Ren K. Heavy hitter estimation over set-valued data with local differential privacy. In: Proc. of the 12th ACM Conf. on Computer and Communications Security. 2016. 192−203.

[33] Lee VE, Jin R, Agrawal G. Frequent pattern mining in data stream. Data Streams, 2014,31:199−224.

[34] McSherry F. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In: Proc. of the 2009 ACM Conf. on Management of Data. 2009. 19−30.

附中文参考文献:

[2] 郑玉艳,田莹,石川.一种元路径下基于频繁模式的实体集扩展方法.软件学报,2018,29(10):2915−2930. http://www.jos.org.cn/1000-9825/5549.htm [doi: 10.13328/j.cnki.jos.005549]

[8] 孟小峰,林东岱.数据开放与隐私管理专题前言.软件学报,2016,27(8):1889−1890. http://www.jos.org.cn/1000-9825/5097.htm [doi: 10.13328/j.cnki.jos.005097]

[13] 张啸剑,王淼,孟小峰.差分隐私保护下一种精确挖掘 Top-*k* 频繁模式方法.计算机研究与发展,2014,51(1):104−114.

[23] 张啸剑,孟小峰.基于差分隐私的流式直方图发布方法.软件学报,2016,27(2):381−393. http://www.jos.org.cn/1000-9825/4863.htm [doi: 10.13328/j.cnki.jos.004863]

王金艳(1982－),女,广西资源人,博士,副教授,CCF 专业会员,主要研究领域为数据安全,自动推理,不确定性理论.

刘陈(1993－),女,学士,主要研究领域为数据安全,数据挖掘.

傅星珵(1988－),男,硕士,主要研究领域为推荐系统,隐私保护,机器学习.

罗旭东(1963－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为人工智能,管理科学和工程,逻辑学.

李先贤(1969－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为数据安全,分布式系统安全,可信软件,形式化理论.