基于本地化差分隐私保护的频繁项目挖掘算法

朱美琪[1,2],杨 庚[1,2],白云璐[1,3]
(1. 南京邮电大学 计算机学院、网络空间安全学院,江苏 南京 210023;
2. 江苏省大数据安全和智能处理重点实验室,江苏 南京 210023;
3. 南京市医药大学 信息技术学院,江苏 南京 210023)

摘 要:频繁项目挖掘是数据挖掘的研究热点之一,若数据集包含敏感信息,不作处理地发布挖掘结果会有隐私泄露的风险。目前已有本地化差分隐私的频繁项目挖掘算法,但还无法满足处理大数据时的实时性和数据可用性要求。针对这些问题,该文提出了一种新的面向本地化差分隐私保护的频繁项目挖掘算法—GFIM (group-based frequent items mining)。该算法把用户随机划分为不相交且大小相等的两组用户,整个运行过程分为两个阶段。第一阶段主要根据全部用户提交的信息挖掘出频繁项目的候选集 C,而在第二阶段,两组用户分别通过设置冗余项把自身修剪为 $O(k)$ 发送给数据收集者,最终的 top-k 频繁项目将利用上述两个阶段的结果。采用分阶段的思想减少了计算时遍历数据集的次数,加快了整体的运行速度。通过理论证明了该算法满足 ε -本地化差分隐私,在多个真实数据集上的实验也验证了该方法的性能。

关键词:频繁项目挖掘;本地化差分隐私;集值数据;隐私保护;随机响应
中图分类号:TP309 **文献标识码**:A **文章编号**:1673-629X(2021)08-0092-08
doi:10.3969/j.issn.1673-629X.2021.08.016

Frequent Items Mining for Local Differential Privacy Protection

ZHU Mei-qi[1,2], YANG Geng[1,2], BAI Yun-lu[1,3]
(1. School of Computer Science and Software, Nanjing University of Posts and Telecommunications, Nanjing 210023, China;
2. Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, Nanjing 210023, China;
3. School of Information Engineering, Nanjing University of Chinese Medicine, Nanjing 210023, China)

Abstract: Frequent items mining is one of the research hotspots of data mining. If the data set contains sensitive information, publishing mining results without processing risks privacy leakage. At present, there are heavy hitter estimation algorithms that meet ε - local differential privacy, but they still cannot meet the real-time and data availability requirements when processing big data. In response to these problems, we propose a new frequent itemset mining algorithm for local differential privacy protection—GFIM (group-based frequent items mining), which divides users into two disjoint and equal-sized parts randomly. The whole operation process is divided into two stages. In the first stage, GFIM digs out the candidate set C of frequent items based on the information submitted by all users. In the second stage, two groups of users trim themselves to $O(k)$ by setting redundant items and send them to the data collector. The final top-k frequent items will use the results of the above two stages. According to the two-stage idea, the times of traversal of data set can be reduced and the overall running speed can be accelerated. Theoretically, it has been proved that the algorithm satisfies ε -differential privacy, and experiments on multiple real data sets have also verified the performance of the method.

Key words: frequent items mining; local differential privacy; set-valued data; privacy protection; randomized response

0 引 言

频繁项目挖掘(frequent items mining)是当前数据挖掘研究的热点问题之一,其算法的核心是找出数据集中频繁出现的项。top-k 频繁项目挖掘[1]是挖掘出前 k 个频繁出现的项。该思想已广泛运用到现实生活中。例如,视频网站可以通过对所有用户观看的影片进行记录、分析,然后向用户推荐本周最受欢迎的前十个电影。在记录用户信息的过程中,如果不做任何隐

收稿日期:2020-08-11 **修回日期**:2020-12-15
基金项目:国家自然科学基金项目(61872197,61972209)
作者简介:朱美琪(1996-),女,研究生,研究方向为网络与信息安全、隐私保护;通讯作者:杨 庚(1961-),男,教授,博导,CCF 高级会员(E20-0006325S),研究方向为物联网安全、隐私保护、云计算与安全。

私保护措施,最后的推荐结果可能会有很高的准确性,但会严重侵犯用户的隐私。因此,在保证用户隐私性的同时要保证挖掘结果的准确性已经成为数据挖掘领域亟待解决的问题之一。

差分隐私[2]作为一个有效的隐私保护机制,现已广泛运用到浏览器、系统等应用中。例如 Apple 的 IOS 系统和 Google 的 Chrome 都运用了差分隐私的思想来保护用户的隐私。差分隐私又分为中心化差分隐私与本地化差分隐私,其中中心化差分隐私技术中,算法的隐私性通过临近数据集来定义,因此其要求一个可信的第三方数据收集者对数据分析结果进行隐私化处理,而对于本地化差分隐私技术而言,每个用户能够独立地对个体数据进行处理。目前已经有了许多关于本地化差分隐私的频繁项目挖掘算法,例如 Zhan Qin 提出的 LDPMiner[3-4]算法,该算法在保护用户隐私的同时,比较了当前已有的满足本地化差分隐私的保护算法,并对其进行优化,在一些真实数据集上有较好的表现。但该算法在面对大量用户的真实数据挖掘的问题时,面临了一些新的挑战:(1)因为用户量的增大,挖掘的时间复杂度也随之增大;(2)可以对用户进行分组挖掘来降低时间复杂度,但同时需要保持挖掘结果的可用性。所以,如何在保证时间复杂度降低的同时也能提高挖掘频繁项目的可用性就成了研究的关键所在。因此,该文通过基于本地化差分隐私保护,对用户进行分组挖掘的思想,设计了一种频繁项目挖掘的算法 GFIM(group-based frequent items mining)。

主要贡献如下:

(1)为了提高挖掘频繁项目的可用性,设计了一种基于分组思想的满足本地化差分隐私挖掘算法,并在理论上证明了该算法满足 ε -本地化差分隐私,多个真实数据集上的实验表明该算法的性能要优于 LDPMiner 算法。

(2)在 GFIM 算法中,采用将整个运行过程分成两个阶段、用户数据分为两组的策略,在保证高可用性的同时,减少了挖掘时计算的次数,从而加快了挖掘数据的时间,达到了优化算法时间复杂度的效果。

1　相关工作

在 Warner 首次对随机响应的方法进行研究[5]之后,研究者们开始探索其他扰动机制。Hsu 等[6]集中在基于随机投影和测度集中的技术来估计频繁项目。继这项工作,Bassily 等[7]提出了一种有效的协议,用于 SH(succinct histogram)估计与信息理论上的误差。为了处理隐私预算的问题,提出了 RAPPOR[8-9],SH 和 RAPPOR 的相关信息将在第二节进行介绍。此外,关于频繁项集挖掘的文献也很丰富。其中,有几篇与文中的研究方向有关。Bhaskar 等[10]提出了一种基于两阶段的方法,该方法使用截短的频率阈值来缩小频繁项集的候选列表。算法可以概括为如下两步:(1)计算出一个 m 值,从所有长度不大于 m 的候选项组成的集合 C 中挑选出 top-k 频繁项集;(2)对挑选出的 k 个项集的真实支持度添加拉普拉斯噪声后发布。该算法的问题在于第一步,因为其候选项集合 C 呈指数规模增大,即 $|C|=|I|^m$($|I|$ 表示项集域的大小),若遍历 C 中所有项集,则每个项集能分到的隐私预算将会很少,计算结果将会很不准确。TF 应用截断频率技术对 C 中项集进行筛选,只需遍历 C 中支持度大于 $f_k-\gamma$ 的项集(其中 f_k 表示第 k 频繁项集的支持度真实计数,γ 是调节参数)。在一般情况下,使用该技术可以对候选项集合 C 进行有效筛选,但是随着 k 值的增加,该筛选条件会被弱化,甚至失效。丁哲[11]为了从不确定的数据集中挖掘出基于期望支持度的前 k 个最频繁的频繁项集,并且保证挖掘结果满足差分隐私,提出了 FIMUDDP 算法。该算法利用差分隐私的指数机制和拉普拉斯机制确保从不确定数据中挖掘出的基于期望支持度的前 k 个最频繁的频繁项集和这些频繁项集的期望支持度满足差分隐私小的项,从而降低发布的频繁项集的支持度误差。

然而,所有上述机制都需要对数据集有全局了解,这使得它们不适用于本地化差分隐私。尽管上述已有工作并非所有针对中心化差分隐私的技术都适合于本地化差分隐私,但这些技术背后的思想仍有助于笔者设计符合 LDP 的算法。

2　理论基础

2.1　本地化差分隐私

近年来,本地化差异隐私作为一种区别于中心化差分隐私的隐私保护模式,引起了人们的广泛关注。它的隐私化处理发生在用户的本地设备中,用户对自己的个人数据进行加噪再发给数据收集者。数据收集者得到的是不准确的用户数据,这样能够避免不可信的第三方造成隐私的泄露。下面给出正式的定义:

定义 1(ε -本地化差分隐私):假设有一个随机算法 A,A 的所有输出构成集合 O,A 的所有取值构成集合 I,如果对于任意两条记录 $R\in I$ 和 $R'\in I$ 以及任意一个输出 $S\in O$,存在:

$$\Pr(A(R)\in S)<e^{\varepsilon}\times\Pr(A(R')\in S) \quad (1)$$

则称算法 A 满足 ε -本地化差分隐私,其中 ε 称为隐私参数或隐私预算。

定义 2(序列组合性)[12]:假设有随机算法 $A_1,A_2,\cdots,A_n$,其隐私参数分别为 $\varepsilon_1,\varepsilon_2,\cdots,\varepsilon_n$,当这些算法作用于同一数据集时,这些算法构成的组合算法

$A(A_1(I),A_2(I),\cdots,A_n(I))$，提供 $\sum_{i=1}^{n}\varepsilon_i$ -差分隐私保护。该性质表明，当多个随机算法作用于同一个数据集时，它们共同提供的隐私保护水平为各算法提供的隐私参数的总和。特殊地，当 $A_1 = A_2 = \cdots = A_n$ 时，$\varepsilon = \varepsilon_i \times n$，即当同一个算法多次作用于一个数据集时，总的隐私参数成倍增大，隐私保护水平成倍下降。

定义 3（并行组合性）：假设有随机算法 $A_1, A_2, \cdots, A_n$，其隐私参数分别为 $\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n$，当这些算法作用于不相交的数据集 $D_1, D_2, \cdots, D_n$ 时，这些算法构成的组合算法 $A(A_1(I),A_2(I),\cdots,A_n(I))$ 对这些数据集提供 $\max(\varepsilon_i)$ -差分隐私保护。

此性质表明，当多个随机算法作用的数据集两两之间互不相交时，它们对所有数据集提供的隐私保护水平取决于 $\max(\varepsilon_i)$。特殊地，当 $A_1 = A_2 = \cdots = A_n$ 时，$\varepsilon = \varepsilon_i$，即当同一个算法多次作用于不相交的数据集时，隐私保护水平不变。

2.2 频繁项目挖掘

频繁项目挖掘是数据挖掘的研究热点问题之一，旨在找出频繁出现在事务数据集中的 top-k 项目。具体描述如下：如果一个数据流 $\sigma = \{a_1, a_2, \cdots, a_m\}$，其中 m 为数据流的大小，$a_i \in \{1,2,\cdots,n\}$。可以定义每个元素出现的次数为 $F = (f_1, f_2, \cdots, f_n)$，其中 f_i 为第 i 个项目出现的次数。如果给定参数 k，求 top-k 频繁项目，那么可以对 F 进行分析和统计，然后输出的前 k 个项目就是 top-k 频繁项目挖掘的过程。

2.3 LDP 解决方案

2.3.1 随机响应解决方案

随机响应（randomized response，RR）向每个用户询问一个敏感问题，答案可以是“是”或“否”。例如“您是否感染了乙肝?”每个用户对此进行响应，用户的答案为是或否，但出于隐私性考虑，用户不会直接回答真实答案。假设其借助于一枚非均匀的硬币来给出答案，其正面向上的概率为 p，反面向上的概率为 $1-p$。抛出该硬币，若正面向上，则回答真实答案，反面向上，则回答相反的答案。已经证明当 $p = \frac{e^{\varepsilon}}{1+e^{\varepsilon}}$ 时，RR 满足 ε -差分隐私[4]。

2.3.2 RAPPOR 算法

RAPPOR 将 RR 扩展到更复杂的数据类型以及数据收集者收集更复杂的统计信息。令 n 为用户总数，每个用户 $u_i(1 \leqslant i \leqslant n)$ 在包含 d 个可能项的域中拥有一个项 v_i，数据收集者的目标是估计 d 中每个项的频率。在 RAPPOR 中，用户 u_i 长度为 d -bit 向量表示 v_i，该向量除了第 v_i 位数为 1 外，其余位皆为 0。然后，对于 d -bit 向量的每一位，用户 u_i 分别以概率 $p = \frac{e^{\frac{\varepsilon}{2}}}{1+e^{\frac{\varepsilon}{2}}}$ [8] 独立应用 RR。数据收集者从 n 个用户中接收 d -bit 向量后，通过独立应用 RR 为 d 个项目的每一个计算无偏频率估计。

2.3.3 Succinct Histogram 算法

Succinct Histogram（SH）算法[13]中，每个用户在 d 个可能项目中仅拥有一个项目，数据收集者来计算每个项目的频率。在 SH 算法中，数据收集者仅发布具有较高频率的项目（高于给定阈值），对于其他项目，SH 将其视为 0。SH 不是和 RAPPOR 一样报告 d-bit 向量，而是每个用户随机报告一位（$d \leqslant n$）[13]，n 为用户总数。由于只报告一位，可直接运用当 $p = \frac{e^{\varepsilon}}{1+e^{\varepsilon}}$ 的 RR。当 $d > n$ 时，SH 在数据预处理步骤中对数据应用随机投影[14]，将其维数从 d 减少至 $m = O(n)$。具体的实现方法是 SH 生成一个 $d \times m$ 的矩阵 $\boldsymbol{\varphi}$，每个元素从两个可能的值：$\frac{1}{\sqrt{m}}$ 和 $-\frac{1}{\sqrt{m}}$ 中独立且均匀地随机选择。然后每个用户将 d -bit 向量乘以 $\boldsymbol{\varphi}$，得到长度为 m 的向量。最后再运用上文提到的当 $d \leqslant n$ 的方法。

3 GFIM 算法

本节包括 GHHE 算法的概述及具体实现细节。GFIM 分为两个阶段，并将隐私预算也分为两个部分用来完成这两个阶段，整个过程满足 ε -LDP。第一阶段中，每个用户拥有 l 项，并在 ε -DP 下向数据收集者报告数据，数据收集者根据用户提交的信息挖掘出一个大小为 $k_{\max} = O(k)$ 的候选集 C。第二个阶段，首先是对用户进行分组，第一组根据挖掘出的候选项集 C，把自身拥有的却不在 C 中的项目设置为冗余项，然后把挖掘出的项集 E 报告给数据收集者；第二组是根据第一组挖掘出的项集 E 进行二次挖掘，最终得到 top-k 频繁项目。

3.1 GFIM 算法概述

算法 1 总结了 GFIM 算法完整的框架。其中 1 ~ 2 行属于预处理部分，3 属于 GFIM 算法的第一阶段，4 ~ 6是 GFIM 算法的第二阶段。

算法 1：GFIM 算法。

Input：事务数据集 D，k，隐私预算 ε；

Output：top-k 频繁项目。

1. 将隐私预算 ε 分为 ε_1 和 ε_2；

2. 计算真实的 top-k 频繁项集；

3. 使用 ε_1 的隐私预算来获取候选项集 C；

4. 将总用户数随机分成两组；

5. 第一组数据使用 ε_2 的隐私，以 C 为候选项集预算来获取项集 E；

6. 第二组数据同样使用 ε_2 的隐私预算,以 E 为候选项集来获取 top-k 频繁项目。

3.2　GFIM 算法中阶段一分析设计

阶段一是找出频繁项的候选集的过程。上文提到的 RAPPOR 和 SH 是两个经典的频繁项目挖掘算法,但它们不能直接运用于阶段一的场景,因为传统的 RAPPOR 和 SH 算法均要求每个用户的输入为一个项目,而在文中的场景中用户输入的是一个经过处理后的大小为 l 的项集。一个直观的解决方案是调用 l 次 RAPPOR 或 SH,再把每次调用得到的估计频率累加得到最终的估计频率。这个想法是可行的,但也是低效的。对这种想法的一种改进方案是从每个用户的项集 S_i 中随机选取一个项作为输入,然后采用 RAPPOR 或 SH 算法。已有的工作[2]证明了该想法的合理性,证明了其优于前面所说的调用 l 次 RAPPOR 或 SH。但是,因为每个用户只向数据收集者发送一个随机的项而不是所有 l 个项,这样直接的随机抽取会导致有偏频率估计。为了达到无偏估计,需要将估计的频率乘以 l。根据上文描述的 RAPPOR 和 SH 算法的对比分析,出于节约通信带宽和提高计算效率的考虑,这里将以 SH 算法为基础对其进行改造分析,改造后的算法称为抽样 SH 算法,抽样 SH 算法将作为阶段一和阶段二的基本算法。抽样 SH 算法与传统 SH 算法大体一致。同时,文中采用了一种正交矩阵生成的方法,假设 $d \ll n$,参考文献[9]证明了在 $d \ll n$ 的情况下采用正交矩阵取代完全随机矩阵能够提高 SH 的准确性。

算法 2:抽样 SH 正交矩阵的生成。

Input:项的取值集合大小 d;

Output:正交矩阵 $\boldsymbol{\varphi}$。

1. 计算 $m = 2^{\lceil \log_2 d \rceil}$
2. $S = \{[1, -1],[1,1]\}$
3. while $|S| < m$ do
4. $\quad S' = \boldsymbol{\varphi}$
5. $\quad$ for $v \in S$ do
6. $\quad\quad S' = S' \cup \{v \| v, v \| (-v)\}$
7. $\quad$ end for
8. $\quad\quad S \leftarrow S'$
9. end while
10. $N = S[1:m]$
11. $\boldsymbol{\varphi} = N^{\mathrm{T}}$
12. return $\boldsymbol{\varphi}$

算法 3:抽样 SH LR(local randomizer)(用户端部分)。

Input:d-bit 向量 $X \in \{\frac{-1}{\sqrt{m}}, \frac{1}{m}\}$,隐私参数 ε,用户 u_i 的项集 S_i;

Output:干扰后的向量 z_i。

1. 从项集 S_i 中随机选取一个项 i;
2. 计算 $c_\varepsilon = \frac{e_\varepsilon + 1}{e_\varepsilon - 1}$;
3. if $i_i = \perp$ then
4. 随机选取 $z_i \in \{c_\varepsilon \sqrt{m}, -c_\varepsilon \sqrt{m}\}$;
5. else
6. $\quad$ 生成一个标准的基向量 $e_{i_i} \in \{0,1\}$;
7. $\quad$ 计算 $X_{i_i} = X \top e_{i_i}$;

概率 $p = \begin{cases} \frac{e_\varepsilon}{e_\varepsilon + 1}, z_i = c_\varepsilon m x_{i_i} \\ \frac{1}{e_\varepsilon + 1}, z_i = -c_\varepsilon m x_{i_i} \end{cases}$

8. end if
9. return z_i

算法 2 与传统的 SH 算法的不同主要体现在 1、3、4 行。当随机选取的项为冗余项时,用户端从 $\{c_\varepsilon \sqrt{m}, -c_\varepsilon \sqrt{m}\}$ 随机选取一个发给数据收集者,这样做的好处是在用户数量很大的情况下,由于冗余项引入的噪声能够达到正负抵消的效果。

3.3　GFIM 算法中阶段二分析设计

阶段二的场景与阶段一有所不同,不同之处有二:一是对总用户数进行了随机分组;二是每一组进行挖掘时的候选项集是不同的。针对这两点不同,需要设计出相应的解决方案。阶段二中依旧采用抽样 SH 作为算法的基础。显然,当候选项集合缩小时,SH 中的随机矩阵也会相应缩小。这一方面减少了噪音,另一方面也降低了运算开销,最终会提高估计频率的准确性。针对第二个不同点,在用户端上的 LR(local randomizer)做了以下调整:(1)第一组 LR 收到候选集 C 后,先求出用户原本的项集 S_i 与候选集 C 的交集 T_i;(2)如果交集 N 的大小小于 $k_{\max}$,补充若干个冗余项使其大小变为 $k_{\max}$,得到新项集 N_i;(3)从 N_i 中随机选取一个项 i_i 应用随机响应技术。第二组数据也要经历这样的过程,区别是第二组用户收到的不是候选项集 C,而是第一组用户响应后的数据。这样处理的好处是能有效增大候选集中的项被抽中的概率,进而提高候选集中的项的估计频率准确性。一个例子能很好解释其中的原因:假设用户 u_i 的原项集 S_i 包含候选集 C,S_i 的大小为 $l = 60$,C 的大小为 $k_{\max} = 30$;在未经过以上处理前,从 S_i 随机抽取一个项,该项属于候选集的概率为 1/2,但经过(1)~(3)步处理后,被抽中的项属于候选集的概率为 1。可见,这样的处理是很有效的。并且,如果不对长度小于的交集 T_i 填充冗余项,T_i 的大小直接暴露给用户,会给攻击者提供额外的信息,存在着隐私泄露的风险。阶段二中的抽样 SH 只能计算候选集中各项的估计频率。然而,真实的 top-k 项并不一定恰好都落到候选集中,为了解决这个问题,需要把两个阶段的结果综合利用起来。这里按以

下公式得到最终各项的估计频率：

$$\hat{f}=\begin{cases}\hat{f}_1, i_i \notin C\\ [\hat{f}_1+\dfrac{(l-1)\hat{f}_2}{l}], i_i \in C\end{cases} \tag{2}$$

需要说明的是，阶段二中处理的两组用户数据并不相交，为了保证两组数据与总数据满足同分布，必须保证划分数据时是随机划分的。

算法 4：GFIM LR(用户端)。

Input：$k_{\max}$ -bit 向量 $X \in \{\frac{-1}{\sqrt{m}}, \frac{1}{m}\}$，隐私参数 ε，用户 u_i 的项集 S_i 以及候选项集 C；

Output：干扰后的向量 z_i。

1. 求交集 $T_i = S_i \cap C$；

2. if $|T_i| < k_{\max}$ then

3. 往 T_i 中加入若干个冗余项得到新的用户项集 $|N_i| = k_{\max}$；

4. end if

5. 从项集 N_i 中随机选取一个项 i_i；

6. 计算 $c_\varepsilon = \dfrac{e_\varepsilon + 1}{e_\varepsilon - 1}$；

7. if $i_i = \perp$ then

8. 随机选取 $z_i \in \{c_\varepsilon\sqrt{m}, -c_\varepsilon\sqrt{m}\}$；

9. else

10. 生成一个标准的基向量 $e_{i_i} \in \{0,1\}$；

11. 计算 $X_{i_i} = X \top e_{i_i}$；

概率 $p = \begin{cases}\dfrac{e_\varepsilon}{e_\varepsilon+1}, z_i = c_\varepsilon m x_{i_i}\\ \dfrac{1}{e_\varepsilon+1}, z_i = -c_\varepsilon m x_{i_i}\end{cases}$

12. end if

13. return z_i

4 GFIM 算法隐私保护性能分析

本地化差分隐私蕴含着两个极其重要的性质：序列组合性和并行组合性[15]。利用这两个性质，可以很容易地证明某种算法是否满足本地化差分隐私。

GFIM 算法将隐私预算 ε 分为两部分，分配给算法的两个主要步骤：阶段一生成候选集 ε_1、阶段二分组计算频繁项目 ε_2。其中，选择 $\varepsilon_1 = 0.6 \times \varepsilon$，$\varepsilon_2 = 0.4 \times \varepsilon$。

定理：GFIM 算法满足 ε -本地化差分隐私。

证明：

(1)阶段一安全性证明。

假设 S_1 和 S_2 为任意两个用户项集，有 $|S_1| = |S_2| = l$。A 表示抽样 SH 算法，z_i 表示 A 的任意可能的输出，$z_i \in \{c_\varepsilon\sqrt{m}, -c_\varepsilon\sqrt{m}\}$。要证 SH 满足 ε_1-本地化差分隐私，即证：

$$\frac{\Pr[A(S_1)=z_i]}{\Pr[A(S_2)=z_i]} \leqslant e^{\varepsilon_1} \tag{3}$$

使 $\Pr(i_i' | S_i)$ 表示输入为 S_i 时，抽样 SH 算法第一步的输出为 i_i' 的条件概率。可知对任意 i_i'，$\Pr(i_i' | S_i) = 1/L$。使 $\Pr(z_i | i_i' \cap S_i)$ 表示输入为 S_i，随机选取到 i_i'，最终输出为 z_i 的条件概率，有：

$$\frac{1}{e^{\varepsilon_1}+1} \leqslant \Pr(z_i | i_i' \cap S_i) \leqslant \frac{e^{\varepsilon_1}}{e^{\varepsilon_1}+1} \tag{4}$$

进而有：

$$\frac{\Pr[A(S_1)=z_i]}{\Pr[A(S_2)=z_i]} = \frac{\Pr(z_i | i_i \cap S_1) \times \Pr(i_i \cap S_1)}{\Pr(z_i | i_i \cap S_2) \times \Pr(i_i \cap S_2)} = \frac{\Pr(z_i | i_i \cap S_1)}{\Pr(z_i | i_i \cap S_2)} \leqslant e^{\varepsilon_1} \tag{5}$$

所以抽样 SH 算法满足 ε_1-本地化差分隐私，即阶段一满足 ε_1-本地化差分隐私。阶段一生成一个大小为 $k_{\max}$ 的候选集。候选项目集大小的选取至关重要。候选项目集太小的话，真实的频繁项可能会没有落在候选集内，导致估计的误差会增大；候选集过大，$k_{\max}$ 有可能会大于 l，同样会降低估计频率的准确性。

(2)阶段二安全性证明。

假设 B 为阶段二中第一组用户对数据的处理算法。B 的输入是隐私参数 ε_2、用户 u_i 的项集 S_i 和候选集 C。B 首先对 S_i 进行修剪得到 N_i。之后 B 对 N_i 采用抽样 SH 算法，该处理过程用 C 表示。由阶段一的证明可知，C 满足本地化差分隐私。假设 S_1，S_2 为任意两个未修剪前的用户项集，o 表示 B 的任意输出。要证第一组用户对数据的处理算法满足 ε_2-本地化差分隐私，即证：

$$\frac{\Pr[B(S_1,\varepsilon_2,C)=o]}{\Pr[B(S_2,\varepsilon_2,C)=o]} \leqslant e^{\varepsilon_2} \tag{6}$$

因为修剪过程是确定的，所以有：

$$\Pr[B(S_1,\varepsilon_2,C)=o] = \Pr[C(N_1,\varepsilon_2)=o] \tag{7}$$

又 C 满足本地化差分隐私，有：

$$\frac{\Pr[B(S_1,\varepsilon_2,C)=o]}{\Pr[B(S_2,\varepsilon_2,C)=o]} = \frac{\Pr[C(N_1,\varepsilon_2)=o]}{\Pr[C(N_2,\varepsilon_2)=o]} \leqslant e^{\varepsilon_2} \tag{8}$$

由上述可知第一组用户对数据的处理算法满足 ε_2-本地化差分隐私。因为第二组用户对数据的处理与第一组原理相同，所以第二组对数据的处理也满足 ε_2-本地化差分隐私。又因为，这两组数据不相交，根据差分隐私的并行组合原理可知，整个阶段二满足 ε_2 -本地化差分隐私。

由于阶段一，阶段二分别满足 ε_1 和 ε_2-本地化差分隐私，由差分隐私的序列组合性知，$\varepsilon = \varepsilon_1 + \varepsilon_2$。即，GFIM 满足 ε -本地化差分隐私，证毕。

5　实　验

5.1　实验设置

实验环境为 Inter(R) Core(TM) i5 - 3230M CPU2.60 GHz,4 GB 内存,Windows10 操作系统。算法均使用 Python 来实现。在三个真实数据集上进行了测试,这些数据集可从 Spmf 上下载。分别是 USCensus(US Census 1990 dataset),Mushroom(UCI mushrooms dataset)和 Connect(UCIconnect - 4 dataset)。表 1 给出了每个数据集中的几种特征,包括事务数量、项集域大小以及平均事务长度。通过实验将 LDPMiner 算法与文中提出的 GFIM 算法进行比较,验证该算法的性能。

表 1　三种数据集信息描述

Dataset	Size	$\|I\|$	Avg$\|t\|$
USCensus	1 000 000	396	68.0
Mushroom	8 416	119	23.0
Connect	67 557	129	43.0

5.2　实验结果与分析

为了评估两种算法的性能,采用了两个广泛使用的评价标准,即相对误差(RE)[16]和折扣累计增益(DCG),定义如下:

(1)相对误差(RE):用来测量相对于频繁项目实际频率的估计频率的误差。具体来说,令 $V=\{v_1,v_2,\cdots,v_k\}$ 是前 k 个频繁项目的集合。

$$\mathrm{RE}=\mathrm{Median}_{v_i\in V}\frac{|f_{\mathrm{estimated}}(v_i)-f_{\mathrm{actual}}(v_i)|}{f_{\mathrm{actual}}(v_i)} \tag{9}$$

(2)折扣累计增益(DCG):DCG 测量数据收集者计算的频繁项目的质量[17]。频繁项目的频率排名列表中的项目 v_i 的相关性或增益是通过相关性计算公式得出的:

$$\mathrm{rel}_{v_i}=\log_2[\,|d-|\mathrm{rank}_{\mathrm{actual}}(v_i)-\mathrm{rank}_{\mathrm{estimated}}(v_i)||\,] \tag{10}$$

可以直观地发现,v_i 的估计频率排名与真实频率排名越接近,相关性就越大。给定一个真实的前 k 个频繁项目 $V=\{v_1,v_2,\cdots,v_k\}$,估计频率的排名的 DCG 计算为:

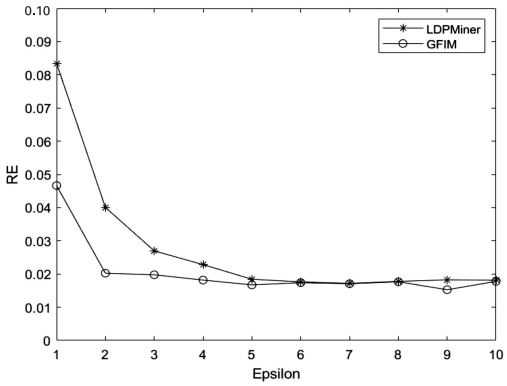
$$\mathrm{DCG}_k=\mathrm{rel}_{v_1}+\sum_{i=2}^{k}\frac{\mathrm{rel}_{v_i}}{\log_2(i)} \tag{11}$$

折扣因子 $\log_2(i)$ 可以赋予较高排名的项目更高的权重。通过将估计的排名列表与理想的 DCG(IDCG,与实际的频繁项目排名完全一致)进行比较来进行归一化。

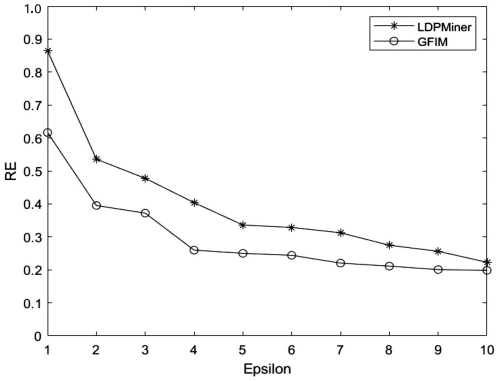
$$\mathrm{NDCG}_k=\frac{\mathrm{DCG}_k}{\mathrm{IDCG}_k} \tag{12}$$

很容易从公式得出,NDCG 的结果是在 0 到 1 之间,能够比较不同 k 值上计算的频繁项目的质量。

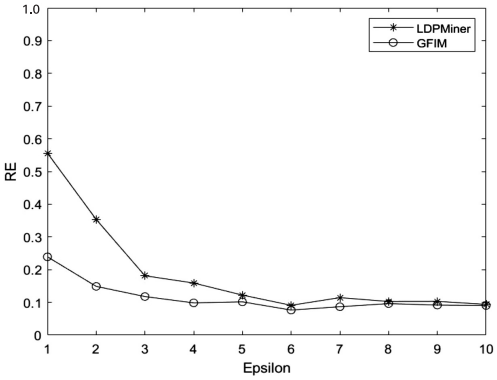
实验分为两个部分:(1)设定 $k=10$,查看 GFIM 算法在不同的隐私预算下与 LDPMiner 算法的 RE 和 NDCG 性能对比。USCensus,Mushroom,Connect 实验结果按顺序如图 1(a)、(b)、(c)和图 2 的(a)、(b)、(c)所示。



(a)USCensus 数据集



(b)Mushroom 数据集

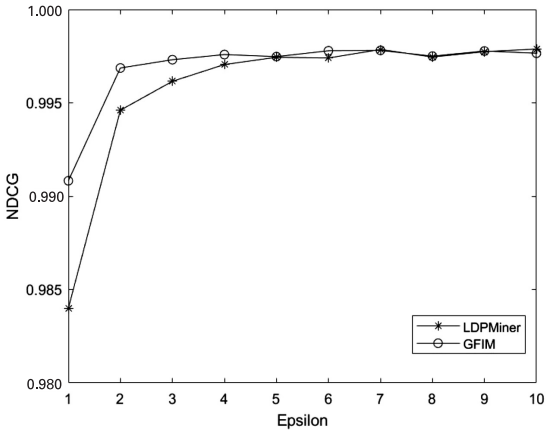


(c)Connect 数据集

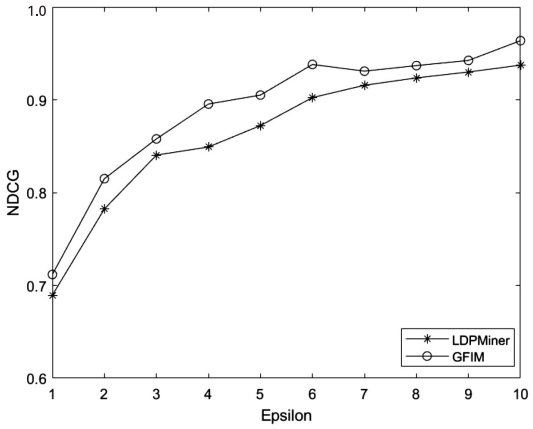
图 1　三个数据集随 ε 变化时 RE 的变化情况

从图 1 可以看出,随着隐私预算 ε 的增大,相对误差 RE 在三个数据集上都是呈总体下降趋势,这符合差分隐私思想中隐私预算越大相对误差越小的性质。同时可以看出,隐私预算参数 ε 从 0 变化到 10 的过程中,改进后的 GFIM 算法在相对误差上的性能优于

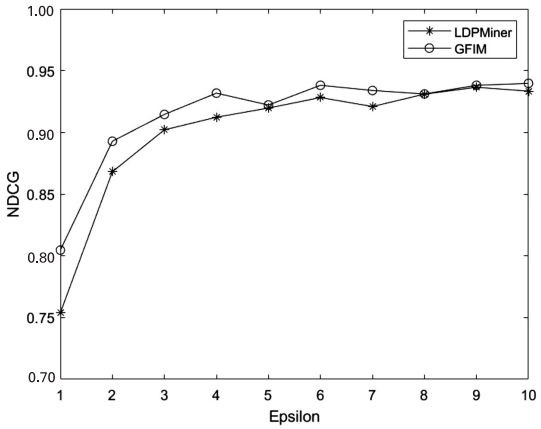
LDPMiner 算法。



(a) USCensus 数据集



(b) Mushroom 数据集

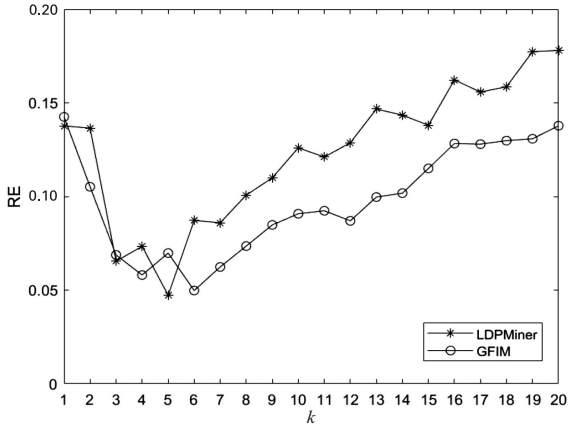


(c) Connect 数据集

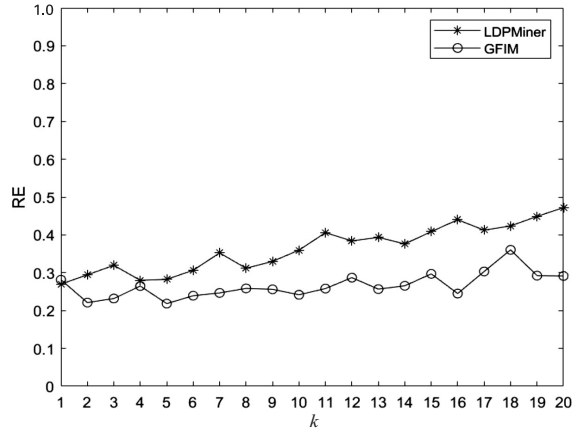
图 2 三个数据集随 ε 变化时 NDCG 的变化情况

从图中可以看出,随着隐私预算 ε 的增大,计算频繁项目的质量在三个数据集上都是呈总体上升的趋势,这与 NDCG 标准定义的理论结果相符。同时,隐私预算参数 ε 从 0 变化到 10 的过程中,改进后的 GFIM 算法在频繁结果质量上优于 LDPMiner 算法。

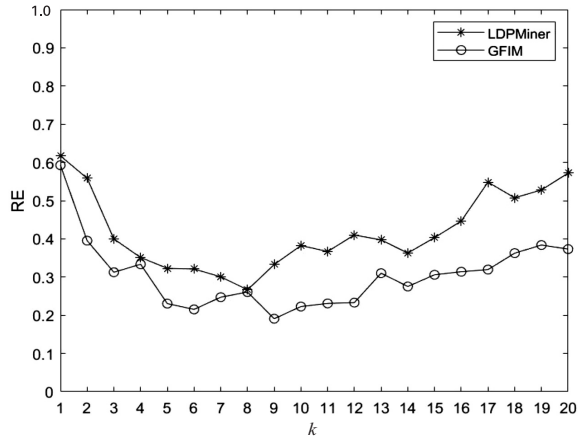
(2)设定隐私预算 ε 为固定值,观察 GFIM 算法在不同 k 值下的 RE 和 NDCG 与 LDPMiner 算法的对比。USCensus,Mushroom,Connect 实验结果如图 3(a)、(b)、(c)和图 4(a)、(b)、(c)所示。



(a) USCensus 数据集



(b) Mushroom 数据集

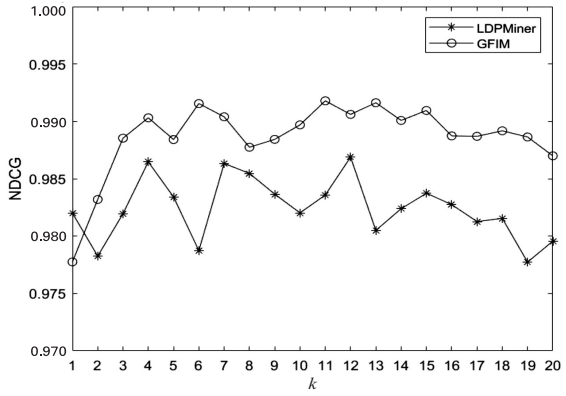


(c) Connect 数据集

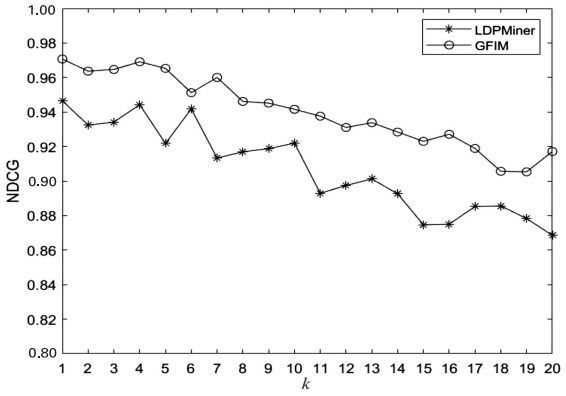
图 3 三个数据集随 k 变化时 RE 的变化情况

图 3 中,根据不同的数据集大小设置了不同的隐私预算值。其中 Mushroom 的隐私预算 ε 设置为 3,Connect 和 USCensus 数据集隐私预算 ε 设置为 1。根据曲线图可以看出,改进后的 GFIM 算法在相对误差上的性能优于 LDPMiner 算法。

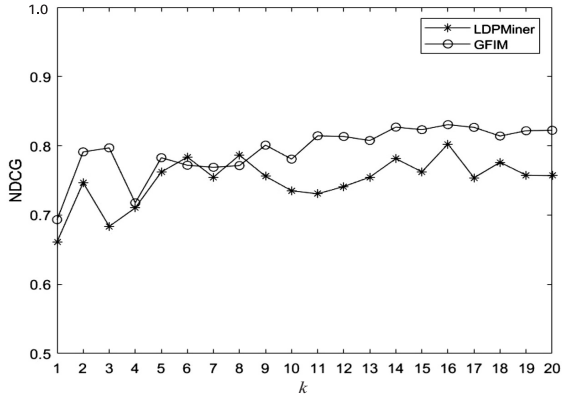
从图 4 中可以明显看出,改进后的 GFIM 算法计算出的频繁项目的质量要高于 LDPMiner 算法计算出的频繁项目的质量。



(a) USCensus 数据集



(b) Mushroom 数据集



(c) Connect 数据集

图 4　三个数据集随 k 变化时 NDCG 的变化情况

6　结束语

该文研究出一个既满足本地化差分隐私又有较高可用性的 GFIM 算法。整个加噪和挖掘频繁项目的过程划分为两个阶段。在第一个阶段,GFIM 实现对整体用户的隐私保护和频繁项候选集的筛选;在第二个阶段,GFIM 把用户划分为随机等大小的两组用户,把候选集发送给第一组用户,让用户对自身的项集重新进行打包和加噪,挖掘候选集内各项的频率,再把结果当成候选项集发送给第二组用户。最后,GFIM 综合两个阶段得到最后频繁项目和对应频率。为了验证算法 GFIM 的可行性和对已有方案的改善,选取了 LDPMiner 算法进行对比,并在三个真实数据集进行多次实验。结果表明,GFIM 能够较为准确地挖掘出频繁项目,并且在 RE 和 NDCG 这两个指标上的性能表现均优于 LDPMiner 算法。同时,在实验中发现,针对不同的数据集和不同的 k 值, $k_{\max}$ 有对应的最适合的大小。$k_{\max}$ 的大小会对实验结果有重要的影响,一个合适的 $k_{\max}$ 将极大地提高实验结果的准确性。另外,用户的分组问题也是个可以讨论的研究方向,文中方法是将用户随机等分成两组,可以尝试在用户分组上面再进行研究,观察是否能对频繁项目挖掘的性能有进一步的提高。

参考文献:

[1] 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[J]. 计算机研究与发展,2014,51(1):104-114.

[2] DWORK C. Differential privacy:a survey of results[C]// International conference on theory and applications of models of computation. Berlin,Heidelberg:Springer,2008:1-19.

[3] QIN Z,YANG Y,YU T,et al. Heavy hitter estimation over set-valued data with local differential privacy[C]//Proceedings of the ACM conference on computer and communications security (CCS). Vienna, Austria: ACM, 2016: 192-203.

[4] KASIVISWANATHAN S P, LEE H K, NISSIM K,et al. What can we learn privately? [J]. SIAM Journal on Computing,2008,40(3):793-826.

[5] WARNER S L. Randomized response:a survey technique for eliminating evasive answer bias[J]. Journal of the American Statistical Association,1965,60(309):63-69.

[6] HSU J,KHANNA S,ROTH A. Distributed private heavy hitters[M]//Automata, languages, and programming. Berlin, Heidelberg:Springer,2012.

[7] BASSILY R,SMITH A. Local,private,effcient protocols for succinct histograms[C]//Forty-seventh annual ACM symposium on theory of computing (STOC ′15). Portland,Oregon,USA:ACM,2008:127-135.

[8] LIN C T,TSAI C Y,KAO C K. Lower power data transport protection for internet of things (IoT)[C]//2017 IEEE conference on dependable and secure computing. Taipei:IEEE, 2017:468-470.

[9] FANTI G,PIHUR V,ERLINGSSON L. Building a RAPPOR with the unknown:privacy-preserving learning of associations and data dictionaries[J]. Proceedings on Privacy Enhancing Technologies,2016,201(3):41-61.

[10] BHASKAR R, LAXMAN S, SMITH A, et al. Discovering frequent patterns in sensitive data[C]//Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining. Washington,DC,USA:ACM,

(下转第 168 页)