

# LDP机制下的快速FIM方法

王家礼

2020 年 3 月 26 日

摘要

this is abstract

## 1 背景简介

2020年2月26日，差分隐私技术被全球知名科技评论期刊《麻省理工学院技术评论》评为“全球十大突破性技术”，并指出该技术将被美国政府用于进行3.3 亿美国居民的2020年人口普查，同时还要对他们的身份数据进行保密。

差分隐私是一种数学技术，它能够在给数据添加噪声的同时，量化计算隐私提升的程度，从而使得增加“噪音”的过程变得更加严谨。苹果和Facebook已经使用这种方法来收集聚合数据，而不需要识别特定的用户。

差分隐私技术因其独特的优势，被学术界及工业界广泛的研究，谷歌、微软、苹果等公司使用该技术在保护用户隐私的同时，手机聚合数据，从而提升服务质量。

然而，传统差分隐私技术在收集聚合数据时，用户先将原始数据提交给第三方，由第三方完成对数据的加噪处理之后，将数据发布。整个过程中，需要认定第三方是可信的。

现实生活中，找到一个可信的第三方是困难的。所以，本地化差分隐私被提出。其一方面继承了传统差分隐私技术的对敏感数据量化处理的优势，另一方面进一步细化隐私保证，将加噪过程由第三方转移至用户端，由用户独立完成对个人敏感数据的加噪处理。

## 2 本文工作

传统的FIM任务是不考虑用户隐私的, Apriori[?]和FP-growth[?]等方法对用户原始数据挖掘且不加任何限制, 极大的损害了用户利益。

为了充分考虑用户的个人隐私, 本文使用本地化差分隐私机制聚合用户信息, 并结合FP-growth方法[?]的快速与高效性, 提出并实现了一种基于本地化差分隐私的快速 $top-k$ 频繁项集挖掘方法。总体框架见算法1

---

**Algorithm 1** 总体框架

---

**Input:**

$n$ 位用户的交易数据集  $DB = \{t_1, t_2, \dots, t_n\}$ ;

正整数 $k$ ;隐私预算 $\epsilon$ ;

**Output:**  $top-k$ 的频繁项集

1: 将数据集 $DB$ 均分为两个组 $DB_1$ 和 $DB_2$ ;

2:  $\tilde{S}_k = SVIM(DB_1, \epsilon, k)$ ; //SVIM[?]方法收集 $top-k$ 的频繁1- $itemset$ 集合 $\tilde{S}_k$

3:  $I\tilde{S}_k = minefp(DB_2, \epsilon, k, \tilde{S}_k)$ ; //本文方案minefp, 详见2.1节

4: **return**  $\tilde{S}_k \cup I\tilde{S}_k$

---

本文在将FP-growth模式挖掘方法用于LDP场景时, 所需要解决的问题主要有:

- 1、如何获得频繁1- $itemset$ ;
- 2、如何构建频繁模式树。

对于问题1, 其等同于LDP的PSFO任务, 本文使用文献[?]所提的SVIM方法, 收集 $top-k$ 的频繁1- $itemset$ 集合 $\tilde{S}_k = \{x^1 : \tilde{\theta}^1, x^2 : \tilde{\theta}^2, \dots, x^k : \tilde{\theta}^k\}$ , , 更多算法细节参见文献;

对于问题2, 通过分析, 我们发现模式树的层次建立能够更好的契合LDP场景, 提出minefp方案收集 $top-k$ 的 $\alpha-itemset(\alpha > 1)$ 集合 $I\tilde{S}_k$ , 详见2.1节。

根据分析, 整个机制是满足 $\epsilon-LDP$ 的, 不会泄露用户隐私。

### 2.1 minefp方案介绍

根据上述对总体框架的介绍可知, 本文主要工作是提出了minefp方案用以收集 $top-k$ 的 $\alpha-itemset(\alpha > 1)$ 集合 $I\tilde{S}_k$ 。

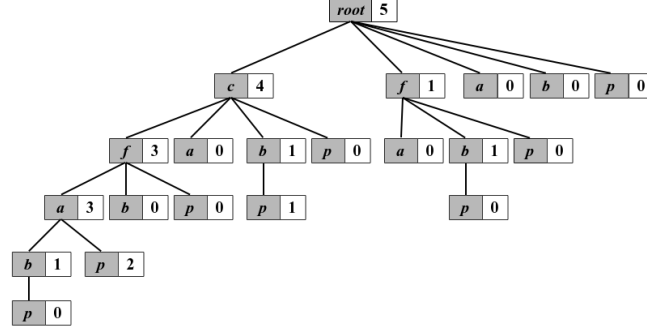


图 1: 模式树

算法2描述了minefp方案的总体框架，首先对整个用户数据进行预处理并分配隐私预算（步骤1、2）；然后为了建立频繁模式树，先聚合用户记录长度估计出树的最大层次（步骤3），最后以最大层次进行频繁模式树的层次建立（步骤4）；在建立模式树之后，挖掘出有效的频繁项集并且对结果进行一定的优化处理后发布（步骤5、6）。

## 2.2 最大树层次

由于本文是在LDP场景下层次建立频繁模式树，树的最大层次与隐私预算的分配直接相关。如图1所示模式树，最大层次为5。由于用户的记录长度也是敏感信息，其长度不固定且具有随机性。本文使用FO协议，聚合用户的记录长度，通过设置临界值  $T = 3.0 \cdot \frac{\sqrt{n}}{\epsilon}$ ，从而估计出有效的最大层次。

$$\varphi(X) = \prod_{x \in X} \mu(x), \mu(x) = \frac{0.8 \times \tilde{\theta}(x)}{\max_{x \in S'} \tilde{\theta}(x)} \quad (9)$$

其过程与文献[?]中对  $L$  值的确定相似，本文对其中一些参数值做了修改，具体见算法??。

## 2.3 层次建树

---

**Algorithm 2** minefp

---

**Input:**

数据集 $DB_2$ ;  
 隐私预算 $\epsilon$ ;  
 正整数 $k$ ;  
 频繁1-itemset集合 $\tilde{S}_k$ ;

**Output:**  $top-k$ 的挖掘结果 $I\tilde{S}_k$ 

- 1: 预处理数据集 $DB_2$ , 删除用户记录中的非频繁项目并重新排序;  
 //预处理数据集,  $\tilde{S}_k$ 中的项目为频繁项目, 并以其中项目顺序进行排序
  - 2: 将 $DB_2$ 按比例分为互斥的两组 $transaction_1$ 和 $transaction_2$ ;
  - 3:  $iterNum = FO\_MaxIter(transaction_1, \epsilon)$ ;  
 //FO协议聚合 $transaction_1$ 中用户记录长度的信息,
  - 4:  $tree = build\_fpTree(transaction_2, \tilde{S}_k, \epsilon, iterNum)$ ;  
 //层次构建频繁模式树,
  - 5:  $\tilde{I}\tilde{S} = FPgrowth(tree)$ ;  
 //FP-growth方法挖掘模式树 $tree$
  - 6: 优化 $\tilde{I}\tilde{S}$ 并选择 $top-k$ 记为 $I\tilde{S}_k$   
 //后处理步骤, 不消耗隐私预算
  - 7: **return**  $I\tilde{S}_k$
- 

### 3 结果优化

### 4 实验

---

**Algorithm 3** FO\_MaxIter

---

**Input:** 数据集 $transaction_1$ ;隐私预算 $\epsilon_1$ ;**Output:** 最大迭代次数MaxIter

- 1: 初始化临界值 $T = 3.0 \cdot \frac{\sqrt{n_1}}{\epsilon}$ ; //  $n_1$ 为 $transaction_1$ 的用户总数
  - 2: FO协议聚合用户记录长度, 并且将长度人数不大于 $T$ 的视作无效估计, 置为0;
  - 3: 利用公式(8)在 $\gamma = 0.8$ 时的整数值 $M$ ;
  - 4: **return** 最大迭代次数 $M$
- 

---

**Algorithm 4** 建树

---

**Input:**数据集 $DB_2$ ;隐私预算 $\epsilon$ ;正整数 $k$ ;频繁1-itemset集合 $\tilde{S}_k$ ;**Output:**  $I\tilde{S}_k$ 

- 1: 初始化树根结点 $root$ 为空;
  - 2: 初始化候选值域 $I'_1$ 为 $S'$ 中项目, 作为第一层次的LDP聚合值域;
  - 3: 将 $DB_2$ 均分为 $M$ 个人数相同的子组  
 $G_1, G_2, \dots, G_M$ ;
  - 4: 初始化FO协议标准差为 $standard\_error$  ;
  - 5: **for** 层次 $j = 1$  to  $M$  **do**
  - 6: FO协议聚合 $estimation = Aggregate(I'_j, j, G_j)$ ;  
//用户使用前 $level$ 个项目参与, 若无记录, 则以无效值 $\perp$ 参与。
  - 7: 删除 $estimation$ 中估计值不大于 $standard\_error$ 的结果;  
//无效值 $\perp$ 不用于建树。
  - 8: 以 $estimation$ 聚合结果更新第 $j$ 层树 $root$ 节点;
  - 9: 根据第 $j$ 层节点, 生成所有可能的孩子节点, 作为下一层初始值域 $I_{j+1}$ ;
  - 10: **end for**
  - 11: **return**  $root$
-