

# LDP机制下的快速FIM方法

王家礼

2020 年 4 月 1 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。

传统的FP-growth算法通过两次直接扫描数据集，将用户记录信息压缩存储在频繁模式树中，用以挖掘频繁模式。如图1为5位用户的交易记录，图2为所建频繁模式树。

TID	Transaction
01	<i>a, f, c, g, p</i>
02	<i>a, b, c, f, l, o</i>
03	<i>b, f, h, o</i>
04	<i>b, c, p</i>
05	<i>f, a, c, l, p, n</i>

图 1: 交易数据集

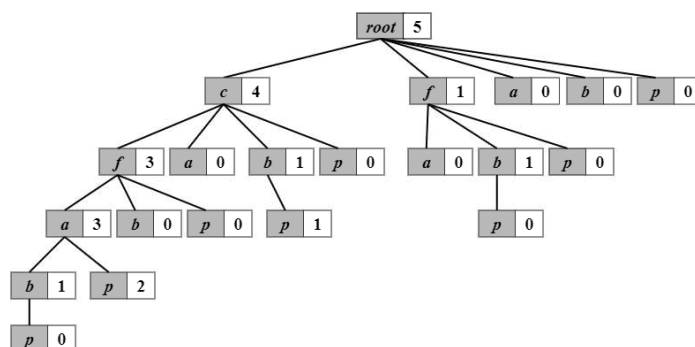


图 2: 模式树

## 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

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：

**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**  $\tilde{S}_k \cup I\tilde{S}_k$

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  $\tilde{S}_k = \{x^1 : \tilde{\theta}(x^1), x^2 : \tilde{\theta}(x^2), \dots, x^k : \tilde{\theta}(x^k)\}$ , 其中  $\tilde{\theta}(x^i)$  是项  $x^i$  的估计频数,  $1 \leq i \leq k$ , 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  $I\tilde{S}_k$ 。

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**

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2所示模式树, 最大层次为5。用户的记录长度是敏

**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;  
 //预处理步骤
- 2: 分配隐私预算  $\epsilon_1 + \epsilon_2 = \epsilon$ ;
- 3:  $iterNum = FO\_MaxIter(transaction_1, \epsilon_1)$ ;  
 //FO协议聚合transaction\_1中用户记录长度的信息,
- 4:  $tree = build\_fpTree(transaction_2, \tilde{S}_k, \epsilon_2, iterNum)$ ;  
 //层次构建频繁模式树,
- 5:  $I\tilde{S} = FPgrowth(tree)$ ;  
 //FP-growth方法挖掘模式树tree
- 6: 优化  $I\tilde{S}$  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**  $I\tilde{S}_k$

感信息, 其长度不固定且具有随机性。

为了估计出有效的最大用户记录长度, 本文使用FO协议对用户的长度进行一次聚合, 并找到第80百分位长度值为  $iterNum$ , 然后使用该值建立频繁模式树 (2.3)。

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3。

### 2.3 层次建树

本节详细介绍了minefp方案的具体建树过程, 与FP-growth方法不同的是, 本文在LDP场景下采用层次建树方式, 构建模式树, 如图2。具体步骤如下 (详见算法4): (记树中根节点为第  $level = 0$  层, 向下依次递增)

- 1、初始化树的根节点  $root$ , 为树的  $level = 0$  层;
- 2、第  $level = 1$  层树节点聚合; 对预处理后的交易记录进行一次LDP频

**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: 初始化误差临界值 $error = 3.0 \cdot \frac{\sqrt{n_1}}{\epsilon_1}$ ; //  $n_1$ 为 $transaction_1$ 的用户总数
- 2: 对所有长度 $l \in [1, 2, \dots, k]$ 以预算 $\epsilon_1$ 使用FO协议聚合用户数 $\Phi(l)$ , 并且剔除估计结果不大于 $error$ 的项
- 3: 计算出 $iterNum$ 使得 $\frac{\sum_{l=1}^{iterNum} \Phi(l)}{\sum_{l=1}^k \Phi(l)} > 0.8$ 成立;
- 4: **return** 最大迭代次数 $iterNum$

率估计, 如下

如用户 $u_1$ 预处理后的交易为 $c, f, a, p$ , 则其当前阶段的隐私信息为其前 $level = 1$ 个项, 即为 $c$ ;

用户 $u_2$ 预处理后的交易为 $c, f, a, b$ , 则其隐私信息为 $c$ ;

用户 $u_3$ 预处理后的交易为 $f, b$ , 则其隐私信息为 $f$ ; 用户 $u_4$ 的隐私信息为 $c$ ; 用户 $u_5$ 的隐私信息为 $c$ 。

则经过第三方聚合后, 得到结果为 $\{(c, 4), (f, 1)\}$ , 然后建第 $level = 1$ 层树节点, 如图3所示, 其中由于FO协议的特性, 需要已知聚合的值域, 也就是当前层的所有可能节点 (见??), 图中计数为0的节点为无效节点。

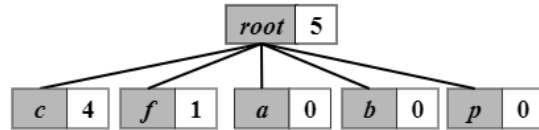


图 3: 第 $level = 1$ 层树节点

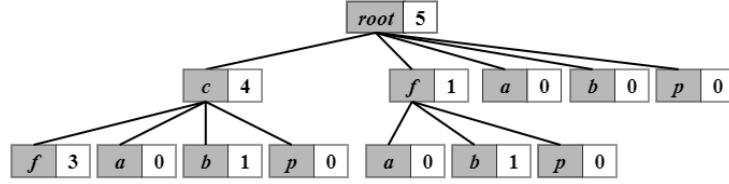
### 3、第 $level = 2$ 层树节点聚合;

用户 $u_1$ 预处理后的交易为 $c, f, a, p$ , 则其当前阶段的隐私信息为其前 $level = 2$ 个项, 即为 $(c, f)$ ;

用户 $u_2$ 的隐私信息为 $(c, f)$ , 用户 $u_3$ 的隐私信息为 $(f, b)$ , 用户 $u_4$ 的隐私信息为 $(c, b)$ , 用户 $u_5$ 的隐私信息为 $(c, f)$

第三方聚合结果为 $\{(c, f, 3), (c, b, 1), (f, b, 1)\}$ , 根据路径建立第 $level = 2$ 层树节点, 如图4所示:

### 4、依层次 $level$ 递增顺序, 直到达到树的最大层数, 建立模式树如图2所

图 4: 第 $level = 2$ 层树节点

示。

---

**Algorithm 4** build\_tree

---

**Input:**

数据集 $transaction_2$ ;  
 $top - k$ 频繁1-itemset集合 $\tilde{S}_k$ ;  
 隐私预算 $\epsilon_2$ ;  
 最大层数 $iterNum$ ;

**Output:** 模式挖掘结果 $I\tilde{S}_k$ 

- 1: 初始化树根结点 $root$ 为空;
  - 2: **for** 层次 $j = 1$  to  $iterNum$  **do**
  - 3:   均匀分配隐私预算为 $\epsilon'$ ;
  - 4:   根据第 $j - 1$ 层保留的树节点生成孩子为第 $j$ 层树节点, 并初始化节点计数为0;
  - 5:   以预算 $\epsilon'$ 使用FO协议聚合得到第 $j$ 层节点信息信息;
  - 6:   初始化误差临界值为协议标准差 $standard\_error$ ;
  - 7:   删除估计的计数值不大于 $standard\_error$ 的节点;
  - 8:   更新节点信息;
  - 9: **end for**
  - 10: **return**  $root$
- 

算法4介绍了层次建树的基本流程, 其中步骤3至步骤8介绍了树层次节点的建立与信息更新。为了保证模式树的有效性与完整性, 在使用LDP协议聚合用户信息以及更新树中节点时, 需要对具体步骤进行细节优化。以下详细介绍了算法中的各步骤及其优化过程。

### 3 建树优化

#### 3.1 隐私预算分配

层次建树过程中，本文均分隐私预算对树的层次节点LDP聚合。若给定最大层次 $iterNum$ （见2.2节），共计有两种预算分配方式：

a、将 $\epsilon$ 均分为 $\epsilon' = \epsilon / iterNum$ ，所有用户共计参与LDP聚合 $iterNum$ 次；

b、将数据集均分为 $iterNum$ 组，每组使用全部隐私预算 $\epsilon' = \epsilon$ 参与其中一次LDP聚合，且不重复参与。

文献[?, ?]表明，分配方式b优于方式a，且能有效降低 $O(\sqrt{iterNum})$ 倍误差。

#### 3.2 层次节点初始化

在使用FO协议聚合第 $j$ 层节点信息时，根据FO协议特性，其输入输出值域 $I$ 和隐私预算 $\epsilon$ 需要是已知的。对于隐私预算的分配，见3.1节，故而本节主要介绍协议在第 $j$ 层聚合时，输入输出值域 $I_j$ 的确定以及优化。

**基本方法** 根据模式树层次建立的特性，第 $j$ 层的节点信息，不仅与FO协议的聚合结果相关，而且与第 $j - 1$ 层保留的节点有关（因为无效节点——节点计数为0——不会生成孩子节点）。所以，一种生成 $I_j$ 的直接方法是根据上一层（第 $j - 1$ 层）保留的节点，生成其所有可能的孩子节点作为初始值域 $I_j$ ，用以FO协议聚合。

以图2模式树为例，根节点 $root$ 为第 $j = 0$ 层：

当 $j = 1$ 时，其上一层为根节点，则 $I_1$ 初始为 $\tilde{S}_k$ 中所有的频繁项目，即 $I_1 = \{(c), (f), (a), (b), (p)\}$ ；设此时FO协议聚合结果为 $\{(c, 4), (f, 1)\}$ ，则保留节点 $\{(c), (f)\}$ 为有效节点。

当 $j = 2$ 时，第 $j = 1$ 层保留的有效节点为 $\{(c), (f)\}$ ，则所有可能的孩子节点路径为 $\{(c, f), (c, a), (c, b), (c, p), (f, a), (f, b), (f, p)\}$ 即为 $I_2$ ；（孩子节点生成规则：根据 $\tilde{S}_k$ 的项目顺序 $c, f, a, b, p$ ，保证模式树中的节点的孩子所对应项目，其次序要小于其父节点对应项目的次序）。

同理可得， $I_3 = \{(c, f, a), (c, f, b), (c, f, p), (c, b, p)\}$ ；

$I_4 = \{(c, f, a, b), (c, f, a, p)\}$ ； $I_5 = \{(c, f, a, b, p)\}$ 。

**优化方法** 上述基本方法的缺点在于，当 $\tilde{S}_k$ 中确定的频繁项目较多或者上一层有效节点个数较多时，所生成的孩子节点会明显增多，从而使得

值域 $I_j$ 较大（即 $|I_j|$ 较大）。而根据FO协议的特性，其估计误差与 $|I_j|$ 直接相关，如GRR方法的估计方差与 $d_j$ 呈线性关系，尽管OLH方法的方差与 $|I_j|$ 无直接关系，但是其使用了 $hash$ 函数将 $|I_j|$ 映射至 $g = \lceil e^\epsilon + 1 \rceil$ ，随着 $|I_j|$ 的增大，在统计聚合时，会导致 $hash$ 碰撞明显增多，从而准确度降低。

为了有效缩小 $|I_j|$ ，从而提高估计准确度，本文提出值域筛选方法，能够在不影响结果准确性的同时将 $I_j$ 限制在大小为 $3k \ll |I_j|$ 范围，且整个过程不消耗隐私预算。

---

**Algorithm 5** prune candidate

---

**Input:**

候选值域 $I_j$ ;  
 $top-k$ 的 $\tilde{S}_k$ ;  
 正整数 $k$ ;  
 参数 $\xi$ ;

**Output:** 筛选后值域 $I'_j$

```

1: if  $|I_j|$ 不大于 $\xi \cdot k$  then
2:    $I'_j = I_j$ ;
3: else
4:   初始化 $I'_j = \emptyset$ ;
5:   for each 候选值 $X \in I_j$  do
6:      $\varphi(X) = \prod_{x \in X} \tilde{\theta}(x)$ ;  $\tilde{\theta}(x)$ 表示 $\tilde{S}_k$ 中项目 $x$ 的估计频数。
7:      $I'_j = I'_j \cup X$ , 并且记录 $\varphi(X)$ 
8:   end for
9:   以 $\varphi(X)$ 为标准, 将 $I'_j$ 排序, 最后选择前 $\xi \cdot k$ 个赋值给 $I'_j$ 
10: end if
11: return  $I'_j$ 

```

---

算法5给出了值域筛选的具体过程，其中步骤6类似于文献[?]所提SVSM的推测频率方法，而在本算法中，候选值 $X$ 中项目个数是相同的，故而本算法对项目频数直接相乘即可得到相对排序。



### 3.3 FO协议聚合

### 3.4 筛选有效聚合结果并更新树

## 4 结果优化

## 5 实验