LDP机制下的快速FIM方法

王家礼

2020 年 3 月 25 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??

Algorithm 1 总体框架

Input:

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

正整数k;隐私预算ϵ;

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??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$\tilde{S_k} = \left\{x^1 : \tilde{\theta}^1, x^2 : \tilde{\theta}^2, \ldots, x^k : \tilde{\theta}^k\right\}$,，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??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$I\tilde{S}_k$。

Algorithm 2 minefp

Input:

数据集DB_2;

隐私预算ϵ;

正整数k;

频繁$1-itemset$集合$\tilde{S}_k$;

Output: $top-k$的挖掘结果$I\tilde{S}_k$

for each $t_i \in DB_2$ **do**

  //预处理数据集

  $trans = t_i \cap \tilde{S}_k$;

  $sort(trans, key = \tilde{S}_k)$; //以$\tilde{S}_k$的项目顺序对$t$中项目进行排序;

  将$trans$中记录保留作为该用户的隐私记录t_i;

end for

将DB_2按比例分为互斥的两组$transaction_1$和$transaction_2$;

$iterNum = FO\_MaxIter(transaction_1, \epsilon)$;

//FO协议聚合transaction_1中用户记录长度的信息,

$tree = build\_fpTree(transaction_2, \tilde{S}_k, \epsilon, iterNum)$;

//层次构建频繁模式树,

$\tilde{IS} = FPgrowth(tree)$;

//FP-growth方法挖掘模式树tree

优化$\tilde{IS}$并选择$top-k$记为$I\tilde{S}_k$

//后处理步骤，不消耗隐私预算

return $I\tilde{S}_k$

算法**??**描述了minefp方案的总体框架，其中最主要是层次建立模式树tree（步骤**??**）。

步骤**??**是以LDP方式聚合得到树的最大层次iterNum，用于层次建树时更好的分配隐私预算，详见**??**节；

步骤**??**用于建立频繁模式树，详见**??**节；

步骤**??**是FP-growth的挖掘方法，更多细节参见文献[**?**]；

步骤**??**为后处理步骤，用以优化最终发布结果，能够使结果更加准确。

2.2 最大树层次

2.3 层次建树

Algorithm 3 建树

Input:

数据集DB_2;

隐私预算ϵ;

正整数k;

频繁$1-itemset$集合$\tilde{S}_k$;

Output: $I\tilde{S}_k$

初始化树根结点$root$为空;

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;

将DB'_3均分为M个人数相同的子组 $G_1, G_2, ..., G_M$;

初始化FO协议标准差为$standard_error$;

for 层次$j = 1$ to M **do**

  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。

  删除$estimation$中估计值不大于$standard_error$的结果;
  //无效值$\perp$不用于建树。

  以$estimation$聚合结果更新第j层树$root$节点;

  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};

end for

return $root$

3 结果优化

4 实验