LDP机制下快速的频繁模式挖掘方法

王家礼

2020 年 3 月 26 日

1 简介

目前，本地化差分隐私主要研究方向有频率估计、均值估计、heavy hitter、频繁模式挖掘和键值数据收集等。其中，对频繁模式挖掘的工作相对较少，文献[5]首次提出LDPMiner方法解决set-Valued场景下本地化数据聚合，并将频繁模式挖掘任务作为开放性问题提出，并没有实际解决该问题。由于本地化差分隐私具有强隐私保护性，任意第三方不会持有用户的敏感数据，所以，如何在噪声数据上进行频繁模式挖掘，并且同时保证LDP的隐私性以及估计结果的有效性，成为了当前一大难点问题。文献[8]首次提出了基于频率推测的候选集构造SVSM方法，其解决了频繁模式挖掘top-k问题。该方法的缺点在于，随着k值得不断增大，频率推测构造候选集的额外开销明细增大（详见5.1节），极大得限制了方法的应用。

为了解决SVSM方法额外开销的问题，本文通过引入FPgrowth[2]模式挖掘算法，一方面实现了基于LDP的频繁模式挖掘方法，另一方面在挖掘结果有效的前提下，大大降低了所需的额外开销。

本文贡献：

1、 LDP机制下频繁模式树的层次建立；

2、 对模式树及挖掘结果的优化；

3、 大大降低SVSM[8]方法的额外计算开销。

2 背景知识介绍

2.1 频繁模式挖掘

频繁模式是指频繁地出现在数据集中的模式，如：项集、子序列或子结构，本文主要针对频繁项集的挖掘（FIM问题）。自文献[1]首次研究在数据库中发掘项集，FIM已被广泛的研究。FIM任务的定义如下。给定一个用户交易数据库，FIM的目的是发掘数据库中频繁出现的项目和一起出现项目集合（项集）。

频繁项集挖掘（FIM）:令$DB = \{t_1, t_2, \ldots, t_n\}$表示$n$位用户的交易数据集，其中用户$u_i$的交易记录$t_i \subset I = \{x_1, x_2, \ldots, x_d\}$，$I$为总项目集合，$x_j$表示项目（或商品），$1 \le i \le n$，$1 \le j \le d$。FIM任务的结果在于挖掘出$DB$中所有频繁出现的项集。如图1所示，为$n = 5, I = \{a, b, c, f, g, h, l, n, o, p\}$时的交易数据集。

定义、频繁项集（$frequent\ itemset$）：若$X \subset I$，则X表示一个项集。当其支持度（在数据集中出现次数）$support(X)$不低于给定阈值时，则X是一个频繁项集。

定义、$\alpha - itemset$：项集X的项目个数记为$\alpha = |X|$，则X是一个$\alpha - itemset$。

2.1.1 FPgrowth模式挖掘方法[2]

本节是对非隐私保护情况下，FPgrowth模式挖掘方法的简介。FPgrowth主要分为两个步骤：建立模式树存储用户信息和挖掘模式树获得频繁模式。

TID	Transaction
01	*a,f,c,g,p*
02	*a,b,c,f,l,o*
03	*b,f,h,o*
04	*b,c,p*
05	*f,a,c,l,p,n*

图 1: 交易数据集

TID	Transaction
01	*c,f,a,p*
02	*c,f,a,b*
03	*f,b*
04	*c,b,p*
05	*c,f,a,p*

图 2: 预处理数据集

一、建立频繁模式树FP（本文主要工作是，在LDP场景下，构建有效的FP树）

1、建立项头表 ——（第一次扫描数据集，得到项头表，也即频繁$1-itemset$）

扫描交易数据集一遍，记录每个项出现的次数，根据给定的最小支持度计数（或最小支持度）筛选得到频繁$1-itemset$及它们的支持度计数，按计数值从大到小依次排序得到项头表。

如：图1交易数据集（每行为一个交易），在给定最小支持度计数为3得到项头表如图3

item	freq
c	4
f	4
a	3
b	3
p	3

图 3: 项头表

2、预处理数据集 ——（根据所得项头表，预处理数据集，将非频繁的项删除，只保留频繁的项，并且进行排序）

因为原始的交易数据集中的交易可能包含频繁$1-itemset$中没有的项（即除项头表以外的项），所以对于每个事务要把非频繁$1-itemset$中的项过滤掉，同时为了方便FP树的建立，需要把每个交易中的项(过滤后)按照其支持度大小排序，得到新交易数据集。预处理后新的交易数据集如图2。

处理规则： 将原始的交易数据集中的每一个交易(每一行)删除项头表中没有的项，并且剩下项按照项头表中每一项的支持度计数从大到小排序（所有交易共用一个项目排序，c, f, a, b, p）。

3、FP树的建立 —— （第二次扫描数据集，根据每个处理后交易记录，构建模式树）

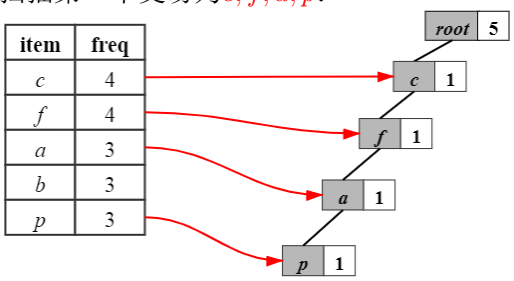
1)、FP树的根节点默认为$root$，其相应的计数为总用户数n；

2)、将新交易数据集中的每个交易变成FP树中的一条路径，并统计每个项出现的次数；

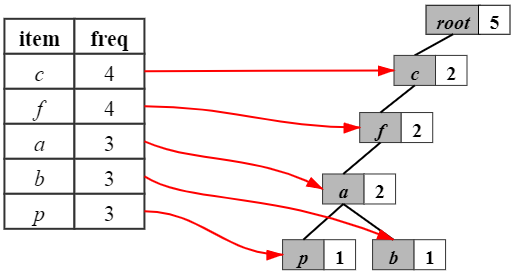
3)、对于后插入的交易先从树的根节点开始找与其相同的部分，从第一个不重合的项开始建立一个新的分支。

例：新的交易数据集如图2

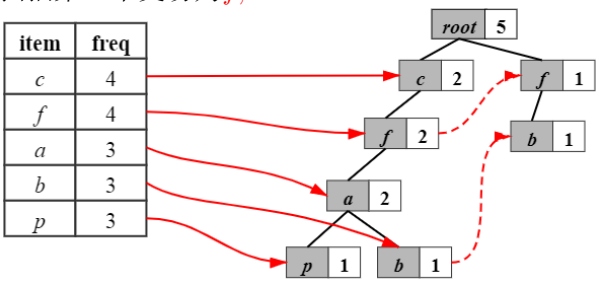
扫描第一个交易为c, f, a, p：



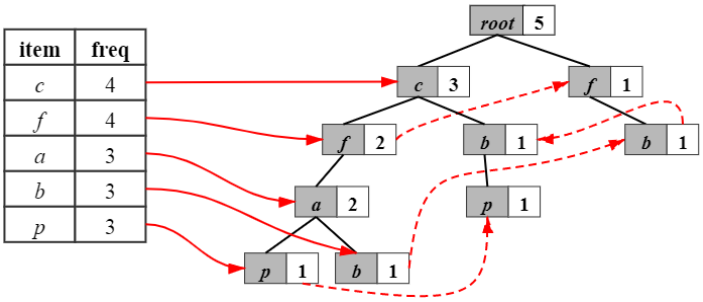
扫描第二个交易为c, f, a, b：



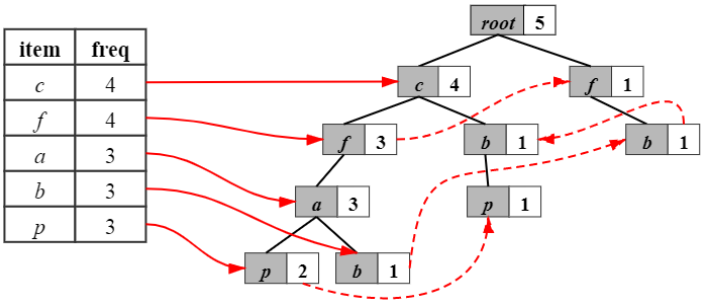
扫描第三个交易为f, b：



扫描第四个交易为c, b, p：



扫描第五个交易为c, f, a, p：



其中用项头表记录树中对应的项的节点位置，用以进行后续的挖掘步骤。

二、FP树的挖掘（本文直接使用该挖掘过程，未修改）

通过建立的FP树，从项头表的最后一项从下到上开始挖掘频繁项集。

挖掘步骤：

从项头表的最后一项从下到上开始，如当前项为i

①得到以i结尾的所有路径的前缀路径，前缀路径是指每个路径删掉该项后的路径。通常前缀路径的最后一项是一个数字，用来记录该路径出现的次数，该数字以项i的计数为准。

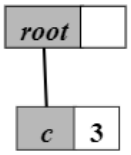
如：项a的前缀路径为$(c, f, 3)$，由以项F结尾的路径(c, f, a)删掉a得到，并且项a的计数为3，所以(c, f, a)路径出现的次数为3。

②根据项i的所有前缀路径得到条件模式基，条件模式基是将项i的所有前缀路径根据计数合并，过滤掉小于最小支持度的项。

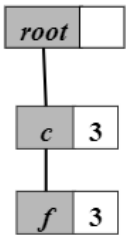
如：项p的前缀路径为$(c, f, a, 2)$，$(c, b, 1)$，根据p的前缀路径最后的计数很容得到前缀路径中每一项出现的次数$(c:2, f:2, a:2)$，$(c:1, b:1)$，然后将所有的前缀路径合并得到$(c:3, f:2, a:2, b:1)$因为$f\Delta a\Delta b$的计数小于最小支持度计数3，所以将其过滤得到最终的条件模式基$(c:3)$

③根据项i的条件模式基画出项i的条件FP树。

如项p的条件FP树为：



项a的条件FP树为：



④根据项i的条件FP树得到项 i 的频繁项集。从左到右遍历项 i 的条件FP树的每一条路径，用路径中的每单个节点和项 i组合得到项i的频繁$2-itemset$，用路径中每2个节点与项 i组合得到频繁$3-itemset$，以此类推用路径中的每k个节点与项i 组合得到项i的频繁$k+1-itemset$。

如项p的频繁项集为:

频繁$2-itemset$：$(c,p,3)$

项a的频繁项集为：

频繁$2-itemset$：$(c,a,3),(f,a,3)$

频繁$3-itemset$：$(c,f,a,3)$

⑤继续挖掘剩余项的频繁项集，得到最小支持度为3的所有频繁项集结果为：

$$(c,p,3),(c,a,3),(f,a,3),(c,b,3),(c,f,3),(c,f,a,3)$$

总结：FPgrowth方法需要扫描两次原始数据集，分别用于建立项头表（频繁$1-itemset$）和建立模式树。

2.2 本地化差分隐私（LDP）

LDP定义

2.3 频数估计协议

LDP频数估计的目的是获得任意指定项目$x\in I=\{x_1,x_2,\ldots,x_d\}$在$n$位用户中真实出现次数$\theta(x)$的无偏估计$\tilde{\theta}(x)$。

2.3.1 FO协议

最基本场景——每位用户的记录中只有一个项目x，最终估计所有项目的频数值$\tilde{\theta}(x)$

一、GRR（Generalized Randomized Response）

用户端——扰动加噪：

$$\forall_{y\in I}Pr\left[f_{GRR-\epsilon}(x)\right]=\begin{cases}p=\frac{e^{\epsilon}}{e^{\epsilon}+d-1}, & \text{if } y=x\\ q=\frac{1}{e^{\epsilon}+d-1}, & \text{if } y\neq x\end{cases} \tag{2}$$

第三方——统计聚合：

$$\tilde{\theta}_{GRR-\epsilon}(x)=\frac{C(x)-n\cdot q}{p-q} \tag{3}$$

估计误差——方差：

$$Var\left[\tilde{\theta}_{GRR-\epsilon}(x)\right] = n \cdot \frac{d-2+e^{\epsilon}}{(e^{\epsilon}-1)^2} \tag{4}$$

二、OLH（Optimized Local Hashing） 为了改进GRR方法的估计误差与d呈线性关系的缺点，OLH使用$hash$函数将d映射到$g = \lceil e^{\epsilon}+1 \rceil$范围内，在$g$中使用GRR方法。

用户端——扰动加噪：

$$f_{OLH-\epsilon}(x) := \left\langle hash, f_{GRR-\epsilon}\big(hash(x)\big) \right\rangle \tag{5}$$

第三方——统计聚合：

$$\tilde{\theta}_{OLH-\epsilon}(x) = \frac{C(x)-n/g}{p-1/g} \tag{6}$$

其中$g = \lceil e^{\epsilon}+1 \rceil, p = \frac{e^{\epsilon}}{e^{\epsilon}+g-1}$。

估计误差——方差：

$$Var\left[\tilde{\theta}_{OLH-\epsilon}(x)\right] = n \cdot \frac{4e^{\epsilon}}{(e^{\epsilon}-1)^2} \tag{7}$$

关于GRR与OLH的更详细介绍见文献[7]。由于FO协议是在用户记录仅有一个项目的情况，但是本文考虑set-valued[5, 8]场景，即用户记录有多个项目且个数未知，所以文献[8]使用了PSFO协议（2.3.2节）。

2.3.2 PSFO协议——用户的项目个数未知且随机

本文场景——每位用户持有记录中有多个项目，并且个数未知且无规律，如图1所示为$n = 5$位用户及其交易记录

填充或截取：Padding and Sampling（PS）：给定一个正整数l和一条记录$V \subset I$，记$\perp_1, \perp_2, \ldots, \perp_l \notin I$为$l$个无效值。$PS_l(V)$步骤如下：

1、若$|V| < l$，则向V中随机添加（均匀随机）$l - |V|$个不同的无效值；若$|V| > l$，则在V中随机选择l个项目保留，其余项目丢弃；

2、在处理之后的记录V中随机选择一个项目，并且输出该项目。

PSFO协议即是先进行PS操作，然后以FO协议进行后续LDP聚合。后续FO协议的值域应为$I' = I \cup \{\perp_1, \perp_2, \ldots, \perp_l\}$，且第三方需要将FO协议的聚合结果乘以因子$l$，从而保证其为无偏估计。

PSFO协议的形式表述为：

$$PSFO(l, FO, \epsilon) := \left\langle f_{FO-\epsilon}\big(PS_l(\cdot)\big), \tilde{\theta}_{FO-\epsilon}(\cdot) \times l \right\rangle \tag{8}$$

2.3.3 LDP机制下的$top-k$频繁模式聚合

在满足LDP隐私保证的同时，聚合有效的频繁模式，并且发布出$top-k$的频繁模式及其支持度。

2.4 现有方案

文献[8]首次提出LDP机制下的$top-k$频繁项集挖掘方法，分别为SVIM和SVSM。其中SVIM方法用于获得$top-k$的频繁$1-itemset$结果S'（即LDP的$top-k$频率估计），然后利用所得S'信息，通过SVSM方法获得$top-k$的$\alpha-itemset(\alpha > 1)$结果。然而，SVSM在进行$k$值较大时的挖掘任务时，会产生非常大的额外计算开销。所以，本文针对频繁项集挖掘任务，同样利用SVIM获得S'，然后通过引入FPgrowth方法进行$\alpha-itemset(\alpha > 1)$挖掘任务。

2.4.1 方案介绍

1、SVIM用于发布$top-k$的$1-itemset$（SVIM方法实现了LDP机制的$top-k$频率估计任务）；

2、SVSM用于发布$top-k$的$\alpha-itemset(\alpha>1)$。

SVSM步骤：SVIM获得$1-itemset$结果S' $\Rightarrow$ 以S'构建并选择$top-2k$的候选集IS $\Rightarrow$ 以IS进行SVIM方法获得$top-k$的$\alpha-itemset(\alpha>1)$。

SVSM候选集IS构建——SVSM的主要步骤

$$\varphi(X)=\prod_{x\in X}\mu(x),\mu(x)=\frac{0.9\times\tilde{\theta}(x)}{\max\limits_{x\in S'}\tilde{\theta}(x)} \tag{9}$$

x为S'中的某个项目，$\tilde{\theta}(x)$为S'中项目x的估计频率。

例：假设$k=4$通过SVIM方法获得的$1-itemset$结果为$S'=\{c:4,f:4,a:3,b:3\}$，则构建候选集IS步骤如下：

1）计算$t=\lfloor\log_2 k\rfloor=2$；

2）$k=4$个项目共计有2^k种组合，先筛选项目个数大于1且不小于t的组合，并且利用公式(9)分别组合的推测频数；如：通过计算组合(c,f)的推测频率为0.81、组合(c,a)的推测频率为0.6075

3）以推测频率进行排序，选择频率的$top-2k$个组合作为候选集IS。

2.4.2 方案缺点

SVSM以SVIM的$top-k$频繁$1-itemset$结果S'作为输入，然后使用公式（9）计算出所有可能项集的推测频数，用于构建出候选集IS。在IS的构建过程中，k个项目生成共计有2^k-k个$\alpha-itemset(\alpha>1)$项目集合

$$2^k=\binom{k}{1}+\binom{k}{2}+\ldots+\binom{k}{t}+\ldots+\binom{k}{k} \tag{10}$$

而SVSM方法通过将t限制为$t=\lfloor\log_2 k\rfloor$，保证其推测频数的计算空间为$O\left(\binom{k}{2}+...+\binom{k}{t}\right)$。然而，随着$k$值的增大，计算开销也会明显增大（详见5.1节）。

3 本文工作

通过SVSM方法的介绍可知，该方法在k值较大时，开销会很大。本文利用频繁模式树对用户数据的压缩存储特性，结合FPgrowth的挖掘算法，实现了快速且高效的LDP模式挖掘方法。总体流程如算法1所示。

算法1简介

步骤1使用分组的方式将总用户分为三组，让不同的组内用户以全部隐私预算参与不同的聚合过程，3.4节所述，这样会降低误差；

步骤2利用SVIM方法在LDP场景下聚合$top-k$的$1-itemset$集合$S'=\left\{x^1:\tilde{\theta}^1,x^2:\tilde{\theta}^2,\ldots,x^k:\tilde{\theta}^2\right\}$，然后公开S的信息；

步骤3至8的目的是将用户原始记录中非频繁的项目剪枝删除，因为频繁项集的Apriori属性，非频繁项目，不可能产生频繁项集，故删除是合理的；

步骤9进行最大迭代次数的估计，即树的最大深度估计，见3.5节；

步骤10是本文主要工作部分，即在LDP场景下建立频繁模式树，详见3.1节；

步骤11是对模式树的挖掘过程，由于过程于文献[2]相同，故不详细介绍。

Algorithm 1 LDP建树总步骤

Input:

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

d个项目组成的值域$I = \{x_1, x_2, \ldots, x_d\}$;

正整数k;隐私预算ϵ;

Output: $top-k$的频繁项集

1: 将数据集DB按比例分为三个互斥的子组DB_1、DB_2和DB_3:
$|DB_1| = n_1 = 0.5n, |DB_2| = n_2 = 0.1n, |DB_3| = n_3 = 0.4n$

2: DB_1的用户执行$SVIM$方法得到$top-k$频繁$1-itemset$集合
$S' = \left\{x^1 : \tilde{\theta}^1, x^2 : \tilde{\theta}^2, \ldots, x^k : \tilde{\theta}^k\right\}$

3: **for** each $t_j \in DB_2 \cup DB_3$ **do**

4: //预处理数据集

5: $t = t_j \cap S$;

6: 以S'中项目顺序对t进行排序;

7: 将处理后的记录t作为用户j的隐私数据;

8: **end for**

9: M = 算法3估计最大迭代次数(DB_2,k,ϵ);

10: tree = 算法2建树(DB_3,S',ϵ,M);

11: $top-k$ = FPgrowth(tree); //挖掘过程详见文献

12: 对$top-k$结果进行优化;//见4.4节

13: **return** $top-k$

步骤12是差分隐私的后处理步骤，不消耗隐私预算，也不会泄露用户隐私。

总结： 算法1是整个机制的总体流程，步骤2与步骤9过程与文献[8]所用方法相似，为已有方法；本文只是在步骤9时，修改了部分参数的值，具体如下：

1、长度估计时，SVIM设置参数$\gamma = 0.9$，本文设置$\gamma = 0.8$用于估计树的最大深度；

2、在临界值的设置中，SVIM设置T $= F^{-1}\left(1 - \frac{0.05}{2k}\right)\sqrt{Var}$，其中$F$为标准正态分布的CDF，将所得的估计结果不大于T的记为无效估计，令为0；本文设置$T = \frac{\sqrt{n}}{\epsilon}$，祥见3.5节。

3.1 LDP机制下的频繁模式树层次建立

根据算法1可知，建立频繁模式树的前提，是在已知$1-itemset$结果S'，以及预处理之后的数据集DB'，即以图2所示数据记录建立模式树，LDP场景下的建树步骤具体如下：（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4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3.2），图中计数为0的节点为无效节点。

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)；

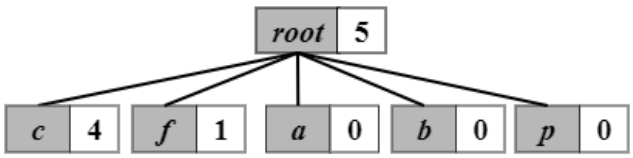


图 4: 第1层树节点

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5所示：

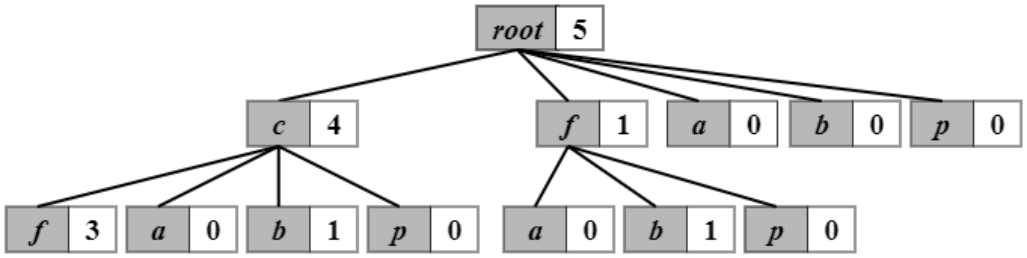


图 5: 第2层树节点

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6所示。

总结：算法2为层次建树具体过程。

存在问题：上述建树过程仍然存在以下问题，对问题的具体解决过程见下文：

Q1、 LDP机制下如何进行每层节点的频数聚合；

Q2、 set-valued场景用户记录长度不固定且随机，聚合过程中，用户记录可能为空；

Q3、 隐私预算ϵ的分配；

Q4、 树的最大层次。

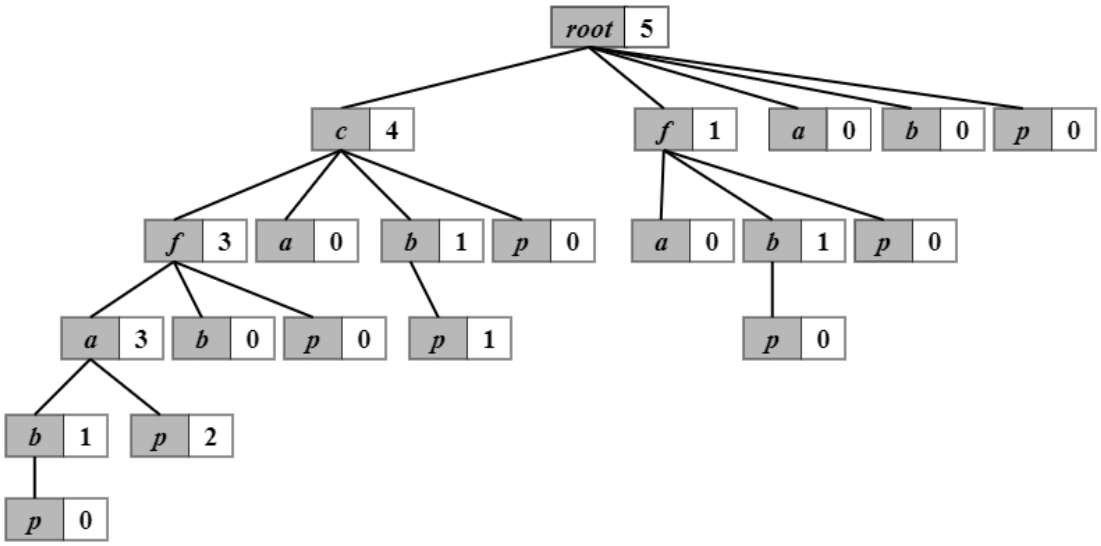


图 6: 模式树

Algorithm 2 建树

Input:

频繁$1-itemset$集合$S'=\left\{x^1:\tilde{\theta}^1,x^2:\tilde{\theta}^2,\ldots,x^k:\tilde{\theta}^2\right\}$;

预处理后的数据集DB_3';

正整数k;隐私预算ϵ;

最大迭代次数M;

Output: 模式树

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};

  $I_{j+1}'=$算法4(I_{j+1},S',k,ξ)

end for

return $root$

3.2 Q1——模式树的LDP机制下层次建立

在使用FO频率估计协议进行当前第j层数据聚合时，首先需要解决的问题是如何确定输出值域I_j?

基本方法 由模式树层次建立的特性，第j层节点的信息，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（因为无效节点不会产生下一层的节点）。所以，一种生成I_j的直接方法是依据第$j-1$层保留节点的信息，生成所有可能的孩子节点作为j层初始节点，然后将其作为I_j使用FO协议聚合。

以图6为例，第0层初始为根节点$root$

当$j=1$时，其上一层为根节点，则I_1初始为所有$1-itemset$，即$I_1=\{(c),(f),(a),(b),(p)\}$；

当$j=2$时，第1层保留的有效节点为$\{(c),(f)\}$，所有可能的孩子节点路径为$\{(c,f),(c,a),(c,b),(c,p),(f,a),(f,b),($
为I_2；（$1-itemset$的项目顺序为c,f,a,b,p，模式树中的节点的孩子，对应项目的顺序要小于其父节点对应项目）

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

上述方法的缺点在于，当S'中的项目增多或者上一层保留节点个数较多时，进行下一层节点聚合所得值域的大小$\|I_j\|=d_j$会明显变大。而LDP的FO协议的估计精确度与d_j呈正相关，如GRR方法的估计方差与d_j为线性关系，而OLH方法的方差虽然与d_j无关，但是其使用了$hash$函数将d_j映射到$g=\lceil e^\epsilon+1\rceil$范围，随着$d_j$的增大，在进行统计时，$hash$碰撞也会明显增多，导致精确度的降低。

为了减小d_j，从而提高估计准确度，本文提出值域筛选的进一步方法，将d_j进一步限制在大小为$3k\ll d_j$，且整个过程不消耗隐私预算。**详见4.1节。**

3.3 Q2——用户记录为空或项目个数少于当前树的层次

若用户在当前阶段的记录项目个数小于层次j或所得值域I_j中不包含用户的记录，那么，则该用户以给定无效值$\perp$为原始数据进行扰动，然后提交扰动结果。此时的值域则为$I_j \cup \perp$

3.4 Q3——隐私预算分配

文献[6]指出，若给定最大迭代次数M，可将用户分为M个互不相交的小组，每组以全部隐私预算ϵ参与一次聚合过程，且不重复参与。(**对于M的确定，见3.5节**)

其同时指出，上述分配方式，与均分隐私预算为ϵ/M，用户参与所有过程（M次）的方式相比，减小了误差为$O(\sqrt{M})$倍.

3.5 Q4——树最大层次

若不考虑用户隐私，最大迭代次数即为用户预处理后的项目记录最大长度，如图2所示，记录最大长度为5，则树的最大层次为5（根结点为第0层）。

但是由于记录的长度也属于敏感信息，在LDP机制下，由于长度是不固定且随机的。本文以FO协议，先对用户的记录长度进行一次聚合，从所得结果中确定最大层次。其过程与文献[8]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。

Algorithm 3 估计最大迭代次数

Input: 预处理后的数据集DB_2';正整数k;隐私预算ϵ;

Output: 最大迭代次数

1: 初始化临界值$T = 3.0 \cdot \frac{\sqrt{n}}{\epsilon}$; /*$n$为$DB_2'$的用户总数*/
2: **for** each $t_i \in DB_2'$ **do**
3: //用户端扰动
4: $x = |t_i|$;
5: 利用公式(5)得到扰动结果$y = f_{OLH-\epsilon}(x)$，然后将y提交给第三方;
6: **end for**
7: **for** each $m \in \{1, 2, \ldots, k\}$ **do**
8: //第三方聚合
9: 利用公式(6)计算出长度为m的用户估计人数$\tilde{\theta}_{OLH-\epsilon}(m)$;
10: **if** $\tilde{\theta}_{OLH-\epsilon}(m) \leq T$ **then**
11: $\tilde{\theta}_{OLH-\epsilon}(m) = 0$;
12: **end if**
13: **end for**t
14: 利用公式(8)在$\gamma = 0.8$时的整数值M;
15: **return** 最大迭代次数M

类似文献[6]所设置最大误差方法，由伯恩斯坦不等式可得，OLH聚合算法，对任意项目x的估计频数$\tilde{C}(x)$与真实频数$C(x)$，以至少$1-\beta$的概率，当$\lambda = O\left(\frac{\sqrt{n}\cdot\sqrt{\ln(1/\beta)}}{\epsilon}\right)$时，不等式$\left|\tilde{C}(x) - C(x)\right| < \lambda$成立。

本文设置$T = 3.0 \cdot \frac{\sqrt{n}}{\epsilon}$为最大误差临界值，用于筛选估计结果，见算法3步骤1。

算法3步骤14中，$\gamma = 0.8$的取值时一个权衡，根据实际情况进去修改（文献[8]使用值为0.9）。若γ越大，迭代次数M越大，会保留更多的用户信息，其估计值的累计误差（均值平方误差）会降低，但是由于迭代次数的增多，结果的命中率会有所降低。

4 优化

本章介绍了对模式树构建方法的进一步优化过程

4.1 值域筛选

本文最终目的在于发布$top-k$的频繁$\alpha-itemset(\alpha>1)$结果，如3.2节所述，值域大小直接影响着估计结果的准确度，本节详细介绍了对值域的进一步筛选过程。

在值域筛选过程中，利用文献[8]的SVSM方法思想，即根据$1-itemset$结果计算项集X的推测频率（公式(9)），然后排序选择$top-2k$个结果作为候选集进行LDP的聚合。

类似于上述方法，本文在进行第j层聚合时，对其初始值域I_j进行了一次的频数推测，然后筛选出$top-\xi\cdot k$个作为最终值域I'_j（实验中$\xi=3$）。由于在该过程中，初始值域I_j中项集的长度必然为j，即属于$j-itemset$，所以公式（9）可直接省略为 $\varphi(X)=\prod_{x\in X}\tilde{\theta}(x)$。具体过程见算法4。

Algorithm 4 值域筛选

Input:
候选值域I_j;
1-itemset结果S';
正整数k;
参数ξ;
Output: 筛选后值域I'_j

```
if I_j的长度不大于4k then
    I'_j = I_j;
else
    初始化I'_j = ∅;
    for each 项集X ∈ I_j do
        φ(X) = ∏_{x∈X} θ̃(x); //θ̃(x)为1-itemset集合S中估计频数值。
        I'_j = I'_j ∪ X,并记录φ(X)
    end for
    以φ(X)为标准，对I'_j中项集排序;
    选择top − ξ · k赋值给I'_j
end if
return I'_j
```

4.2 临界值设置

根据4.1节所述方法确定值域后，以FO协议聚合出该阶段用户的信息，即为树当前层节点所需保留的信息。然而由于估计误差的存在，聚合结果中存在许多无效的估计（如：估计频数为负数等），本文通过设置临界值$error$对聚合结果进一步筛选，将所有估计频数不大于$error$的结果删除。$error$值的设置一方面要尽可能多的删除无效节点，另一方面也要将有效节点尽可能多的保留。本文设置的误差临界值为$error=\sqrt{Var}$，即LDP聚合协议的估计标准差。

(类似误差值的设置有三种方法：)

1、$error_1=\sqrt{Var}$，即标准差；

2、文献[7, 9, 8]，$error_2=\mathrm{T}=F^{-1}\left(1-\frac{0.05}{2k}\right)\sqrt{Var}$，$F^{-1}$为标准正态分布的CDF的倒数。

3、文献[6]，$error_3=3.0\cdot\frac{\sqrt{n}}{\epsilon}$，其根据伯恩斯坦不等式求得最大误差边界（3.5节）

（三者的关系为$error_1<error_2<error_3$）

4.3 约束推理——一致性矫正，该部分仍然达不到预期效果，导致具体实验结果的均值平方误差与对比方案相比偏高

设v表示树中某节点，其相应的计数值为$v.count$，则对于节点v应有以下约束：

1）、 无偏估计：$\mathbb{E}\left[\tilde{C}(v.count)\right] = \mathbb{E}\left[C(v.count)\right]$;即对节点的计数值的估计保证无偏（FO协议）

2）、 父节点计数值不小于孩子节点计数之和，记$v.children$表示v的所有孩子节点集合，假设有r个：

$$\mathbb{E}\left[\tilde{C}(v.count)\right] \geq \sum_{y \in v.children} \mathbb{E}\left[\tilde{C}(y.count)\right] \tag{11}$$

文献[3, 6, 4]等均研究了不同场景下的约束推理方式，其中文献[6]对两次的聚合结果进行矫正，不适用于本文场景；而文献[3, 4]均是DP机制下的约束推理。

具体矫正步骤：

4.4 加权组合

根据本文算法构建模式树，以最小支持度为0对树使用FP-growth算法[2]所得到的所有频繁项集结果（不含$1-itemset$）为$IS' = \left\{X_1 : \tilde{\theta}(X_1), X_2 : \tilde{\theta}(X_2), \ldots, X_s : \tilde{\theta}(X_s)\right\}$，共$s$个为模式树中挖掘的项集及其频数。文献[8]指出，对某项集X可通过公式（9）计算出其推测频数$\varphi(X)$，然后得到其相对顺序。

本文同样利用该推测频数信息，对估计结果，进行进一步的优化：

1、对频数结果加权组合

$$\tilde{\theta}'(X) = \omega \cdot \tilde{\theta}(X) + (1-\omega) \cdot \varphi(X) \tag{13}$$

2、以加权结果的$top-k$项集作为最终输出的项目集合

5 实验

记$X_{estimate} = \{X_1, X_2, \ldots, X_k\}$与$X_{correct}$分布表示$top-k$估计结果与真实结果，则$X_{cap} = X_{estimate} \cap X_{correct}$表示算法正确命中的项集。

评价指标[8]：

1、**NCR**(命中率)：

$$NCR = \frac{\sum_{X \in X_{estimate}} q(X)}{\sum_{X \in X_{correct}} q(X)}$$

其中$q(X)$是评分，文献[8]将其根据$X_{correct}$的排序，依次设置为$k, k-1, \ldots, 1$，其他均为0。

2、**MSE**(误差——均值平方误差)：

$$MSE = \frac{1}{\|X_{cap}\|} \sum_{X \in X_{cap}} \left(\theta(X) - \tilde{\theta}(X)\right)^2$$

其中$\theta(X)$与$\tilde{\theta}(X)$分布是X的真实频数与估计频数。

数据集：

1、**Kosarak**：990002位用户的网页点击数据，总项目个数为41270，用户平均项目长度8.0，最大项目长度2498。

2、**BMS-POS**：515597位用户的商品购买数据，总项目个数为1657，用户平均项目长度6.5，最大项目长度164。

5.1 计算开销

以下对比的开销只针对在值域筛选的搜索空间（搜索空间内排序并筛选得到值域的额外计算开销），其中对比方案的搜索空间大小只与k有关，根据公式(9)计算得到，而本文方案的搜索空间是在实验过程中总搜索空间（本文方案搜索空间与k, ϵ以及数据集都有关，其中ϵ和数据集会影响树中每层节点的保留数，保留节点越多，搜索空间越大）。以下结果是在$\epsilon = 2.0$时的搜索空间对比。

k	计算开销		
	对比方案	本文方案($\epsilon = 2$)	
		POS数据集	Kosarak数据集
50	2369885.0	6100.4	5537.5
75	219904690.0	12807.15	10916.0
100	1271427795.0	22979.6	17107.0
125	4935173650.0	33532.8	24959.5
150	309019152705.0	46129.3	33036.5

表 1: 开销对比

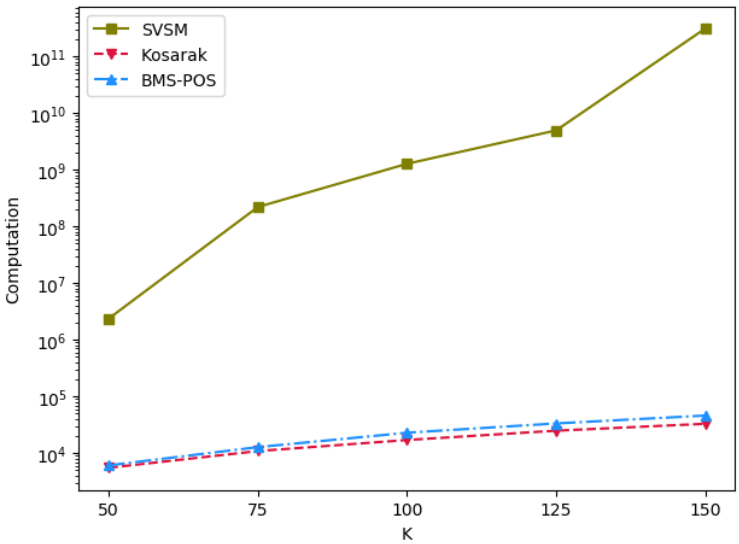


图 7: 计算开销

5.2 评分结果——该部分结果显示在高预算情况下，均值平方误差比对比方案高

图中Origin线是未进行加权组合结果，即权值$\omega = 1$；而SVSM线是本文的对比方案；Linear线是在$\omega = 0.8$取值下的结果。

Kosarak数据集

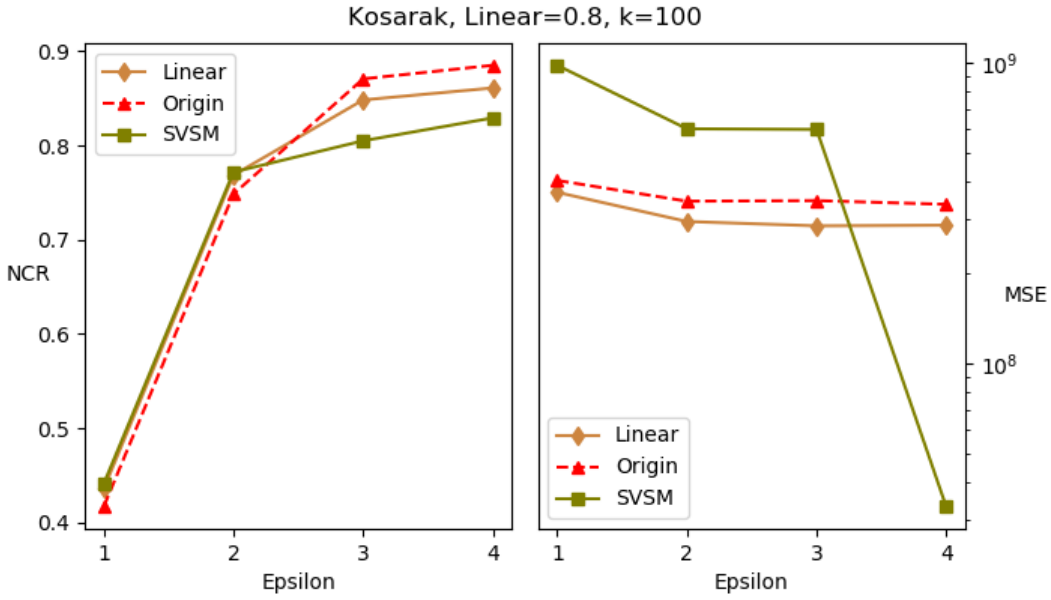


图 8: Kosarak数据集，$Linear = 0.8, k = 100$时不同ϵ对比

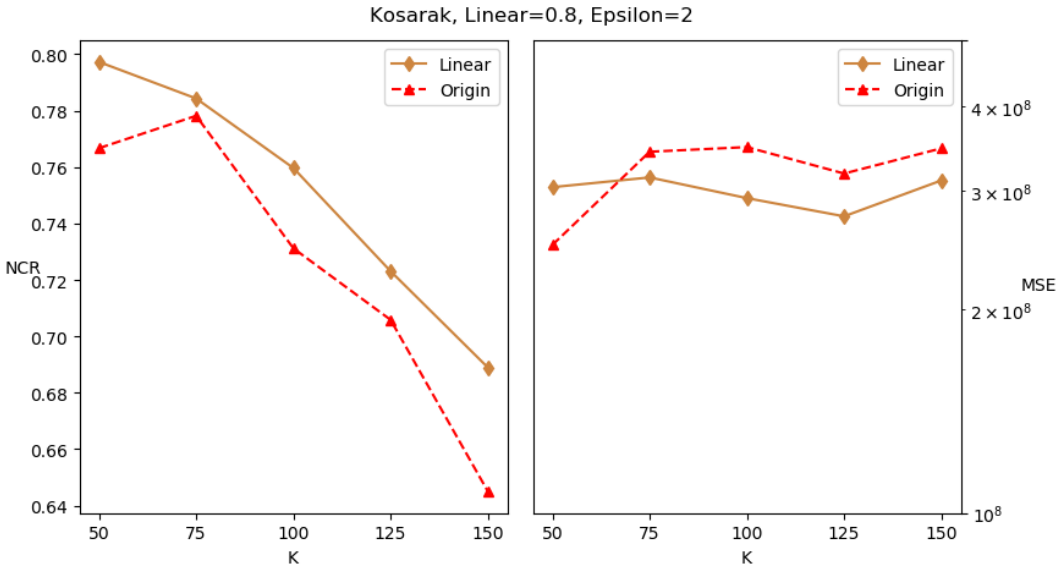


图 9: Kosarak数据集，$Linear = 0.8, \epsilon = 2$时不同$k$对比

BMS-POS数据集

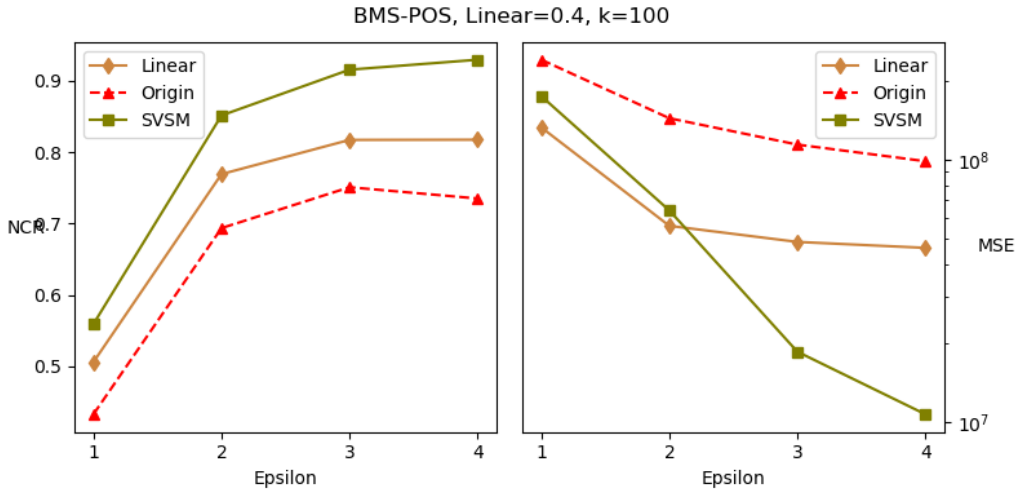


图 10: BMS-POS数据集，$Linear = 0.4, k = 100$时方案对比

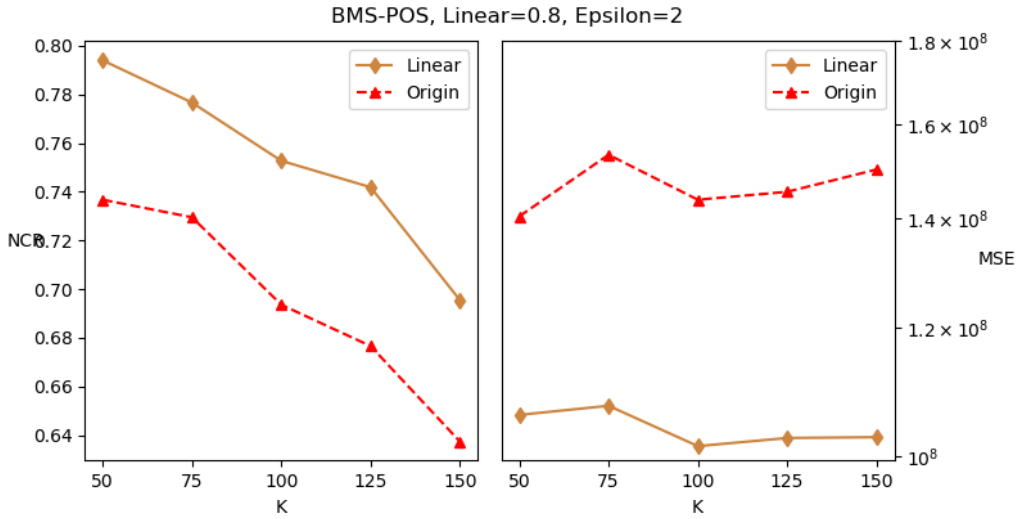


图 11: BMS-POS数据集，$Linear = 0.8, \epsilon = 2$时不同$k$对比

5.3 加权组合

图中Origin线是未进行加权组合结果，即权值$\omega = 1$；而SVSM线是本文的对比方案；Linear线不同ω取值下的结果。

5.3.1 BMS-POS数据集不同ϵ取值下的对比

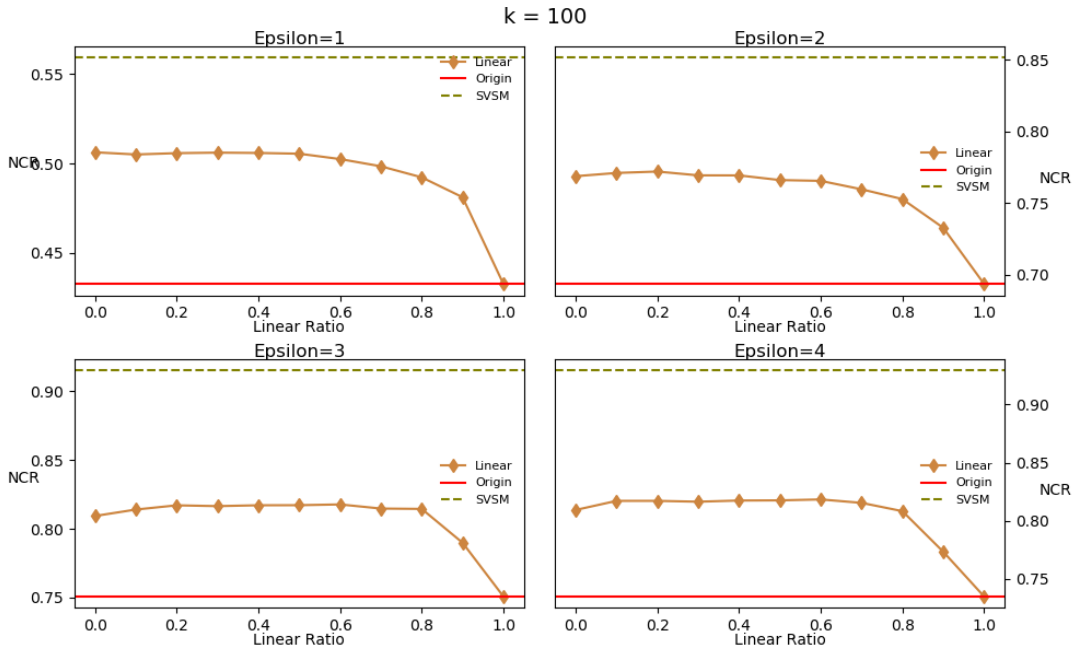


图 12: BMS-POS命中率NCR对比

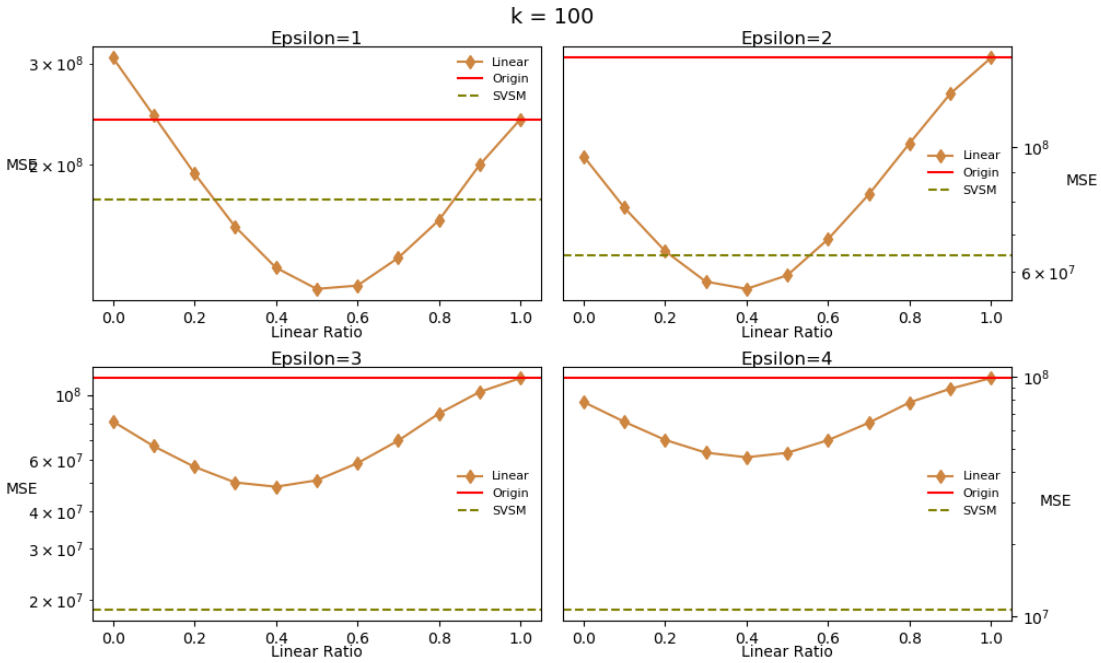


图 13: BMS-POS误差MSE对比

5.3.2 Kosarak数据集不同ϵ取值下的对比

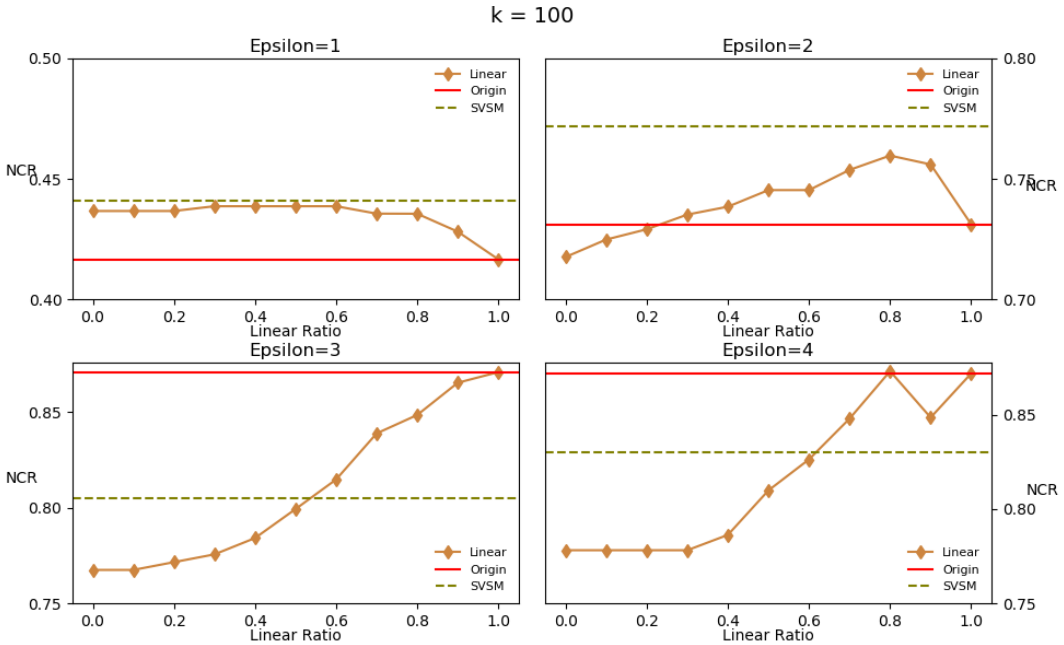


图 14: Kosarak命中率NCR对比

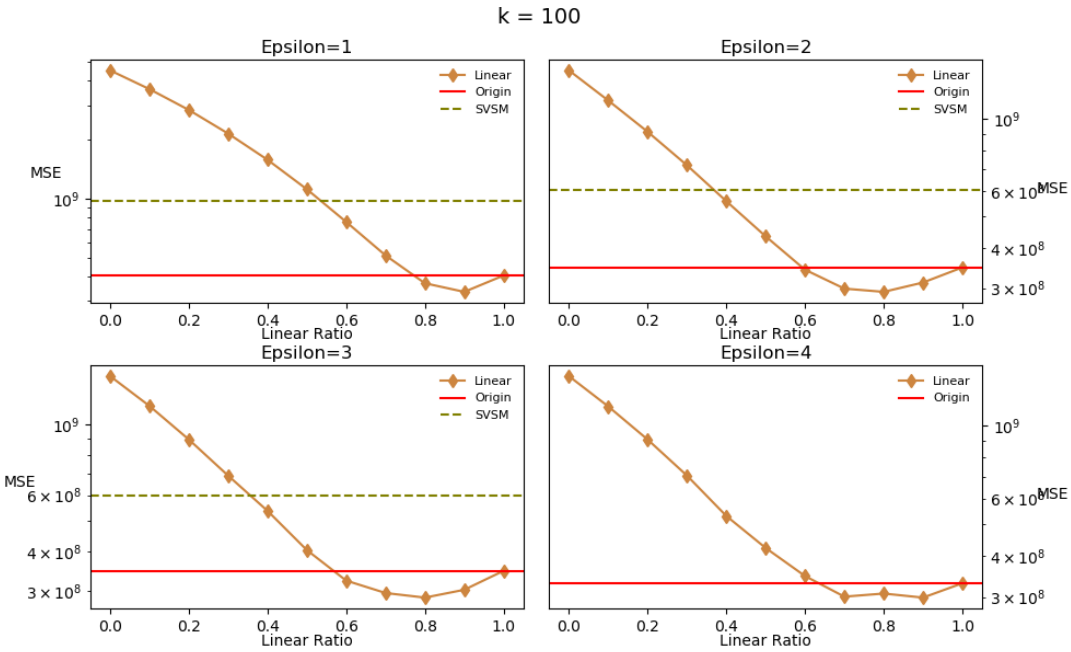


图 15: Kosarak误差MSE对比

Reference

[1] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[2] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM sigmod record*, volume 29, pages 1–12. ACM, 2000.

[3] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1-2):1021–1032, 2010.

[4] Jaewoo Lee and Christopher W Clifton. Top-k frequent itemsets via differentially private fp-trees. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 931–940. ACM, 2014.

[5] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 192–203. ACM, 2016.

[6] Ning Wang, Xiaokui Xiao, Yin Yang, Ta Duy Hoang, Hyejin Shin, Junbum Shin, and Ge Yu. Privtrie: Effective frequent term discovery under local differential privacy. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 821–832. IEEE, 2018.

[7] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 729–745, 2017.

[8] Tianhao Wang, Ninghui Li, and Somesh Jha. Locally differentially private frequent itemset mining. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 127–143. IEEE, 2018.

[9] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.