AHEAD:Adaptive Hierarchical Decompositon Decomposition for Range Query under Local Differential Privacy

AHEAD:本地差分隐私下针对范围查询的自适应层次分解方法

摘要：

为了保护用户的数据隐私，本地差分隐私被用来向范围查询提供隐私保护。但是现存的基于本地隐私查询的方法都受限于一种静态的预定义的收集结构。当隐私预算 ε 值低的时候，这将会导致聚合数据的时候产生过量的噪声。所以在本篇文章中作者提出了一种自适应的控制收集结构构建的方案，这将会降低噪音所带来的误差从而提升范围查询的精确度。

文章主要结构：

1.介绍背景

2.介绍论文涉及的主要概念

3.现存范围查询方法及误差分析

4.AHEAD 方案误差分析及参数选择

5.实验结果

6.总结和展望

2.介绍论文涉及的主要概念

2.1 本地差分隐私

若随机算法 A 对任意的不同输入 X 和 X' 返回相同结果 Y 的概率满足公式（0），则 A 满足 ε −本地化差分隐私。

$$\Pr[A(X)=Y]\le \exp(\varepsilon)\times \Pr[A(X')=Y].....(0)$$

当严格满足上述公式的范围查询结果 Ψ 且当且仅当 ε >0，所有的 $v_1,v_2\in D$ 我们将其满足下列公式:

$$\forall \mathrm{T}\in Range(\Psi):\Pr\Psi(v_1)\in T\le e^{\varepsilon}\Pr\Psi(v_2)\in T$$

2.2 频率估计协议（噪声扰动机制）Frequency Oracle

论文中会介绍两种先进的协议，其大致步骤分为：编码、扰动、聚合。

2.2.1 随机响应机制（GRR）

编码：传统的 RR 机制只适用于二进制数据，即{0,1}，而 GRR 将其扩展到 d 元数据，即 x∈{1,2,....,d},根据需要将数据进行编码，就用二进制形式为例。

扰动：当我们编码为一组由{0,1}构成的向量时，采用 p 的概率保留第 i 位上的数字不变，采用 q 的概率翻转第 i 位上的数字。公式如下：

$$\Pr[M(x)=y]=\begin{cases}p=\dfrac{\varepsilon}{e^{\varepsilon}+|D|+1}, y=x\\ q=\dfrac{1}{e^{\varepsilon}+|D|+1}, y\neq x\end{cases}$$

聚合：聚合过程会统计一个特征值 v 出现的次数 count[v]，同时

将其纠正为无偏估计 $\hat{f}[v]=\frac{count[v]-Nq}{N(p-q)}$

估计误差：无偏估计 $\hat{f}_v$ 与真实估计 f_v 之间的误差为：

$$var_{GRR(\varepsilon)}=\frac{|D|-2+e^{\varepsilon}}{N(e^{\varepsilon}-1)^2}$$

2.2.1 改进一元编码机制（OUE）

编码：对于 d 元数据 $x\in\{1,2,....,d\}$,首先对输入 v 进行独热（one-hot）编码,构造一个长度为 d 的零向量 B，然后把 v 对应特征位设为 1，其他位置为 0 不变。

扰动：然后对每一位进行扰动，扰动概率如下，p 对应 1 是否翻转的概率，q 为 0 是否翻转的概率。

$$\Pr[B'[i]=1]=\begin{cases}p, B[i]=1\\ q, B[i]=0\end{cases}$$

聚合：统计每一个向量中不同特征位中 1 出现的次数 count[v]，纠正为无偏估计：$\hat{f}[v]=\frac{count[v]-Nq}{N(p-q)}$

估计误差：$var_{OUE(\varepsilon)}=\frac{4e^{\varepsilon}}{N(e^{\varepsilon}-1)^2}$

3.现存范围查询方法及误差分析

	Age	Salary	Loan amount
v_1	18	150	0
v_2	42	5400	49192
v_3	27	2310	2194
...	...	...	...
v_N	69	3820	1982

上图为一个数据库的例子，不同类型的特征值有三个：Age、Salary、Loan amount.每一个特征值所含有的最大不同特征值数为特征值域 D 的值域长度|D|。

3.1 层次分隔优化(HIO)

采用一种静态的预定义的树结构（大部分特征查询区间划分都采用树的形式进行划分）统计频率区间分布，根节点代表了整个特征域 D，然后每个子区间都会计算得到由 OUE 扰动机制（或者其他的扰动机制）扰动后的频率估计。当响应一个查询的时候，HIO 会最小区间划分来囊括住整个查询范围。

例如，当|D|=8，树的分支数取 B=2，即为二叉树。则范围查询[2,7]的区间划分为[2,3]∪[4,7]。然后 HIO 将统计到的这两个区间的经过扰动后的区间频率放进这两个区间中去。

HIO 的缺点在于会往所有区间中加入同样等级的噪声，而扰动噪声往往会覆盖某些含有真实值频率分布的区间，并且在一种多维查询的应用场景下，静态树的层数会随着多维的查询而提升，最终导致查询误差的累积增大。

4.AHEAD 方案误差分析及参数选择

本篇文章提出的 AHEAD 自适应的动态方案就是为了解决两个问题：

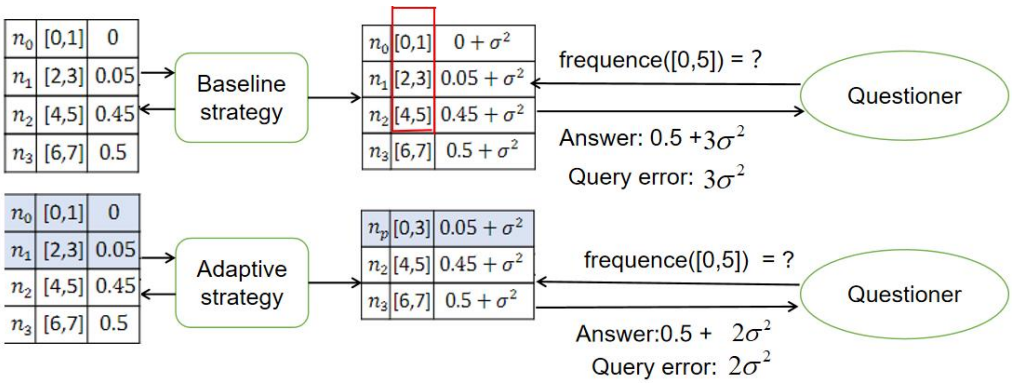
（1）实际查询情况下，由于查询的前提条件是假设数据的分布在区间内是均匀分布的，而真实情况下数据在区间内的分布不是均匀的，所以会带来一种因非均匀分布在均匀分布下所引起的误差。

（2）区间的划分会使得噪声扰动数据后带来的误差增大，从而降低查询的精度。

如何理解这两个问题之间的关系？重点在于区间的划分上面，当区间划分多了也就是每个区间的宽度（域）更小了，会增大噪声累积误差（2），但是可以降低第（1）类中的实际非均匀分布的误差，而将区间划分少了，也就是每个区间的宽度更大了，可以降低（2）的误差但是会增大（1）中的误差。

所以AHEAD方案就是为了平衡这两者之间的错误以提升查询的精度。

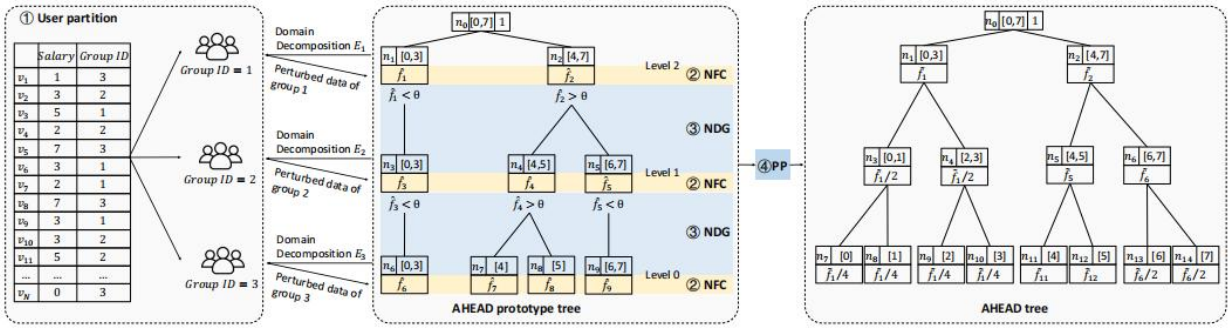
4.1 在具体看 AHEAD 的工作流程之前我们再举一个例子来帮助了解：



在上述情况中上半部分代表的是静态查询方案下的一种查询过程，可以看到n_0代表的是[0,1]的区间，而 0 代表的是区间内是没有真实数据分布在其中，频率分布为 0。假设在经过 OUE 扰动机制之后采用 HIO 的静态方案重新统计区间频率统计分布可以知道在n_0区间中这时候频率分布是 $0+\sigma^2$，σ^2为噪声带来的误差，而此时查询区间[0,5]的统计结果是 $0.5+3\sigma^2$。但是如果使用的是 AHEAD 的方案，则查询结果为 $0.5+2\sigma^2$，误差降低了 30%多。

AHEAD 这种自适应的方案是如何更加合理的划分区间，在降低噪声累积误差的同时平衡因此引入的非均匀分布误差的？主要关键在于每一次划分区间数 B（树往下一层的分叉数）和决定图中所展示的那样是否不继续划分区间的阈值θ。这两个参数将决定 AHEAD 方案的优越性。

4.2 AHEAD 方案的工作流程（Workflow of AHEAD）



接下来我们将会按照上图所示讲述 AHEAD 的工作流程，我们的范围查询是对 Salary 这个特征进行的一维查询，其域|D|为 8，用户 v∈{1,2,...,N},用户的分组 ID 为{0,1,2,3}。我们选取查询树的分支数 B=2，即树中每一个非叶子节点都含有两个孩子节点。并且，值得注意的是 AHEAD 在划分用户的时候，每个用户组所使用到的隐私预算都是 ε，ε 无需跟随用户组一起划分，这与静态的方案是不一样的。

Step 1：用户组划分（User Partition）。如上图所示，用户组的划分取决于树的高度 $c = \log_B |D|$，每一个用户组对应树中的一层。在这里 c=3。同时在用户组划分结束后，每个用户会被随机的分配进组内也可以根据一些其他的要素来进行分配（例如用户的 ID 等）。这样随机分组的形式可以使得每一子区间的频率估计使用完整的隐私预算 ε，从而降低噪声，提高查询精确性。

Step 2：噪声扰动（Noisy Frequency Construction）。看到图中第一个图形与第二个图形之间的那些连线，就是我们第二步所需要做的事。例如图中根节点 n_0 划分为两个区间 n_1 和 n_2 后，NFC（第二步）的作用就是，计算每个区间中经过噪声扰动之后的数据分布频率 $\hat{f}$。

Step 3：划分子区间（New Decomposition Generation）。这一

步将在我们 NFC 后根据计算出的$\hat{f}$来决定当前子区间是（1）继续按照参数 B 来进行生成孩子节点还是（2）保持不变该节点到下一层。具体细节是比较我们区间的$\hat{f}_n$和阈值θ。若$\hat{f}_n$>θ则生成下一层的子孩子，例如区间n_2中的频率$\hat{f}_2$>θ则生成孩子n_4和n_5。

以上就是我们生成整个查询树的过程，我们重复上述的 3、4 步直到所有的$\hat{f}_n$<θ就停止。

但此时注意到，有些节点的生成是保持父节点到下一层的，所以我们此时生成好的二叉树是不满足孩子节点的频率之和等于父节点的频率，而且注意到有些具体的特征值在经过噪声扰动之后的取值是可能跳变为负值的，这有可能导致$\hat{f}_n$为负。所以我们要对此时形成的树进行非负化和加权平均。

Step 4:后处理（Post-processing）。我们会将区间中为负的频率$\hat{f}_n$取 0。每一个区间的频率$\hat{f}_n$都会减去总的平均差。重复这两步直到不出现负值。之后根据以下公式来更新每个区间中的频率值$\hat{f}_n$。

$$\widetilde{f}(n)=\begin{cases}\lambda_1\hat{f}(n)+\lambda_2\sum_{u\in child(n)}\hat{f}(u), leaf\\ \lambda_1\hat{f}(n)+\lambda_2\sum_{u\in child(n)}\widetilde{f}(u), o.w.\end{cases}$$

其中$\lambda_1=\frac{\mathrm{Var}_{child(n)}}{Var_{child(n)}+Var_{(n)}}$和$\lambda_2=\frac{\mathrm{Var}_{(n)}}{Var_{child(n)}+Var_{(n)}}$，$\tilde{f}$是$\hat{f}$经过后处理后的频率。

最后，从上到下，AHEAD 在均匀分布的假设下严格的划分区间以形成一颗满二叉树。

4.2 误差分析

假设我们的扰动机制是采用先进的办法 OUE 进行的，那我们的噪声误差将由 OUE 带来。因为用户组的分配是按照随机组 c 进行分配的，所以每个组内有$\frac{\mathrm{N}}{c}$个用户，那么由 OUE 带来的误差 var 也应变为$\sigma^2 = c \cdot \frac{4e^{\varepsilon}}{N(e^{\varepsilon}-1)^2}$。同时，由于阈值θ的设定，那些已经无法继续划分孩子的节点的孩子节点的频率值应该由父节点进行平均分配，由层数来决定，即父节点的区间宽度是孩子节点的 k 倍。则孩子节点的频率值为最后一个无法划分的父节点的$\frac{1}{k}$。因此噪声带来的误差也应该是$\frac{1}{k}$。

而对于非均匀分布带来的误差，在数据的分布真的满足均匀分布的情况下，对于那些不满足阈值θ的区间，其频率值就是$\frac{1}{k}$的f_{p}，即$\mathrm{f}_n = \frac{f_p}{k}$。实际上我们的数据分布也不可能是均匀分布的，当父节点f_{p}的分布越均匀，则误差则越小，否则越大。因此，非均匀分布的误差取决于真实频率f_{p},i.e.$|f_p - \frac{f_p}{k}|$。

4.3 参数 B 与θ的取值

有了 4.2 的误差分析，那么我们就能得出 B 与θ是如何取的。首先对于参数**θ**，θ的取值目的是为了使得 AHEAD 的划分方案的误差总是优于静态分解方案的。那么我们可以举个例子，对于 AHEAD 方案方面，根据我们之前的误差分析，使用$f_1, f_2 \ldots\ldots, f_B$表示真实频率 f 的孩子频率，当 f 的分布越远离均匀分布的情况下，η越趋近于 0 或 1，反之

η越趋近于$\frac{1}{\mathrm{B}}$。则其误差期望为：

$$\mathrm{E}[Err_2]=E[(\frac{\hat{f}}{B}-f_1)^2+(\frac{\hat{f}}{B}-f_2)^2+\cdots+(\frac{\hat{f}}{B}-f_B)^2]$$

$$=E[(\frac{f+X_1}{B}-\eta f)^2+\cdots+(\frac{f+X_B}{B}-f_B)^2]$$
$$=E[\eta^2 f^2+\sum_{i=2}^{B}f_i^2-\frac{f^2}{B}]+E[\frac{X^2}{B}]\le\frac{1}{B}((B-1)f^2+Var).....②$$

反之，考虑例先进的静态分解方案 HIO，他也结合 OUE 的扰动机制，则我们可以根据其子区间得到一个总体的误差$\mathrm{E}rr_1$的期望：

$$\mathrm{E}[Err_1]=\mathrm{E}[(\hat{f}_1-f_1)^2+(\hat{f}_2-f_2)^2+\cdots+(\hat{f}_B-f_B)^2]$$
$$=\mathrm{E}[(f_1+X_1-f_1)^2+\cdots+(f_B+X_B-f_B)^2]$$
$$=\mathrm{E}[B\cdot X^2]=B\cdot E[X^2]=B\cdot Var.....①$$

则我们总是让②<①，即我们总是让 AHEAD 方案的误差小于静态方案的误差，最后可以求得：

$$f<\sqrt{(B+1)Var}=\theta$$

我们让算得的常数值$\sqrt{(B+1)Var}$等于θ，则当$f<\sqrt{(B+1)Var}=\theta$的时候我们总是能够使得 AHEAD 的误差小于 HIO 静态方案的误差，此时θ就决定了我们是否应该继续进行划分。

参数 B 的选择不同于θ的选择，θ的选择是知道我们的树结构的，而 B 的计算是不知道树结构的前提下进行的。我们将误差$\mathrm{E}rr_3$记作非均匀分布的误差和噪声误差之和的总体误差，则其期望可以表示为：

$$E[Err_3]=E[(\frac{X}{B}+(f-\frac{f}{B}))^2]=\frac{c\sigma^2}{B}+(f-\frac{f}{B})^2$$

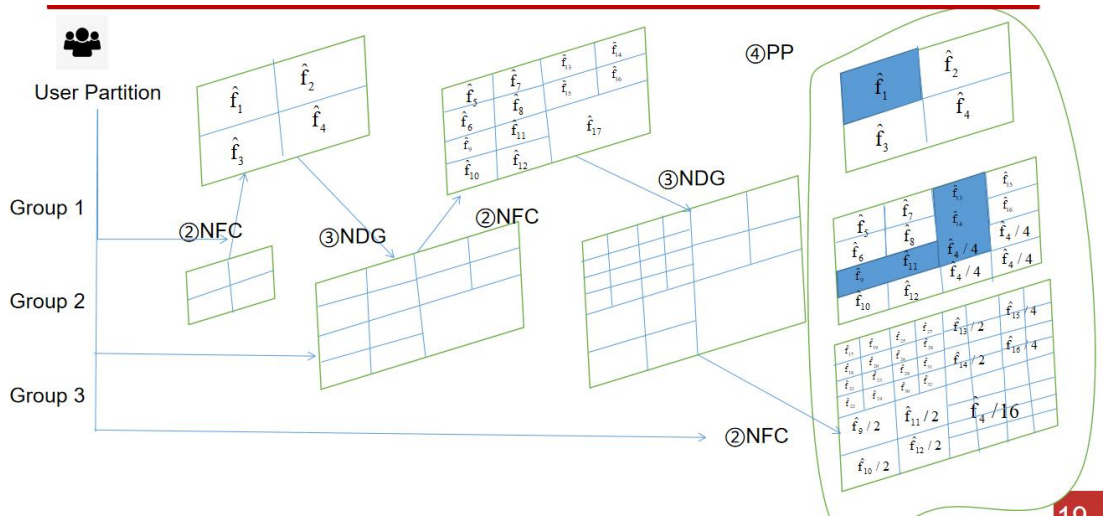
$$=\frac{c\sigma^2}{B}+(\frac{B-1}{B})^2(B+1)c\sigma^2$$

$$=(\sigma^2 In|D|)\frac{B+(B-1)^2(B+1)}{B^2 InB}$$

我们对最后求得的公式求导取极值可得 B=2.2 和 B=0.6。至于 B 的取值应该为什么，我们可以通过实验比较来得出。

4.4 扩展到高维查询

当一维查询时，区间是[1,|D|],而到了二维查询，则区间变为[1,|D|]X[1,|D|],可以形象的比作是 X，y 轴组成的坐标系，那么类比到三维查询可以是一个正方体方块。这时可能会有一个疑惑就是为什么值域的宽度都设为|D|,特征值的总体区间不可能都是|D|。对于这个问题我们可以添加一些空白噪声以满足每个特征值的值域都是|D|。那么用户分组数到了二维条件下则由 $c=\log_B|D|^2$ 决定。同时整个 AHEAD 的工作流程依旧是按照那 4 个工作流程进行。



对于高维查询条件下的工作流程和查询过程可以查看上图或查看附带的 PPT，能有一个更为直观的印象。

同时，AHEAD 的方案与 PrivTreee 是相当相近的，只不过 PrivTree 是作用于中心化差分隐私的。

5.实验结果

5.1 实验的设置我们可以参考下表：

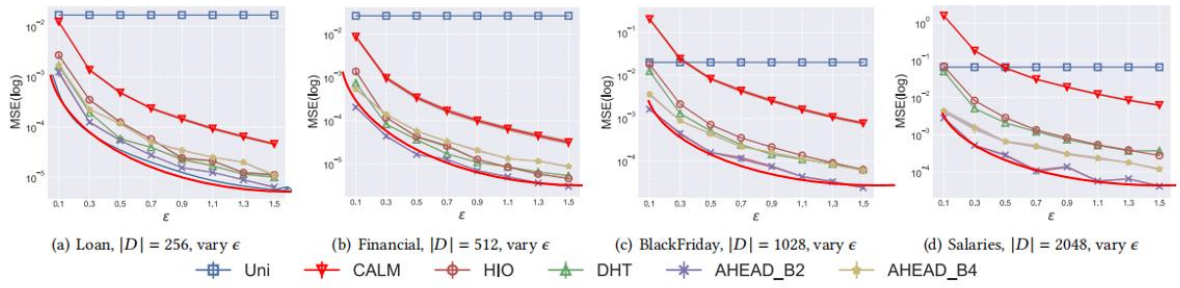
Dataset	Distribution	Scale	Field	Type
Salaries	–	148,654	employee salary	real
BlackFriday	–	537,577	shopping	real
Loan	–	2,260,668	online loan	real
Financial	–	6,362,620	fraud detection	synthetic
Cauchy	Cauchy	–	–	synthetic
Zipf	Zipf (power-law)	–	–	synthetic
Gaussian	Gaussian	–	–	synthetic
Laplacian	Laplacian	–	–	synthetic

在一维查询下我们比较不同的方法例如：HIO,DHT,CALM,Uni

在二维查询下我们比较不同的方法例如：CALM，HDG

评估的标准是采用：均方误差 MSE。

5.2 比较一维查询下各方案的性能。



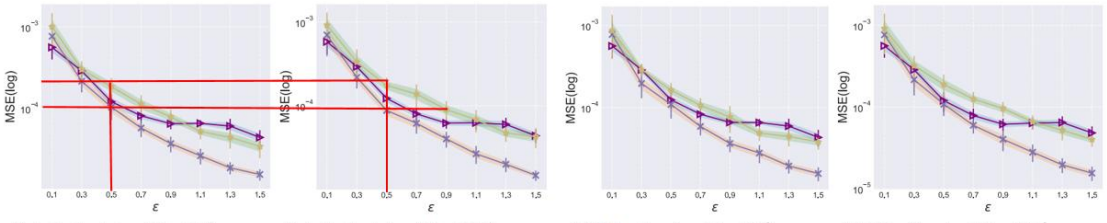
AHEAD_B2: 分支 B = 2　　AHEAD_B4: 分支 B = 4

Observation

- AHEAD在不同数据集下的MSE始终低于其他同类算法，这代表这更加精确的范围查询
- AHEAD对ε或是|D|具有良好的健壮性

从图中可以很仔细的分辨出，不管是什么特征下，什么规模的域|D|下 AHEAD 方案在 B 去 2 的情况下都能保持一个比其他方案都低的 MSE。

5.3 比较二维查询下各方案的性能。

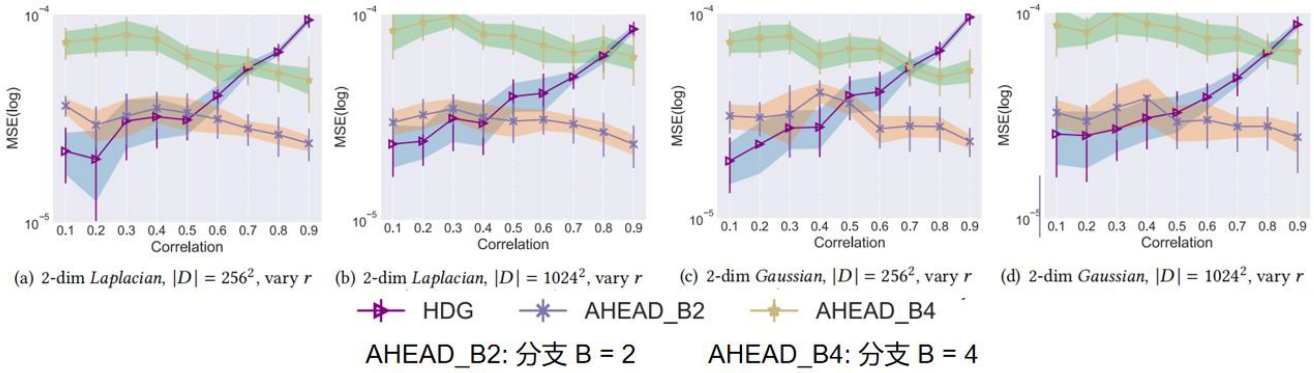


AHEAD_B2: 分支 B = 2　　AHEAD_B4: 分支 B = 4

Observation

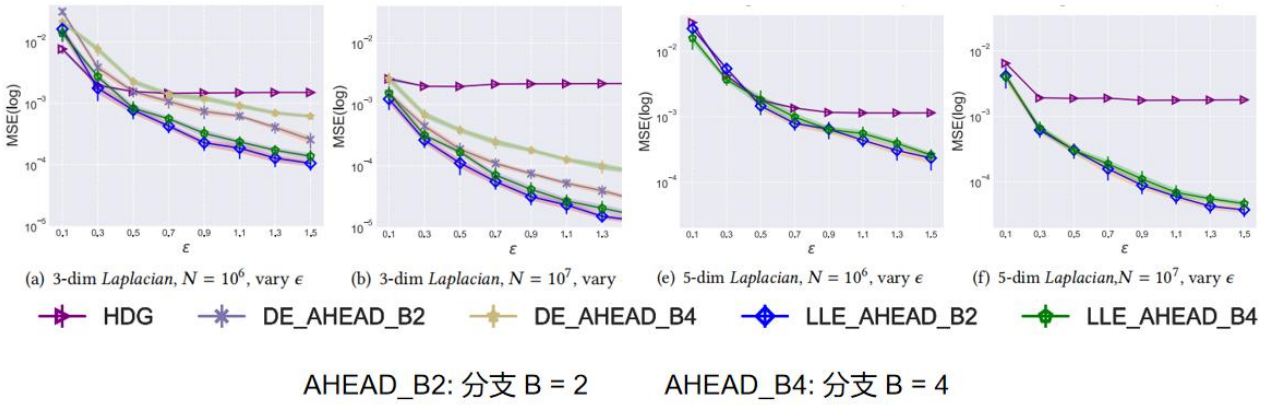
- B对AHEAD的影响很明显要大于|D|对AHEAD的影响
- 当B=2时，AHEAD对区间粒度的自适应分解的优势特性很明显得展示了出来

同时，对于 AHEAD 在 B=2、B=4 的条件下，不管是满足拉普拉斯机制还是高斯机制的合成数据集，AHEAD 有较好的健壮性，同时数据规模|D|的影响是不如 B 的。



AHEAD 在关系数据集上的健壮性是明显的，相较于静态方案 HIO,AHEAD 方案的 MSE 之中保持在几个指数级范围内，不会像 HIO 那样出现 MSE 上升的情况。

5.3 比较高维查询下各方案的性能。



Observation

- 在高维查询下，AHEAD更像是在自我竞争，HDG对于高ε的情况下由于粗粒度分解不再具备优势
- 对比LLE和DE：AHEAD使用LLE具有更低的MSE
- AHEAD的MSE对层次分解的深度不敏感，拥有更好的鲁棒性

这里只有 AHEAD 的方案在高维查询下才能依旧保持住较低的 MSE。

6.总结和展望

对于论文的局限：(1)对于高维范围查询，AHEAD 对用户记录的规模很敏感。AHEAD 需要将用户划分 2 次，即将 2-dim 树划分为不同的属性组合和不同的层。因此，当组用户数量较大时，AHEAD 需要足够的用户记录，以确保每个 2 暗层的频率估计的准确性。意思是θ的设置，在高维查询下需要更多的用户数据去进行预设置，这就导致了在高维查询下θ对于用户数据规模的依赖性。

(2) 与仅使用两个网格的 HDG 相比，即细粒度 1 维区间和稀疏粒度 2 维区间，AHEAD 生成具有不同粒度的 2 暗间隔来分解整个域。因此，在实践中，与 HDG 相比，AHEAD 需要更长的频率值搜索时间和更大的内存使用量。可以为每棵 2 维的树设计一个层融合策略来压缩树的高度。

(3) AHEAD 是一个完全动态的框架，它使用阈值来确定每个子域的划分。在域规模较大的情况下，每个子域都需要与阈值进行比较，这不可避免地增加了计算开销。因此可以采用“静态”和“动态”混合树结构。例如，前几个层可以使用静态框架，而其余的层可以利用阈值根据动态框架分解子域

(4) AHEAD 方法严格满足 LDP 保证，同时通过理论推导的参数获得了有利的效用性能。具体代码可以通过

https://github.com/linkzju/ccs21-AHEAD. 获取