Online Query Answering with Differential Privacy: a Greedy Approach using Bayesian Inference

ABSTRACT
Data privacy issues frequently and increasingly arise for data sharing and data analysis tasks. In this paper, we study the problem of online query answering under the rigorous differential privacy model. The existing interactive mechanisms for differential privacy can only support a limited number of queries before the accumulated cost of privacy reaches a certain bound. This limitation has greatly hindered their applicability, especially in the scenario where multiple users legitimately need to pose a large number of queries.

We propose a greedy algorithm using Bayesian statistical inference for online query answering, which minimizes the use of privacy budget when answering each query with a utility requirement. The key idea is to keep track of the query history and use Bayesian inference to answer a new query using previous query answers if the inference result satisfies the utility requirement; Otherwise the query is answered with the minimal privacy budget corresponding to its utility requirement. The Bayesian inference algorithm provides both optimal point estimation and optimal interval estimation given the observations of the query. We show that our approach maintains lower privacy budget usage, achieves a longer system lifetime and provides more accurate estimations than traditional approaches through extensive experiments on different real-life data sets.

1. INTRODUCTION

Data privacy issues frequently and increasingly arise for data sharing and data analysis tasks. Among all the privacy-preserving mechanisms, differential privacy has been widely accepted for its strong privacy guarantee [6, 7]. It requires that the outcome of any computations or queries is formally indistinguishable when run with and without any particular record in the dataset.

The existing interactive mechanisms [9, 5, 2, 17] of differential privacy can only support a limited number of queries under a certain bound of privacy budget. We define the privacy bound as overall privacy budget and such system bounded differentially private system. To answer a query, the mechanism allocates some privacy budget (also called privacy cost of the query) for it. Once the overall privacy budget is exhausted, the database has to be shut down or any further query would be rejected. This limitation has greatly hindered their applicability, especially in the scenario where multiple users legitimately need to pose a large number of queries.

Most recent work [16, 12, 22, 4] about differential privacy mainly focused on two settings of the problem: first, how to maximize utility given a bunch of “known” queries; second, how to boost accuracy of point estimation given a bunch of “known” query answers. While these approaches can provide high accuracy for the whole bunch of queries, a prerequisite question is how to allocate privacy budget to these queries. Because both the utility of released data and the life span of the system are related to the problem of privacy budget allocation, how to minimize the privacy cost or expand the lifetime of an online system becomes an important issue.

We formulate the question as an online resource allocation problem. Given the constraints of limited privacy budget as the resource and the utility requirements of each query, the objective function is to answer these queries with least privacy cost, or to answer as many queries as possible. To solve this problem, following assumptions are made in the first place:

First, we do not predict user queries, especially for multiple users posing a large number of queries. This assumption and problem is realistic in two ways: 1. Data may be used for many purposes. It’s difficult to know what tasks the data will be needed to tackle in the future; 2. Data may be demanded by multiple users. Each user may have different interest on the data. Therefore, data manager may have a set of queries issued by users, but knowing future queries is too difficult to be feasible. Otherwise, the matrix mechanism [16] can be used to derive the optimal query strategy.
and answers\(^1\).

Second, we assume different queries have different utility requirements. This assumption holds when the data is used by multiple users for multiple purposes. For example, adding a noise 20 to the value 100 may be acceptable for a user; adding the same noise 20 to a value 500 may be too big to be tolerable for another user or mission. Therefore, different users or different tasks may require different utility. Using the Bayesian inference technique in our model, we let users demand the credible interval and confidence level\([11, 1]\) to guarantee the utility for each query.

Having these assumptions, we propose a greedy algorithm to minimize the use of privacy budget when answering each query with its utility requirement. Because future queries are unpredictable, then only current minimization of privacy budget can be implemented for current system; Furthermore, the computational complexity is too high to achieve global minimization of privacy budget (shown in later sections).

The main process of answering a target query includes two parts: 1. How to estimate the answer of target query using query history; 2. How to allocate privacy budget if the utility of estimation does not satisfy requirement. The key of our investigation is the correlated queries\([16]\). A set of queries are correlated if one of them can be linearly represented by others. If so, it becomes possible that the answer can be statistically inferred without access to the differentially private query mechanism (so that no privacy cost is needed to answer this query) as long as the derived answer satisfies the required utility.

1.1 Contributions and main features

We summarize the contributions and main features as below.

1. We have developed an efficient query answering system with differential privacy. The system minimizes the privacy cost for each query. Therefore, it maintains lower privacy budget usage, can answer more queries and achieve a longer system lifetime.

2. The framework of our model is illustrated in Figure 1. The data manager maintains a history of issued queries and answers, which are returned from the differentially private query mechanism. For any user query with its utility requirement, we first estimate its answer from the query history using statistical inference. If the estimate satisfies the utility requirement, then the estimate would be returned as query answer. Otherwise, we invoke the query mechanism with minimal privacy cost to obtain answer from the differentially private query mechanism, and save the returned answer to the query history.

2.1 Recent researches\([12, 22, 4]\) about differential privacy mainly focus on the problem how to boost accuracy of point estimation given a bunch of “known” query answers. While these approaches can provide high accuracy for the whole bunch of queries, they may not achieve high utility for each query individually, especially when each query has different utility requirement.

Our system is driven by utility, which is demanded by users. Using the Bayesian inference technique, we let users demand the credible interval and confidence level\([11, 1]\), which are guaranteed in the returned answers. Furthermore, we also provide the optimal point estimation, which achieves minimal mean square error \(E[(\hat{\theta} - \theta)^2]\) where \(\theta\) is the true value and \(\hat{\theta}\) is the estimate.

3. In the field of statistics, differential privacy and statistics have been researched\([19, 20, 8, 10]\). It’s shown that accuracy of query answers can be boosted by statistical inference, like maximal likelihood estimator\((MLE)\)[19], least square\(\text{(LS)}\) solution\([12, 16]\) or the probabilistic inference\([20]\).

However, Bayesian estimator has not been adopted in the field of differential privacy. Note that the notion of differential privacy is initially defined in Bayesian probability. It’s proven in \([18]\) that to some extent differential privacy is equivalent to adversarial privacy, which is also a function of Bayesian probability. Therefore, differential privacy should be considered with Bayesian statistics, which is a theoretical way to understand and develop differential privacy.

To our knowledge, this is the first paper that achieves the optimal point estimation and interval estimation using Bayesian method. We formally derive the error of the estimate which allows us to reject the estimate if it does not satisfy the utility requirement. With the Bayesian method of this paper, further researches can be conducted in the future, like disclosure risk analysis, parameter estimation and other statistical inferences.

2. PRELIMINARIES AND DEFINITIONS

We use bold characters to denote vector or matrix, normal character to denote one row of the vector or matrix; subscript \(i\), \(j\) to denote the \(i\)th row, \(j\)th column of the vector or matrix; \(\theta, \theta, \bar{\theta}\) to denote the true answer, estimated answer and the noisy answer (with any noise) respectively; \(\mathcal{A}_2\) to denote the differentially private answer of Laplace mechanism. Note that \(\mathcal{A}_2\) is a special case of \(\bar{\theta}\).

We will introduce our notations as we introduce the definitions. For reference purposes, Appendix 10.1 gives an overview of the key notations used in this paper.

2.1 Differential privacy and Laplace mechanism

**Definition 2.1** (\(\alpha\)-Differential Privacy \([5]\)). A data access mechanism \(\mathcal{A}\) satisfies \(\alpha\)-differential privacy\(^2\) if for any neighboring databases \(D_1\) and \(D_2\), for any query function:

\[^1\text{It’s the least square solution.}\]

\[^2\text{Our definition is consistent with the unbounded differential privacy in\([14]\).}\]
tion Q, r \subseteq \text{Range}(Q)\textsuperscript{3}, A_Q(D) \text{ is the mechanism to return an answer to query } Q(D),
\[
\sup_{r \subseteq \text{Range}(Q)} \frac{\Pr[A_Q(D) = r | D = D_1]}{\Pr[A_Q(D) = r | D = D_2]} \leq e^\alpha \tag{1}
\]
α is also called privacy budget which is discussed in section 2.3.

Note that it’s implicit in the definition that the privacy parameter α can be made public.

**Definition 2.2 (Sensitivity).** For arbitrary neighboring databases D\textsubscript{1} and D\textsubscript{2}, the sensitivity of a query Q is the maximum difference between the query results of D\textsubscript{1} and D\textsubscript{2},
\[
S_Q = \max(Q(D_1) - Q(D_2))
\]

For example, for these two queries:

- **Q1:** If one is interested in the population size of 20 ~ 30 old and income \(\geq 20\)K, then we can pose this query: select count(*) from data where 20 \(\leq\) age \(\leq\) 30 and income \(\geq 20\)K ...

- **Q2:** If we know people with income \(< 10\)K should pay 100 dollars each year, people with income \(10\)K to \(< 20\)K pay 1000 dollars, people with income \(\geq 20\)K pay 2000 dollars, and from the total tax revenue 20K dollars are invested for education each year, then to calculate the total tax this query can be issued: select 100*count() where income \(< 10\)K + 1000*count() where income \(\geq 20\)K - 50K

Q1 has sensitivity 1 because any change of 1 record can only affect the result by 1 at most; while Q2 has sensitivity 2000 because any change of 1 record can affect the result by 2000 at most. Note that the constant 50K does not affect the sensitivity of Q2 because it won’t vary for any record change. This feature is important for privacy budget composition discussed in section 3.1.

**Definition 2.3 (Laplace mechanism).** Laplace mechanism\textsuperscript{9} is such a data access mechanism that given a user query Q whose true answer is θ, the returned answer θ is
\[
A_Q(D) = \theta + \tilde{N}(\alpha/S)
\]  
(2)

\(\tilde{N}(\alpha/S)\) is a random noise of Laplace distribution. If S = 1, the noise distribution is like equation (3).
\[
\Pr(x, \alpha) = \frac{\alpha}{2} \exp(-\alpha|x|);
\]  
(3)

For S ≠ 1, replace α with \(\alpha/S\) in equation (3).

**Theorem 2.1.** Laplace mechanism achieves \(\alpha\)-differential privacy, that means adding \(\tilde{N}(\alpha/S_Q)\) to \(Q(D)\) guarantees \(\alpha\)-differential privacy.

\(\text{Range}(Q)\) is the domain of the differentially private answer of Q.

### 2.2 Utility

In contrast to point estimation, on which recent researches mainly focused to provide a single value with high accuracy, interval estimation specifies instead a range, like the \((\epsilon, \delta)\)-usefulness\textsuperscript{3}, within which the parameter is estimated to lie. We introduce \((\epsilon, \delta)\)-usefulness first, then show that it’s actually a special case of credible interval, which will be used in this paper.

For a given query Q, the true answer of Q is θ. From the query system, users obtain some observations\textsuperscript{4} \(\tilde{\theta}_1, \tilde{\theta}_2, \ldots\), represented as a vector \(\tilde{\Theta}\). All the observations of θ has the form:
\[
\tilde{\theta} = \theta + \tilde{N}
\]  
(4)

Different from equation (2), here \(\tilde{N}\) can be any noise\textsuperscript{5}. With these observations, an estimate \(\hat{\theta}\) about θ can be calculated.

**Definition 2.4 \((\epsilon, \delta)\)-usefulness\textsuperscript{3}).** A query answering mechanism is \((\epsilon, \delta)\)-useful for query Q if \(\Pr(\vert\hat{\theta} - \theta\vert \leq \epsilon) \geq 1 - \delta\).

**Definition 2.5 (credible interval\textsuperscript{11}).** Credible interval with confidence level 1 - \(\delta\) is a range \([L(\tilde{\Theta}), U(\tilde{\Theta})]\) with the property:
\[
\Pr(L(\tilde{\Theta}) \leq \theta \leq U(\tilde{\Theta})) \geq 1 - \delta
\]  
(5)

In this paper, we denote \(2\epsilon = U(\tilde{\Theta}) - L(\tilde{\Theta})\) the length of credible interval.

Note when θ is the midpoint of \(L(\tilde{\Theta})\) and \(U(\tilde{\Theta})\), these two definitions are the same and have equal \(\epsilon\). So \((\epsilon, \delta)\)-usefulness is a special case of credible interval. But in general, we don’t assume θ is the midpoint, which means the posterior probability function \textsuperscript{6} around θ may not be symmetric. In this paper, we adopt credible interval as the utility metric for users. The utility requirement is the pair of parameters \(\epsilon\) and \(\delta\), which describe the length of credible interval and confidence level. In addition, we also return the optimal point estimation \(\hat{\theta}\) which achieves minimal mean square error.

### 2.3 Privacy cost

For an online query answering system, it’s important to know the security status of current state. Although it’s widely accepted that query answering should not exceed the predefined privacy budget, how to calculate the used privacy cost is not very clear besides the composition theorem\textsuperscript{17}. We define local privacy cost and system privacy cost to measure the security of a differentially private system. Detailed computation of privacy cost is discussed in section 3.3.2.

Here we briefly introduce the composition theorem.

**Theorem 2.2 (Sequential Composition \textsuperscript{17}).** Let \(M_i\) each provide \(\alpha_i\)-differential privacy. The sequence of \(M_i\) provides \(\sum \alpha_i\)-differential privacy.

\textsuperscript{4}Also called samples, or linear representations in this paper
\textsuperscript{5}The noise may also be composed Laplace noise as shown in Section 3.2.1.
\textsuperscript{6}For convenience and clearance, probability function only has two meanings in this paper: probability density function for continuous variables and probability mass function for discrete variables.
3. GREEDY ONLINE ALGORITHM

This section explains the query-answering mechanism. As shown in Figure 2, for any query \( Q \), we need a utility requirement in \((\epsilon, \delta)\). At the beginning, we know the query history \( H \), perturbed answers \( y \) providing \( A \)-differential privacy\(^7\) and a target query \( Q \). First, (1) we find the observations(linear representations), denoted as \( \hat{\Theta} \) (section 3.2.1); (2) An estimate \( \hat{\theta} \) and posterior probability function can be derived from \( \Theta \) to satisfy minimal mean square error(section 3.2.2); (3) Next we can compute the error of \( \hat{\theta} \) and the credible interval(section 3.2.3); if the utility is better than required, then output the estimate; (4) Else we generate the necessary privacy cost(section 3.3.1). (5) Then we evaluate the system privacy cost(section 3.3.2); (6) If answering this query will not overspend the predefined privacy budget, then invoke the query mechanism to return a differentially private answer \( \hat{\theta} \); else the system cannot provide an answer for \( Q \) to satisfy the required utility.

In summary, the input is query \( Q \) and utility requirement \((\epsilon, \delta)\). The output can be one of the following: the estimated answer \( \hat{\theta} \) and credible interval(probably better than required), or the differentially private answer \( \hat{\theta} \) and credible interval. This query answering flow is summarized in Algorithm 1.

We postpone the technical details of probability function computation in section 3.2.2 to section 4.

3.1 Data model and example

In this online model, the data manager needs to know the following information: query history matrix \( H \), returned answers \( y \) of \( H \) and used privacy cost \( A \) for \( y \).

3.1.1 Data

We use a “data cube” to represent the data space \( D \). We denote any sub-cube that is not divided by any more dimensions by “cell”, meaning it’s the “smallest” sub-cube. For any multi-dimensional data cube, we can map data into one-dimensional vector, denoted by \( x \). For example, we use the original data shown in Figure 3 with attributes age and income. The domain value of age is 20~30, 30~40 and 40~50; the domain value of income is 0~10K, 10K~20K and >20K. We can represent the data in a multi-dimensional OLAP (Online Analytical Processing) data cube. Figure 3 shows the data represented in a two-dimensional count cube or histogram, in which each cell represents the population count corresponding to the age and income values. We can also represent the cells in the original data cube by a vector \( x \) as equation (6).

Algorithm 1 Online query-answering framework

Require: \( Q, \epsilon, \delta \)
if \( \exists \) linear representation of \( Q \) then
(1) Find observations \( \Theta \) by Algorithm 2;
(2) Estimate \( \hat{\theta} \) by Algorithm 3;
(3) Compute the utility of \( \hat{\theta} \) by Algorithm 4;
end if
if utility is better than required then
return \( \hat{\theta}, L(\hat{\theta}), U(\hat{\theta}) \);
else
Predict necessary privacy cost \( \alpha \) by Theorem 3.9;
if answering this query won’t overspend budget then
Return direct query answer \( \hat{\theta} \);
\( L(\hat{\theta}) \leftarrow \hat{\theta} - \epsilon; \)
\( U(\hat{\theta}) \leftarrow \hat{\theta} + \epsilon; \)
Update \( H, A, B, y \);
else
No answer can be returned;
end if
end if
return \( \hat{\theta}, L(\hat{\theta}), U(\hat{\theta}) \)

Figure 3: Example original data represented in a relational table (left) and a 2-dimensional count cube (right)

\[ x = [\begin{array}{cccccccc} 10 & 21 & 37 & 20 & 0 & 0 & 53 & 0 \end{array}]^T \] (6)

3.1.2 Query

Then for any linear counting queries, the query answer is the product of two vectors: query vector and data vector \( x \). For the query “Q1” in section 2.1, it asks the count of \( x_1 \). Then it can be written in a vector as equation (7).

\[ Q_1 = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] \] (7)

\[ Q_2 = [2k \ 2k \ 2k \ 1k \ 1k \ 1k \ 100 \ 100 \ 100] \] (8)

Figure 4 shows the value of Q1. The original answer Q1·x is the product of query vector Q1 and data vector x.
Note that the constant 50K in query Q2 affects neither the S of Q2 nor the accuracy of returned answer. For discussion simplicity, we ignore all the constant in all queries. Then Q2 can be written as equation (8).

\[ N = 0 \text{ if } \alpha = 0 \]

Also, we write the above equations in matrix form. Let the sensitivity of each query vector may not be necessarily the same. For example, \( S_{Q1} = 1 \) while \( S_{Q2} = 2K \). Therefore, we need to compute the sensitivity of each query by Lemma 3.1.

**Lemma 3.1.** For a linear counting query \( Q \), the sensitivity of \( Q \) is \( \max(\text{abs}(Q)) \).

We use the function \( \text{abs()} \) to denote the absolute value of a vector, meaning that \( \forall j, \text{abs}(Q)_j = |Q_j| \). \( \max(\text{abs}(Q)) \) means the maximal value of \( \text{abs}(Q) \).

If many queries have been issued already, then we can write the above equations in matrix form. Let \( y \), \( H \), \( A \) and \( S \) be the matrix form of \( A_Q \), \( Q \), \( \alpha \) and \( S \) respectively. According to Lemma 3.1, \( S = \max_{\text{row}}(\text{abs}(H)) \) where \( \max_{\text{row}}(\text{abs}(H)) \) is a column vector whose item is the maximal value of each row in \( \text{abs}(H) \).

Equations (11, 12) summarize the relationship of \( y, H, x, N \) and \( A \).

\[ y = Hx + \bar{N}(A/S) \]  
\[ S = \max_{\text{row}}(\text{abs}(H)) \]  

where operator \(./\) means \( \forall i, |A_i/S_i| = A_i/S_i \).

### 3.1.4 Running example

Suppose we have a query history matrix \( H \) as equation (13), \( x \) as equation (6) and the privacy parameter \( A \) used in all query vector shown in equation (14).

\[ \bar{A}_Q = 50K \text{ would be the same with } \bar{A}_Q = Q2 \cdot x - 50K + \bar{N}(\alpha/S_{Q2}) \text{, so we ignore the constant in } Q2. \]

\[ y = Hx + \bar{N}(A/S) \]

\[ S = \max_{\text{row}}(\text{abs}(H)) \]

\[ \text{Let } \bar{A}_Q = 50K, \bar{A}_Q = Q2 \cdot x - 50K + \bar{N}(\alpha/S_{Q2}), \text{so we ignore the constant in } Q2. \]

\[ H = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 4 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix} \]

\[ A = [0.05 \ 0.1 \ 0.05 \ 0.1 \ 0.05 \ 0.05 \ 0.05]^T \]

Then according to Lemma 3.1, \( S \) can be derived in equation (15).

\[ S = \begin{bmatrix} 1 & 1 & 4 & 2 & 1 & 1 & 2 \end{bmatrix}^T \]

Therefore, the relationship of \( y, H, A \) and \( S \) in equation (11) can be illustrated in equation (16).

\[ \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 4 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix} \times \bar{N} = \begin{bmatrix} 0.05/1 \\ 0.1/1 \\ 0.05/4 \\ 0.1/2 \\ 0.05/1 \\ 0.05/3 \\ 0.05/2 \end{bmatrix} \]

With above \( y, H, A \) and \( S \), assume the target query \( Q \) is \( x_5 + x_6 \) with utility requirement as (15, 0.2). Then

\[ Q = [0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0] \]  

### 3.2 Statistical inference

#### 3.2.1 Linear representation of a query

To find the linear representation of a query \( Q \), it means to derive the answer of \( Qx \) from \( H \) and \( y \). Simply put, whether the equation \( \text{TH} = Q \) has a solution for \( T \) can solve this. Actually this is the LS[16] solution which allocate different weight for each \( y \).

Why don’t we just use the LS[least square] solution to solve the linear equation? Generally speaking, we have two goals: first, we want to return an credible interval; second, we want to find the point estimation to achieve minimal mean square error. However, LS solution only provides one point estimation with no probability feature nor mean square error. And we show in the experiment that LS solution is insensitive to privacy cost \( \alpha \) and utility requirement \( (\epsilon, \delta) \). Therefore, it cannot guarantee the required utility.

We need to find independent observations from query history \( H \). Equations (18) and (19) are our goal.

\[ \bar{Q} = Qx + \bar{N}_i \]

\[ \bar{N}_i = T_i\bar{N}_i \]  

If multiple observations can be found, it’s an overdetermined linear problem. However, finding all the possible linear representations is NP-hard. Even if we assume the rows of \( H \) are linearly independent, there are \( k^n \) \text{ sets of solutions where } \( H \in \mathbb{R}^{m \times n} \).

Theorem 3.1 proves the computational complexity, which justify the greedy algorithm: if finding the optimal estimate for one query is NP-hard, then finding optimal estimates for a bunch of queries are also NP-hard.

**Theorem 3.1.** Finding all the possible linear representations of query \( Q \) from query history \( H \) is NP-hard.
Proof. This problem can be transformed to partition problem. Let \( H \) be the set of candidates, the criterion of each partition is to linearly solve the \( Q \) by the elements in the partition. Because partition problem is NP-hard, this problem is NP-hard. \( \square \)

For computation simplicity, we use a greedy approach to derive the linear representations. From the first row to the last row of \( H \), we solve \( TH = Q \). If a \( T_k \) can be computed, then add \( T_k \) to the observation \( \Theta \). Next we remove the used rows of \( H \), which are the rows involved in the computation of \( TH = Q \). In another word, if \( T_{kj} \neq 0 \), then \( H_j \) is used. Next we find \( T_{k+1} \) similarly. Algorithm 2 summarizes the process. \( H(1 : i,:) \) means the first \( i \) rows of \( H \).

**Algorithm 2** Search observations \( \Theta \)

Require: \( Q, H, y, A \)

\[
[m, n] = \text{size}(H); \\
k \leftarrow 1; \\
\text{for } (i=1; i \leq m; i = i + 1) \text{ do} \\
\text{if } T \ast H(1 : i,:) = Q \text{ has solution for } T \text{ then} \\
T_k = Q/H_{\text{copysym}}(1 : i,:); \\
\text{Remove involved rows from } H:\n\forall j, \\
\text{if } T_{kj} \neq 0 \text{ then} \\
\text{Replace } H_j \text{ with row } 0; \\
\text{end if} \\
k \leftarrow k + 1; \\
\text{end if} \\
\text{end for} \\
\text{return } \Theta, T
\]

**Theorem 3.2.** Let \( \hat{\Theta} \) be the observations of target query \( Q \), then

\[
\hat{\Theta} = Qx + T\hat{N}
\]  
(20)

**Proof.**

\[
T_y = T(THx + \hat{N}) = THx + T\hat{N} = Qx + T\hat{N}
\]  
(21)

It’s the sum of original answer \( Qx \) and a composed noise. So \( \hat{\Theta} = Qx + T\hat{N} \). \( \square \)

Let’s consider the running example \( Q \) in equation (17). First \( i = 1 \). When \( i \) increases to 4, \( T \ast H(1 : 4,:) = Q \) has solution for \( T_1, T_1 = [0 \ 0 \ 1/2 \ -1/2 \ 0 \ 0 \ 0 \ 0] \). Denote the first observation as \( \hat{\theta}_1, \hat{\theta}_1 = T_1y \) in equation (22). The used rows of \( H \) is shown in the yellow rows in equation (23).

\[
\hat{\theta}_1 = T_1y = y_3/2 - y_4/2 \\
= (H_3x + \hat{N}(A_3/S_3))/2 - (H_4x + \hat{N}(A_4/S_4))/2 \\
= Qx + \left( \frac{\hat{N}(A_3/S_3) - \hat{N}(A_4/S_4)}{2} \right)
\]  
(22)

Finally, \( T, \Theta \) are shown in the yellow part of equation (31) and equation 32.

\[
TH = \begin{bmatrix} 0 & 0 & 1/2 & -1/2 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & -1 & 1/2 & 0 \end{bmatrix} \\
H = \begin{bmatrix} Q \\
Q \\
Q \end{bmatrix}
\]  
(31)

\[
\Theta = \theta + \begin{bmatrix} \frac{\hat{N}(0.05/4) - \hat{N}(0.1/2)}{2} \\
\frac{\hat{N}(0.1/1) - \hat{N}(0.1/1)}{2} \\
\frac{\hat{N}(0.05/1) - \hat{N}(0.05/1) + \hat{N}(0.05/2)}{2} \end{bmatrix} \begin{bmatrix} -16.7 \\
1.1 \\
-11.5 \end{bmatrix}
\]  
(32)

**3.2.2 Query result estimation**

We have all observations of the query in this step. To estimate the answer, we formulate the problem as a Bayesian estimation problem to seek \( \text{argmax}_{\hat{\theta}}(E(\hat{\theta} - \theta)^2) \) where operator \( \text{argmax}_{\hat{\theta}} \) means to find the argument \( \hat{\theta} \) to achieve the minimal value of \( E((\hat{\theta} - \theta)^2) \). \( \hat{\theta} \) is the true value of query and \( \theta \) is the estimate of the query. According to the MMSE estimator[1], our goal is to obtain

\[
\hat{\theta} = \mathbb{E}(\theta|\hat{\Theta}) = \mathbb{E}(\theta|\hat{\theta}_1, \hat{\theta}_2, \cdots, \hat{\theta}_n).
\]  
(33)

So the next question is how to compute the expectation value given the observed answers? There are several ways to solve the problem. Because we need not only the estimate but also the credible interval, we should calculate the probability function of the estimate \( \hat{\theta} \). Theorem 3.3 gives a way to compute the probability function iteratively.

**Theorem 3.3.** Let \( Pr_i(\theta) = Pr(\theta|\hat{\theta}_1, \hat{\theta}_2, \cdots, \hat{\theta}_i) \),

\[
Pr_i(\theta) = \frac{Pr(\hat{\theta}_i|\theta)Pr(\hat{\theta}_i)}{Pr(\hat{\theta}_i)}
\]  
(34)
Therefore, we can calculate $Pr_n(\theta)$ from $\tilde{\theta}_1$ to $\tilde{\theta}_n$ iteratively. Note that when $i = 1$, we assume $Pr_0(\theta)$ is uniformly distributed[11], meaning that $Pr_1(\theta) = Pr(\tilde{\theta}_1|\theta)$. So the next question is how to derive $Pr(\tilde{\theta}_i|\theta)$. Theorem 3.4 shows a way to solve this.

In the rest of this section, we need to process many probability functions. So we denote $Pr_z(x)$ a probability function of $z$, $Pr_i(z)$ a probability function of $z$ with input variable $x$. For example, if $Pr_N(x)$ is the probability function of $N(\alpha)$ whose probability function is shown in equation (3), $Pr_N(z)$ is shown in the following equation.

$$Pr_N(z) = \frac{\alpha}{2} \exp(-\alpha|z|);$$

(35)

**Theorem 3.4.** Let $Pr_N(.)$ be the probability function of $\tilde{N}$ in Equation (18), meaning $\tilde{N}$ is the composed noise of each observation, then

$$Pr(\tilde{\theta}_i|\theta) = Pr(\tilde{\theta}_i - \theta)$$

(36)

**Proof.**

$$Pr(\tilde{\theta}_i|\theta) = Pr(\tilde{\theta}_i) = \theta + \tilde{N}_i|\theta) = Pr(\tilde{N}_i) = \tilde{\theta}_i - \theta|\theta$$

(37)

Note that the shape of $Pr(\tilde{\theta}_i|\theta)$ doesn’t change with $\theta$, therefore,

$$Pr(\tilde{N}_i - \tilde{\theta}_i - \theta|\theta) = Pr(\tilde{\theta}_i - \theta)$$

(38)

$\square$

**Theorem 3.5.** Equations (39, 40) are the Bayes estimator achieving MMSE.

$$Pr_n(\theta|\tilde{\theta}) = \prod_{i} Pr_N(\tilde{\theta}_i - \theta)$$

(39)

$$\hat{\theta} = \int \theta \prod_{i} Pr_N(\tilde{\theta}_i - \theta) d\theta$$

(40)

where $Pr(\tilde{\theta}_i) = \int Pr(\tilde{\theta}_i|\theta)d\theta$.

Given the above theorems, we will be capable to derive the Bayes estimate to achieve MMSE as long as we know the probability function of $Pr_N(.)$ in equation (18). Note that $\tilde{N}_i$ is a composed noise which may not be Laplace distributed. For example, in equation (32), $\tilde{N}_i = (\tilde{N}(0.05/4) - \tilde{N}(0.1/2))$ is a linear result of Laplace noises. Due to the computational complexity, we postpone the technical details to section 4. Here we assume we have the result of $Pr_N(.)$.

Then by Theorem 3.4, $Pr(\tilde{\theta}_i|\theta)$ can be derived; by Theorem 3.3, $Pr_i(\theta)$ can be derived iteratively; by Theorem 4 the $\hat{\theta}$ can be derived. Algorithm 3 summarizes the estimation process.

In our example, $T$ and $\tilde{\Theta}$ are shown in equations (31, 32). There are 3 observations. So $Pr(\tilde{\theta}_1|\theta)$, $Pr(\tilde{\theta}_2|\theta)$ and $Pr(\tilde{\theta}_3|\theta)$ can be calculated. The probability function curves are shown in Figure 5. And the probability function of $Pr_n(\theta|\tilde{\Theta})$ is shown in the solid line. By equation (33), we obtain $\hat{\theta} = 1.72$.

![Figure 5: Query answering example](image)

### 3.2.3 Credible interval and error quantification

In this step, we have the posterior probability function, the question is how to compute the credible interval and the error of our MMSE point estimation.

For credible interval, we formulate the problem as following: given the confidence level $1 - \delta$, how to find the narrowest interval containing probability $1 - \delta$? Many classical solutions can be found in [11, 1, 15]. To reduce the computational complexity, we assume the credible interval will contain the point estimation $\hat{\theta}$. Then by integrating the probability mass(density) to $1 - \delta$, we can find the credible interval. Algorithm 4 summarizes the process, which uses probability function in discrete domain.

**Algorithm 4 credible interval**

**Require:** $\hat{\theta}$, probability function: $Pr_0(\theta|\tilde{\Theta})$, $\delta$

1. $a \leftarrow \hat{\theta}$; $b \leftarrow 0$; $P_m \leftarrow 0$;
2. **while** $P_m < 1 - \delta$ **do**
   **if** $Pr_0(a) \geq Pr_0(a + b)$ **then**
   $P_m \leftarrow P_m + Pr_0(a)$; $a \leftarrow a - 1$; $b \leftarrow b + 1$;
   **else**
   $P_m \leftarrow P_m + Pr_0(a + b)$; $b \leftarrow b + 1$;
   **end if**
3. $L(\theta) \leftarrow a$; $U(\tilde{\Theta}) \leftarrow b$.
**return** $L(\theta)$, $U(\tilde{\Theta})$.

In our example, recall that the required range size is $15^\circ 2 = 30$ with probability 80%. However, the credible interval size is $12 - (-22) = 34$. Thus the error is bigger than required. Unfortunately we need to invoke the differentially private query mechanism, which is discussed in section 3.3. Following theorems show the error of the MMSE point estimation. Theorem 3.6 shows the unbiasedness. Theorem

![Algorithm 3 Bayes estimation to achieve MMSE](image)

**Require:** $\tilde{\theta}$, $T$, $y$

1. $[m, n] = size(\tilde{\Theta})$;
2. for $(i = 1; i < n; i = i + 1)$ do
   **Compute** $Pr_n(i)$ by Algorithm ?? or 5;
   **Compute** $Pr(\tilde{\theta}_i|\theta)$ by Theorem 3.4;
3. **end for**
4. Derive $\hat{\theta}$ by equation (40), $Pr_0(\theta|\tilde{\Theta})$ by equation (39);
5. **return** $\hat{\theta}$, $Pr_0(\theta|\tilde{\Theta})$.
3.7 and 3.8 give the absolute error and mean square error.

**Theorem 3.6.** MMSE estimator is unbiased.

**Proof.** \( E[\hat{\theta}] = E[\theta|\hat{\theta}] = \theta \) □

**Lemma 3.2.** Denote \(|\hat{\theta} - \theta|\) as \( \gamma \), then

\[
Pr_{\gamma}(\gamma) = 2 Pr_{\theta}(\theta + \gamma) = 2 Pr_{\theta}(\theta - \gamma) \tag{41}
\]

**Proof.** \( Pr_{\gamma}(\gamma) = Pr_{\gamma}([\theta - \theta]) = Pr_{\theta}(\theta - \theta) + Pr_{\theta}(\hat{\theta} + \theta) = 2 Pr_{\theta}(\theta + \gamma) = 2 Pr_{\theta}(\theta - \gamma) \) □

**Theorem 3.7.** The absolute error of our estimator is

\[
E[\gamma] = 2 \int_{-\infty}^{\theta} (\theta - \gamma) Pr_{\theta}(\theta) d\theta = 2 \int_{\theta}^{\infty} (\theta - \theta) Pr_{\theta}(\theta) d\theta \tag{42}
\]

**Proof.** Using equation (41) can get the direct result. □

**Theorem 3.8.** MSE of our estimator is

\[
MSE(\hat{\theta}) = 2 \int_{0}^{\infty} \gamma^2 \prod_{i} \frac{Pr_{\gamma}(\hat{\theta}_i - \theta - \gamma)}{Pr_{\hat{\theta}_i}(\hat{\theta}_i)} d\gamma \tag{43}
\]

**Proof.**

\[
MSE(\hat{\theta}) = E[\gamma^2] = \int_{0}^{\infty} \gamma^2 Pr_{\gamma}(\gamma) d\gamma \tag{44}
\]

From equations (41) and (39), we can derive equation (43). □

### 3.3 Differentially private query mechanism

Only when the utility of estimate does not meet the requirement, we need to invoke the differentially private query mechanism, return \( A(D) \) and interval \( |A(D) - \epsilon, A(D) + \epsilon| \) to user, add the query \( Q \) to query history \( H \), noisy answer \( A(D) \) to \( y \) and budgeted \( \alpha \) to \( A \).

#### 3.3.1 Budgeted query answering

The first step is to decide how much necessary privacy cost should be allocated. Theorem shows the solution which spends \( S_{Q} \) privacy cost.

**Theorem 3.9.** Given the utility requirement \( \epsilon, \delta \) of a query \( Q \), it’s enough to allocate \( \alpha = S_{Q} \frac{\epsilon}{\delta} \).

**Proof.** Let \( A_{Q}(D) = Q(D) + \bar{N}(\alpha/S_{Q}) \). Because probability function of Laplace distribution is symmetric, in definition 2.5, \( |A(D) - \epsilon, A(D) + \epsilon| \) gives the most probability mass.

\[
Pr(-\epsilon \leq A_{Q}(D) - Q(D) \leq \epsilon) = Pr(-\epsilon \leq \bar{N} \leq \epsilon) = \int_{-\epsilon}^{\epsilon} \frac{\alpha/S_{Q}}{2} \exp(-\alpha/S_{Q}|x|) dx \]

\[
= 2 \int_{-\epsilon}^{\epsilon} \frac{\alpha/S_{Q}}{2} \exp(\alpha/S_{Q}x) dx \]

\[
= 1 - \exp(-\frac{\alpha \epsilon}{S_{Q}}) \geq 1 - \delta \tag{45}
\]

So we have \( \alpha \leq S_{Q} \frac{\epsilon}{\delta} \). □

In our example, we obtain that \( \alpha = 0.108 \) would be enough, then we assume \( A_{s} = 0.108 \) and \( H_{s} = Q \).

#### 3.3.2 Privacy cost evaluation

Next we solve the privacy budget cost of the system to test whether the privacy budget cost is within privacy budget in a bounded differentially private system if answering the query \( Q \). If no, any answer can be returned to user for security reason; otherwise, we return the differentially private answer from Laplace mechanism using the derived necessary privacy cost and update the system parameters \( H, A \) and \( y \) for the next coming query.

In this paper, we assume the cells of the data are independent or only have negative correlations [18]. Generally speaking, this means adversaries cannot infer the count answer of a cell from any other cells, specifically, cells are independent if they do not have any correlation at all; cells are negative correlated\(^9\) if the inference result is within the previous knowledge of the adversary. So the inference from correlation does not help.

With this assumption, we state the system privacy cost in Theorem 3.10.

**Theorem 3.10.** Given current differentially private system with query history, denoted by \( H \) and privacy cost of all query, denoted by \( A \), the system privacy cost, denoted by \( \tilde{a} \), can be derived from equation (46), (47).

\[
\tilde{a} = \max(B) \tag{46}
\]

\[
B = \text{Sum}_{row}[\text{diag}(A/S_{abs}(H))] \tag{47}
\]

where \( B \) is vector of used privacy budget of each cell, function \( \text{diag()} \) transforms vector \( A \) to a matrix with each of element of \( A \) in diagonal, \( \text{Sum}_{row} \) means the sum of rows.

**Proof.** First, we prove for each query \( H_{i} \) in \( H \), the privacy budget cost of each cell \( x_{ij} \) is \( A_{ij}/S_{i} \ast \text{abs}(H_{i}) \); then add up each cell and each row, we get the result.

For any query \( Q \), the differentially private answer is returned as \( A_{Q} \). Assume an adversary knows all the records of the data set except one record in cell \( x_{ij} \), we define the local privacy cost for cell \( x_{ij} \) as \( \alpha_{ij} \) if \( A_{Q} \) is \( \alpha_{ij} \)-differentially private where

\[
\exp(\alpha_{ij}) = \sup_{r \in \text{Range}(Q)} \frac{Pr[A_{Q}(D) = r|D = D_{1}]}{Pr[A_{Q}(D) = r|D = D_{2}]} \tag{48}
\]

where \( D_{1} \) and \( D_{2} \) are neighboring databases the same as definition 2.1. Next we compute the local privacy cost of each cell for each query.

For each query \( H_{i} \), the Laplace noise is \( \bar{N}(A_{i}/S_{i}) \) according to equation (2). By lemma 3.1 and equation (12), it is \( \bar{N}(A_{i}/S_{i}) \). For the cell \( x_{ij} \), assume an adversary knows all but one record in \( x_{ij} \), then the count numbers of other cells become constant in this query. Thus this query becomes \( y_{i} = H_{i}x_{ij} + \bar{N}(A_{i}/S_{i}) + \text{constant} \). The \( S \) of this query is \( H_{i} \) and the noise of this query is \( \bar{N}(A_{i}/S_{i}) \). By Theorem 2.1, it guarantees \( H_{i}A_{i}/S_{i} \)-differential privacy. Then the budget cost of \( x_{ij} \) is \( H_{i}A_{ij}/S_{i} \).

For example, in equation (13), \( H_{i} = [0 0 4 0 0 0 0 0 0 0 \frac{4}{2 0 0} 0 \frac{2}{0 0}] \), \( x_{5} = 0, x_{6} = 0 \) by equation (6), \( A_{3} = 0.05 \) by equation (14). According to equation (11), \( y_{3} = H_{i}x_{ij} + \bar{N}(A_{i}/S_{i}) \). For the cell \( x_{6} \) with coefficient 2, assume an adversary knows all but one record in \( x_{6} \), then the count numbers of other cells become constant in this query.

\(^{9}\)For two records \( v_{1} \) and \( v_{2} \), it’s negative correlated if \( Pr(v_{1} = t | v_{2} = t \geq Pr(v_{1} = t | v_{2} = t) \).
So $y_3 = 4 \ast x + 2x_5 + \hat{N}(0.05/4) = 2x_5 + \hat{N}(0.05/4)$. The $S$ of this query is 2 and the noise of this query is $\hat{N}(0.05/4)$. According to Theorem 2.1, it guarantees 0.025-differential privacy. Then the local privacy cost of $x_5$ is 0.025. Similarly for $x_3$ with coefficient 4, $y_3 = 4x_3 + \hat{N}(0.05/4)$, so the local privacy cost of $x_3$ is 0.05; for $x_1$ with coefficient 0, $y_3 = 0 \ast x_1 + \hat{N}(0.05/4)$, so the local privacy cost of $x_1$ is 0.

By theorem 2.2, the local privacy cost composes linearly for all queries. Then the local privacy costs can add up by each query.

Because we assume cells are independent or negatively correlated in this paper, by Theorem 2.3, the privacy costs of all cells donot affect each other. Therefore, we prove equation (47).

The highest privacy cost indicates the current system privacy cost, denoted by $\alpha$. So we prove equation (46).

In our example, we add $H_s = Q$ and $A_\alpha = 0.108$ to $H$ and $A$. Then by Theorem 3.10 $B$ is in Equations (49). system privacy $\alpha$ is 0.408 which is $\text{max}(B)$. Assuming the privacy budget for this system is 1, this would be far less than the upper bound. Therefore, we return the differentially private answer from Laplace mechanism to user. Finally, we update the parameters $H$, $A$ and $y$ for next coming query.

$$B = [0 \ 0.1 \ 0 \ 0.408 \ 0.283 \ 0 \ 0.2 \ 0.2] \quad (49)$$

4. SIMULATION APPROACH OF PROBABILITY FUNCTION

Section 4.1 introduces the theoretical probability function, which is difficult to calculate in practice, especially for equation (52). Then we use a simulation approach to compute the probability function. Detailed and sophisticated techniques about simulation approach can be found in [11, 1, 15]. To achieve more accuracy, we compute the linear composition of probability function, discussed in Section 4.2.

4.1 Laplace related probability function

We denote probability function of exponential distribution for $x \geq 0$ and $\alpha \geq 0$ as

$$Pr(x, \alpha) = a \exp(-\alpha x); \quad (50)$$

Probability function of Laplace distribution as 3; Probability function of Gamma distribution as

$$Pr(x, n, \alpha) = x^{n-1} \frac{\exp(-\alpha x)}{\Gamma(n)} \alpha^n. \quad (51)$$

where $\Gamma(n) = (n - 1)!$.

**Theorem 4.1** ([13]). The probability function of sum of $n$ i.i.d Laplace noises is

$$f_n(z) = \frac{\alpha^n}{2n\Gamma(n)} \exp (-\alpha |z|) \int_0^\infty e^{-v(n-1)/2} e^{-v} dv. \quad (52)$$

Note that above equations are just the cases when each $\alpha$ is the same. More complicated equations with different $\alpha$ can be found in [15].

4.2 Linear computation of probability function

We define this problem as following:

$$\hat{N} = [\hat{N}_1, \hat{N}_2, \cdots, \hat{N}_n]$$

are $n$ noises and we know their probability functions (may not necessarily be Laplace distribution). $T\hat{N}$ is a linear computation of $\hat{N}$ as equation (53). we want to know the probability function of $T\hat{N}$.

$$T\hat{N} = t_1\hat{N}_1 + t_2\hat{N}_2 + \cdots t_n\hat{N}_n \quad (53)$$

To solve this problem, first we eliminate all coefficients by theorem 4.2. Then As long as we solve operator “+”, $T\hat{N}$ can be solved by iteration. Algorithm 5 describes the process of linear computation of probability function.

**Theorem 4.2.** Probability function of $T\hat{N}$ in equation (53) is the same as probability function of $Z$ in equation (54).

$$Z = \hat{N}_1' + \hat{N}_2' + \cdots \hat{N}_n' \quad (54)$$

where $\hat{N}_j' = t_j\hat{N}_j$ and $Pr_{\hat{N}_j'}(\hat{N}') = Pr_{\hat{N}_j}(\hat{N}'/t_j)$.

**Algorithm 5** Linear computation of probability function

Require: $\hat{N}$ with discrete probability function $Pr_{\hat{N}}(), T$

1. $L_1(z) \leftarrow Pr_{\hat{N}_1}(z/t_1)$ where $L$ is the result;

2. for each $t_j\hat{N}_j$ in equation (53), $j \geq 2$ do

   $Pr_{\hat{N}_j'}(\hat{N}') = Pr_{\hat{N}_j}(\hat{N}'/t_j)$ by theorem 4.2;

   for each $k \in \mathbb{R}$ do

      $Pr_{\hat{N}_j}(\hat{N}_j') = \frac{l_j(k) = L_j-1(u)Pr_{\hat{N}_j'}(v);}{end for}$

   end for

return probability function: $Pr_{T\hat{N}}() = L(z)$

Note that in algorithm 5 we will not compute all $k \in \mathbb{R}$ in practice, need to break the loop when $L_j(k) < \text{negligible}$ where $\text{negligible}$ is any predefined negligible small number. We denote the range of $L_j(k) \geq \text{negligible}$ as the domain size of discrete probability function by $m_q$.

4.3 Computational complexity

**Theorem 4.3.** If we consider $m_d$ as the domain size of each probability function, $n$ as a constant, Algorithm 5 takes time $O(m_d^2)$.

**Proof.** For each operator $\pm$, for each domain value of the one probability function, it must traverse all the domain value of the other probability function. Therefore, each operator $\pm$ takes time $m_d^2$. Then the algorithm takes time $O(m_d^2)$.

5. EXPERIMENTAL STUDY

5.1 Experiment setup

**Environment** All experiments were run on a computer with Intel P8600(2 * 2.4 GHz) CPU and 2GB memory.

**Data.** We use the three data sets: net-trace, social network and search logs, which are the same data sets with [12].

Net-trace is an IP-level network trace collected at a major university; Social network is a graph of friendship relations...
in an online social network site. Search logs is a set of search query logs over time from Jan. 1, 2004 to 2011.

**Query.** User queries may not necessarily obey a certain distribution. So the question arises how to generate the testing queries. In our setting, queries are assumed to be multinomial distributed, which means the probability for the query to involve cell $x_i$ is $P_i$, where $\sum_{i=1}^{n} P_i = 1$. For efficiency and practicality, we assume there exist some regions of interest, where the cells are asked more frequently than others. Equation (55) shows the probability of each cell.

$$P_j = 0.9 \times 10^{-Jloor(\frac{j-1}{n})}$$

where the function $Jloor()$ rounds the input to the nearest integers less than it. To generate a query, we first generate a random number $n_j$ in $1 \sim 10$, which is the $\#$ of independent trials of multinomial distribution. Then we generate the query from the multinomial distribution, equation (56) shows the probability of $Q$.

$$Pr(Q_1 = q_1, Q_2 = q_2, \cdots, Q_n = q_n) = \frac{n_1! q_1! \cdots q_n!}{q_1! \cdots q_n!} P_1^{q_1} P_2^{q_2} \cdots P_n^{q_n}$$

where $\sum_{i=1}^{n} q_i = n_i$.

Finally, we generate 1000 queries for each setting, denoted by $H$.

**Utility Requirement $\epsilon$ and $\delta$.** We need to select the utility requirement $\epsilon$ and $\delta$ in definition 2.5 carefully to run the experiment due to the computation complexity. Intuitively, the larger $\epsilon$ (or the smaller $\delta$), the smaller $\alpha$ allocated because of theorem 3.9, then the wider probability function of noise. To simulate the estimation process with high accuracy, $m_4$ in theorem 4.3 becomes larger. Finally, the computation complexity may become too high. To run the experiment, we assume $\epsilon$ for the 1000 queries are uniformly distributed in $[1, 10^3]$, $\delta = 0.2$.

**Settings and Metrics.** We have conducted three sets of experiments to demonstrate the effects of dynamic $\alpha$ allocation, inference, the online system. In each setting, different metrics are used. Besides system privacy level $\bar{\alpha}$ in theorem 3.10 and credible interval $(\epsilon, \delta)$, following metrics are used.

For a bounded system, once the system privacy cost $\bar{\alpha}$ reaches the overall privacy budget, then the system cannot answer further queries. To measure this, we define the ratio of answered queries as below. In another word, it indicates the life span of a system.

**Definition 5.1.** Under the bound of system privacy $\bar{\alpha}$, given a set of query $Q$, the answer ratio $R_a$ is

$$R_a = \frac{\# \text{ of answered queries}}{\# \text{ of all queries}}$$

Among all the answered queries, we want to know the accuracy of the returned answer. We use following two metrics. 1. $R_t$ is defined to show whether the returned interval really contain the original answer; 2. $E$ is the defined to indicate the distance between returned answer and true answer. It’s easy to prove that if $\# \text{ of answered queries}$ approaches to infinity, $R_t$ converges to confidence level.

**Definition 5.2.** For a set of queries with $\alpha$ utility requirements, the answers are returned according to certain mechanisms. Ratio of satisfaction $R_s^{10}$ is defined as follows:

$$R_s^{10} = \frac{\# \text{ of answers} \leq \theta \leq U}{\# \text{ of all answered queries}}$$

**Definition 5.3.** For each query with $(\epsilon, \delta)$-usefulness requirement, a returned answer $\hat{\theta}$ is provided by the query answering system. If the original answer is $\theta$, the relative error is defined as:

$$E = \frac{\hat{\theta} - \theta}{\epsilon}$$

In summary, $\bar{\alpha}$ indicates the system privacy cost; $R_a$ indicates the life span of a system; $R_t$ indicates the confidence level; $E$ indicates the accuracy of the point estimation.

**5.2 Evaluation of dynamic $\alpha$ allocation.**

Let $\epsilon$ for the 1000 queries are uniformly distributed in $[50, 10^4]$, $\delta = 0.2$. For each query with different utility requirement, we can either allocate a static $\alpha$ to all queries or dynamically allocate different $\alpha$ with respect to its utility requirement. As the number of query increases, the system privacy cost in Figure .

![Figure 6: test](attachment:figure_6.png)

**5.3 Evaluation of inference on unbounded system.**

[50,10000] range, known queries 0.3-differential privacy. Let $\epsilon$ for the 1000 queries are uniformly distributed in $[1, 10^3]$, $\delta = 0.2$. First we show the system privacy level in Figure . X-axis is the # of queries.

The “improved baseline” in Figure ?? means $\alpha$ is derived by theorem 3.9 to guarantee the required utility without inference. The “inference” line means $\alpha$ is derived by theorem 3.9 and inference is used.
Next we show the utility result about inference in Figure ?? As number of queries increases, more and more observations can be derived to make inference of a target query, the error declines gradually as the “inference” line.

5.4 Evaluation of online bounded system

Because the bound of $\tilde{\alpha}$, only limited queries can be answered by the online system. If adopting our statistical inference, queries can also be inferred by the query history; otherwise for ordinary system, only the same query in query history can be answered after the system privacy reaches the bound.

Fixing the bound of system privacy $\tilde{\alpha}$ as 4, we can calculate the answer ratio $R_\alpha$. To compare the effect of statistical inference, we allocate constant $\alpha = 0.1$ for each query and fix $\epsilon = 100$. For each coming query, if it can be answered or inferred by our approach, $R_\alpha = \frac{1}{|H|+1};$ else $R_\alpha = \frac{1}{|H|+1}$, where $|H|$ is the rows of $H$, the size of query history. Figure ?? gives the result. X-axis is the # of queries.

Figure ?? shows the utility result of bounded system. For “improved baseline”, once the privacy budget is exhausted, further queries cannot be answered. Therefore, the query error is will not change after budget is out. For “inference”, even if budget is exhausted, some queries can still be estimated from query history. When the number of queries in query history increases, the error declines gradually to a lower level than “improved baseline”.

Reason: 1)At first, system needs to build the query history; 2)random queries history may be as better as delicate history.

5.5 Impact of parameters

Impact of $\alpha$

Impact of query distribution

The result is a little bit worse than figure 5.4, 5.4, 5.4 because the range starts from 1.
5.6 Experiment conclusion

System privacy $\bar{a}$ increases slower for two reasons. First, once a query can be estimated from query history, $\bar{a}$ won’t increase at all; Second, we only allocate the necessary budget for the query, which saves privacy budget. For statistical inference, note that sometimes the utility of estimate would be better than required.

6. RELATED WORKS

Many works on differential privacy and utility measurement were proposed so far. [23] gave an sufficient upper bound of $\alpha$. [16] provides a matrix mechanism to release data when the workload of queries is already known. [12] hierarchically partitions one-dimensional data and release the count of each cell. By checking the noisy counts with the requirement of consistency, it boosts the accuracy of release counts. However, the techniques seek least square solution, not minimal mean square error.

Online query answering system with differential privacy has been proposed for a long time[9, 5, 2, 17]. However, researches about how to deal with the privacy budget was not conducted so much. [21] gives a useful techniques, called query relaxation, which returns an alternative answer of the original answer when the original query is similar to some queries in history.

Differential privacy and statistics has been connected. [10] is a survey about differential privacy and statistics. [19] provides maximal likelihood estimator(MLE). [8] talks about robust statistics. All the related works are about point estimator, different from our utility measurement(interval estimator).

We assume nobody knows the true answer except the data manager. Therefore, neither for users nor for adversaries, they cannot predict $Pr(\#0)$. Otherwise, [20] gives the bounds of this equation derived from Jensen’s Inequality.

7. CONCLUSION

We propose a greedy algorithm using statistical inference techniques for online query answering, which minimizes the use of privacy budget for answering each query given a utility requirement. The minimal mean square error estimator is combined with differential privacy for the first time. We show the techniques of implementing MMSE estimator to estimate query answer. Also the statistical inference algorithm can be used to measure the posterior distribution w.r.t. adversarial privacy.

Our greedy algorithm only optimize local budget usage. Future works can research on how to allocate least budget for a set of queries because the computational complexity is too high. Future works can also find out how to maximize the utility given certain privacy budget. Given the bound of posterior distribution of adversarial privacy in this paper, we believe it’s useful for a series data publication with differential privacy.

8. ACKNOWLEDGEMENT

This research was supported in part by NSF grant CNS-1117763, a Cisco Research Award, and an Emory URC grant.

9. REFERENCES

10. APPENDIX

10.1 Denotation

<table>
<thead>
<tr>
<th>Denotation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>matrix of query history</td>
</tr>
<tr>
<td>X or Y</td>
<td>random samples from a distribution</td>
</tr>
<tr>
<td>hist()</td>
<td>histogram function; $\text{hist}(t) = \sum_{i=1}^{m} b(X_i = t) / m$</td>
</tr>
<tr>
<td>b()</td>
<td>bool function with binary output, 1 for true and 0 for false;</td>
</tr>
<tr>
<td>argx()</td>
<td>find the argument x in the equation to satisfy the requirement;</td>
</tr>
<tr>
<td>diag(v)</td>
<td>a function that returns a matrix with each element of vector v in diagonal;</td>
</tr>
<tr>
<td>abs()</td>
<td>a function that returns the absolute value of the input vector or matrix;</td>
</tr>
<tr>
<td>α</td>
<td>differential privacy budget;</td>
</tr>
<tr>
<td>γ</td>
<td>absolute error: $\gamma =</td>
</tr>
<tr>
<td>A</td>
<td>used privacy budget in H;</td>
</tr>
<tr>
<td>B</td>
<td>privacy budget table of each cell;</td>
</tr>
<tr>
<td>S</td>
<td>the sensitivity vector of H;</td>
</tr>
<tr>
<td>θ or Q(D)</td>
<td>original answer of query Q;</td>
</tr>
<tr>
<td>ˆθ</td>
<td>returned answer of Laplace mechanism: $\hat{\theta} = \theta + \tilde{N}$;</td>
</tr>
<tr>
<td>ˆθi</td>
<td>the vector of $\hat{\theta}$, the set of observations for a query;</td>
</tr>
<tr>
<td>N(α)</td>
<td>random variable from Laplace distribution with parameter α;</td>
</tr>
<tr>
<td>Prκ()</td>
<td>PDF of $N_i$ in equation (18);</td>
</tr>
</tbody>
</table>