Integrating Historical Noisy Answers for Improving Data Utility under Differential Privacy

Shixi Chen and Shuigeng Zhou
Fudan University, China
{chensx,sgzhou}@fudan.edu.cn

Sourav S Bhowmick
Nanyang Technological University, Singapore
assourav@ntu.edu.sg

ABSTRACT

Differential privacy is a robust principle for privacy preserving data analysis tasks, and has been successfully applied to a variety of applications. However, the number of queries that can be answered is limited for preventing privacy disclosure. Once the privacy budget is exhausted, all succeeding queries must be rejected. Therefore, each of the historical query answers is valuable and it is important to exploit them together to learn more about the data. We propose to integrate all available linear query answers into a consistent form that embodies our knowledge learned from the noisy answers, obtaining more accurate answers to past queries and even new queries, improving the data utility. Two distinct approaches are developed for this purpose, one via principle component analysis, and another via maximum entropy method. The second approach also generates a synthetic database, which is useful for differentially private data publishing. One important goal of our work is to ensure that the running time of our approaches does not grow with the cardinality of the universe of a data tuple, so that high-dimensional data with very large domain can still be tackled efficiently.

Categories and Subject Descriptors

H.2.8 [**DATABASE MANAGEMENT**]: Database Applications

General Terms

Algorithms, Security

Keywords

differential privacy, private data analysis

1. INTRODUCTION

The problem of privacy preservation is to enable public analysts studying useful knowledge from sensitive databases, while protecting personal privacy. Recently, differential privacy [10] has received considerable attention because it provides a rigorous paradigm to protect privacy. The notion of privacy principle states that the released information should not tell whether or not any personal data is included in the database. This is achieved by requiring that the output of a differentially private mechanism is insensitive to a small change in the input database. Differential privacy has been applied successfully to a variety of data analysis tasks and has shown good utility [2, 4, 5, 8, 9, 11–15, 17, 19, 21, 22, 24, 25].

Differential privacy is achieved by injecting randomness into query answers. In the typical Laplace mechanism [10], a random noise drawn from a Laplace distribution is added to each query answer. The magnitude of the noise is adjusted by the strongness of privacy protection required and is calibrated to the maximum possible change to the true answer caused by an atomic change to the input database. For simple queries such as counting queries, the noisy answers often show very good accuracy, and the absolute error does not grow with the cardinality of the database.

However, answering more queries over time may result in disclosure of more private information. To control the amount of private information exposed, it is important not to answer too many queries. The privacy budget, which determines the number of queries that can be answered, is limited jointly by the required privacy level and the accuracy of query answers. Once the privacy budget is exhausted, all succeeding queries have to be rejected.

In practice, analysts may have many different query tasks. Each query task studies some properties of the database by issuing some queries. Because the number of answerable queries is very limited, every answer to historical queries is valuable. When we are processing new queries, we should not forget the historical query answers. Instead, it is important to exploit them together to learn more about the data. In particular, if the answer to an incoming query is derivable from the historical query answers, we can afford to retrieve answer to the query without paying of privacy budget. Motivated by this, we propose to integrate all available *linear* query answers into a consistent form that embodies our knowledge learned from the noisy answers, for improving data utility.

The main purpose of integrating all available query answers into a consistent form is to obtain more accurate answers to queries. The gain in accuracy often comes from the inconsistency of noisy answers, due to redundant or dependent queries. As an example, suppose the analysts are studying a census database and are curious to know how much the *Weight* of a person raises on average when his *Height* increases by 1. The correlation coefficient between *Height* and *Weight* is demanded for this purpose. Three queries are then been issued, probably by different analysts — the first query computes SUM($Height \times Weight$) over people with $Age < 18$[1], the

EDBT 2012, March 26–30, 2012, Berlin, Germany.

[1]More rigorously, the analyst issues a query that computes SUM($Height \cdot Weight \cdot \delta(Age < 18)$), where δ is an indicator function

second query with $Age \geq 18$, and the third query over people of any age. Here, three queries are linear dependent, and the answer to each query can be improved by incorporating the noisy answers to other two, provided that injected noise is independent. In practice, dependence among a large collection of queries is expected to appear commonly, especially when issued by different analysts.

A potential solution to the above problem is to apply a powerful query mechanism to all queries, obtaining the best possible noisy answers within the limited privacy budget. However, there are at least two reasons for this solution to be often infeasible. First, in many situations, some query tasks appear earlier than other query tasks, and the later query tasks can be adaptively generated based on the results from the earlier query tasks. But most query mechanisms are unable to incorporate historical noisy answers to compute new queries, they require all queries in the workload are given at once. Second, the database T may be high-dimensional, even if each individual query task may operate on a low-dimensional projection of T. Because most, if not all, query mechanisms [14, 17] that support arbitrary linear queries require running time at least quadratic in the size of the data domain, the computation cost is prohibitively expensive.

Contributions. In this paper we study two post-processing approaches to integrate noisy answers to all linear queries into a consistent form, which can improve the accuracy of the estimate answers to all past or even new queries. One is based on principle component analysis, the other is based on the maximum entropy method. A typical use of our approaches is to incorporate the knowledge learned from historical query answers to refine the noisy answers output by a query mechanism for the current query workload. Purely based on past query answers, our approaches can also give meaningful answers to new queries that are highly correlated with past queries, which are very useful when the privacy budget has been exhausted. An important advantage of our approaches is that explicit computation of the base histogram over the entire universe U of a data tuple is circumvented, and the computation cost does not explicitly depend on the cardinality of U. Hence it is possible to tackle high-dimensional data with very large domain (or even continuous data with infinitely large domain) efficiently.

Our second approach also generates a synthetic database that approximately matches the query answers, which is useful for differentially private data publishing. Some studies [4, 11] have dedicated to differentially private data publishing, but most have polynomial, or even super-polynomial computation cost in the cardinality of U. Moreover, they adopt the difference between the answers to the synthetic database and to the input database as the only objective for optimization, which may not be preferable because the synthetic database can be very skew and unnatural. Our approach uses entropy as a secondary objective, leading to more natural data distributions and can give meaningful answers to new queries.

Our approaches use a parameter α to trade off the bias and the variance of estimate answers. Our first approach returns unbiased estimate answers when α is set to $+\infty$, but we are more interested in biased answers because that often allows more accurate results. We also present an approach to automatically select appropriate values for α for obtaining the best possible answers. When prior belief about the data distribution is available, the prior knowledge is often helpful for improving query accuracy when the past noisy answers do not provide sufficient information. In this paper we also show how to incorporate the prior belief into our approaches.

Organization. Section 2 gives notation and preliminaries; Section 3 and 4 introduce the two proposed approaches respectively; Section 5 presents experimental evaluation; Section 6 surveys related work; Section 7 concludes the paper.

2. NOTATION AND PRELIMINARIES

Suppose an input database T has d attributes $A_1 \dots A_d$ and n tuples $t_1 \dots t_n$, $t_i \in U$. We denote by $t_i[j] \in A_j$ the j'th attribute value of t_i. The universe or domain U of a tuple is the Cartesian product of all attribute domains A_j. We consider a collection of m queries $q_1 \dots q_m$, each described by a feature function $f_i : U \to \mathbb{R}$ and computing the sum of feature values over T

$$q_i(T) = \sum_{1 \leq j \leq n} f_i(t_j) \tag{1}$$

We refer to queries that can be expressed in the above form as *linear* queries. Note that a variety of important query types is covered in this expression, such as counting queries (including range queries as a special case) and moment estimation (mean, variance, correlation coefficient, etc). Many data mining tasks can be built on linear queries.

To simplify notations and discussion, we assume that all data are discrete and have finite domains. But it is possible to extend our approaches to continuous data that have infinitely large domains.

A common (but not the only) way to obtain differentially private answers to linear queries is via Laplace mechanism [10], which introduces an independent Laplace noise $e_i \sim Lap(b_i)$ with variance $2b_i^2$ to the answer to each query q_i. We have the following m noisy answers (together with their noise magnitudes b_i) released

$$a_i = \sum_{1 \leq j \leq n} f_i(t_j) + e_i \quad 1 \leq i \leq m \tag{2}$$

where the probability density of $e_i \sim Lap(b_i)$ at x is

$$Lap(x; b_i) = \frac{1}{2b_i} \exp\left(-\frac{|x|}{b_i}\right) \tag{3}$$

In this paper, we will simply assume that the database cardinality n is known to analysts.[2]

For better query accuracy, many query strategies introduce correlated noises into the query answers. However, all query strategies [2, 8, 15, 17, 25] built upon Laplace mechanism can be converted into the above model. These query strategies work by computing and submitting the *strategy* queries, some linear combinations of the user queries, to the Laplace mechanism. We can simply assume that the strategy queries and their noisy answers are known, which will not violate differential privacy.

Our principle component analysis approach is in fact based on a more relaxed setting — it merely assumes that the noisy answers are unbiased and the covariance matrix is known. This will support query mechanisms [14] that are not built upon Laplace mechanism.

that predicates whether $Age < 18$ is true.

[2]We can use ϵ-indistinguishability, the original concept of differential privacy, which permits release of n.

The output of our approaches can be regarded as the estimate answers to the space of all possible linear queries. These estimate answers are consistent in the sense that they must correspond to a base histogram $\hat{h}$ that comprises the estimate counts for each element in the data universe U. The estimate counts are real numbers and can be negative. Once the base histogram $\hat{h}$ is decided, the estimate answers to all linear queries are determined. However, we will avoid the explicit construction of the base histogram $\hat{h}$, otherwise the computation cost is at least proportional to the size of data domain U. The principle component analysis approach represents $\hat{h}$ as a set of eigen-queries along with the least squares estimates to their answers. In contrast, the maximum entropy approach represents $\hat{h}$ as a parameterized probability distribution or a sufficiently large sample drawn from that distribution. The maximum entropy approach affords stronger consistency, in the sense that the output always correspond to a valid database.

The goal of our approaches is to improve the query accuracy, which can be measured by the mean squared error of the estimate answers. Our approaches aim at obtaining the best possible answers to the past queries, but they are also able to give meaningful answers to new queries that are highly correlated with past queries.

When some prior belief about the data distribution is available, it is often helpful to incorporate that *a priori* knowledge into the computation for obtaining better query answers. The knowledge often comes from unequal chances of value assignments to an attribute. For example, some diseases are more common than others, and an address value at the level of city should be weighted more than an address value as the level of street. Our approaches can refine the estimate answers according to a prior distribution $p_0(t) = \prod p_{0,i}(t[i])$ that assumes strong independency between attributes. If no a priori knowledge is given, an uniform distribution $p_0(t) = \frac{1}{|U|}$ is used as the uninformative prior.

It is possible to obtain the prior distribution p_0 by computing some extra queries to the database T. Each marginal distribution $p_{0,i}$ can be obtained by computing $|A_i|$ marginal queries that compute the counts for each possible assignment to A_i. Since the sensitivity of all 1-D marginal queries is only d, a little cost of privacy budget can afford very accurate results.[3]

3. THE PRINCIPLE COMPONENT ANALYSIS APPROACH

Let $U = \{1, \ldots, N\}$, and we represent $x = [x_1, \ldots, x_N]^T$ as a column vector of counts: $x_i = |\{j | t_j = i, 1 \le j \le n\}|$. Each linear query is a length-N row vector $q_i = [q_{i,1}, \ldots, q_{i,N}]$ with each $q_{i,j} = f_i(j)$. We organize the m queries q_i into the rows of a $m \times N$ query matrix Q, the m noisy answers a_i and m noise variables e_i into column vectors a and e. We denote by $\Sigma = E(ee^T)$ the covariance matrix of the noisy answers. Then the best linear unbiased estimate for the true answer Qx is given by the weighted least squares solution to the linear regression problem

$$a = Qx + e \quad E(e) = 0 \quad Cov(e) = \Sigma \tag{4}$$

[3]With little chances, the noisy answers to the marginal queries may be invalid (negative), thus we need to convert them into a proper prior. A simple way is to truncate negative answers to 0. A more elegant way is to postulate a parameterized prior of the prior $p_{0,i}$ and then perform Bayesian inference with the noisy answers as evidence. This goes beyond the scope of this paper.

The solution to this problem is $\hat{x} = (Q^T\Sigma^{-1}Q)^+Q^T\Sigma^{-1}a = (\Sigma^{-\frac{1}{2}}Q)^+\Sigma^{-\frac{1}{2}}a$, where the superscript $+$ denotes the Moore-Penrose pseudo-inverse. $\hat{x}$ is an unbiased estimate of x only if $rank(Q) = N$. In any case, $Q\hat{x}$ gives the least squares estimate to the true answer Qx. In general, the subspace spanned by the rows of Q, made up of all linear combinations of input queries $\{q_i\}$, gives the set of all estimable functions[4]. For any estimable query q, $q\hat{x}$ gives the least squares estimate to qx.

However, the above standard process requires explicit construction of the $m \times N$ matrix Q, and takes $O(Nm^2)$ time if $N > m$. This is prohibitively expensive when the data universe U is extremely large, say, for high-dimensional data. In this section, we will present a more efficient approach that is suitable to the case of $m \ll |U|$ and is applicable for a range of important queries.

3.1 The Kernel PCA Approach

Our idea is to use PCA (Principle Component Analysis) to compute a collection of orthonormal queries $\{v_i\}$, called *eigenqueries*, each a linear combination of the input queries $\{q_i\}$. Moreover, we also compute $\{z_i\}$, the least squares estimates to the eigenqueries' true answers. Those least squares estimates are guaranteed being pairwise uncorrelated. Once $\{v_i\}$ and $\{z_i\}$ are obtained, the estimate answers to any other queries can be derived from them.

Specifically, we are looking for a linear transformation operation W that achieves three goals: WQ returns an orthogonal matrix $V = WQ$ whose rows give a set of $r = rank(Q)$ orthonormal queries $\{v_i\}$; Wa obtains the least squares estimate to Vx, i.e., $Wa = WQ\hat{x}$; Wa, or its errors We, are pairwise linearly uncorrelated, i.e., $E(We(We)^T) = W\Sigma W^T$ is a diagonal matrix.

Such matrix W can be found by computing the SVD (Singular Value Decomposition) of $\Sigma^{-\frac{1}{2}}Q$: $\Sigma^{-\frac{1}{2}}Q = U\Lambda V$, where $U = (u_{i,j})_{m\times m}$ and $V = (v_{i,j})_{N\times N}$ are orthonormal matrices, and Λ is an $m \times N$ diagonal matrix with $r = rank(Q)$ nonnegative singular values $\lambda_1 \ldots \lambda_r$ on the diagonal in descending order. Then $W = \Lambda^+U^T\Sigma^{-\frac{1}{2}}$ suffices our need. Only the r nonzero rows of W are useful, and other zero rows are discarded.

An alternative way to find W is by computing the eigendecomposition of $\Sigma^{-\frac{1}{2}}QQ^T\Sigma^{-\frac{1}{2}} = U\Lambda^2U^T$, where U is an orthonormal matrix and Λ^2 is a diagonal matrix with eigenvalues on the diagonal in decreasing order. This way can produce the same U and Λ (except the extra rows or columns with full zeros) as the former way. Thus we also obtain $W = \Lambda^+U^T\Sigma^{-\frac{1}{2}}$.

Finally, we obtain r transformed queries that come from the r nonzero rows of $V = WQ$. We refer to these transformed queries as eigenqueries, denoted by row vectors $v_1 \ldots v_r$. In fact they are the principle components of the input queries (without centering at zero) weighted by $1/\sigma_1 \ldots 1/\sigma_m$, when the noisy answers are independent variables with variances $\{\sigma_i^2\}$.

PROPOSITION 1. *The eigenqueries $v_1 \ldots v_r$ are a set of orthonormal vectors that forms a orthonormal basis for the subspace spanned by the input queries $q_1 \ldots q_m$.*

[4]The term *estimable* means that an unbiased estimate can be derived.

PROPOSITION 2. *The least squares estimate to the answer to the eigenqueries is Wa.*

PROOF.

$$WQ\hat{x} = WQ(\Sigma^{-\frac{1}{2}}Q)^{+}\Sigma^{-\frac{1}{2}}a \quad (5)$$

$$= W\Sigma^{\frac{1}{2}}U\Lambda V(V^T\Lambda^{+}U^T)\Sigma^{-\frac{1}{2}}a \quad (6)$$

$$= \Lambda^{+}U^T\Sigma^{-\frac{1}{2}}\Sigma^{\frac{1}{2}}U\Lambda\Lambda^{+}U^T\Sigma^{-\frac{1}{2}}a \quad (7)$$

$$= \Lambda^{+}U^T\Sigma^{-\frac{1}{2}}a = Wa \quad (8)$$

□

PROPOSITION 3. *The errors We of the estimate answers to eigenqueries are pairwise linearly uncorrelated (but they are usually dependent in terms of probabilities).*

PROOF. $W\Sigma W^T = \Lambda^{+}U^T\Sigma^{-\frac{1}{2}}\Sigma\Sigma^{-\frac{1}{2}}U(\Lambda^{+})^T = (\Lambda^T\Lambda)^{+}$ □

Thus, the eigenqueries $\{v_i\}$ and their estimate answers $\{z_i\}$ collectively give the least squares estimates for all estimable queries in a consistent form, where $z = (z_i)_{r\times 1}$ comprises the first r entries of Wa. Each estimable query q is a unique combination $c_1v_1 + \cdots + c_rv_r$ of the eigenqueries. The coefficients $c_1 \ldots c_r$ can be simply derived by computing the inner products $c_i = qv_i^T$, and we obtain the least squares estimate answer $c_1z_1 + \cdots + c_rz_r$. Because z_i's have pairwise uncorrelated errors of variances $1/\lambda_i^2$, the variance of q's estimate answer is $c_1^2/\lambda_1^2 + \cdots + c_r^2/\lambda_r^2$.

For non-estimable query q, the above process actually finds an estimable query q' that is closest to q (measured by $\|q - q'\|_2$) and returns the least squares estimate answer to q'.

When the database cardinality n is known to analysts, it is useful to add a virtual query $u = [1 \ldots 1]_{1\times N}$ with sufficiently low error variance into the input queries Q to embody our knowledge about n.[5] The virtual query u counts the total number of tuples in T, and we call u the *universal* query. Inclusion of u in Q results in that u becomes the most significant eigenquery $v_1 = N^{\frac{1}{2}}u$ with infinitely large λ_1.

3.2 Computing the Kernel Matrix

The benefit of the PCA approach is that, in essence, all calculations can be carried out with the inner products $\langle q_i, q_j\rangle$. It is unnecessary to explicitly construct the length-N vectors q_i and v_i if the inner products $\langle q_i, q_j\rangle$ are easy to compute. We first compute the *kernel matrix* $K = QQ^T$ that involves with m^2 inner products. Based on eigendecomposition of $\Sigma^{-\frac{1}{2}}K\Sigma^{-\frac{1}{2}} = U\Lambda^2U^T$, we obtain $W = \Lambda^{+}U^T\Sigma^{-\frac{1}{2}}$, where each nonzero row gives the coefficients of an eigenquery v_i in combination of input queries. Those coefficients are used for finding the coordinates $c_1 \ldots c_r$ of any query q in the basis of eigenqueries: $c_i = \langle q, v_i\rangle = \sum_j w_{i,j}\langle q, q_j\rangle$. Thus, the least squares estimate answer to any query q can be obtained by computing nr inner products. Overall, the output of our approach are the coefficient matrix W, the least squares estimate $z = Wa$ to the answers to r eigenqueries, and the eigenvalues $\{\lambda_i^2\}$, which can be computed in $O(m^2r)$ time once the kernel matrix K is available.

[5]or we can use constrained linear regression to avoid use of the infinitely low variance.

In the remainder of this section, we redefine the inner product $\langle q_i, q_j\rangle$ as the dot product $q_iq_j^T$ divided by N. This scales the kernel matrix K by a factor $1/N$, and makes the norm of the universal query u be $||u|| = 1$ (so $v_1 = u$). Because PCA is a scale and rotation invariant procedure, the output W and $\{\lambda_i^2\}$ will not change.

The key problem is to find a way to compute the inner products without traversing every location in the universe U. Recall that the tuple representation t that consists of d attribute values. To simplify discussion, we assume that the size of every attribute domain is bounded by a number C: $|A_i| \le C$ for all i.

We propose to write a query $q = (f(t))_{1\times N}$ in the following *polynomial* form

$$(q)_t = f(t) = \sum_{1\le i\le l_q}\prod_{j\in S_{q,i}} g_{q,i,j}(t[j]) \quad (9)$$

that comprises l_q product terms $\prod_{j\in S_{q,i}} g_{q,i,j}(t[j])$, where $S_{q,i} \subseteq \{1 \ldots d\}$ and $g_{q,i,j}(t[j])$ are arbitrary functions that can be computed in constant time. We define the size of q, denoted by s_q, the number of times the $g_{q,i,j}$ functions appear in the polynomial form: $s_q = \sum_i |S_{q,i}| \le l_qd$. We let $g_{q,i,j}(t[j]) = 1$ for all $j \notin S_{q,i}$, so that we also have $(q)_t = \sum_{1\le i\le l_q}\prod_{1\le j\le d} g_{q,i,j}(t[j])$.

The reason for the use of polynomial form is that many queries can be written in a polynomial form of very short lengths $l_q \ll N$, which helps us to compute the inner products in an efficient way.

THEOREM 1. *The inner product of q_1 and q_2 can be computed in $O(l_{q_1}s_{q_2}C + l_{q_2}s_{q_1}C) \le O(l_{q_1}l_{q_2}dC)$ time.*

PROOF.

$$\langle q_1, q_2\rangle = \sum_{t\in U}(q_1)_t(q_2)_t/N \quad (10)$$

$$= \sum_{1\le i\le l_{q_1}}\sum_{1\le j\le l_{q_2}}\prod_{1\le k\le d}\sum_{\theta\in A_k}\frac{1}{|A_k|}g_{q_1,i,k}(\theta)g_{q_2,j,k}(\theta) \quad (11)$$

$$= \sum_{1\le i\le l_{q_1}}\sum_{1\le j\le l_{q_2}}\prod_{\substack{k\in S_{q_1,i}\\ \cup S_{q_2,j}}}\sum_{\theta\in A_k}\frac{1}{|A_k|}g_{q_1,i,k}(\theta)g_{q_2,j,k}(\theta) \quad (12)$$

Eq. 12 has at most $l_{q_1}s_{q_2}C + l_{q_2}s_{q_1}C$ calls to g functions. □

The factor C appears because we need to traverse every element of A_k for computing $\sum_{\theta\in A_k} g_{q_1,i,k}(\theta)g_{q_2,j,k}(\theta)$. For many kinds of queries (like range queries), the g functions have special structure that allows us to compute $\sum_{\theta\in A_k} g_{q_1,i,k}(\theta)g_{q_2,j,k}(\theta)$ in constant time. In that case, the factor C does not appear.

COROLLARY 1. *The kernel matrix K can be computed in $O(\sum_i l_{q_i}\sum_i s_{q_i}C) \le O((\sum_i l_{q_i})^2dC)$ time.*

Many query types can be written in a polynomial form with short lengths, thus support efficient computation of the inner products. Some typical examples are as follows.

A **range query** q counts the number of tuples whose attribute values fall into specified intervals. A k-D range query can be

written as $(q)_t = \delta(a_1 \le t[i_1] \le b_1) \times \cdots \times \delta(a_k \le t[i_k] \le b_k)$, where δ denotes an indicator function that returns 1 if the predicate is true and returns 0 otherwise. We have $l_q = 1$ and $s_q = k$.

A **counting query** q with predicate function given by a boolean expression can be written in a polynomial form, by summing over all possible assignments to the boolean variables in the expression. The length l_q can be exponentially large in the number of boolean variables. However, many user queries only contain a few boolean variables.

Mean, variance, covariance, and higher order moments can be written in the form $(q)_t = (t[i_1])^{c_1} \times \cdots \times (t[i_k])^{c_k}$ for some constants $\{c_k\}$. $l_q = 1$ and $s_q \le k'$ for a k'-order moment.

A query strategy proposed in [2] addresses binary attributes ($|A_i| = 2$) and computes the **Fourier transformation** of the database. The query to compute a Fourier coefficient is in the form: $(q)_t = (-1)^{t[i_1]} \times \cdots \times (-1)^{t[i_k]}$. $l_q = 1$ and $s_q \le d$.

The work in [25] extends the Fourier transformation to the **wavelet transformation**. The strategy query is in the form: $(q)_t = \prod_i (-1)^{c_i(t[i])}$ for some functions c_i. $l_q = 1$ and $s_q = d$.

Some queries does not belong to any of the above examples. However, we can show that the inner products can still be efficiently computed if each of the queries is relevant to only a few attributes. Because most existing query strategies [2,8,15,17,25] require computation cost at least proportional to the domain size, the queries generated by those strategies usually operate on a low-dimensional projection of T on only a few attributes.

Suppose a query q is relevant to a set of d_q attributes $h_q = \{h_{q,1}, \ldots, h_{q,d_q}\}$. Let Dom_q denote the Cartesian product $\prod_i A_{h_{q,i}}$, and let $q(t')$ for $t' \in Dom_q$ denote $(q)_t$ when t' and t match on the relevant attributes. Then q can be regarded as the sum of $|Dom_q|$ counting queries, each specifies a location in Dom_q. We can write q as

$$(q)_t = \sum_{t' \in Dom_q} \prod_{1 \le i \le d_q} \delta(t[h_{q,i}] = t'[i]) q(t') \qquad (13)$$

We have $l_q = |Dom_q|$ and $s_q = l_q d_q$. This immediately implies that the inner product of two such queries q_1 and q_2 can be computed in at most $O(|Dom_{q_1}||Dom_{q_2}|(d_{q_1} + d_{q_2})) \le O(C^{d_{q_1}+d_{q_2}}(d_{q_1} + d_{q_2}))$ time. We can further reduce the computation cost to only $O(|Dom_{q_1}|d_{q_1} + |Dom_{q_2}|d_{q_2})$.

THEOREM 2. *The inner product of q_1 and q_2 can be computed in $O(|Dom_{q_1}|d_{q_1} + |Dom_{q_2}|d_{q_2}) \le O(C^{d_{q_1}} d_{q_1} + C^{d_{q_2}} d_{q_2})$ time, assuming the computation cost for $q_i(t')$ for $t' \in Dom_{q_i}$ is $O(d_{q_i})$.*

PROOF.

$$\langle q_1, q_2 \rangle = \sum_{t \in U} (q_1)_t (q_2)_t / N \qquad (14)$$

$$= \sum_{t_0 \in Dom_{h_{q_1} \cap h_{q_2}}} \left(\sum_{t_1 \in Dom_{h_{q_1} \setminus h_{q_2}}} q_1(t_0, t_1) \right) \left(\sum_{t_2 \in Dom_{h_{q_2} \setminus h_{q_1}}} q_2(t_0, t_2) \right) / |Dom_{h_{q_1} \cup h_{q_2}}| \qquad (15)$$

where Dom_h denotes $\prod_{i \in h} A_i$. Eq. 15 has $|Dom_{q_1}|$ calls to $q_1(\cdot)$ and $|Dom_{q_2}|$ calls to $q_2(\cdot)$. □

COROLLARY 2. *The kernel matrix K can be computed in $O(\sum_i |Dom_{q_i}| d_{q_i} m) \le O(\sum_i C^{d_{q_i}} d_{q_i} m)$ time, assuming the computation cost for $q_i(t')$ for $t' \in Dom_{q_i}$ is $O(d_{q_i})$.*

In practice, we can use either Eq. 12 or Eq. 15 to compute the inner product of a pair of queries, and choose the faster one based on the types of queries.

3.3 Trade-off between Bias and Variance

Usually, if we can tolerate a small bias, and change the query a little, we may obtain more accurate estimate answer with smaller error. Suppose there are two queries q_1 and q_2, which compute the number of elements that fall into the interval $[0, 99]$ and $[1, 100]$, respectively. Because these two queries are very similar and highly correlated, we are often able to obtain a more accurate answer, to any of them, by averaging their noisy answers. The resulting answer $a = (a_1 + a_2)/2$ is the least squares estimate answer to the query $q = (q_1 + q_2)/2$ (provided that two noisy answers have equal variances). Because q is very similar to q_1 and q_2, the estimate bias $(q - q_1)x$ is hopefully very small. Because $E(error^2) = bias^2 + variance$, as long as the reduction in variance is more than the square of estimate bias, a is a better answer to q_1 (or q_2) in terms of squared error, though the estimate bias is unknown to analysts.

To answer a query q, we propose to return the least squares estimate $q'\hat{x}$ to another estimable query q', and minimize the objective function

$$R_q(q') = \alpha^2 \|q - q'\|^2 + Var(q'\hat{x}) \qquad (16)$$

The parameter $\alpha \ge 0$ controls the trade-off between bias and variance. In the following we present a simple way to solve this problem.

Any query q can be decomposed into orthogonal components $q = c_1 v_1 + \cdots + c_r v_r + q_{bias}$, in which $c_i = \langle q, v_i \rangle$ and the inherent bias q_{bias} denotes the difference from q to the estimable space spanned by $\{v_i\}$. Then we represent q' in the similar way: $q' = y_1 c_1 v_1 + \cdots + y_r c_r v_r$, where $0 \le y_i \le 1$. Through simple calculation, we obtain $\|q - q'\|^2 = \sum_i (1 - y_i)^2 c_i^2 + ||q_{bias}||^2$ and $Var(q'\hat{x}) = Var(\sum_i y_i c_i z_i) = \sum_i y_i^2 c_i^2 / \lambda_i^2$, where $1/\lambda_i^2$ is the variance of z_i. Hence,

$$R_q(q') = ||q_{bias}||^2 + \sum_i ((1 - y_i)^2 \alpha^2 c_i^2 + y_i^2 c_i^2 / \lambda_i^2) \qquad (17)$$

The optimal solution can be obtained by setting the derivative to zero, and we get $y_i = \frac{\alpha^2}{\alpha^2 + 1/\lambda_i^2}$.

Finally, for each query q we output $y_1 c_1 z_1 + \cdots + y_r c_r z_r$ as its estimate answer that minimizes the objective function R_q. y_i serve as the weights to every eigenqueries v_i, and are irrelevant to the query q. Obviously, the output estimate answers to the space of all possible queries are consistent.

THEOREM 3. *Let $y_i = \frac{\alpha^2}{\alpha^2+1/\lambda_i^2}$. For any query q, the output $y_1c_1z_1 + \cdots + y_rc_rz_r$ is the least squares estimate answer to another query q' that minimizes $R_q(q')$, where $c_i = \langle q, v_i\rangle$. For any fixed assignment to α, the output estimate answers to the space of all possible queries are consistent.*

The weights of eigenvectors are assigned according to the variances of their least squares estimate answer: higher variance, less important. The parameter α controls how much the weights are affected by the variances. If $\alpha = +\infty$, all eigenqueries are equally weighted, so we always output unbiased estimate answer to any estimable query q. When α approaches 0, only the universal query $u = v_1$ is used. In latter case, the estimate answer to a query q is $\langle q, u\rangle z_1 = qu^T n/N$, as if computed on a uniformly distributed database. This is like a smooth version of dimension reduction. A simple observation is that, if we restrict each y_i to be either 0 or 1, then $y_i = 1$ if and only if $1/\lambda_i^2 < \alpha^2$, and all eigenqueries v_i with $1/\lambda_i^2 > \alpha^2$ are discarded.

3.4 Incorporating Prior Belief about Data Distribution

So far, we measure the difference between two queries by the squared norm $\|q - q'\|^2$ when we trade off between bias and variance. Because the inner product is set to proportional to the dot product, all locations in U are equally weighted in measuring the difference between q and q'. This may not be preferable, because some assignments to a tuple can be more likely to appear than other assignments. When a prior distribution p_0 is given, we propose to use a new inner product function, which is defined as

$$\langle q_1, q_2\rangle = \sum_{t\in U} p_0(t)(q_1)_t(q_2)_t \tag{18}$$

With this definition, each member t of U is weighted by it's prior probability $p_0(t)$.

To intuitively understand the effect of this change, consider the extreme case that α approaches 0. In that case, the estimate answer to a query q is $\langle q, u\rangle z_1 = n\sum_{t\in U} p_0(t)(q)_t(u)_t$, as if computed on a database that follows the prior distribution p_0. So the parameter α indicates to what extent we believe the prior distribution.

We assume that the prior distribution is given in the form $p_0(t) = \prod p_{0,i}(t[i])$, which makes strong independency assumption. Then Eq. 12 now becomes

$$\langle q_1, q_2\rangle = \sum_{t\in U} p_0(t)(q_1)_t(q_2)_t = \tag{19}$$

$$\sum_{1\le i\le l_{q_1}} \sum_{1\le j\le l_{q_2}} \prod_{\substack{k\in S_{q_1,i}\\ \cup S_{q_2,j}}} \sum_{\theta\in A_k} p_{0,k}(\theta) g_{q_1,i,k}(\theta) g_{q_2,j,k}(\theta) \tag{20}$$

This is equivalent to having every $g_{q,i,j}$ function in Eq. 12 and Eq. 15 to be substituted with a new one $g'_{q,i,j}(t[j]) = \sqrt{|A_j| p_{0,j}(t[j])}\, g_{q,i,j}(t[j])$. Hence, the computation cost for inner products remains the same.

3.5 Selecting an Appropriate α

For improving query accuracy, it is desirable to choose an α that leads to estimate answers with minimal mean squared error (MSE) to a collection of queries. The best choice of α depends on the data distribution and the queries. A simple way to choose α is to try many possible assignments to α, compute for each trial the resulting MSE in a differentially private manner, and choose the assignment that leads to the smallest noisy MSE. However, this approach will expend considerable amount of privacy budget, because a) it has to compute many differentially private MSE queries for making a good choice; b) the sensitivity of a MSE query is large, which makes it difficult to obtain an accurate estimate to MSE.

We assume that all input queries are independently drawn from an underlying population Ω, and the noises added to every query answer are also independent (though it is still possible to apply our approach to the case of dependent noises). We propose to find an α that minimizes the following quantity

$$MSE_m(\alpha) = \tag{21}$$
$$E_{q_i\sim\Omega, q\sim\Omega}((\mathcal{A}(\{(q_1,a_1)\ldots(q_m,a_m)\}, q, \alpha) - qx)^2)$$

where $\mathcal{A}$ denotes the algorithm that receives m input queries q_i and noisy answers a_i and outputs the estimate answer to q given parameter α, and qx denotes the true answer to q. So MSE_m is the expected squared error of the estimate answers to new queries that come from the same distribution from which the input queries are drawn. We present an approach to estimate this MSE resulted by a choice of α, purely based on the past noisy answers to Q.

Our idea is based on cross-validation. We randomly split all m queries in Q into k groups. At each stage we choose a group as test set Q_{test}, and the union of other groups as training set $Q_{training}$. We use the PCA approach to process the training queries in $Q_{training}$ together with their noisy answers. Then we compute the estimate answers a'_{test} to the test queries in Q_{test}, according to the current choice of α. The estimate answers a'_{test} are then compared to Q_{test}'s noisy answers a_{test}, so as to obtain an unbiased estimate to $MSE_{|Q_{training}|}$. The above stage is repeated k times, each time choosing a different group as the test set. The k estimates are averaged to obtain the final (biased) estimate to MSE_m.

Now we focus on a single stage. To obtain an unbiased estimate to $MSE_{|Q_{training}|}$, we will compare the estimate answers a'_{test} with the noisy answers a_{test}. We first compute $\widehat{mse} = ||a_{test} - a'_{test}||^2/|Q_{test}|$, which uses a_{test} as if true answers to derive the mean squared error of a'_{test}. Then we subtract from $\widehat{mse}$ the mean variance of a_{test} to obtain the estimate $MSE_{|Q_{training}|}$ for this stage. If all noisy answers to Q are independent, then a_{test} and a'_{test} are independent, and the obtained estimate $MSE_{|Q_{training}|}$ is unbiased, as supported by the following theorem.

THEOREM 4. *Let $a_1\ldots a_n$ be some unknown quantities, $b_1\ldots b_n$ be some estimates to $\{a_i\}$, and $c_1\ldots c_n$ be some unbiased estimates to $\{a_i\}$. Let σ_i^2 denote the variance of c_i. Assuming that $\{b_i\}$ and $\{c_i\}$ are independent random variables. Then, an unbiased estimate to the expected mean squared error of $\{b_i\}$ is $\frac{1}{n}\sum_i (b_i - c_i)^2 - \frac{1}{n}\sum_i \sigma_i^2$.*

PROOF. Because $E(c_i) = a_i$ and $E(c_i^2) = a_i^2 + \sigma_i^2$,

$$E\left(\sum_i (b_i - c_i)^2 - \sum_i \sigma_i^2\right) \tag{22}$$
$$= \sum_i E(b_i^2 - 2b_ic_i + c_i^2 - \sigma_i^2) \tag{23}$$

$$= \sum_i E(b_i^2 - 2b_i a_i + a_i^2) = E\left(\sum_i (b_i - a_i)^2\right) \quad (24)$$

□

The parameter k is chosen by the user. Larger k gives better estimate to MSE_m but requires more computation cost. Usually a small k (e.g., $k \leq 10$) is sufficient to provide good performance. We can fix the partition of the queries Q, so that we does not need to recompute the PCA for each trial of α. Hence, in the preprocessing phase, we pre-compute PCA for each training set $Q_{training}$, which expends roughly $O(km^3)$ time in total. Then, given any choice of α, the resulting estimate answers a_{test} to all test sets Q_{test} can be computed in $O(m^2)$ time, from which we can simply derive the estimate MSE_m.

To find the best possible assignment to α, we can make many different trials of α and select the one that leads to minimal estimate MSE_m. A more efficient way is to use ternary search algorithm, assuming that the MSE_m is a unimodal function of α.

Once the best assignment to α is selected, out approach can even be used to answer new queries if they are drawn from the same population that generates past queries, without submitting the new queries to a differentially private mechanism that expends privacy budget. If new queries are drawn from a different population, however, we still need to invoke a differentially private mechanism on the new queries. In the latter case, our approach is still useful as a postprocessing algorithm that can be applied on the union of past and new queries to improve the query accuracy.

4. THE MAXIMUM ENTROPY APPROACH

In this section, we will present an approach that finds a data distribution p that is most likely to generate the noisy answers.

4.1 The Objective Function

Our goal is to estimate the distribution of T. Two criterions are of our interest for choosing the estimate distribution p: $U \to [0, 1]$. The first is the information entropy, because distribution with greater entropy makes less assumptions about data. The second is the likelihood given the noisy answers, which indicates how probable for the distribution to generate the answers. Let $\pi_i = a_i/n$, we propose the following objective function to make leverage between the two criterions.

$$\min_{p \in \Delta} \quad Q(p) = -H(p) - \alpha L(p|\{\pi_i\}) \quad (25)$$

$$= p[\ln p] + \sum_i \beta_i |\pi_i - p[f_i]| + \text{const} \quad (26)$$

where

$$L(p|\{\pi_i\}) = \ln Pr(\{\pi_i\}|p) \quad (27)$$

$$= \ln \prod_i Lap(\pi_i - p[f_i]; \frac{b_i}{n}) \quad (28)$$

$$= -\sum_i \left(\frac{n}{b_i}|\pi_i - p[f_i]| + \ln \frac{2b_i}{n}\right) \quad (29)$$

$$\beta_i = \frac{\alpha n}{b_i} \quad (30)$$

We denote by $p[f] = p[f(t)]$ the expectation of f w.r.t. probability distribution p, and $\Delta \subset [0, 1]^U$ the simplex of the probability distributions on U. The penalty functions $\beta_i|\pi_i - p[f_i]|$ give soft constraints to the standard maximum entropy model. We are unaware of any prior study for this type of soft constraints to maximum entropy model. Note that our goal is to estimate the distribution of the true database T, rather than learning the underlying distribution that generates T.

$\alpha \geq 0$ is a parameter that trades off between the entropy $H(p)$ and the log-likelihood $L(p|\{\pi_i\})$. If α is small, then distribution with large entropy is favored. If α gets to infinite, then the estimate distribution is required being exactly consistent to the noisy answers if possible.

The objective function Q can also be interpreted in another way. Consider $p \in \Delta$ as an unknown length-N random vector with prior distribution Φ: $\Delta \to [0, 1]$, and as our prior belief we presume that the expected entropy of p is given by $\tilde{h}$: $\Phi[H(p)] = \tilde{h}$. Then the solution to the problem Eq. 26 corresponds to the maximum a posteriori probability estimate to p given evidence $\{\pi_i\}$ and prior Φ formed by the principle of maximum entropy subject to the condition $\Phi[H(p)] = \tilde{h}$ for a certain $\tilde{h}$.

THEOREM 5. *Let the prior $\Phi_{\tilde{h}}$: $\Delta \to [0, 1]$ be the maximum entropy distribution subject to $\Phi_{\tilde{h}}[H(p)] = \tilde{h}$. Then the solution to Eq. 26 maximizes the posterior probability density $\Phi_{\tilde{h}}(p)Pr(\{\pi_i\}|p)$ for a certain $\tilde{h}$.*

PROOF. (Sketch) Because $\Phi_{\tilde{h}}$ is the maximum entropy distribution, it is the solution to the following problem

$$\max_{\Phi} \quad H(\Phi) = -\Phi[\ln \Phi] \quad (31)$$

$$\text{subject to} \quad \Phi[H(p)] = \tilde{h} \quad (32)$$

Using Lagrange multipliers [7], it can be shown that $\Phi_{\tilde{h}}$ can be expressed as

$$\Phi_{\tilde{h}}(p) \propto \exp(\frac{H(p)}{\alpha}) \quad (33)$$

for a certain α. Because

$$\ln(\Phi_{\tilde{h}}(p)Pr(\{\pi_i\}|p)) = \ln \Phi(p) + L(p|\{\pi_i\}) \quad (34)$$

$$\propto H(p) + \alpha L(p|\{\pi_i\}) + \text{const} \quad (35)$$

Maximizing the logarithm of the posterior probability density is the same as minimizing Q. □

By fixing $p[f_i]$, thus also fixing the log-likelihood $L(p|\{\pi_i\})$, via Lagrange multipliers, it can be shown that the optimal p that minimizes Q or maximizes the entropy $H(p)$ can be expressed as [7]

$$p_w(t) = \frac{1}{Z_w} \exp \sum_i w_i f_i(t) \quad (36)$$

which is a Gibbs distribution parameterized by $w = \{w_i\}$.[6] The Lagrange multipliers $w = \{w_i\}$ are also called the weights or factors of the features, and $Z_w = \sum_t \exp \sum_i w_i f_i(t)$ is a partition function that ensures p_w sums to 1.

[6]We ignore distributions that have exactly zero probabilities, because these distributions are never optimal provided that the answers are noisy.

The solution to the problem Eq. 26 can be obtained by solving its dual program, as shown in the following theorem. A sketch of the proof is presented in [6].

THEOREM 6. *The optimal p that minimizes $Q(p)$ is the Gibbs distribution (Eq. 36) parameterized by $w = \{w_i\}$, where w is the solution to the following convex program*

$$\max_{-\beta_i \le w_i \le \beta_i} Q'(w) = \sum_i w_i \pi_i - \ln Z_w \tag{37}$$

Eq. 37 describes a constrained convex optimization problem, which is often easier to solve than its primal form as shown in Eq. 26. It can also be viewed as a constrained maximum likelihood problem. Suppose that there exists a database with empirical distribution $\tilde{p}$, whose feature expectations $\tilde{p}[f_i]$ exactly match the noisy answers π_i (assuming that the noisy answers are consistent). Because

$$\sum_i w_i \pi_i - \ln Z_w = \tilde{p}[\sum_i w_i f_i - \ln Z_w] = \tilde{p}[\ln p_w] \tag{38}$$

Q' is proportional to the log-likelihood of p_w w.r.t. $\tilde{p}$. Thus, the solution to the dual problem (Eq. 37) is the maximum likelihood Gibbs distribution with box constraints $w_i \in [-\beta_i, \beta_i]$.

4.2 Learning the Maximum Entropy Model

The convex program Eq. 37 for Q' is a constrained version of the well known problem of parameter estimation for a general Markov random field or conditional random field [3,16,23]. Since extensive work have devoted to this learning problem, we just present here a simple algorithm that suffices our need.

Through simple calculations, it can be shown that

$$\frac{\partial Q'(w)}{\partial w_i} = \pi_i - p_w[f_i] \quad \text{and} \quad \frac{\partial^2 Q'(w)}{\partial w_i \partial w_j} = Cov_{p_w}(f_i, f_j) \tag{39}$$

However, exact inference for the marginal $p_w[f_i]$ is intractable for general graphical models. Thus, we resort to MCMC (Markov chain Monte Carlo) methods to approximate the gradients. Due to the approximation, it is not appropriate to use optimization methods that are sensitive to gradients or to the Hessian. Instead, we use stochastic gradient descent, which fits well with approximate marginal.

The framework of our algorithm is as follows. We start with an initial parameter estimate $w = 0$. In each iteration we draw a sample t from the current estimate distribution p_w. Then we update every w_i by $w_i \leftarrow w_i + \gamma(\pi_i - f_i(t))$. We also truncate w_i once we discover $|w_i| > \beta_i$. The above steps are repeated until convergence or until a specified number of iterations is reached. The learning rate γ is set to a small enough number.

The Gibbs sampling [18] is used to generate a sequence of (dependent) samples from p_w. To draw a new sample from a previous sample t, it generates an instance $t[k]$ from the distribution $p_w(A_k | t \setminus \{t[k]\})$ of each attribute A_k in turn, conditioned on the current assignments to other attribute values. The sequence of samples t forms a Markov chain, and t's distribution will converge to the target p_w. We update the parameter w after every new sample t is generated. Gibbs sampling is often a slow-mixing process, due to the auto-correlation between adjacent samples. Thus, t could be lagging in catching the latest p_w, but this issue can be alleviated by choosing a smaller learning rate γ.

To generate an instance from the conditional distribution $p_w(A_k | t \setminus \{t[k]\})$, we employ the Metropolis-Hastings algorithm [18]. In short, it first draws an proposal instance o from A_k with proposal distribution $q(o)$, then update $t[k]$ to o with probability $\min(1, \frac{p_w(o|t\setminus\{t[k]\})q(t[k])}{p_w(t[k]|t\setminus\{t[k]\})q(o)})$. Without additional knowledge, q can be chosen as the uniform distribution. This process involves the calculation of the ratio $\frac{p_w(t \text{ with } t[k]=o)}{p_w(\text{old } t)}$, which can be derived from Eq. 36. Not all feature functions are necessary to calculate, but only those feature functions f_j whose results are affected by a change to $t[k]$ are relevant. Generating instances for all attributes of a tuple will need at most $2md'$ calls to feature functions, where d' denotes the average number of attributes a query is relevant to.

To update all weights $w_i \leftarrow w_i + \gamma(\pi_i - f_i(t))$, we need at most m calls to feature functions. In total, every iteration will need $O(md')$ calls to feature functions.

An interesting special case is, when doing privacy-preserving data publishing, the set of m queries are often formed by all k-D marginal queries. In this case, for every tuple t there are only $C(d,k) = \left(\begin{array}{c} d \\ k \end{array} \right)$ feature functions f_i with non-zero output $f_i(t) \neq 0$. With some simple improvement, every iteration will need only $O(kC(d,k))$ calls to feature functions, though the total number m of queries is very large. Due to space limitation, we omit the details here.

Finally, we obtain the approximate solution p_w, which integrates all noisy answers into an estimate distribution that gives consistent estimate answers to all queries. In practice, we will generate a sufficiently large sample from p_w to support queries and other data mining tasks. This sample also serves as a synthetic database for differentially private data publishing. The weakness of our algorithm is that, like all other learning algorithms for general Markov random field, the convergence rate is not guaranteed. In practice, we may simply set a fixed number of iterations.

4.3 Incorporating the Prior Distribution p_0

If a prior distribution p_0 is given, we can use relative entropy instead of the information entropy, and obtain a slightly different objective function

$$\min_{p \in \Delta} \quad Q(p) = D_{KL}(p||p_0) - \alpha L(p|\{\pi_i\}) \tag{40}$$

$$= p[\ln \frac{p}{p_0}] + \sum_i \beta_i |\pi_i - p[f_i]| + \text{const} \tag{41}$$

where $D_{KL}(p||p_0) = p[\ln \frac{p}{p_0}]$ is the relative entropy or Kullback-Leibler divergence. When p_0 is the uniform distribution, Eq. 41 differs from Eq. 26 only by a constant. The optimal p for Eq. 41 can be expressed as $p_w(t) = \frac{1}{Z_w} p_0(t) \exp \sum_i w_i f_i(t)$, where Z_w is again a partition function that ensures p_w sums to 1. The dual program Eq. 37 remains the same form, but the Z_w function in Eq. 37 changes due to change of p_w. The previous learning algorithm for estimating p_w still applies.

5. EXPERIMENTS

In this section, we evaluate the performance of our approaches based on the Adult dataset[7], which contains personal information of about $500,000$ American adults and has 9 attributes. The list of attributes is presented in Fig. 1. All numeric attribute values are integers.

[7]available at http://www.ipums.org

Attribute	Type	Domain
Age	numeric	[16,94]
Gender	categorical	2
Education	numeric	[1,17]
Birth-place	categorical	51
Race	categorical	9
Work-class	categorical	7
Marital	categorical	6
Occupation	categorical	50
Salary	numeric	[0,49]

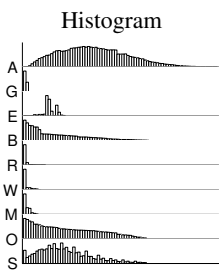


Figure 1: Adult dataset used in our experiments

We have conducted two sets of experiments. In the first set of experiments, we provide a collection of randomly generated queries and noisy answers to our approaches, then study their ability in improving accuracy. In the second set of experiments, we use the maximum entropy approach as a privacy preserving data publishing tool and compare its performance with a few existing techniques. As default setting for the maximum entropy approach, we set the learning rate γ to 10^{-4} and perform roughly about 10^6 iterations for each run.

Although there are some relevant techniques that also process a collection of differentially private queries that may be given arbitrarily, such as [4, 11, 13, 14, 17, 22], we did not test these techniques because their computation cost are extremely high even for datasets of moderate size. These techniques have running time at least linear or polynomial in the size of data universe U.

5.1 Integrating Past Query Answers and Improving Accuracy

In these experiments, we consider two kinds of queries. The first kind is 2-D range query, which specifies an interval for each of a pair of randomly chosen attributes. Every attribute is chosen with equal probability. The predicate intervals for a range query are also randomly generated. The second kind is arbitrary 2-D counting query. Each 2-D counting query q is defined by a binary function $f : A_i \times A_j \to \{0, 1\}, i \neq j$. All entries $f(x, y)$ are generated at random.

We first generate $m = 5000$ random queries, and compute the answers based on the $500,000$ tuples in the Adult dataset. Every answer is then introduced with an independent Laplace noise $Lap(b)$ with scale $b = 5000$. Because the sensitivity of m random queries is at most $m = 5000$, we guarantee to satisfy at least 1-differential privacy.

In Fig. 2a and 2b, we study the performance of our approaches with varying assignments to α. The performance is measured by the RMSE (root mean square error) of the estimate answers to past queries. For each of our approaches, we have tested two versions — the version that does not utilize prior distribution (amounts to a uniform prior distribution) and the version that incorporates the prior distribution into the computation. The prior distribution is obtained by computing all 1-D marginal queries from the dataset, and each marginal is added by some noises so that the set of all marginal queries satisfies 0.02 differential privacy.

Fig. 2a and 2b reveal that an appropriate setting to α can give very good estimate answers. When α is infinitely large, our approaches, in particular the kernel PCA approach, can only utilize the linear dependency among the past queries to reduce the noises. With smaller α, high correlation between past queries is also helpful to improve the accuracy. But if we set α too small, then the estimate answers will be too biased and the RMSE grows. We also observe that the incorporation of prior distribution gives significant boost to the query accuracy, especially for the kernel PCA approach. But the effect of the prior distribution vanishes when α grows.

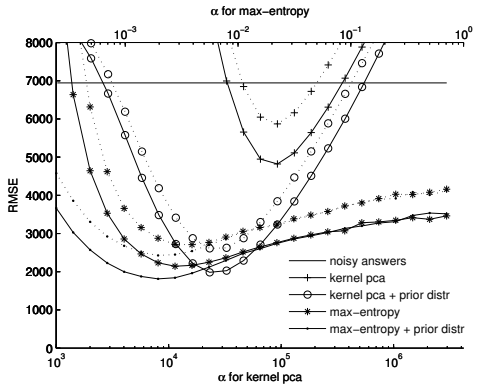


Figure 3: Estimating RMSEs by cross-validation. The dashed lines denote the estimate RMSEs obtained by 5 cross-validation on $m = 5000$ 2-D range queries. The solid lines denote the true RMSEs for the following setting — the past queries are 2-D general queries, and the RMSEs are computed on randomly generated new 2-D range queries. The solid lines match the result presented in Fig. 2c.

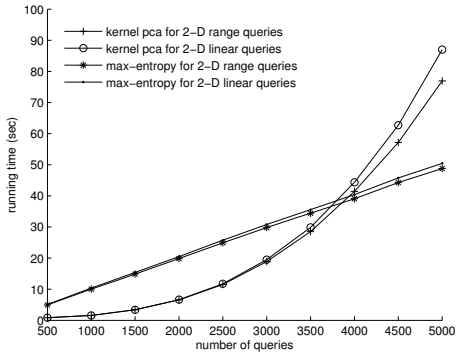
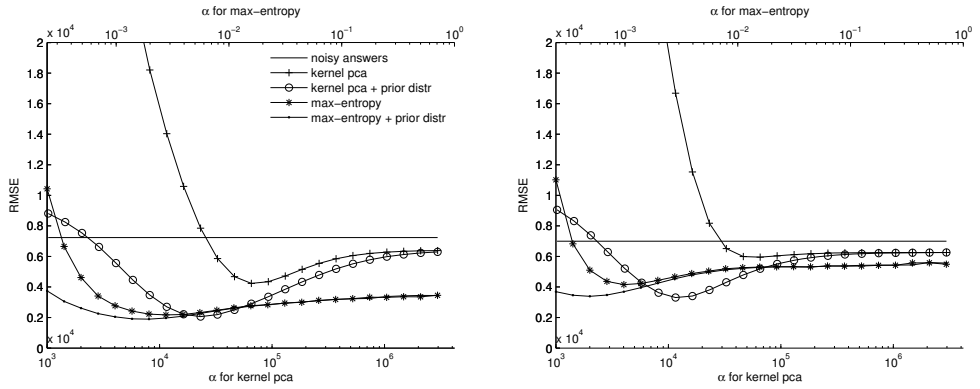


Figure 5: Running time with growing number of input queries

In above experiments we only address the accuracy of estimate answers to past queries. But the experimental results also make sense when we are more concerned about new queries, because by applying our approaches to the union of past queries and new queries, the accuracy of answers to new queries also improves.

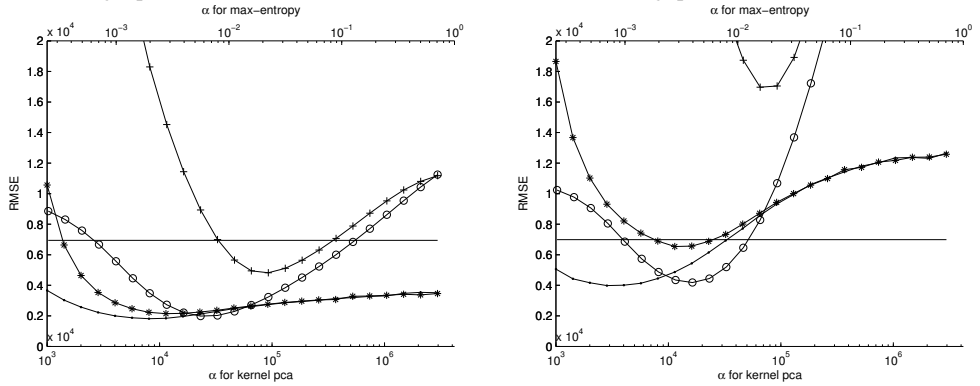
In next experiments, we consider another scenario where we derive the estimate answers to new queries purely based on noisy answers to past queries. This scenario is important when the privacy budget is already exhausted but we still want to make some inference about the data. The experimental results are presented in Fig. 2c and 2d. The past queries and new queries are independently generated, hence they are expected to be very different. Moreover, in Fig. 2d the kind of new queries is not the same as the past queries. It is shown that for the kernel PCA approach to obtain reasonable answers to new queries, we have to use a small and appropriate α and tolerate biased answers. Overall, our approaches still work fine in this scenario.

Because α is important for obtaining good estimate answers, we



(a) 2-D range queries

(b) 2-D counting queries



(c) Past queries: 2-D range queries
New queries: a different set of 2-D range queries

(d) Past queries: 2-D range queries
New queries: 2-D counting queries

Figure 2: RMSE with varying α. Fig. (a) and (b) study the ability of our approaches in improving the accuracy of estimate answers to past queries. Fig. (c) and (d) investigate whether our approaches can give reasonable answers to new queries that are different from the past queries, and the derivation of the estimate answers to new queries are purely based on the noisy answers to past queries. In Fig. (c) and (d), the RMSEs of the estimate answers to new queries are presented.

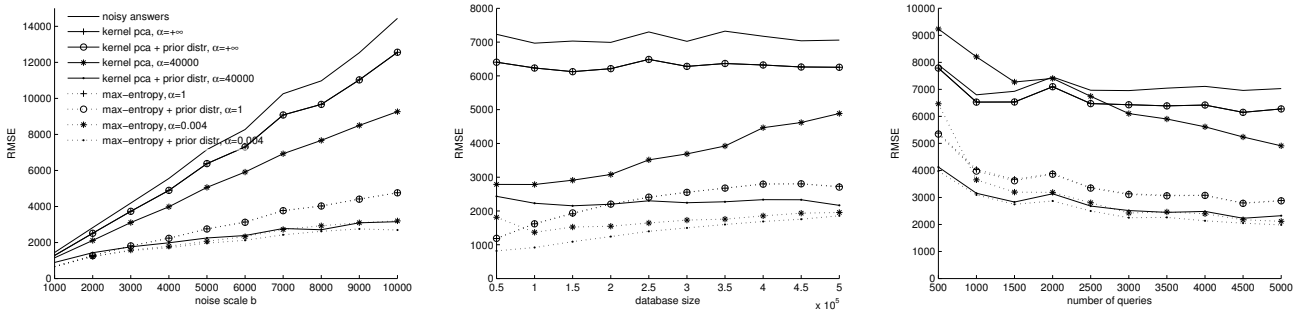
should set α very carefully. To do this, we need to know how good an assignment to α is, and choose the best possible assignment. In Fig. 3, we estimate the RMSEs via 5 cross-validation on $m = 5000$ 2-D range queries, and study how the estimate RMSEs are different from the true RMSEs. It is shown that, the curves of the estimate RMSEs are almost identical to the curves of the true RMSEs, except that the cross-validation often overestimates the RMSEs. The overestimation is because in each stage of cross-validation only a fraction $\frac{k-1}{k}$ of the m queries is used. Nonetheless, the α that minimizes the estimate RMSE usually minimizes the true RMSE too. Hence, it is often a good way to adopt estimate RMSE as the performance measure to select α. Moreover, from Fig. 2 we observe that the best choice of α is often irrelevant to whether we focus on new queries or past queries.

In Fig. 4, we show how RMSE changes with different settings to the experimental parameters. Overall, the maximum entropy approach often gives lower RMSE, and the kernel PCA approach can achieve comparable performance if we utilize prior distribution and set α appropriately. Fig. 5 presents the running time of both approaches, with growing number of input queries. Because the kernel PCA approach has $O(n^2 r)$ computation cost, it is inapplicable when there are too many queries. The computation costs for both approaches are insensitive to the database size and to the size of data universe.

5.2 Data Publishing

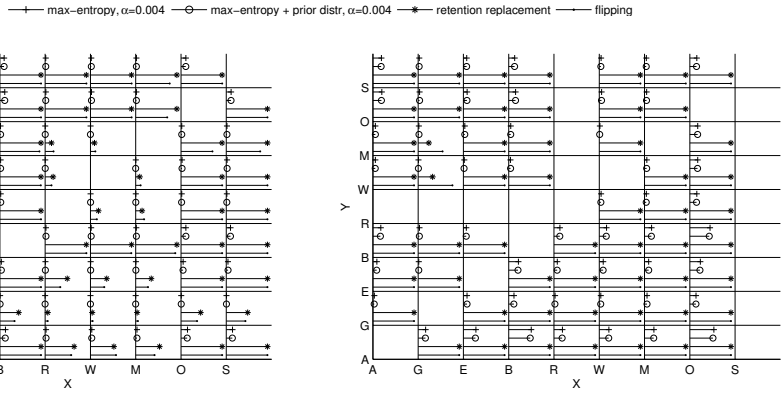
In this set of experiments, we test the maximum entropy approach as a privacy-preserving data publishing tool. Given the input database, we will first issue a collection of 2-D marginal queries to capture all bivariate relations. Specifically, for every pair of attributes and for every possible assignment to them, a counting query is issued. For each query answer, we add a Laplace noise with scale $b = d(d-1)/2 = 36$, where $d = 9$ is the dimension of the Adult dataset. Because the sensitivity of the set of all 2-D marginal queries is only $d(d-1)/2$, we achieve 1-differential privacy. There are totally 29570 2-D marginal queries. Given the noisy answers to those queries as input, the maximum entropy approach generates a synthetic database as the output, on which succeeding queries are computed for evaluating the data utility.

We compare our approach with two *randomized response* methods — *retention replace* perturbation [1] and *flipping* perturbation [20]. They guarantee γ-amplification privacy by independently perturbing every attribute value. *Retention replacement* replaces each value by another value with certain probability, and *flipping* represents each value as a bit-vector and randomly flips every bit. We set $\gamma = e^{\epsilon}$ so that γ-amplification is implied by ϵ-differential privacy, but the converse is not true. Hence, the experiments are not only fair, but even favorable to competing algorithms because our approach offers stronger privacy guarantee. We did not test other differentially private data publishing methods [4, 11] due to the prohibitively expensive computation cost for large data universe $|U|$.



(a) RMSE vs noise scale (b) RMSE vs database size (c) RMSE vs number of input queries

Figure 4: RMSE with varying parameters for 2-D range queries



(a) $Err(X, Y)$ (b) left-top: $Err(X, Y, Race)$; right-bottom: $Err(X, Y, Salary)$

Figure 6: Errors of estimate bivariate and trivariate joint distributions. Each row and column specifies an attribute as a dimension of the joint distribution. Those error values greater than 1 have been truncated to 1.

To measure the utility of the output database, we compute the estimate joint distribution for a chosen set of attributes, then examine how different the answer is from the true distribution. The error of an estimate (bivariate) joint distribution is defined as

$$Err(X, Y) = \sum_{x,y} |p(x, y) - \tilde{p}(x, y)| \tag{42}$$

where p is computed based on the output database, and $\tilde{p}$ denotes the true distribution of the input database. The same definition also applies to multivariate case. If the attributes have many possible values, the data universe is partitioned into smaller granularity with fewer supporting samples at each location, resulting in greater error due to sampling variance.

We first investigate whether the bivariate relations can be preserved in the output database, as presented in Fig. 6a. It is shown that the maximum entropy approach achieves the best utility in all cases. Randomized response methods behave well only when attribute domains are small. When an attribute has too many possible values, randomized response methods give extremely poor performance.

Comparison based on the ability for capturing bivariate relations might be unfair for randomized response methods, because in the experiments our approach has used bivariate statistics as training features. We also examined the errors of estimate trivariate joint distributions, as reported in Fig. 6b. We can observe that randomized response methods no longer give meaningful results, while the performance of the maximum entropy approach is still reasonable. Finally, in Fig. 7, we present the RMSEs for randomly generated k-D range queries. The maximum entropy approach shows remarkable advantage.

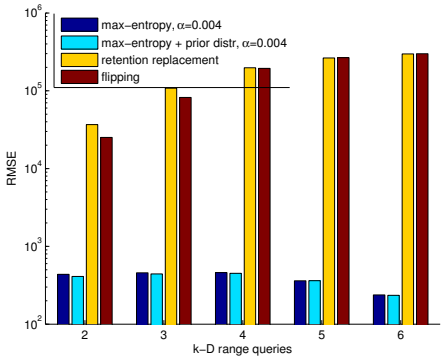


Figure 7: RMSE of k-D range queries

6. RELATED WORK

Since differential privacy was introduced [10], a lot of work has been dedicated to data publishing mechanisms and data analysis techniques that suffice differential privacy [2, 4, 5, 8, 9, 11–15, 17, 19, 21, 22, 24, 25]. Due to linear queries' simplicity and importance, most of existing work focuses on answering linear queries and seeks to develop techniques that can outperform Laplace mechanism. For example, a class of work [4, 11, 13, 22] studied data publication mechanisms that output a synthetic dataset to simultaneously answer all or most counting queries in a given concept. Another work, matrix mechanism [17], optimizes a batch of linear queries by invoking Laplace mechanism on a carefully

chosen query sequence that is called a query strategy. K-norm mechanism [14] proposes to draw noise from a more complex high dimensional joint distribution that is tailored to the geometry of the query sequence. However, all of these techniques (except Laplace mechanism) require computation cost that is at least polynomial or even super-polynomial in the size of data domain and the number of queries (unless in a very special situation). In particular, matrix mechanism [17] tries to solve two semidefinite programs iteratively, and each semidefinite program has size $O(mN)$ and can be solved in roughly $O(m^3N^3)$ time. Furthermore, the number of iterations required for convergence is unknown, and the algorithm does not guarantee to terminate in polynomial time. K-norm mechanism [14] and its improved version are based on uniformly sampling from a m-dimensional convex polytope that has $O(N)$ vertices, which is very time consuming. The computation cost for obtaining an almost uniform sample from the polytope is roughly $O(m^3N^3)$, while the improved version demands $O(m^3)$ such approximately independent samples. The multiplicative weights mechanism [13] is one of the most efficient techniques in the literature, which has $\tilde{O}(mN|T|)$ running time. The time of this mechanism can be further reduced to sublinearity in N if the data is drawn from a so called smooth distribution — a distribution that does not place too much weight on any single data item (that is, it can not be too skewed and should be close to a uniform distribution). Because real datasets often have very large data domains, especially for datasets that have many attributes or dimensions, the computation cost of such techniques is prohibitively expensive for practical use.

Meanwhile, many studies concentrate on more specific data mining tasks for pursuing better performance. One mostly studied subject is the release of low dimensional histograms or marginals from private data. Such work includes but not limited to [2, 8, 15, 25], where consistency is often an important issue because consistency can improve query accuracy and data utility. Usually, computation cost for these techniques is (near-)linear to the number of queries, since more specific problems allow better and simpler solutions.

7. CONCLUSIONS

In this paper, we propose the kernel PCA approach and the maximum entropy approach to integrate all available linear query answers into a consistent form that embodies our knowledge learned from the noisy answers, so that more accurate answers to past queries are obtained. The running time of our approaches does not explicitly depend on the cardinality of data universe, hence they can be applied to high dimensional data with very large domain.

8. ACKNOWLEDGEMENT

This work was supported by the National Natural Science Foundation of China (NSFC) under grant No. 60873070.

9. REFERENCES

[1] R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving olap. In *SIGMOD Conference*, pages 251–262, 2005.

[2] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS*, pages 273–282, 2007.

[3] A. L. Berger, S. D. Pietra, and V. J. D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[4] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.

[5] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *NIPS*, pages 289–296, 2008.

[6] S. Chen and S. Zhou. Integrating historical noisy answers for improving data utility under differential privacy. Technical Report #2011-100, School of Computer Science, Fudan University, 2011.

[7] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[8] B. Ding, M. Winslett, J. Han, and Z. Li. Differentially private data cubes: optimizing noise sources and consistency. In *SIGMOD Conference*, pages 217–228, 2011.

[9] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[11] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, pages 381–390, 2009.

[12] M. Gupte and M. Sundararajan. Universally optimal privacy mechanisms for minimax agents. In *PODS*, pages 135–146, 2010.

[13] M. Hardt and G. N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, pages 61–70, 2010.

[14] M. Hardt and K. Talwar. On the geometry of differential privacy. In *STOC*, pages 705–714, 2010.

[15] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1):1021–1032, 2010.

[16] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[17] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, pages 123–134, 2010.

[18] D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambrdige University Press, 2003. http://www.inference.phy.cam.ac.uk/mackay/itprnn/book.html.

[19] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

[20] N. Mishra and M. Sandler. Privacy via pseudorandom sketches. In *PODS*, pages 143–152, 2006.

[21] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.

[22] A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *STOC*, pages 765–774, 2010.

[23] C. Sutton and A. McCallum. An Introduction to Conditional Random Fields. *ArXiv e-prints*, Nov. 2010.

[24] X. Xiao, G. Bender, M. Hay, and J. Gehrke. ireduct: differential privacy with reduced relative errors. In *SIGMOD Conference*, pages 229–240, 2011.

[25] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, pages 225–236, 2010.