Maximum Likelihood Postprocessing for Differential Privacy under Consistency Constraints

Jaewoo Lee
jlee@cse.psu.edu

Yue Wang
yuw140@cse.psu.edu

Daniel Kifer
dkifer@cse.psu.edu

Department of Computer Science and Engineering
Penn State University

ABSTRACT

When analyzing data that has been perturbed for privacy reasons, one is often concerned about its usefulness. Recent research on differential privacy has shown that the accuracy of many data queries can be improved by post-processing the perturbed data to ensure consistency constraints that are known to hold for the original data.

Most prior work converted this post-processing step into a least squares minimization problem with customized efficient solutions. While improving accuracy, this approach ignored the noise distribution in the perturbed data.

In this paper, to further improve accuracy, we formulate this post-processing step as a constrained maximum likelihood estimation problem, which is equivalent to constrained L_1 minimization. Instead of relying on slow linear program solvers, we present a faster generic recipe (based on ADMM) that is suitable for a wide variety of applications including differentially private contingency tables, histograms, and the matrix mechanism (linear queries).

An added benefit of our formulation is that it can often take direct advantage of algorithmic tricks used by the prior work on least-squares post-processing.

An extensive set of experiments on various datasets demonstrates that this approach significantly improve accuracy over prior work.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Statistical Database

Keywords

Differential privacy; ADMM; PPDM; Post-processing

1. INTRODUCTION

Many publicly available datasets have been altered to protect privacy. In many cases, users were encouraged to use this perturbed data in exactly the same way they would have used the original data. However, recent work has shown that post-processing the data with custom statistical inference algorithms can significantly improve accuracy of the results [20, 11, 14, 15, 6].

One of the biggest sources of improvement is enforcing *consistency constraints* — the modification of perturbed data so that it satisfies constraints that are known to hold in the original data. For example, suppose C_{nj} is the number of cancer patients in New Jersey, C_{ny} is the number in New York, and $C_{\text{all}} = C_{\text{nj}} + C_{\text{ny}}$. Suppose independent noise is added to each variable and the resulting noisy quantities $\widetilde{C}_{\text{ny}}$, $\widetilde{C}_{\text{nj}}$, $\widetilde{C}_{\text{all}}$ are released. The original variables satisfy some obvious constraints: $C_{\text{nj}} \geq 0$, $C_{\text{ny}} \geq 0$, $C_{\text{all}} = C_{\text{nj}} + C_{\text{ny}}$ but that is not necessarily true of their noisy versions. Thus we could seek new estimates, C'_{ny}, C'_{nj}, C'_{all} that are close to $\widetilde{C}_{\text{ny}}$, $\widetilde{C}_{\text{nj}}$, $\widetilde{C}_{\text{all}}$ and satisfy those constraints.

The most common approach is to formulate such problems in terms of least squares estimation [11, 13, 17]: choose the C'_{ny}, C'_{nj}, C'_{all} that minimize

$$(C'_{\text{ny}} - \widetilde{C}_{\text{ny}})^2 + (C'_{\text{nj}} - \widetilde{C}_{\text{nj}})^2 + (C'_{\text{all}} - \widetilde{C}_{\text{all}})^2 \qquad (1)$$

subject to $C'_{\text{ny}} + C'_{\text{nj}} = C'_{\text{all}}$ (sometimes dropping the inequality constraints). The resulting estimates $C'_{\text{nj}}, C'_{\text{ny}}$ of the number of cancer patients will generally be more accurate than the unprocessed noisy values $\widetilde{C}_{\text{ny}}$, $\widetilde{C}_{\text{nj}}$.

Such a formulation has two drawbacks – it drops the inequality constraints and it ignores the distribution of noise that was added to each variable. In the case of differential privacy [7], the noise often comes from the Laplace distribution with density $f(x;\alpha) = \frac{1}{2\alpha}e^{-|x|/\alpha}$. If we want constrained maximum likelihood (instead of least squares) estimates C'_{ny}, C'_{nj}, C'_{all} for C_{ny}, C_{nj}, C_{all}, the optimization problem is to find C'_{ny}, C'_{nj}, C'_{all} that minimize

$$|C'_{\text{ny}} - \widetilde{C}_{\text{ny}}| + |C'_{\text{nj}} - \widetilde{C}_{\text{nj}}| + |C'_{\text{all}} - \widetilde{C}_{\text{all}}| \qquad (2)$$

subject to $C'_{\text{ny}} + C'_{\text{nj}} = C'_{\text{all}}$, $C'_{\text{ny}} \geq 0$, $C'_{\text{nj}} \geq 0$.

Why was the least squares approach preferred in prior work? First, it had a unique solution (not guaranteed in Equation (2)). Second without inequality constraints, the problem becomes differentiable and fast custom algorithms (that do not use matrix inversion or decomposition) can be developed [11].

In this paper, we leverage the alternating direction method of multipliers [3] to provide a *generic* recipe for developing maximum likelihood post-processing algorithms for many differential privacy applications, including releasing contingency tables [2], histograms [11, 1, 21], frequent itemsets [12], batch query processing [13, 22], etc.

KDD'15, August 11–14, 2015, Sydney, NSW, Australia
Copyright 2015 ACM 978-1-4503-3664-2/15/08 ...$15.00.
DOI: http://dx.doi.org/10.1145/2783258.2783366.

We specialize our approach for three target applications: differentially private contingency tables, histograms, and linear queries (via the matrix mechanism [13]). The advantages of our proposed algorithms include the following.

- They outperform (in terms of accuracy) prior work that uses least squares estimation.
- They outperform (in terms of speed) equivalent linear programs.
- They can directly take advantage of many algorithmic tricks that were used to speed up least squares algorithms in other differential privacy applications.
- They can be easily modified to provide unique solutions.

The rest of the paper is organized as follows. In Section 2, related works are discussed. Section 3 defines the notations used throughout the paper and provides background knowledge about differential privacy. We discuss the applications of the proposed algorithm to contingency tables, histogram and matrix mechanism and introduce the least squares formulation. Section 4 describes the details of the proposed algorithm. In Section 5, we provide algorithms for our target applications and their performance evaluation on a variety of datasets. Finally, Section 6 concludes our work.

2. RELATED WORKS

The problem of finding an estimate of data or query answers that satisfies consistency constraints was first considered by Barak et al. [2]. They transform the data into Fourier domain and then add random noise to Fourier coefficients. They use linear programming to obtain non-negative entries in contingency tables. Their objective function, however, is designed not to maximize the likelihood but to minimize the tolerance on the constraint violation. Although they do not formulate their problem as maximum likelihood estimation, the resulting linear program can be solved using the techniques discussed in this paper.

Hay et al. [11] introduced a post-processing algorithm customized for releasing histograms. To release a histogram, they build a query sequence such that linear relationships are established between query answers. Those linear relationships are used as a consistency constraint during post-processing. The post-processing algorithm finds an estimate that minimizes the L_2 distance to the noisy answer among those which satisfy the constraint; in other words, it performs least squares under the linear constraint. This formulation, however, does not utilize the knowledge of noise distribution (another source of improvement).

The least squares based post-processing technique has been widely adopted in the literature for generating consistent estimates. For example, [18] used it to maintain the consistency of marginal counts between multiple contingency tables, [5, 12] to generate consistent hierarchical tree data structures, [6] to derive consistent marginal counts between data cuboids, and [16] to produce consistent spatial grids.

Lin and Kifer [14] applied a Bayesian approach to estimate sorted histograms and pointed out that substantial improvements in accuracy can be obtained if the estimation procedure makes use of knowledge on the noise distribution. They proposed a sophisticated estimation algorithm that views sorted histograms as a Markov chain and imposes ordering constraints on the estimates. The employment of such a complex model, however, incurs high computational complexity and limits the general applicability of their approach to other problems.

Symbol	Description
$\tilde{\mathbf{x}}$	noisy version of $\mathbf{x}$, $\mathbf{x} + \mathsf{Lap}(\lambda)$
$(\mathbf{x})_+$	vector with components $(x_i)_+ = \max\{0, x_i\}$
$\mathbf{x} \succeq 0$	$\forall i, x_i \geq 0$
$\mathbf{I}$	an identity matrix

Table 1: Notations

Instead of maximum likelihood, Williams and McSherry [20] used a variational bayesian approximation to make estimates about the data. However, it cannot make use of linear and non-negativity constraints.

Li et al. [13] introduced a general framework, called matrix mechanism, for answering linear queries. Given a strategy query $\mathbf{A}$, the mechanism reconstructs the original input database from the noisy answer to $\mathbf{A}$ by solving a least squares problem. However, the use of least square solution does not take the noise distribution into account.

Smith [19] showed that, for statistical estimators T with asymptotic normality, there exists a differentially private estimator A_T with the same asymptotic distribution as T. His result applies to maximum likelihood estimators for certain parametric model families, which is different from the problem considered in this paper in that we estimate the data or query answers under consistency constraints.

We now review two algorithms used as baseline performance measure in our experiments.

Privlet [21] uses the domain transform technique for histogram publishing. Given a histogram, it applies Haar wavelet transform on the data to get wavelet coefficients and adds noise to the coefficients. The count of any arbitrary range query is represented as a linear sum of noisy coefficients.

The multiplicative weights mechanism (MW) [10] is a synthetic database based approach. It start with a poor estimates to the true database and iteratively refines the estimate. In each round, using the exponential mechanism, it identifies a query on which their estimate differs most significantly from the true database, and then the noisy answer to that query is used to improve the estimate.

3. PRELIMINARIES

3.1 Notation

To denote vectors, we use boldface lowercase letters, e.g., $\mathbf{v}$. All vectors are assumed to be column vectors, unless noted otherwise. For a vector $\mathbf{v}$, the i^{th} component of $\mathbf{v}$ is denoted by v_i. Matrices are written by boldface uppercase letters, e.g., $\mathbf{A}$. The transpose of a vector $\mathbf{v}$ and a matrix $\mathbf{A}$ are represented by $\mathbf{v}^\intercal$ and $\mathbf{A}^\intercal$, respectively. For $p \in [1, \infty)$, the L_p norm of a vector $\mathbf{v} \in \mathbb{R}^n$ is defined as $\|\mathbf{v}\|_p = (\sum_{i\in[n]} |x_i|^p)^{1/p}$. The notation $(\mathbf{v})_+$ is used to represent a vector obtained by replacing negative values with 0, i.e., $\forall i$, its component $(v_i)_+ = \max\{v_i, 0\}$. An identity matrix is denoted by $\mathbf{I}$. For a vector $\mathbf{x}$, $\tilde{\mathbf{x}}$ denotes $\mathbf{x}$ with Laplace noise added to each component. We use the notation $\mathbf{x} \succeq 0$ to denote that $\mathbf{x}$ contains no negative components (i.e., $\forall i, x_i \geq 0$). All notations are summarized in Table 1.

3.2 Differential Privacy

A database $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ is a (multi)set of records, where each $\mathbf{x}_i \in \mathcal{U}$ corresponds to an individual and each element of $\mathbf{x}_i$ represents an attribute value. Two databases X and X' are referred to as *neighboring* if one can be obtained by adding or removing one record from the other, i.e., $|(X - X') \cup (X' - X)| = 1$. Informally, differential privacy protects individuals by ensuring that the output distribution of an algorithm is not significantly affected by the presence/absence of any individual in the database. The formal definition of differential privacy is as follows:

DEFINITION 1 (ϵ-DIFFERENTIAL PRIVACY). *Given a query function q, a randomized algorithm $\mathcal{K}$ satisfies ϵ-differential privacy if for every possible pair of neighboring databases X and X' and $\forall S \subseteq \mathrm{Range}(\mathcal{K})$,*

$$\mathrm{P}\left[\mathcal{K}_q(X) \in S\right] \leq e^{\epsilon}\, \mathrm{P}\left[\mathcal{K}_q(X') \in S\right] .$$

A mechanism well-known to achieve ϵ-differential privacy is the Laplace mechanism [7], which adds random noise to the output of a query function. The magnitude of noise to add depends on how sensitive the output of a query function is to a small change in the input. This notion is captured by the concept of sensitivity [8]. A query q is a function that maps a database X to a vector in $\mathbb{R}^d$, i.e., $q : \mathcal{U}^{|X|} \to \mathbb{R}^d$.

DEFINITION 2 (SENSITIVITY). *For two neighboring databases X and X', the sensitivity of a query function q is defined as*

$$\Delta q = \max_{X, X' \in \mathcal{U}} \left\| q(X) - q(X') \right\|_1 .$$

The sensitivity is the upper bound on the impact any individual can have on the outcome of function and this amount of uncertainty is required to make two neighboring databases indistinguishable. It is a function of query type and not dependent on the data.

DEFINITION 3 (LAPLACE MECHANISM). *Given a query function q :, the algorithm $\mathcal{A}$ that adds i.i.d. noise to each element of $q(X)$ achieves ϵ-differentially privacy.*

$$\mathcal{A}(X) = q(X) + \langle \mathsf{Lap}\,(\Delta q/\epsilon) \rangle^d$$

3.3 Applications

Contingency tables. Let $K = \{1, \cdots, k\}$. Consider a database D with k categorical attributes $(A_1, \cdots, A_k)$ where, for $j \in K$, each A_j can take values from $\mathcal{I}_j = \{1, \cdots, I_j\}$. A k-way contingency table with entries $\mathbf{x} = \{x(i_1, \cdots, i_k)\}$ is a k-dimensional array of non-negative integers. With an ordering function $\Psi : \mathcal{I}_K \to \mathbb{N}$ that maps a multi-index to a one-dimensional index lexicographically, a contingency table can be viewed as a vector $\mathbf{x} \in \mathbb{N}^n$ where $n = \prod_{j=1}^{k} I_j$. For $B = \{i_1, \cdots, i_b\} \subseteq K$, let $\mathcal{I}_B = \mathcal{I}_{i_1} \times \cdots \times \mathcal{I}_{i_b}$. Given $b \in \mathcal{I}_B$, b-marginal is obtained by summing over all other attributes.

$$\mathfrak{m}(b) = \sum_{j \in K \setminus B} x(b, j) = \sum_{j \in K \setminus B} x_{\Psi(b,j)} \tag{3}$$

Notice that a marginal count $\mathfrak{m}(b)$ is a linear constraint on the contingency table $\mathbf{x}$: $a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = \mathfrak{m}(b)$. where a_i is 1 if the projection of $\Psi^{-1}(i)$ on $\mathcal{I}_B = b$ and 0 otherwise. Let $\mathfrak{m}(B) = \{\mathfrak{m}(b) \mid b \in \mathcal{I}_B\}$. Given $\mathfrak{m}(B)$ and a noisy contingency table $\tilde{\mathbf{x}}$, the objective of post-processing is to find a more accurate table $\bar{\mathbf{x}}$ that is consistent with $\mathfrak{m}(B)$.

Histograms. Let $H = (h_1, h_2, \cdots, h_d)$ be a histogram of length $d = k^{\ell}$. As in [11], given a histogram H, we build a complete k-ary tree such that each leaf node corresponds to a unit interval and a parent node covers the union of intervals of its k children. A complete k-ary tree can easily be converted to a vector of length $n = \frac{k^{\ell+1}-1}{k-1}$ by storing the tree in breadth-first order where ℓ is the height of tree. Let $\mathbf{x} = (x_1, \cdots, x_n)$ be a vector representing the tree. The j^{th} child of x_i has the index $k(i-1) + j + 1$. Thus, $\mathbf{x}$ satisfies the following linear constraint.

$$x_i = \sum_{j=1}^{k} x_{k(i-1)+j+1}, \quad \text{for } i = 1, \cdots, \frac{k^{\ell} - 1}{k - 1} \tag{4}$$

Given a noisy tree $\tilde{\mathbf{x}}$, the goal of post-processing is to find a new tree (with non-negative counts) $\bar{\mathbf{x}}$ that is closest to $\tilde{\mathbf{x}}$ and, at the same time, satisfies the constraint (4).

Matrix mechanism (MM). Suppose a database $\mathbf{x}$ that contains a set of non-negative integers. Given the strategy matrix $\mathbf{A}$, MM produces the noisy answer $\tilde{\mathbf{r}} = \mathbf{A}\mathbf{x} + \mathsf{Lap}\,(\Delta_{\mathbf{A}}/\epsilon)$ according to the Laplace mechanism. From $\tilde{\mathbf{r}}$, the original input database is conveniently approximated by the *least squares solution*:

$$\hat{\mathbf{x}} = \arg\min_{\bar{\mathbf{x}}} \frac{1}{2}\|\tilde{\mathbf{r}} - \mathbf{A}\bar{\mathbf{x}}\|_2^2 = \mathbf{A}^{\dagger}\tilde{\mathbf{r}} = (\mathbf{A}^{\intercal}\mathbf{A})^{-1}\mathbf{A}^{\intercal}\tilde{\mathbf{r}}$$

where $\mathbf{A}^{\dagger}$ is the pseudo-inverse of $\mathbf{A}$. To achieve maximum likelihood, the post-processing should solve the following optimization problem:

$$\bar{\mathbf{x}} = \arg\min_{\hat{\mathbf{x}} \succeq 0} \|\tilde{\mathbf{r}} - \mathbf{A}\hat{\mathbf{x}}\|_1 . \tag{5}$$

3.4 Least Squares Formulation

The equality constraints in all of the above applications are linear and can be represented by

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = b .$$

A set of p such linear equations can be expressed concisely in matrix notation as $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A} \in \mathbb{R}^{p \times n}$ and $\mathbf{b} \in \mathbb{R}^p$. The post-processing problem for the above applications shares the common form, and previous approaches [11, 13, 5, 12] formulated it as a constrained least squares problem:

$$\begin{array}{ll} \underset{\mathbf{x}}{\text{minimize}} & \|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2 \\ \text{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b} . \end{array} \tag{6}$$

For algorithmic simplicity, the inequality constraint is often excluded. The main disadvantage of the above formulation is that it doesn't achieve maximum likelihood. Provided that the query answers are perturbed with Laplace noise, maximizing the likelihood function is equivalent to minimizing $\sum_{i=1}^{n} |x_i - \tilde{x}_i| = \|\mathbf{x} - \tilde{\mathbf{x}}\|_1$, as shown in Section 4.1.

4. THE PROPOSED ALGORITHM FOR CONSTRAINED INFERENCE

In this section, we describe each step of the proposed ADMM algorithm in detail. For details on ADMM, refer to [3] and the references therein.

4.1 L_1 Formulation

We use maximum likelihood to estimate the unknown dataset $\mathbf{x}$, not model parameters, from the noisy data $\tilde{\mathbf{x}}$.

In most cases, $\tilde{\mathbf{x}} = \mathbf{x} + \mathsf{Lap}(\lambda)$ (independent multivariate Laplace noise). Thus, the log likelihood function is

$$\ln L(\mathbf{x}) = \ln \mathrm{P}[\tilde{\mathbf{x}} \mid \mathbf{x}] = \ln \prod_{i=1}^{n} \left(\frac{1}{2\lambda} \exp\left(-\frac{|x_i - \tilde{x}_i|}{\lambda} \right) \right)$$
$$\propto -\sum_{i=1}^{n} |x_i - \tilde{x}_i| + \mathsf{const} = -\|\mathbf{x} - \tilde{\mathbf{x}}\|_1 + \mathsf{const}\,.$$

Therefore, the maximum likelihood estimator $\bar{\mathbf{x}}$ that satisfies consistency constraints known to hold over original data is the solution of optimization problems such as

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \|\mathbf{x} - \tilde{\mathbf{x}}\|_1 && \text{(7a)} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} && \text{(7b)} \\ & \mathbf{x} \succeq 0 && \text{(7c)} \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^n$. The above problem arises from various applications in differential privacy, where the constraints (7b) and (7c) represent the auxiliary partial information that data users may have before the data release. We note that having the constraints in our setting is not a requirement but a choice.

To derive an ADMM algorithm, we reformulate the problem (7) as follows:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y},\mathbf{z}}{\text{minimize}} \quad & \|\mathbf{y}\|_1 + \mathbb{I}_{\mathcal{C}}(\mathbf{z}) && \text{(8a)} \\ \text{subject to} \quad & \mathbf{x} - \tilde{\mathbf{x}} - \mathbf{y} = 0 && \text{(8b)} \\ & \mathbf{x} - \mathbf{z} = 0 && \text{(8c)} \\ & \mathbf{A}\mathbf{x} - \mathbf{b} = 0 && \text{(8d)} \end{aligned}$$

where $\mathcal{C} = \{\mathbf{x} \mid \mathbf{x} \succeq 0\}$ and $\mathbb{I}_{\mathcal{C}}(\mathbf{z})$ denotes an indicator function defined as $\mathbb{I}_{\mathcal{C}}(\mathbf{x}) = 0$ if $\mathbf{x} \in \mathcal{C}$, ∞ otherwise. The first term in the objective function measures L_1 distance between the solution and the noisy answer while the second term, if included, ensures the non-negativity of the solution. By introducing a new variable $\mathbf{z}$, the inequality constraint has been absorbed into the objective function.

With the scaled dual variables [3], the dual augmented problem of (8) can be written as

$$\max_{\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{\eta}} \min_{\mathbf{x},\mathbf{y},\mathbf{z}} \Big(\|\mathbf{y}\|_1 + \mathbb{I}_{\mathcal{C}}(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{x} - \tilde{\mathbf{x}} - \mathbf{y} + \boldsymbol{\mu}\|_2^2 + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z} + \boldsymbol{\nu}\|_2^2 + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b} + \boldsymbol{\eta}\|_2^2 \Big)$$

where $\rho > 0$ is a penalty parameter. Note that $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are primal variables and $\boldsymbol{\mu}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ are dual variables. The ADMM algorithm for solving the above problem consists of iterating the following steps.

$$\mathbf{y}^{k+1} = \arg\min_{\mathbf{y}} \left(\|\mathbf{y}\|_1 + \frac{\rho}{2}\|\mathbf{x}^k - \tilde{\mathbf{x}} - \mathbf{y} + \boldsymbol{\mu}^k\|_2^2 \right) \tag{9}$$

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \left(\mathbb{I}_{\mathcal{C}}(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{x}^k - \mathbf{z} + \boldsymbol{\nu}^k\|_2^2 \right) \tag{10}$$

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \Big(\frac{1}{2}\|\mathbf{x} - \tilde{\mathbf{x}} - \mathbf{y}^{k+1} + \boldsymbol{\mu}^k\|_2^2 + \frac{1}{2}\|\mathbf{x} - \mathbf{z}^{k+1} + \boldsymbol{\nu}^k\|_2^2 + \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b} + \boldsymbol{\eta}\|_2^2 \Big) \tag{11}$$

$$\begin{aligned} \boldsymbol{\mu}^{k+1} &= \boldsymbol{\mu}^k + \mathbf{x}^{k+1} - \tilde{\mathbf{x}} - \mathbf{y}^{k+1} \\ \boldsymbol{\nu}^{k+1} &= \boldsymbol{\nu}^k + \mathbf{x}^{k+1} - \mathbf{z}^{k+1} \\ \boldsymbol{\eta}^{k+1} &= \boldsymbol{\eta}^k + \mathbf{A}\mathbf{x}^{k+1} - \mathbf{b}\,. \end{aligned}$$

The efficiency of this iterative algorithm is dependent on how efficiently each subproblem can be solved.

4.2 Solving subproblems

The subproblems in our formulation are in the form of

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left(\phi(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x} - \boldsymbol{\upsilon}\|_2^2 \right).$$

These subproblems have been actively studied in other fields and, for many types of function ϕ, there exists a closed form solution. Here we only introduce solutions for the subproblems that are shared by all of applications considered in this paper. For more examples, refer to [4].

(i) When $\phi(\mathbf{x}) = \mathbb{I}_{\mathcal{C}}(\mathbf{x})$, the subproblem reduces to a *Euclidean projection* of $\boldsymbol{\upsilon}$ onto a closed convex set $\mathcal{C}$ and it is denoted by

$$\mathbf{x}^{k+1} = \Pi_{\mathcal{C}}(\boldsymbol{\upsilon})\,. \tag{12}$$

See [3, 4] for examples of how to calculate the projections on various convex sets.

(ii) When $\phi(\mathbf{x}) = \|\mathbf{x}\|_1$, the solution yields

$$\mathbf{x}^{k+1} = S_{1/\rho}(\boldsymbol{\upsilon}) = \left(S_{1/\rho}(\upsilon_1), \cdots, S_{1/\rho}(\upsilon_n) \right)$$

where $S_\tau(\cdot)$ is an element-wise *soft-thresholding* operator defined as

$$S_\tau(\upsilon_i) = \begin{cases} \upsilon_i - \tau\,\mathrm{sign}(\upsilon_i) & \text{if } |\upsilon_i| > \tau \\ 0 & \text{if } |\upsilon_i| \le \tau \end{cases} \tag{13}$$

(iii) When $\phi(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \boldsymbol{\omega}\|_2^2$, the subproblem reduces to a least squares problem:

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left(\frac{1}{2} \left\| \begin{bmatrix} \mathbf{A} \\ \sqrt{\rho}\mathbf{I} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \boldsymbol{\omega} \\ \sqrt{\rho}\boldsymbol{\upsilon} \end{bmatrix} \right\|_2^2 \right).$$

The solution to the above least squares problem is given by solving the following linear equation

$$(\mathbf{A}^\intercal\mathbf{A} + \rho\mathbf{I})\mathbf{x} = \mathbf{A}^\intercal\boldsymbol{\omega} + \rho\boldsymbol{\upsilon}\,.$$

(iv) When $\phi(\mathbf{x}) = \alpha\|\mathbf{x}\|_1 + (1-\alpha)\|\mathbf{x}\|_2^2$, the solution is obtained by applying a thresholding operator $\Theta_{\alpha,\rho}$ to each element:

$$\mathbf{x}^{k+1} = \Theta_{\alpha,\rho}(\boldsymbol{\upsilon}) = (\Theta_{\alpha,\rho}(\upsilon_1), \cdots, \Theta_{\alpha,\rho}(\upsilon_n))$$

where $\Theta_{\alpha,\rho}(\upsilon_i)$ is defined as

$$\Theta_{\alpha,\rho}(\upsilon_i) = \begin{cases} 0 & \text{if } |\upsilon_i| \le \frac{\alpha}{\rho} \\ \frac{\rho\left(\upsilon_i - \frac{\alpha}{\rho}\,\mathrm{sign}(\upsilon_i)\right)}{2(1-\alpha)+\rho} & \text{if } |\upsilon_i| > \frac{\alpha}{\rho}\,. \end{cases} \tag{14}$$

4.3 General Recipe

Armed with closed form solutions to subproblems, we now describe the general recipe for the maximum likelihood post-processing under consistency constraints. The first step is to rewrite the original problem in ADMM form. The general rule of thumb is to split terms in the objective function that are difficult to optimize simultaneously by introducing a new variable, so that each subproblem has a simple closed form solution. In (8), variable $\mathbf{y}$ is substituted with terms in L_1 norm in the original problem, which results in the constraint (8b). If a constraint represents a convex set on which

Algorithm 1: The proposed ADMM algorithm

```
    Input: $\mathbf{A}, \mathbf{b}, \tilde{\mathbf{x}}, \rho$
    Output: a private estimate $\bar{\mathbf{x}}$
 1  $\mathbf{x}^0 \leftarrow \tilde{\mathbf{x}}$, $\boldsymbol{\mu}^0 = \boldsymbol{\nu}^0 = \boldsymbol{\eta}^0 \leftarrow 0$, $k \leftarrow 0$
 2  LU $\leftarrow$ chol$(\mathbf{A}^\intercal\mathbf{A} + 2\mathbf{I})$        ▷ Cholesky decomposition
 3  repeat
 4      $\mathbf{y}^{k+1} \leftarrow S_{1/\rho}(\mathbf{x}^k - \tilde{\mathbf{x}} + \boldsymbol{\mu}^k)$        ▷ see (13)
 5      $\mathbf{z}^{k+1} \leftarrow (\mathbf{x}^k + \boldsymbol{\nu}^k)_+$        ▷ see (12)
 6      $\mathbf{x}^{k+1} \leftarrow \mathbf{U}^{-1}(\mathbf{L}^{-1}($ r.h.s. of (15)$))$   ▷ substitution
 7      $\boldsymbol{\mu}^{k+1} \leftarrow \boldsymbol{\mu}^k + \mathbf{x}^{k+1} - \tilde{\mathbf{x}} - \mathbf{y}^{k+1}$
 8      $\boldsymbol{\nu}^{k+1} \leftarrow \boldsymbol{\nu}^k + \mathbf{x}^{k+1} - \mathbf{z}^{k+1}$
 9      $\boldsymbol{\eta}^{k+1} \leftarrow \boldsymbol{\eta}^k + \mathbf{A}\mathbf{x}^{k+1} - \mathbf{b}$
10      $k \leftarrow k + 1$
11  until (22) is met
12  return x
```

an efficient projection algorithm exists, the constraint can be removed by introducing a new variable. The introduction of the indicator function $\mathbb{I}_{\mathcal{C}}(\mathbf{z})$ in the objective function enables to replace the non-negativity constraint (7c) with the simple constraint (8c). Notice that the non-differentiable term, L_1 norm of $\mathbf{x} - \tilde{\mathbf{x}}$, is completely decoupled by introducing auxiliary variables, $\mathbf{y}$ and $\mathbf{z}$.

The second step is to determine the solver for each subproblem. A solver that exploits the structure of the problem is desired.

The $\mathbf{x}$-update in Algorithm 1 is a standard least squares problem and its solution can be found by solving the following linear equation:

$$(\mathbf{A}^\intercal\mathbf{A} + 2\mathbf{I})\mathbf{x} = (\mathbf{y}^{k+1} + \tilde{\mathbf{x}} - \boldsymbol{\mu}^k) + (\mathbf{z}^{k+1} - \boldsymbol{\nu}^k) + \mathbf{A}^\intercal(\mathbf{b} - \boldsymbol{\eta}^k) \tag{15}$$

Notice that $\mathbf{A}^\intercal\mathbf{A} + 2\mathbf{I}$ is symmetric and positive definite. Let $\mathbf{G} = \mathbf{A}^\intercal\mathbf{A} + 2\mathbf{I}$ and $\mathbf{q}$ be the right hand side of the equation (15). A classical and generic method is to use a factorization of $\mathbf{G}$. When the structure of $\mathbf{G}$ or $\mathbf{A}$ (e.g., sparsity) is known, more efficient algorithms which exploit the structure can be employed. Let $\mathbf{G} = \mathbf{L}\mathbf{L}^\intercal$ be the Cholesky factorization of $\mathbf{G}$. Note that there exists a unique such $\mathbf{L}$ since $\mathbf{G} \succ 0$. Given $\mathbf{L}$, using a forward substitution followed by a backward substitution, the equation (15) can be solved directly. We denote it by $\mathbf{x}^{k+1} = \mathbf{U}^{-1}\left(\mathbf{L}^{-1}(\mathbf{q})\right)$ where $\mathbf{U} = \mathbf{L}^\intercal$. Note that, since $\mathbf{G}$ does not change throughout the iterations, the factorization needs to be calculated only once and can be cached for use in subsequent iterations. A summary of the proposed algorithm is presented in Algorithm 1.

The problem (8) can be further split by introducing another variable $\mathbf{w}$ as follows:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \|\mathbf{y}\|_1 + \mathbb{I}_{\mathcal{C}_1}(\mathbf{z}) + \mathbb{I}_{\mathcal{C}_2}(\mathbf{w}) \\ \text{subject to} \quad & \mathbf{x} - \mathbf{y} - \tilde{\mathbf{x}} = 0, \quad \mathbf{x} - \mathbf{z} = 0 \\ & \mathbf{x} - \mathbf{w} = 0 \end{aligned}$$

where $\mathcal{C}_1 = \{\mathbf{x} \mid \mathbf{x} \succeq 0\}$ and $\mathcal{C}_2 = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}\}$. This leads to a slightly different ADMM algorithm:

$$\begin{aligned} \mathbf{y}^{k+1} &= S_{1/\rho}(\mathbf{x}^k - \tilde{\mathbf{x}} + \boldsymbol{\mu}^k) \\ \mathbf{z}^{k+1} &= \Pi_{\mathcal{C}_1}(\mathbf{x}^k + \boldsymbol{\nu}^k) \\ \mathbf{w}^{k+1} &= \Pi_{\mathcal{C}_2}(\mathbf{x}^k + \boldsymbol{\eta}^k) \\ \mathbf{x}^{k+1} &= \frac{1}{3}\left\{(\mathbf{y}^{k+1} + \tilde{\mathbf{x}} - \boldsymbol{\mu}^k) + (\mathbf{z}^{k+1} - \boldsymbol{\nu}^k) + (\mathbf{w}^{k+1} - \boldsymbol{\eta}^k)\right\} \\ \boldsymbol{\mu}^{k+1} &= \boldsymbol{\mu}^k + \mathbf{x}^{k+1} - \mathbf{y}^{k+1} - \tilde{\mathbf{x}} \\ \boldsymbol{\nu}^{k+1} &= \boldsymbol{\nu}^k + \mathbf{x}^{k+1} - \mathbf{z}^{k+1} \\ \boldsymbol{\eta}^{k+1} &= \boldsymbol{\eta}^k + \mathbf{x}^{k+1} - \mathbf{w}^{k+1}. \end{aligned}$$

This algorithm is almost identical to the one derived above, except the feasibility of the linear constraint $\mathbf{A}\mathbf{x} = \mathbf{b}$ is maintained by the projection operation.

REMARK 1. *Geometrically, the Euclidean projection of a vector $\boldsymbol{v} = \mathbf{y}^k + \boldsymbol{\eta}^k$ on the set $\mathcal{C}_2$ is to find a point in $\mathcal{C}_2$ that is closest to $\boldsymbol{v}$. In other words, it is a solution to the problem:* $\underset{\mathbf{x}\in\mathcal{C}_2}{\arg\min}\ \|\mathbf{x} - \boldsymbol{v}\|_2^2$.

Notice that this exactly matches the least squares formulation (6), which implies that we can directly re-use the efficient algorithms developed in prior works on least squares based post-processing.

This algorithm is especially useful when the projection onto the affine set $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be done more efficiently than solving linear equation involving $\mathbf{A}$. We show one such case in Section 5.2.

4.4 When to stop iterating

The optimal solution of (8) satisfies the primal feasibility,

$$\begin{aligned} \mathbf{r}_1 &= \mathbf{x}^\star - \mathbf{y}^\star - \tilde{\mathbf{x}} = 0 \\ \mathbf{r}_2 &= \mathbf{x}^\star - \mathbf{z}^\star = 0 \\ \mathbf{r}_3 &= \mathbf{A}\mathbf{x}^\star - \mathbf{b} = 0 \end{aligned} \tag{16}$$

and dual feasibility,

$$0 \in \partial f_1(\mathbf{y}^\star) - \rho\boldsymbol{\mu}^\star \tag{17}$$

$$0 \in \partial f_2(\mathbf{z}^\star) - \rho\boldsymbol{\nu}^\star \tag{18}$$

$$0 \in \partial f_3(\mathbf{x}^\star) + \rho(\boldsymbol{\mu}^\star + \boldsymbol{\nu}^\star + \boldsymbol{\eta}^\star) \tag{19}$$

where $f_1(\mathbf{y}) = \|\mathbf{y}\|_1$, $f_2(\mathbf{z}) = \mathbb{I}_{\mathcal{C}}(\mathbf{z})$ and $f_3(\mathbf{x}) = 0$.

From $\mathbf{y}$-update of the algorithm, we have

$$\begin{aligned} 0 &\in \partial f_1(\mathbf{y}^{k+1}) - \rho(\mathbf{x}^k - \mathbf{y}^{k+1} - \tilde{\mathbf{x}} + \boldsymbol{\mu}^k) \\ &= \partial f_1(\mathbf{y}^{k+1}) - \rho(\boldsymbol{\mu}^k + \mathbf{x}^{k+1} - \mathbf{y}^{k+1} - \tilde{\mathbf{x}}) + \rho(\mathbf{x}^{k+1} - \mathbf{x}^k) \\ &= \partial f_1(\mathbf{y}^{k+1}) - \rho\boldsymbol{\mu}^{k+1} + \rho(\mathbf{x}^{k+1} - \mathbf{x}^k). \end{aligned} \tag{20}$$

The $\mathbf{z}$-update yields

$$\begin{aligned} 0 &\in \partial f_2(\mathbf{z}^{k+1}) - \rho(\mathbf{x}^k - \mathbf{z}^{k+1} + \boldsymbol{\nu}^k) \\ &= \partial f_2(\mathbf{z}^{k+1}) - \rho(\boldsymbol{\nu}^k + \mathbf{x}^{k+1} - \mathbf{z}^{k+1}) + \rho(\mathbf{x}^{k+1} - \mathbf{x}^k) \\ &= \partial f_2(\mathbf{z}^{k+1}) - \rho\boldsymbol{\nu}^{k+1} + \rho(\mathbf{x}^{k+1} - \mathbf{x}^k). \end{aligned} \tag{21}$$

Since $\mathbf{x}^{k+1}$ is the minimizer of subproblem (11), we have

$$\begin{aligned} 0 &= \rho(\mathbf{x}^{k+1} - \mathbf{y}^{k+1} - \tilde{\mathbf{x}} + \boldsymbol{\mu}^k) + \rho(\mathbf{x}^{k+1} - \mathbf{z}^{k+1} + \boldsymbol{\nu}^k) \\ &\quad + \rho(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{b} + \boldsymbol{\eta}^k) \\ &= \rho(\boldsymbol{\mu}^{k+1} + \boldsymbol{\nu}^{k+1} + \boldsymbol{\eta}^{k+1}). \end{aligned}$$

Therefore, the optimality of Algorithm 1 is attained when (16), (20), and (21) are satisfied. This means that the algorithm can stop iterations when

$$\|\mathbf{r}_i\|_2 \le \tau_1 \text{ for } i = 1, 2, 3 \text{ and } \|\rho(\mathbf{x}^{k+1} - \mathbf{x}^k)\|_2 \le \tau_2 \tag{22}$$

for sufficiently small tolerance values τ_1 and τ_2.

4.5 Unique Solution

One of potential drawbacks of L_1 based approach is lack of unique solution. The L_1 norm in our objective function is convex but not strongly convex, which means that its optimal solution may not be unique and sensitive to an initial guess. To avoid this, we augment the objective function by adding a strongly convex function:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \alpha\|\mathbf{x}-\tilde{\mathbf{x}}\|_1 + (1-\alpha)\|\mathbf{x}-\tilde{\mathbf{x}}\|_2^2 \\ \text{subject to} \quad & \mathbf{A}\mathbf{x}=\mathbf{b}, \quad \mathbf{x}\succeq 0 \end{aligned} \tag{23}$$

where α is a mixing parameter in the range $[0,1]$. The objective function is in the form of *elastic net* regularization function [23], and its solution is proven to converge to an L_1 solution when α is sufficiently large. The problem (23) in ADMM form is

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{w}}{\text{minimize}} \quad & \alpha\|\mathbf{y}\|_1 + (1-\alpha)\|\mathbf{y}\|_2^2 + \mathbb{I}_{\mathbb{R}^n_+}(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{x}-\mathbf{y}-\tilde{\mathbf{x}}=0 \\ & \mathbf{x}-\mathbf{z}=0 \\ & \mathbf{A}\mathbf{x}-\mathbf{b}=0 \end{aligned}$$

and the update steps of ADMM algorithm are

$$\begin{aligned} \mathbf{y}^{k+1} &= \Theta_{\alpha,\rho}(\mathbf{x}^k-\tilde{\mathbf{x}}+\boldsymbol{\mu}^k) \\ \mathbf{z}^{k+1} &= (\mathbf{x}^k+\boldsymbol{\nu}^k)_+ \\ \mathbf{x}^{k+1} &= \text{solve (24) for } \mathbf{x} \text{ using Cholesky decomposition} \\ \boldsymbol{\mu}^{k+1} &= \boldsymbol{\mu}^k+\mathbf{x}^{k+1}-\mathbf{y}^{k+1}-\tilde{\mathbf{x}} \\ \boldsymbol{\nu}^{k+1} &= \boldsymbol{\nu}^k+\mathbf{x}^{k+1}-\mathbf{z}^{k+1} \\ \boldsymbol{\eta}^{k+1} &= \boldsymbol{\eta}^k+\mathbf{A}\mathbf{x}^{k+1}-\mathbf{b} \end{aligned}$$

where the equation (24) is

$$\begin{aligned} (\mathbf{A}^\intercal\mathbf{A}+2\mathbf{I})\mathbf{x} = \mathbf{A}^\intercal(\mathbf{b}-\boldsymbol{\eta}^k) + (\mathbf{y}^{k+1}+\tilde{\mathbf{x}}-\boldsymbol{\mu}^k) \\ + (\mathbf{z}^{k+1}-\boldsymbol{\nu}^k)\,. \end{aligned} \tag{24}$$

4.6 L_2 Formulation

The constrained inference algorithm presented in [11] is efficient but is specific to hierarchical tree constraints. In addition, it is unclear how to adjust the algorithm to incorporate additional constraints, e.g., non-negativity of the solution or box constraints. With our framework, a general L_2 minimization algorithm for Problem (7) that works with any linear constraints can be easily derived. The only difference with (8) in Section 4.1 is that the first term in the objective function is expressed with L_2 norm; hence, the change is only in the $\mathbf{x}$-update step. The $\mathbf{x}$-update step requires to solve

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left(\frac{1}{2}\|\mathbf{x}\|_2^2 + \frac{\rho}{2}\|\mathbf{y}^k-\mathbf{x}-\tilde{\mathbf{x}}+\boldsymbol{\mu}^k\|_2^2\right).$$

The equation above is similar to *Tikhonov's regularization* and has the following closed form solution.

$$\mathbf{x}^{k+1} \leftarrow \frac{1}{1+\rho}\left(\mathbf{y}^k-\tilde{\mathbf{x}}+\boldsymbol{\mu}^k\right) \tag{25}$$

Replacing the $\mathbf{x}$-update step in Algorithm 1 with (25) gives an L_2 solution.

Algorithm 2: ADMM algorithm for contingency table

```
   Input: $\mathbf{A}, \mathbf{b}, \tilde{\mathbf{x}}, \rho, \mathfrak{M}^t$
   Output: a private estimate $\hat{\mathbf{x}}$
 1 $\mathbf{x}^0 \leftarrow \tilde{\mathbf{x}}$, $\boldsymbol{\mu}^0=\boldsymbol{\nu}^0=\boldsymbol{\eta}^0 \leftarrow 0$ $k \leftarrow 0$
 2 LU $\leftarrow \mathsf{chol}(\mathbf{A}^\intercal\mathbf{A}+2\mathbf{I})$        ▷ Cholesky decomposition
 3 repeat
 4     $\mathbf{y}^{k+1} \leftarrow \Theta_{\alpha,\rho}(\mathbf{x}^k-\tilde{\mathbf{x}}+\boldsymbol{\mu}^k)$        ▷ see (14)
 5     $\mathbf{z}^{k+1} \leftarrow (\mathbf{x}^k+\boldsymbol{\nu}^k)_+$        ▷ see (12)
 6     $\mathbf{x}^{k+1} \leftarrow \mathbf{U}^{-1}\left(\mathbf{L}^{-1}\,(\text{r.h.s. of (24)})\right)$
 7     $\boldsymbol{\mu}^{k+1} \leftarrow \boldsymbol{\mu}^k+\mathbf{x}^{k+1}-\mathbf{y}^{k+1}-\tilde{\mathbf{x}}$
 8     $\boldsymbol{\nu}^{k+1} \leftarrow \boldsymbol{\nu}^k+\mathbf{x}^{k+1}-\mathbf{z}^{k+1}$
 9     $\boldsymbol{\eta}^{k+1} \leftarrow \boldsymbol{\eta}^k+\mathbf{A}\mathbf{x}^{k+1}-\mathbf{b}$
10     $k \leftarrow k+1$
11 until Convergence
12 return x
```

5. ALGORITHMS AND EMPIRICAL EVALUATIONS

In this section, algorithms for applications discussed in Section 3.3 are derived and evaluated. To evaluate the performances of algorithms, a series of experiments is conducted on a variety of datasets. All experiments were conducted on a machine with AMD Opteron 2216 dual core and 8 GB RAM. To obtain an average performance estimate, all results are averaged over 10 runs. For ADMM, the mixing parameter α and penalty parameter ρ are fixed to 0.9 and 2.0, respectively.

Before presenting experimental results, we briefly discuss the performance and scalability of linear program (LP) solvers. We applied simplex and interior-point algorithms to solve the histogram estimation problem in Section 5.2 and observed that they fail to run when the length of histogram is greater than 2^{12}. The worst running time for ADMM was 21 sec. while built-in Matlab LP solvers run for several minutes and fail to return results. For the histograms of length greater than 2^{10}, ADMM consistently outperforms LP solvers in terms of both speed and accuracy.

5.1 Contingency Table

A marginal of order h, denoted by $\mathfrak{M}^h$, is a set of marginals $\{\mathfrak{m}(B)|B\subseteq H, |B|=h\}$. Notice that $\mathfrak{M}^0$ is equal to the sample size (or number of individuals in the dataset). Given $\mathfrak{M}^h$ and a noisy contingency table $\tilde{\mathbf{x}}$, finding an estimate consistent with $\mathfrak{M}^h$ can be formulated as

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \alpha\|\mathbf{y}\|_1 + (1-\alpha)\|\mathbf{y}\|_2^2 + \mathbb{I}_{\mathbb{R}^n_+}(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{x}-\tilde{\mathbf{x}}-\mathbf{y}=0, \quad \mathbf{x}-\mathbf{z}=0 \\ & \mathbf{A}\mathbf{x}=\mathbf{b} \end{aligned}$$

where $\alpha\in[0,1]$ and $\mathbf{A}\mathbf{x}=\mathbf{b}$ represents the marginal constraints. The algorithm for solving the above problem is given in Algorithm 2.

To evaluate the performance, we adopt three datasets, described in Table 2, from [9]. The proposed algorithm is applied on the datasets and compared with existing algorithms: the Laplace mechanism (LM) [7], the wavelet method (Privlet) [21] and the multiplicative weight mechanism (MWEM) [10]. For a fair comparison, negative cell counts reported by algorithms that do not generate the consistent output are replaced with 0.

Dataset	Dims.(k)	# samples(N)	Sparsity(%)
Edwards	6	70	65.6
Czech	6	1,841	1.5
Rochdale	8	665	64.4

Table 2: Binary contingency table datasets

Figure 5.1.1 shows the MSE of each algorithm by different values of ϵ. Note that the y-axis is in log scale. The top, middle and bottom rows of the figure correspond to the cases where $\mathfrak{M}^0$, $\mathfrak{M}^1$ and $\mathfrak{M}^2$ are publicly known, respectively. It is common in many settings of empirical studies that the number of individuals participated in the study (sample size), $\mathfrak{M}^0$, is publicly available, and in some cases the distribution of one or more binary variables may be known a priori (e.g., the number of men and women in the sample). Observe that the proposed algorithms, equivalent to the naive Laplace mechanism when there is no publicly available information about the input, consistently outperform other algorithms. One interesting observation is that the performance gap between the proposed and other algorithms becomes noticeable on sparse datasets. This is because of the different behavior of L_1 and L_2 norms. In L_2 minimization, large residuals (noise) are restricted because they are severely penalized. This, however, comes at the price of increasing the amplitude of small residuals. As a result, it tends to produce residuals of equal variance (or magnitude). Hence, the L_2 solution is not likely to have zero values in the places of zero entries in the original contingency table. On the other hand, the L_1 norm encourages small residuals become smaller while causing large residuals to increase.

5.2 Histogram

Given $\tilde{\mathbf{x}}$ corresponding to a noisy k-ary tree, the hierarchical constraint can be expressed in matrix notation as $\mathbf{A}\mathbf{x} = 0$, where

$$\mathbf{A} = \{a_{ij}\} = \begin{cases} 1 & \text{if } i = j \\ -1 & \text{if } x_j \text{ is a child of } x_i \\ 0 & \text{otherwise}. \end{cases}$$

The dimension of $\mathbf{A}$ is (p, n) where p and n correspond to the number of non-leaf nodes and all nodes, respectively. The matrix $\mathbf{A}$ is very sparse; for each row, it contains only $k+1$ non-zero elements. This enables to solve (15) efficiently by employing sparse linear solvers. With $\mathbf{A}$ constructed as above, it is possible to re-use Algorithm 2, but a better algorithm can be derived by slightly modifying the formulation.

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y},\mathbf{z}}{\text{minimize}} \quad & \alpha\|\mathbf{y}\|_1 + (1-\alpha)\|\mathbf{y}\|_2^2 + \mathbb{I}_{\mathcal{C}}(\mathbf{z}) + \mathbb{I}_{\mathbb{R}^n_+}(\mathbf{w}) \\ \text{subject to} \quad & \mathbf{x} - \tilde{\mathbf{x}} - \mathbf{y} = 0, \quad \mathbf{x} - \mathbf{z} = 0 \\ & \mathbf{x} - \mathbf{w} = 0 \end{aligned}$$

where $\mathcal{C} = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} = 0\}$ is the set of vectors that satisfies the hierarchical constraint. The $\mathbf{z}$-update step reduces to the Euclidean projection on the set $\mathcal{C}$, which is defined as

$$\mathbf{z}^{k+1} = \Pi_{\mathcal{C}}(\mathbf{x}^k + \boldsymbol{\nu}^k) = \underset{\mathbf{z}\in\mathcal{C}}{\arg\min}\ \|\mathbf{z} - (\mathbf{x}^k + \boldsymbol{\nu}^k)\|_2 .$$

As discussed in Remark 1, the definition of Euclidean projection exactly matches least squares based formulation. Let

Algorithm 3: ADMM algorithm for histogram release

```
   Input: $\mathbf{A}, \mathbf{b}, \tilde{\mathbf{x}}, \rho, \mathfrak{M}^t$
   Output: a private estimate $\hat{\mathbf{x}}$
1  $\mathbf{x}^0 \leftarrow \tilde{\mathbf{x}}, \mathbf{y}^0 \leftarrow 0, \mathbf{z}^0 \leftarrow \mathbf{x}, \boldsymbol{\mu}^0 = \boldsymbol{\nu}^0 = \boldsymbol{\eta}^0 \leftarrow 0$
2  $k \leftarrow 0$
3  repeat
4      $\mathbf{y}^{k+1} \leftarrow \Theta_{\alpha,\rho}(\mathbf{x}^k - \tilde{\mathbf{x}} + \boldsymbol{\mu}^k)$          ▷ see (14)
5      $\mathbf{z}^{k+1} \leftarrow$ HTREE$(\mathbf{x}^k + \boldsymbol{\nu}^k)$          ▷ see [11]
6      $\mathbf{w}^{k+1} \leftarrow (\mathbf{x}^k + \boldsymbol{\eta}^k)_+$
7      $\mathbf{x}^{k+1} \leftarrow \frac{1}{3}\{(\mathbf{y}^{k+1} + \tilde{\mathbf{x}} - \boldsymbol{\mu}^k) + (\mathbf{z}^{k+1} - \boldsymbol{\nu}^k)$
8                  $+(\mathbf{w}^{k+1} - \boldsymbol{\eta}^k)\}$
9      $\boldsymbol{\mu}^{k+1} \leftarrow \boldsymbol{\mu}^k + \mathbf{x}^{k+1} - \mathbf{y}^{k+1} - \tilde{\mathbf{x}}$
10     $\boldsymbol{\nu}^{k+1} \leftarrow \boldsymbol{\nu}^k + \mathbf{x}^{k+1} - \mathbf{z}^{k+1}$
11     $\boldsymbol{\eta}^{k+1} \leftarrow \boldsymbol{\eta}^k + \mathbf{x}^{k+1} - \mathbf{w}^{k+1}$
12     $k \leftarrow k + 1$
13 until convergence
14 return x
```

Dataset	Size(n)	$\min_i x_i$	$\max_i x_i$	Sparsity(%)
Search Logs	32,768	0	496	52.1
Nettace	65,536	0	1,423	96.6
Social Network	11,342	1	1,678	0

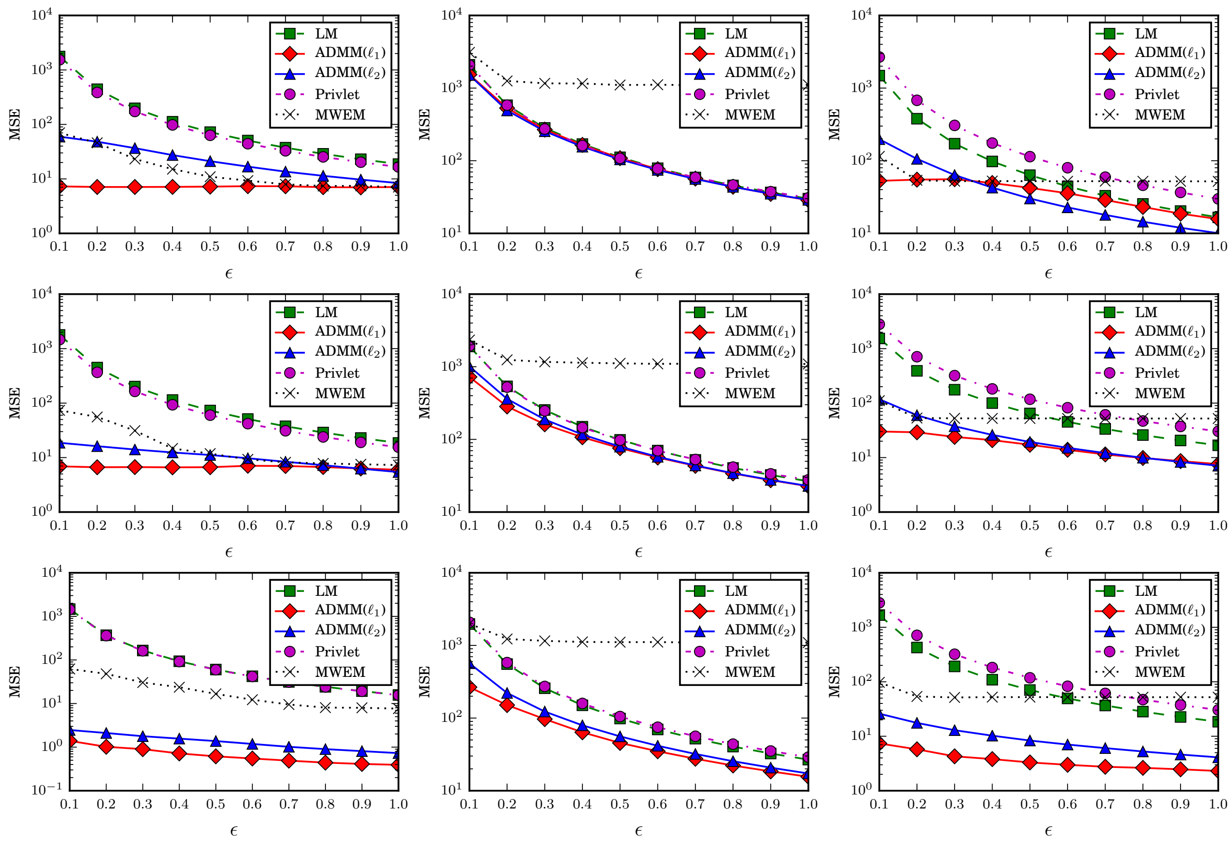
Table 3: Histogram datasets

HTREE($\boldsymbol{v}$) denote the algorithm for universal histogram proposed in [11]. Given $\boldsymbol{v} = \mathbf{x}^k + \boldsymbol{\nu}^k$, the function HTREE finds a new vector in $\mathcal{C}$ that is closest to $\boldsymbol{v}$. This means the function HTREE can replace the Euclidean projection in $\mathbf{z}$-update. The resulting ADMM algorithm for the above formulation is given in Algorithm 3.

To evaluate our algorithm, we use the three datasets, namely Search Logs, Nettrace and Social Network; for descriptions on the datasets, refer to [11]. We consider six existing algorithms for releasing histograms: two versions of the Laplace mechanism, LM(H) and LM(L), the constrained inference algorithm (HTree), the wavelet algorithm (Privlet), the fourier perturbation algorithm (EFPA) [1], and the clustering based algorithm (P-HP) [1]. LM(H) builds hierarchical tree over a histogram while LM(L) is pure Laplace mechanism. The performance of HTree algorithm is largely dependent on the branching factor k. To tune this parameter, we iteratively searched for a value that gives the best result, and that value is used for the experiments. We note that none of the existing algorithms do guarantee the non-negativity of bin counts in the released histogram. When a range query answer is negative, we replace it with 0. This trivial step slightly improves the accuracies of existing algorithms.

Figure 5.2.1 describes the performance of the proposed algorithm on the unit-length queries. On sparse datasets, Search Logs and Nettrace, ADMM(ℓ_1) clearly outperforms HTree which is an L_2 minimization based algorithm while they perform similarly on the Social Network dataset, as it was the case for the contingency table estimation.

In Figure 5.2.2, the performances of algorithms for fixed length range queries are compared. The performance of each algorithm over a set of range queries is measured in terms of MSE. The range sizes are increased from 2^1 to 2^{h-2} where



(a) Edwards (b) Czech (c) Rochdale

Figure 5.1.1: MSE by varying ϵ given (**Top:** $\mathfrak{M}^0$, **Middle:** $\mathfrak{M}^1$, **Bottom:** $\mathfrak{M}^2$)

h is the height of the tree. For each fixed range size, 1000 random ranges are selected at uniformly random. The top and bottom rows correspond to $\epsilon = 0.1$ and $\epsilon = 1.0$, respectively. Given a histogram of length n, both HTree and Privlet are known to have $\mathcal{O}(\log^3 n/\epsilon^2)$ error bound and they show similar performances throughout all range sizes. In contrast, the error of the LM(H) and LM(L) increases linearly with the size of the range. While ADMM(ℓ_1) is much more accurate for smaller ranges than HTree and Privlet, the performance on large ranges is not as good as them. This is because that, while L_1 minimization reduces the amount of noise in most bins of the histogram, the result of that is an increased noise in the small number of bins. As we increase the range size, large noise in those bins will be accumulated and it degrades the accuracy of large range queries. P-HP shows a good performance on Social network dataset. This is because the dataset is sorted. However, it performs poorly on Search Log dataset. Especially, when $\epsilon = 1$, its performance is even worse than that of naive Laplace mechanism.

5.3 Matrix Mechanism

The matrix mechanism (MM) is a general framework for answering linear queries in a differentially private manner.

The problem (5) in ADMM form is

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y},\mathbf{z}}{\text{minimize}} \quad & \alpha\|\mathbf{y}\|_1 + (1-\alpha)\|\mathbf{y}\|_2^2 + \mathbb{I}_{\mathbb{R}^n_+}(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} - \tilde{\mathbf{r}} + \mathbf{y} = 0 \\ & \mathbf{x} - \mathbf{z} = 0\,. \end{aligned}$$

Algorithm 4: ADMM algorithm for matrix mechanism

```
   Input: $\mathbf{A}$, $\tilde{\mathbf{r}}$, $\rho$
   Output: a private estimate $\bar{\mathbf{x}}$
 1 $\mathbf{x}^0 \leftarrow 0$, $\boldsymbol{\mu}^0 = \boldsymbol{\nu}^0 \leftarrow 0$, $k \leftarrow 0$
 2 $\mathbf{LU} \leftarrow \mathsf{chol}(\mathbf{A}^\intercal\mathbf{A} + \mathbf{I})$          ▷ Cholesky decomposition
 3 repeat
 4     $\mathbf{y}^{k+1} \leftarrow \Theta_{\alpha,\rho}(\tilde{\mathbf{r}} - \mathbf{A}\mathbf{x}^k - \boldsymbol{\mu}^k)$          ▷ see (14)
 5     $\mathbf{z}^{k+1} \leftarrow (\mathbf{x}^k + \boldsymbol{\nu}^k)_+$          ▷ see (12)
 6     $\mathbf{x}^{k+1} \leftarrow \mathbf{U}^{-1}\left(\mathbf{L}^{-1}\left(\mathbf{A}^\intercal(\tilde{\mathbf{r}} - \mathbf{y}^{k+1} - \boldsymbol{\mu}^k) + (\mathbf{z}^k - \boldsymbol{\nu}^k)\right)\right)$
 7     $\boldsymbol{\mu}^{k+1} \leftarrow \boldsymbol{\mu}^k + \mathbf{A}\mathbf{x}^{k+1} - \tilde{\mathbf{r}} + \mathbf{y}^{k+1}$
 8     $\boldsymbol{\nu}^{k+1} \leftarrow \boldsymbol{\nu}^k + \mathbf{x}^{k+1} - \mathbf{z}^{k+1}$
 9     $k \leftarrow k + 1$
10 until convergence
11 return x
```

The resulting ADMM algorithm is described in Algorithm 4. For MM, negative range query answers are replaced with 0. In this experiment, the privacy budget ϵ is fixed to 0.1.

Although it is shown in [22] that the matrix mechanism has some limitations and generally does not perform well, we apply our algorithm to show that the proposed algorithm combined with other existing algorithms can enhance the accuracy. Since solving the semi-definite programming provided in [13] is computationally infeasible, we use the approximated version of matrix mechanism provided in [22].

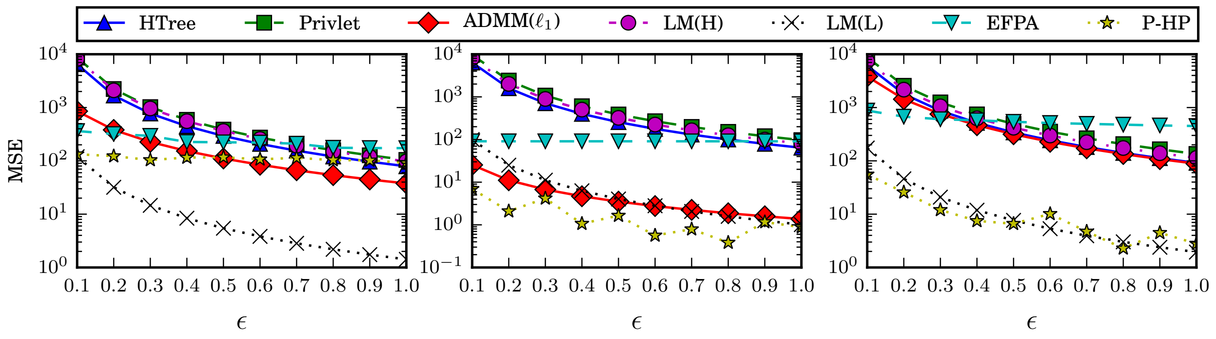


Figure 5.2.1: MSE for unit-length range query by varying ϵ (**Left:** Search Logs, **Middle:** Nettrace, **Right:**Social Network)

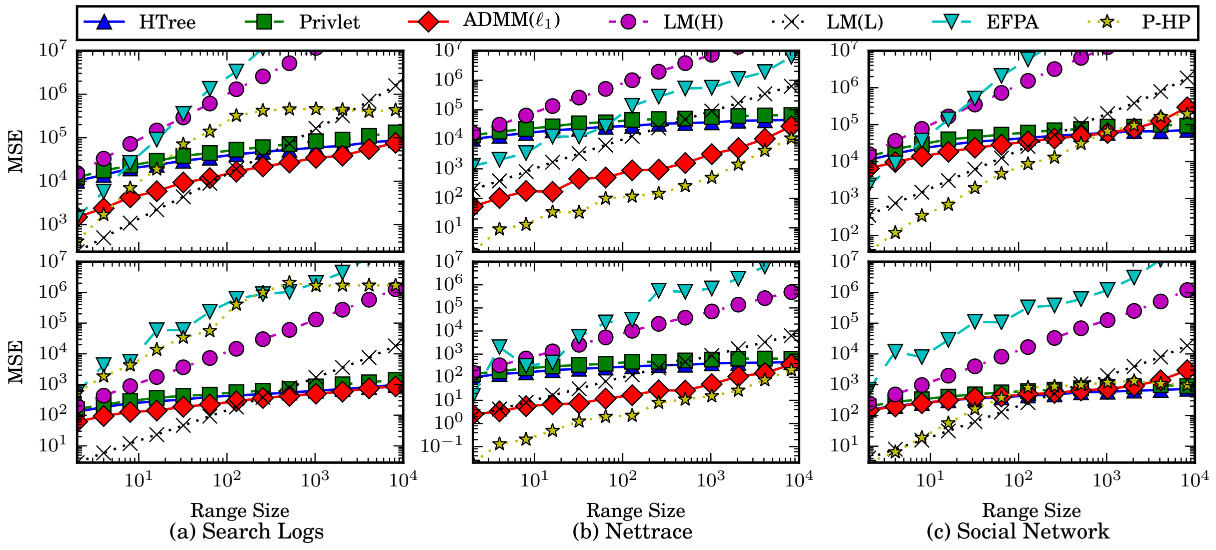


Figure 5.2.2: MSE by varying range size (**Top:** $\epsilon = 0.1$, **Bottom:** $\epsilon = 1.0$)

Due to the computational inefficiency of matrix mechanism, the domain size is fixed to 1024 by aggregating datasets. The matrix mechanism that estimates the true databases using least squares and L_1 minimization are denoted by MM(ℓ_2) and MM(ℓ_1), respectively. We compare MM to the following algorithms: the constrained inference algorithm (HTree), the wavelet algorithm (Privlet), the Laplace mechanism (LM), and the Low-rank mechanism (LRM) [22].

Figure 5.3.1 describes the performance of MM on each dataset. The workload $\mathbf{W}$=$\{w_{ij}\}$ considered is a Hadamard matrix, generated by setting each entry to either 1 or -1 with probability 1/2. As shown in the figure, MM performs worse than other algorithms but replacing its estimation step with our algorithm improves its accuracy on all three test datasets.

In Figure 5.3.2, we consider three different types of workloads, namely identity, predicate and all range. Identity is an identity matrix which represents a set of queries consisting of all unit-length queries. A predicate query $\mathbf{q}$ is a linear combination of unit-length intervals: $a_1x_1 + a_2x_2 + \cdots a_nx_n$ where $a_i \in \{0, 1\}$ and a_i is set to either 0 or 1 with equal probability. The predicate workload contains 512 randomly chosen predicate queries. All range workload consists of queries for all consecutive intervals in the domain. Applying the proposed algorithm to MM improves the accuracy in all cases, except all range workload on Nettrace dataset.

6. CONCLUSION

In this paper, we formulated the post-processing data perturbed with noise as a constrained L_1 minimization problem and used the consistency constraints to improve the accuracy. The decomposition of our formulation using ADMM revealed that one of subproblems involves solving the previous least squares based formulation. This means that our approach can re-use and take advantage of existing algorithms developed for the prior least squares based formulation. The effectiveness of the algorithm in improving accuracy is demonstrated through a series of experiments. Although we only described (and provided algorithms for) three applications of differential privacy, the proposed approach is generic and highly flexible that it can be easily adapted to other problems. We expect the proposed algorithm can help other private algorithms to achieve better accuracy.

Acknowledgments

This work was supported by NSF grant No. CNS-1228669 and a gift from Xerox.

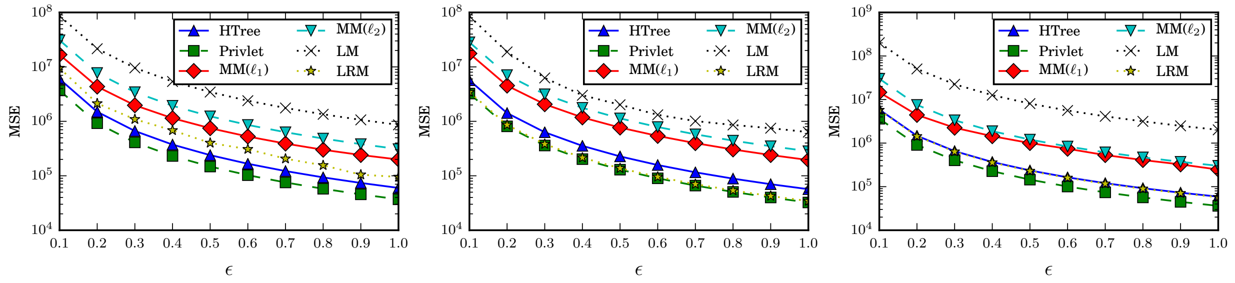


Figure 5.3.1: MSE of matrix mechanism by varying ϵ (**Left:** Search Logs, **Middle:** Nettrace, **Right:** Social Network)

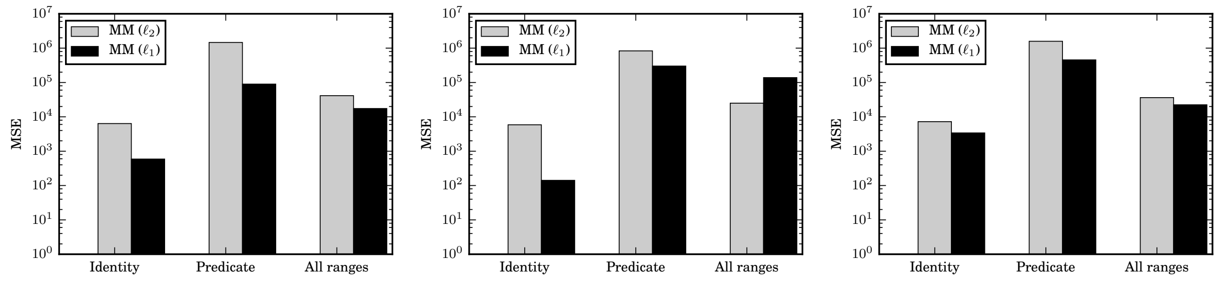


Figure 5.3.2: MSE of MM on three workloads ($\epsilon = 0.1$, **Left:** Search Logs, **Middle:** Nettrace, **Right:** Social Network)

7. REFERENCES

[1] G. Acs, C. Castelluccia, and R. Chen. Differentially private histogram publishing through lossy compression. In *ICDM*, 2012.

[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*, 2007.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, 2011.

[4] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

[5] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially private spatial decompositions. In *ICDE*, 2012.

[6] B. Ding, M. Winslett, J. Han, and Z. Li. Differentially private data cubes: optimizing noise sources and consistency. In *SIGMOD*, 2011.

[7] C. Dwork. Differential privacy. In *ICALP*, 2006.

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

[9] S. E. Fienberg, A. Rinaldo, and X. Yang. Differential privacy and the risk-utility tradeoff for multi-dimensional contingency tables. In *Privacy in Statistical Databases*, 2010.

[10] M. Hardt, K. Ligett, and F. Mcsherry. A simple and practical algorithm for differentially private data release. In *NIPS*, 2012.

[11] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1-2):1021–1032, 2010.

[12] J. Lee and C. W. Clifton. Top-k frequent itemsets via differentially private fp-trees. In *KDD*, 2014.

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

[14] B.-R. Lin and D. Kifer. Information preservation in statistical privacy and bayesian estimation of unattributed histograms. In *SIGMOD*, 2013.

[15] D. Proserpio, S. Goldberg, and F. McSherry. Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. *PVLDB*, 7(8):637–648, 2014.

[16] W. Qardaji, W. Yang, and N. Li. Differentially private grids for geospatial data. In *ICDE*, 2013.

[17] W. Qardaji, W. Yang, and N. Li. Understanding hierarchical methods for differentially private histograms. *PVLDB*, 6(14):1954–1965, 2013.

[18] W. Qardaji, W. Yang, and N. Li. Priview: Practical differentially private release of marginal contingency tables. In *SIGMOD*, 2014.

[19] A. Smith. Privacy-preserving statistical estimation with optimal convergence rates. In *STOC*, 2011.

[20] O. Williams and F. McSherry. Probabilistic inference and differential privacy. In *NIPS*, 2010.

[21] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *TKDE*, 23(8):1200–1214, 2011.

[22] G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao. Low-rank mechanism: Optimizing batch queries under differential privacy. *PVLDB*, 5(11):1352–1363, 2012.

[23] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.