Graphical-model based estimation and inference for differential privacy

Ryan McKenna [1] **Daniel Sheldon** [1] **Gerome Miklau** [1]

Abstract

Many privacy mechanisms reveal high-level information about a data distribution through noisy measurements. It is common to use this information to estimate the answers to new queries. In this work, we provide an approach to solve this estimation problem efficiently using graphical models, which is particularly effective when the distribution is high-dimensional but the measurements are over low-dimensional marginals. We show that our approach is far more efficient than existing estimation techniques from the privacy literature and that it can improve the accuracy and scalability of many state-of-the-art mechanisms.

1. Introduction

Differential privacy (Dwork et al., 2006) has become the dominant standard for controlling the privacy loss incurred by individuals as a result of public data releases. For complex data analysis tasks, error-optimal algorithms are not known and a poorly designed algorithm may result in much greater error than strictly necessary for privacy. Thus, careful algorithm design, focused on reducing error, is an area of intense research in the privacy community.

For the private release of statistical queries, nearly all recent algorithms (Zhang et al., 2017; Li et al., 2015; Lee et al., 2015; Proserpio et al., 2014; Li et al., 2014; Qardaji et al., 2013b; Nikolov et al., 2013; Hardt et al., 2012; Ding et al., 2011; Xiao et al., 2010; Li et al., 2010; Hay et al., 2010; Hardt & Rothblum, 2010; Hardt & Talwar, 2010; Barak et al., 2007; Gupta et al., 2011; Thaler et al., 2012; Acs et al., 2012; Zhang et al., 2014; Yaroslavtsev et al., 2013; Cormode et al., 2012; Qardaji et al., 2013a; McKenna et al., 2018) include steps within the algorithm where answers to queries are *inferred* from noisy answers to a set of *measurement* queries already answered by the algorithm.

Inference is a critical component of privacy algorithms because: (i) it can reduce error when answering a query by combining evidence from multiple related measurements, (ii) it provides consistent query answers even when measurements are noisy and inconsistent, and (iii) it provides the above benefits without consuming the privacy-loss budget, since it is performed only on privately-computed measurements without re-using the protected data.

Consider a U.S. Census dataset, exemplified by the Adult table, which consists of 15 attributes including age, sex, race, income, education. Given noisy answers to a set of measurement queries, our goal is to infer answers to one or more new queries. The measurement queries might be expressed over each individual attribute (age), (sex), (race), etc., as well as selected combinations of attributes (age, income), (age, race, education), etc. When inference is done properly, the estimate for a new query (e.g., counting the individuals with income>=50K, 10 years of education, and over 40 years old) will use many, or even all, available measurements.

Current inference methods are limited in both scalability and generality. Most methods first estimate some model of the data and then answer new queries using the model. Perhaps the simplest model is a full contingency table, which stores a value for every element of the domain. When the measurements are linear queries (a common case, and our primary focus) least-squares (Hay et al., 2010; Nikolov et al., 2013; Li et al., 2014; Qardaji et al., 2013b; Ding et al., 2011; Xiao et al., 2010; Li et al., 2010) and multiplicative-weight updates (Hardt & Rothblum, 2010; Hardt et al., 2012) have both been used to estimate this model from the noisy measurements. New queries can then be answered by direct calculation. However, the size of the contingency table is the product of the domain sizes of each attribute, which means these methods break down for high-dimensional cases (or even a modest number of dimensions with large domains). In the example above, the full contingency table would consist of 10^{19} entries. To avoid this, factored models have been considered (Hardt et al., 2012; Zhang et al., 2017). However, while scalable, these methods have other limitations including restricting the query class (Hardt et al., 2012) or failing to properly account for (possibly varying) noise in measurements (Zhang et al., 2017).

In this work we show that graphical models provide a foun-

[1]College of Computer Science, University of Massachusetts, Amherst. Correspondence to: Ryan McKenna <rmckenna@cs.umass.edu>.

dation for significantly improved inference. We propose to use a graphical model instead of a full contingency table as a model of the data distribution. Doing so avoids an intractable full materialization of the contingency table and retains the ability to answer a broad class of queries. We show that the graphical model representation corresponds to using a maximum entropy criterion to select a single data distribution among all distributions that minimize estimation loss. The structure of the graphical model is determined by the measurements, such that no information is lost relative to a full contingency table representation, but when each measurement is expressible over a low-dimensional marginal of the contingency table, as is common, the graphical model representation is much more compact.

This work is focused on developing a principled and general approach to inference in privacy algorithms. Our method is agnostic to the loss function used to estimate the data model and to the noise distribution used to achieve privacy. We focus primarily on linear measurements, but also describe an extension to non-linear measurements

We assume throughout that the measurements are given, but we show our inference technique is versatile since it can be incorporated into many existing private query-answering algorithms that determine measurements in different ways. For those existing algorithms that scale to high-dimensional data, our graphical-model based estimation method can substantially improve accuracy (with no cost to privacy). Even more importantly, our estimation method can be added to some algorithms which fail to scale to high-dimensional data, allowing them to run efficiently in new settings. We therefore believe our inference method can serve as a basic building block in the design of new privacy algorithms.

2. Background and Problem Statement

Data. Our input data represents a population of individuals, each contributing a single record $\mathbf{x} = (x_1, \dots, x_d)$ where x_i is the i^{th} attribute belonging to a discrete finite domain $\mathcal{X}_i$ of n_i possible values. The full domain is $\mathcal{X} = \prod_{i=1}^d \mathcal{X}_i$ and its size $n = \prod_{i=1}^d n_i$ is exponential in the number of attributes. A dataset $\mathbf{X}$ consists of m such records $\mathbf{X} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)})$. We also consider a normalized contingency table representation $\mathbf{p}$, which counts the fraction of the population with record equal to $\mathbf{x}$, for each $\mathbf{x}$ in the domain. That is, $\mathbf{p}(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^m \mathbb{I}\{\mathbf{x}^{(i)} = \mathbf{x}\}, \forall \mathbf{x} \in \mathcal{X}$, where $\mathbb{I}\{\cdot\}$ is an indicator function. Thus $\mathbf{p}$ is a probability vector in $\mathbb{R}^n$ with index set $\mathcal{X}$ (ordered lexicographically). We write $\mathbf{p} = \mathbf{p_X}$ when it is important to denote the dependence on $\mathbf{X}$.

Queries, Marginals, and Measurements. We focus on the most common case of linear queries expressed over subsets of attributes. We will describe an extension to a generalized class of queries, including non-linear ones, in Section 3.1. A *linear query set* $f_{\mathbf{Q}}(\mathbf{X})$ is defined by a *query matrix* $\mathbf{Q} \in \mathbb{R}^{r \times n}$ and has answer $f_{\mathbf{Q}}(\mathbf{X}) = \mathbf{Q}\,\mathbf{p_X}$. The ith row of $\mathbf{Q}$, denoted $\mathbf{q}_i^T$ represents a single scalar-valued query. In most cases we will refer unambiguously to the matrix $\mathbf{Q}$, as opposed to $f_{\mathbf{Q}}$, as the query set. We often consider query sets that can be expressed on a *marginal* (over a subset of attributes) of the probability vector $\mathbf{p}$. Let $A \subseteq [d]$ identify a subset of attributes and, for $\mathbf{x} \in \mathcal{X}$, let $\mathbf{x}_A = (x_i)_{i \in A}$ be the sub-vector of $\mathbf{x}$ restricted to A. Then the marginal probability vector (or simply "marginal on A") $\boldsymbol{\mu}_A$, is defined by:

$$\boldsymbol{\mu}_A(\mathbf{x}_A) = \frac{1}{m}\sum_{i=1}^m \mathbb{I}\{\mathbf{x}_A^{(i)} = \mathbf{x}_A\}, \quad \forall \mathbf{x}_A \in \mathcal{X}_A := \prod_{i \in A} \mathcal{X}_i.$$

The number of entries of the marginal is $n_A := |\mathcal{X}_A| = \prod_{i \in A} n_i$, which is exponential in $|A|$ but may be considerably smaller than n. Note that $\boldsymbol{\mu}_A(\mathbf{x}_A)$ is a linear function of $\mathbf{p}$, so there exists a matrix $\mathbf{M}_A \in \mathbb{R}^{n_A \times n}$ such that $\boldsymbol{\mu}_A = \mathbf{M}_A\mathbf{p}$. When a query set depends only on the marginal vector $\boldsymbol{\mu}_A$, we call it a *marginal query set* written as $\mathbf{Q}_A \in \mathbb{R}^{r_A \times n_A}$, and with answer $f_{\mathbf{Q}_A}(\mathbf{X}) = \mathbf{Q}_A\,\boldsymbol{\mu}_A$. The marginal query set $\mathbf{Q}_A$ is equivalent to the query set $\mathbf{Q} = \mathbf{Q}_A\mathbf{M}_A$ on the full contingency table, since $\mathbf{Q}_A\boldsymbol{\mu}_A = (\mathbf{Q}_A\mathbf{M}_A)\mathbf{p}$. One marginal query set asks for the marginal vector itself, in which case $\mathbf{Q}_A = \mathbf{I}_{n_A \times n_A}$ (the identity matrix).

In our problem formulation, we consider measurements consisting of a collection of marginal query sets. Specifically, let $\mathcal{C}$ be a collection of *measurement sets*, where each $C \in \mathcal{C}$ is a subset of $[d]$.[1] For each measurement set $C \in \mathcal{C}$, we are given a marginal query set $\mathbf{Q}_C$. The following notation is helpful to refer to combined measurements and their marginals. Let $\boldsymbol{\mu} = (\boldsymbol{\mu}_C)_{C \in \mathcal{C}}$ be the combined vector of marginals, and let $\mathbf{Q}_{\mathcal{C}}$ be the block-diagonal matrix with diagonal blocks $\{\mathbf{Q}_C\}_{C \in \mathcal{C}}$, so that the entire set of query answers can be expressed as $\mathbf{Q}_{\mathcal{C}}\boldsymbol{\mu}$. Finally, let $\mathbf{M}_{\mathcal{C}}$ be the matrix that vertically concatenates the matrices $\{\mathbf{M}_C\}_{C \in \mathcal{C}}$, so that $\boldsymbol{\mu} = \mathbf{M}_{\mathcal{C}}\mathbf{p}$ and $\mathbf{Q}_{\mathcal{C}}\boldsymbol{\mu} = \mathbf{Q}_{\mathcal{C}}\mathbf{M}_{\mathcal{C}}\mathbf{p}$. This shows that our measurements are equivalent to the combined query set $\mathbf{Q} = \mathbf{Q}_{\mathcal{C}}\mathbf{M}_{\mathcal{C}}$ applied to the full table $\mathbf{p}$.

Differential privacy. Differential privacy protects individuals by bounding the impact any one individual can have on the output of an admissible algorithm. This is formalized using the notion of neighboring datasets. Let $\text{nbrs}(\mathbf{X})$ denote the set of datasets formed by replacing any $\mathbf{x}^{(i)} \in \mathbf{X}$ with an arbitrary new record $\mathbf{x}'^{(i)} \in \mathcal{X}$.

Definition 1 (Differential Privacy; Dwork et al., 2006)**.** *A randomized algorithm $\mathcal{A}$ satisfies (ϵ, δ)-differential privacy if for any input $\mathbf{X}$, any $\mathbf{X}' \in$ nbrs$(\mathbf{X})$, and any subset of*

[1] Later, these will comprise the cliques of a graphical model, as the notation suggests.

outputs $S \subseteq \mathit{Range}(\mathcal{A})$,

$$\Pr[\mathcal{A}(\mathbf{X}) \in S] \leq \exp(\epsilon) \Pr[\mathcal{A}(\mathbf{X}') \in S] + \delta$$

When $\delta = 0$ we say $\mathcal{A}$ satisfies ϵ-differential privacy. Differentially private answers to $f_{\mathbf{Q}}$ are typically obtained with a noise-addition mechanism, such as the Laplace or Gaussian mechanism. For ϵ-differential privacy, the noise added to the output of $f_{\mathbf{Q}}$ is determined by the L_1 *sensitivity* of $f_{\mathbf{Q}}$, which, specialized to linear queries, is defined as $\Delta_{\mathbf{Q}} = \max_{\mathbf{X},\mathbf{X}' \in \text{nbrs}(\mathbf{X})} \|\mathbf{Q}\,\mathbf{p}_{\mathbf{X}} - \mathbf{Q}\,\mathbf{p}_{\mathbf{X}'}\|_1$. It is straightforward to show that $\Delta_{\mathbf{Q}} = \frac{2}{m}\|\mathbf{Q}\|_1$ where $\|\mathbf{Q}\|_1$ is the maximum L_1 norm of the columns of $\mathbf{Q}$.

Definition 2 (Laplace Mechanism; Dwork et al., 2006)**.** *Given a query set* $\mathbf{Q} \in \mathbb{R}^{r \times n}$ *of* r *linear queries, the Laplace mechanism is defined as* $\mathcal{L}(\mathbf{X}) = \mathbf{Q}\,\mathbf{p}_{\mathbf{X}} + \mathbf{z}$ *where* $\mathbf{z} = (z_1, \ldots, z_r)$ *and each* z_i *is an i.i.d. random variable from Laplace*$(\Delta_{\mathbf{Q}}/\epsilon)$.

The Laplace mechanism satisfies ϵ-differential privacy. The sequential composition property implies that if we answer two query sets $\mathbf{Q}_1$ and $\mathbf{Q}_2$, under ϵ_1 and ϵ_2 differential privacy, respectively, then the combined answers are $(\epsilon_1 + \epsilon_2)$-differentially private. The *post-processing* property of differential privacy (Dwork & Roth, 2014) asserts that an algorithm that accepts as input the output of an ϵ-differentially algorithm, but does not use the original protected data, is also ϵ-differentially private.

Problem Statement. We assume as given a collection $\mathcal{C}$ of *measurement sets*, and for each $C \in \mathcal{C}$: a marginal query set $\mathbf{Q}_C$, a privacy parameter ϵ_C, and an ϵ_C-differentially private *measurement* $\mathbf{y}_C = \mathbf{Q}_C\boldsymbol{\mu}_C + \text{Lap}(\Delta_{\mathbf{Q}_C}/\epsilon_C)$. The combined *measurements* are $\mathbf{y} = (\mathbf{y}_C)_{C \in \mathcal{C}}$ which satisfy ϵ-differential privacy for $\epsilon = \sum_{C \in \mathcal{C}} \epsilon_C$ by sequential composition. Note that there is no loss of generality in these assumptions; in the extreme case, there may be just a single measurement set $C = [d]$ consisting of all attributes. Formulating the problem this way will allow us to realize computational savings when measurements are not full-dimensional, which is common in practice. We also emphasize that the marginal query set $\mathbf{Q}_C$ is often a complex set of linear queries expressed over measurement set C (not simply a marginal). Many past works (Li et al., 2015; 2014; Qardaji et al., 2013b; Nikolov et al., 2013; Ding et al., 2011; Xiao et al., 2010; Li et al., 2010; Hay et al., 2010; Barak et al., 2007) have shown that it is beneficial, in the presence of noise-addition for privacy, to measure carefully chosen query sets which balance sensitivity against efficient reconstruction of the workload queries.

Our goal is: given $\mathbf{y}$, derive answers to (possibly different) workload queries $\mathbf{W}$. There are multiple possible motivations: $\mathbf{W}$ may include new queries that were not part of the original measurements; or it is possible that $\mathbf{W}$ is a subset of measurement queries, but we can obtain a more accurate answer by combining *all* of the available information to estimate $\mathbf{W}\mathbf{p}$ as opposed to just using the noisy answer we got. We describe an extension to non-linear queries and more general linear queries in Section 3.1; this will be applied to the DualQuery algorithm (Gaboardi et al., 2014) in Section 4.

3. Algorithms for Estimation and Inference

What principle can we follow to estimate answers to the workload query set? Prior work takes the approach of first using all available information to estimate a full contingency table $\hat{\mathbf{p}} \approx \mathbf{p}$ and then using $\hat{\mathbf{p}}$ to answer later queries (Hay et al., 2010; Li et al., 2010; Ding et al., 2011; Qardaji et al., 2013b; Lee et al., 2015). We will call finding $\hat{\mathbf{p}}$ *estimation*, and using $\hat{\mathbf{p}}$ to answer new queries *inference*.

3.1. Optimization Formulation

The standard framework for estimation and inference is:

$$\hat{\mathbf{p}} \in \operatorname*{argmin}_{\mathbf{p} \in \mathcal{S}} L(\mathbf{p}), \qquad \text{(estimation)}$$

$$f_{\mathbf{W}}(\mathbf{X}) \approx \mathbf{W}\,\hat{\mathbf{p}}. \qquad \text{(inference)}$$

Here $\mathcal{S} = \left\{\mathbf{p} : \mathbf{p} \geq 0, \mathbf{1}^T\mathbf{p} = 1\right\}$ is the probability simplex and $L(\mathbf{p})$ is a loss function that measures how well $\mathbf{p}$ explains the observed measurements. In past works, $L(\mathbf{p}) = \|\mathbf{Q}\mathbf{p} - \mathbf{y}\|$ has been used as a loss function, where $\mathbf{Q}$ is the measured query set and $\|\cdot\|$ is either the L_1 norm or L_2 norm. Minimizing the L_1 norm is equivalent to maximum likelihood estimation when the noise comes from the Laplace mechanism (Lee et al., 2015). Minimizing the L_2 norm is far more common in the literature however, and it is also the maximum likelihood estimator for Gaussian noise (Hay et al., 2010; Nikolov et al., 2013; Li et al., 2014; Qardaji et al., 2013b; Ding et al., 2011; Xiao et al., 2010; Li et al., 2010; McKenna et al., 2018). Our method supports both of these loss functions; we only require that L is convex. Both of these loss functions are easily adapted to the situation where queries in $\mathbf{Q}$ may be measured with differing degrees of noise. The constraint $\mathbf{p} \in \mathcal{S}$ may also be relaxed, which simplifies L_2 minimization; additionally, under different assumptions and an alternate version of privacy, the number of individuals may not be known. All existing algorithms to solve these variations of the estimation problem suffer from the same problem: they do not scale to high dimensions since the size of $\mathbf{p}$ is exponential in d and we have to construct it explicitly as an intermediate step even if the inputs and outputs are small (e.g., all measurement queries are over low-dimensional marginals).

Optimization in Terms of Marginals. For marginal query sets, a loss function will typically depend on $\mathbf{p}$ only through its marginals $\boldsymbol{\mu}$. For example, when $\mathbf{Q} = \mathbf{Q}_C\mathbf{M}_C$ we have

$L(\mathbf{p}) = \|\mathbf{Q}\mathbf{p} - \mathbf{y}\| = \|\mathbf{Q}_C\boldsymbol{\mu} - \mathbf{y}\| = L(\boldsymbol{\mu})$ where we now write the loss function as $L(\boldsymbol{\mu})$. More generally, we will consider *any* loss function that only depends on the marginals. A very general case is when $L(\boldsymbol{\mu}) = -\log p(\mathbf{y} \mid \boldsymbol{\mu})$ is the negative log-likelihood of *any* differentially private algorithm that produces output $\mathbf{y}$ that depends only on the marginal vector $\boldsymbol{\mu}$ (see our treatment of DualQuery in Section 4).

The marginal vector $\boldsymbol{\mu}$ may be much lower dimensional than $\mathbf{p}$. How can we take advantage of this fact? An "obvious" idea would be to modify the optimization to estimate only the marginals as $\hat{\boldsymbol{\mu}} \in \operatorname{argmin}_{\boldsymbol{\mu}\in\mathcal{M}} L(\boldsymbol{\mu})$, where $\mathcal{M} = \{\boldsymbol{\mu} : \exists \mathbf{p} \in \mathcal{S} \text{ s.t. } \mathbf{M}_C\mathbf{p} = \boldsymbol{\mu}\}$ is the marginal polytope, which is the set of all valid marginals. There are two issues here. First, the marginal polytope has a complex combinatorial structure, and, although it is a convex set, it is generally not possible to enumerate its constraints for use with standard convex optimization algorithms. Note that this optimization problem is in fact a *generic* convex optimization problem over the marginal polytope, and as such it generalizes standard graphical model inference problems (Wainwright & Jordan, 2008). Second, after finding $\hat{\boldsymbol{\mu}}$ it is not clear how to answer new queries, unless they depend only on some measured marginal $\boldsymbol{\mu}_C$.

Graphical Model Representation. After finding an optimal $\hat{\boldsymbol{\mu}}$ we want to answer new queries that do not necessarily depend directly on the measured marginals. To do this we need to identify a distribution $\hat{\mathbf{p}}$ that has marginals $\hat{\boldsymbol{\mu}}$, and we must have tractable representation of this distribution. Also, since there may be many $\hat{\mathbf{p}}$ that give rise to the same marginals, we want a principled criteria to choose a single estimate, such as the principle of maximum entropy. We accomplish these goals using undirected graphical models.

Definition 3 (Graphical model)**.** *Let* $\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{Z}\exp\left(\sum_{C\in\mathcal{C}} \boldsymbol{\theta}_C(\mathbf{x}_C)\right)$ *be a normalized distribution, where* $\boldsymbol{\theta}_C \in \mathbb{R}^{n_C}$*. This distribution is a graphical model that factors over the measurement sets* $\mathcal{C}$*, which are the cliques of the graphical model. The vector* $\boldsymbol{\theta} = (\boldsymbol{\theta}_C)_{C\in\mathcal{C}}$ *is the parameter vector.*

Theorem 1 (Maximum entropy (Wainwright & Jordan, 2008))**.** *Given any* $\hat{\boldsymbol{\mu}}$ *in the interior of* $\mathcal{M}$ *there is a parameter vector* $\hat{\boldsymbol{\theta}}$ *such that the graphical model* $\mathbf{p}_{\hat{\boldsymbol{\theta}}}(\mathbf{x})$ *has maximum entropy among all* $\hat{\mathbf{p}}(\mathbf{x})$ *with marginals* $\hat{\boldsymbol{\mu}}$*.*[2]

Theorem 1 says that, after finding $\hat{\boldsymbol{\mu}}$, we can obtain a factored representation of the maximum-entropy distribution with these marginals by finding the graphical model parameters $\hat{\boldsymbol{\theta}}$. This is the problem of learning in an graphical model, which is well understood (Wainwright & Jordan, 2008).

Algorithm 1 Proximal Estimation Algorithm

Input: Loss function $L(\boldsymbol{\mu})$ between $\boldsymbol{\mu}$ and $\mathbf{y}$
Output: Estimated data distribution $\hat{\mathbf{p}}_{\boldsymbol{\theta}}$
$\boldsymbol{\theta} = \mathbf{0}$
for $t = 1, \ldots, T$ **do**
  $\boldsymbol{\mu}$ =MARGINAL-ORACLE($\boldsymbol{\theta}$)
  $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta_t \nabla L(\boldsymbol{\mu})$
end for
return $\hat{\mathbf{p}}_{\boldsymbol{\theta}}$

3.2. Estimation: optimizing over the marginal polytope

We need algorithms to find $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\theta}}$. We considered a variety of algorithms and present two of them here. Both are proximal algorithms for solving convex problems with "simple" constraints (Parikh et al., 2014). Central to our algorithms is a subroutine MARGINAL-ORACLE, which is some black-box algorithm for computing the clique marginals $\boldsymbol{\mu}$ of a graphical model from the parameters $\boldsymbol{\theta}$. This is the problem of *marginal inference* in a graphical model. MARGINAL-ORACLE may be any marginal inference routine — we use belief propagation on a junction tree. In the remainder of this section, we assume that the clique set $\mathcal{C}$ are the cliques of a junction tree. This is without loss of generality, since we can enlarge cliques as needed until this property is satisfied.

Algorithm 1 is a routine to find $\hat{\boldsymbol{\mu}}$ by solving a convex optimization problem over the marginal polytope. Due to the special structure of the algorithm it also finds the parameters $\hat{\boldsymbol{\theta}}$. Algorithm 1 is inspired by the entropic mirror descent algorithm for solving convex optimization problems over the probability simplex (Beck & Teboulle, 2003). The iterates of the optimization are obtained by solving simpler optimization problems of the form:

$$\boldsymbol{\mu}^{t+1} = \operatorname*{argmin}_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^T \nabla L(\boldsymbol{\mu}^t) + \frac{1}{\eta_t} D(\boldsymbol{\mu}, \boldsymbol{\mu}^t) \qquad (1)$$

where D is a Bregman divergence that is chosen to reflect the geometry of the marginal polytope. Here we use the following Bregman divergence generated from the Shannon entropy: $D(\boldsymbol{\mu}, \boldsymbol{\mu}^t) = -H(\boldsymbol{\mu}) + H(\boldsymbol{\mu}^t) + (\boldsymbol{\mu} - \boldsymbol{\mu}^t)^T \nabla H(\boldsymbol{\mu}^t)$, where $H(\boldsymbol{\mu})$ is the Shannon entropy of the graphical model $\mathbf{p}_{\boldsymbol{\theta}}$ with marginals $\boldsymbol{\mu}$. Since we assumed above that $\boldsymbol{\mu}$ are marginals of the cliques of a junction tree, the Shannon entropy is convex and easily computed as a function of $\boldsymbol{\mu}$ alone (Wainwright & Jordan, 2008).[3]

[2] If the marginals are on the boundary of $\mathcal{M}$, e.g., if they contain zeros, there is a sequence of parameters $\{\boldsymbol{\theta}^{(n)}\}$ such that $\mathbf{p}_{\boldsymbol{\theta}^{(n)}}(\mathbf{x})$ converges to the maximum-entropy distribution as $n \to \infty$. See (Wainwright & Jordan, 2008).

[3] An alternative would be to use the Bethe entropy as in (Vilnis et al., 2015). The Bethe entropy is convex and computable from $\boldsymbol{\mu}$ alone regardless of the model structure. Using Bethe entropy would lead to approximate marginal inference instead of exact

Algorithm 2 Accelerated Proximal Estimation Algorithm

Input: Loss function $L(\boldsymbol{\mu})$ between $\boldsymbol{\mu}$ and $\mathbf{y}$
Output: Estimated data distribution $\hat{\mathbf{p}}_{\boldsymbol{\theta}}$
$K =$ Lipchitz constant of ∇L
$\bar{\boldsymbol{g}} = \mathbf{0}$
$\boldsymbol{\nu}, \boldsymbol{\mu} =$MARGINAL-ORACLE$(\mathbf{0})$
for $t = 1, \ldots, T$ **do**
  $c = \frac{2}{t+1}$
  $\boldsymbol{\omega} = (1-c)\boldsymbol{\mu} + c\boldsymbol{\nu}$
  $\bar{\boldsymbol{g}} = (1-c)\bar{\boldsymbol{g}} + c\nabla L(\boldsymbol{\omega})$
  $\boldsymbol{\theta} = \frac{-t(t+1)}{4K}\bar{\boldsymbol{g}}$
  $\boldsymbol{\nu} =$MARGINAL-ORACLE$(\boldsymbol{\theta})$
  $\boldsymbol{\mu} = (1-c)\boldsymbol{\mu} + c\boldsymbol{\nu}$
end for
return graphical model $\hat{\mathbf{p}}_{\boldsymbol{\theta}}$ with marginals $\boldsymbol{\mu}$

With this divergence, the objective of the subproblem in Equation 1 can be seen to be equal to a variational free energy, which is minimized by marginal inference in a graphical model. The full derivation is provided in the supplement. The implementation of Algorithm 1 is very simple — it simply requires calling MARGINAL-ORACLE at each iteration. Additionally, even though the algorithm is designed to find the optimal $\boldsymbol{\mu}$, it also returns the corresponding graphical model parameters $\boldsymbol{\theta}$ "for free" as a by-product of the optimization. This is evident from Algorithm 1: upon convergence, $\boldsymbol{\mu}$ is the vector of marginals of the graphical model with parameters $\boldsymbol{\theta}$. The variable η_t in this algorithm is a step size, which can be constant, decreasing, or found via line search. This algorithm is an instance of mirror descent, and thus inherits its convergence guarantees. It will converge for any convex loss function L at a $O(1/\sqrt{t})$ rate,[4] even ones that are not smooth, such as the L_1 loss.

We now present a related algorithm which is based on the same principles as Algorithm 1 but has an improved $O(1/t^2)$ convergence rate for convex loss functions with Lipchitz continuous gradients. Algorithm 2 is based on Nesterov's accelerated dual averaging approach (Nesterov, 2009; Xiao, 2010; Vilnis et al., 2015). The per-iteration complexity is the same as Algorithm 1 as it requires calling the MARGINAL-ORACLE once, but this algorithm will converge in fewer iterations. Algorithm 2 has the advantage of not requiring a step size to be set, but it requires knowledge of the Lipchitz constant of ∇L. For the standard L_2 loss with linear measurements, this is equal to the largest eigenvalue of $\mathbf{Q}^T\mathbf{Q}$. The derivation of this algorithm appears in the supplement.

marginal inference as the subproblems, which is an interesting direction for future work.

[4] That is, $L(\boldsymbol{\mu}^t) - L(\boldsymbol{\mu}^*) \in O(1/\sqrt{t})$.

3.3. Inference

Once $\hat{\mathbf{p}}_{\boldsymbol{\theta}}$ has been estimated, we need algorithms to answer new queries without materializing the full contingency table representation. This corresponds to the problem of inference in a graphical model. If the new queries only depend on $\hat{\mathbf{p}}_{\boldsymbol{\theta}}$ through its clique marginals $\boldsymbol{\mu}$, we can immediately answer them using MARGINAL-ORACLE, or by saving the final value of $\boldsymbol{\mu}$ from Algorithms 1 or 2. If the new queries depend on some other marginals outside of the cliques of the graphical model, we instead use the variable elimination algorithm (Koller & Friedman, 2009) to first compute the necessary marginal, and then answer the query. In Section B of the supplement, we present a novel inference algorithm that is related to variable elimination but is faster for answering certain queries because it does not need to materialize full marginals if the query does not need them. For more complicated downstream tasks, we can generate synthetic data by sampling from $\hat{\mathbf{p}}_{\boldsymbol{\theta}}$, although this should be avoided when possible as it introduces additional sampling error.

4. Use in Privacy Mechanisms

Next we describe how our estimation algorithms can improve the accuracy and/or scalability of four state-of-the-art mechanisms: MWEM, PrivBayes, HDMM, and DualQuery.

MWEM. The multiplicative weights exponential mechanism (Hardt et al., 2012) is an active-learning style algorithm that is designed to answer a workload of linear queries. MWEM maintains an approximation of the data distribution and at each time step selects the worst approximated query $\mathbf{q}_i^T$ from the workload via the exponential mechanism (McSherry & Talwar, 2007). It then measures the query using the Laplace mechanism as $y_i = \mathbf{q}_i^T\mathbf{p} + z_i$ and then updates the approximate data distribution by incorporating the measured information using the multiplicative weights update rule. The most basic version of MWEM represents the approximate data distribution in vector form, and updates it according to the following formula after each iteration:

$$\hat{\mathbf{p}} \leftarrow \hat{\mathbf{p}} \odot \exp\left(-\mathbf{q}_i(\mathbf{q}_i^T\hat{\mathbf{p}} - y_i)/2m\right)/Z, \qquad (2)$$

where $\odot$ is elementwise multiplication and Z is a normalization constant.

It is infeasible to represent $\mathbf{p}$ explicitly for high-dimensional data, so this version of MWEM is only applicable to relatively low-dimensional data. Hardt et al describe an enhanced version of MWEM, which we call *factored MWEM*, that is able to avoid materializing this vector explicitly, in the special case when the measured queries decompose over disjoint subsets of attributes. In that case, $\mathbf{p}$ is represented implicitly as a product of independent distributions over smaller domains, i.e., $\mathbf{p}(\mathbf{x}) = \prod_{C\in\mathcal{C}} \mathbf{p}_C(\mathbf{x}_C)$, and the update is done on one group at a time. However, this en-

hancement breaks down for measurements on overlapping subsets of attributes in high-dimensional data, so MWEM is still generally infeasible to run except on simple workloads.

We can replace the multiplicative weights update with a call to Algorithm 2 using the standard L_2 loss function (on all measurements up to that point in the algorithm). By doing so, we learn a compact graphical model representation of $\hat{\mathbf{p}}$, which avoids materializing the full $\mathbf{p}$ vector even when the measured queries overlap in complicated ways. This allows MWEM to scale better and run in settings where it was previously infeasible. We remark that Equation 2 is closely related to the update equation for entropic mirror descent (Beck & Teboulle, 2003), suggesting that if the update equation is iterated until convergence, it solves the same L_2 minimization problem that we consider. More details on this are given in Section E.2 of the supplement.

PrivBayes. PrivBayes (Zhang et al., 2017) is a differentially private mechanism that generates synthetic data. It first spends half the privacy budget to learn a Bayesian network structure that captures the dependencies in the data, and then uses the remaining privacy budget to measure the statistics—which are marginals—necessary to learn the Bayesian network parameters. PrivBayes uses a heuristic of truncating negative entries of noisy measurements and normalizing to get conditional probability tables. It then samples a synthetic dataset of m records from the Bayesian network from which consistent answers to workload queries can be derived. While this is simple and efficient, the heuristic does not properly account for measurement noise and sampling may introduce unnecessary error.

We can replace the PrivBayes estimation and sampling step with a call to Algorithm 2, using an appropriate loss function (e.g. L_1 or L_2), to estimate a graphical model. Then we can answer new queries by performing graphical model inference (Section 3.3), rather than using synthetic data.

HDMM. The high-dimensional matrix mechanism (McKenna et al., 2018) is designed to answer a workload of linear queries on multi-dimensional data. It selects the set of measurements that minimizes estimated error on the input workload. The measurements are then answered using the Laplace mechanism, and inconsistencies resolved by solving an ordinary least squares problem of the form: $\hat{\mathbf{p}} = \text{argmin} \|\mathbf{Qp} - \mathbf{y}\|_2$. Solving this least squares problem is the main bottleneck of HDMM, as it requires materializing the data vector even when $\mathbf{Q}$ contains queries over the marginals of $\mathbf{p}$.

We can replace the HDMM estimation procedure with Algorithm 2, using the same L_2 loss function. If the workload contains queries over low-dimensional marginals of $\mathbf{p}$, then $\mathbf{Q}$ will contain measurements over the low-dimensional marginals too. Thus, we replace the full "probability" vector $\hat{\mathbf{p}}$ with a graphical model $\hat{\mathbf{p}}_{\boldsymbol{\theta}}$. Also $\hat{\mathbf{p}}$ may contain negative values and need not sum to 1 since HDMM solves an *ordinary* (unconstrained) least squares problem.

DualQuery. DualQuery (Gaboardi et al., 2014) is an iterative algorithm inspired by the same two-player game underlying MWEM. It generates synthetic data to approximate the true data on a workload of linear queries. DualQuery maintains a distribution over the workload queries that depends on the true data so that poorly approximated queries have higher probability mass. In each iteration, samples are drawn from the query distribution, which are proven to be differentially private. The sampled queries are then used to find a single record from the data domain (without accessing the protected data), which is added to the synthetic database.

The measurements — i.e., the random outcomes from the privacy mechanism — are the queries sampled in each iteration. Even though these are very different from the linear measurements we have primarily focused on, we can still express the log-likelihood as a function of $\mathbf{p}$ and select $\mathbf{p}$ to maximize the log-likelihood using Algorithm 1 or 2. The log-likelihood only depends on $\mathbf{p}$ through the answers to the workload queries. If the workload can be expressed in terms of $\boldsymbol{\mu}$ instead, the log-likelihood can as well. Thus, after running DualQuery, we can call Algorithm 1 with this custom loss function to estimate the data distribution, which we can use in place of the synthetic data produced by DualQuery. The full details are given in the supplementary material.

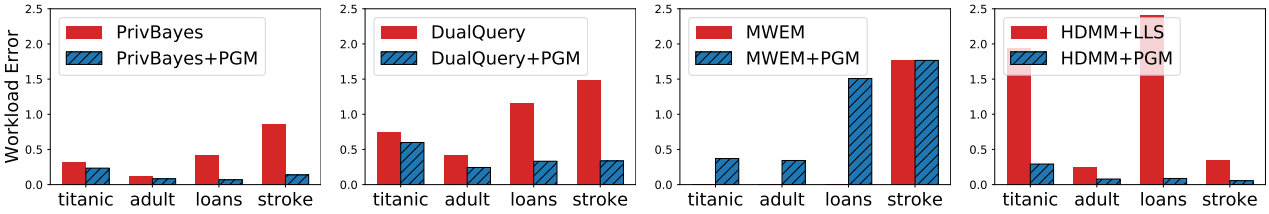
5. Experimental evaluation

In this section, we measure the accuracy and scalability improvements enabled by probabilistic graphical-model (PGM) based estimation when it is incorporated into existing privacy mechanisms.

5.1. Adding PGM estimation to existing algorithms

We run four algorithms: MWEM, PrivBayes, HDMM, and DualQuery, with and without our graphical model technology using a privacy budget of $\epsilon = 1.0$ (and $\delta = 0.001$ for DualQuery). We run Algorithm 1 with line search for DualQuery and Algorithm 2 for the other mechanisms, each for 10000 iterations. We repeat each experiment five times and report the median workload error. Experiments are done on 2 cores of a single compute cluster node with 16 GB of RAM and 2.4 GHz processors.

We use a collection of four multi-dimensional datasets in our experiments, summarized in Table 1. Each dataset consists of a collection of categorical and numerical attributes (with the latter discretized into 100 bins). Note the large domain of each dataset, which is the main property that makes efficient estimation challenging.



(a) PrivBayes (b) DualQuery (c) MWEM (d) HDMM

Figure 1: Workload error of four mechanisms on four datasets, with and without our PGM estimation algorithm for $\epsilon = 1.0$.

Table 1: Datasets used in experiments along with the number of queries in the workload used with the dataset.

Dataset	Records	Attributes	Domain	Queries
Titanic	1304	9	3e8	4851
Adult	48842	15	1e19	62876
Loans	42535	48	5e80	362201
Stroke	19434	110	4e104	17716

For each dataset, we construct a workload of counting queries which is an extension of the set of three-way marginals. First, we randomly choose 15 subsets of attributes of size 3, $\mathcal{C}$. For each subset $C \in \mathcal{C}$, if C contains only categorical attributes, we define sub-workload $\mathbf{W}_C$ to be a 3-way marginal. However, when C contains any discretized numerical attributes, we replace the set of unit queries used in a marginal with the set of prefix range queries. For example, if $C = \langle \text{sex, education, income} \rangle$ then the resulting subworkload $\mathbf{W}_C$ would consist of all queries of the form: sex $= x$, education $= y$, income $\in [0, z]$ where x, y, z range over the domains of the attributes, respectively. The final workload is the union of the 15 three-way subworkloads defined above.

We measure the error on the workload queries as:

$$Error = \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} \frac{\|\mathbf{W}_C \boldsymbol{\mu}_C - \mathbf{W}_C \hat{\boldsymbol{\mu}}_C\|_1}{2\|\mathbf{W}_C \boldsymbol{\mu}_C\|_1}$$

where the summand is related to the total variation distance (and is equal in the special case when $\mathbf{W}_C = \mathbf{I}$).

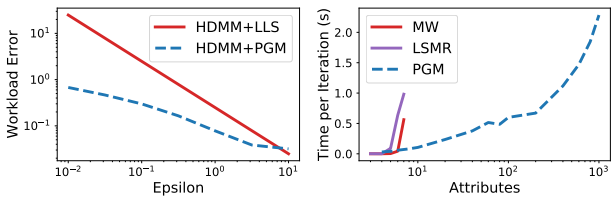
Improved accuracy. PrivBayes and DualQuery are highly scalable algorithms supporting the large domains considered here. Figures 1a and 1b show that incorporating PGM estimation significantly improves accuracy. For PrivBayes, workload error is reduced by a factor of 6× and 7× on the Loans and Stroke datasets, respectively, and a modest 30% for Adult. For DualQuery, we also observe very significant error reductions of 1.2×, 1.8×, 3.5×, and 4.4×.

Replacing infeasible estimation methods. The MWEM and HDMM algorithms fail to run on the datasets and workloads we consider because both require representations too large to maintain in memory. However, incorporating PGM estimation makes these algorithms feasible.

As Figure 1c shows, for the first three datasets, MWEM crashed before completing because it ran out of memory or timed out. For example, on one run of the Adult dataset, the first three chosen queries were on the (race, native-country, income), (workclass, race, capital-gain), and (marital status, relationship, capital-gain) marginals. Since these all overlap with respect to race and capital-gain, factored MW offers no benefit and the entire vector $\mathbf{p}_C$ must be materialized over these attributes, which requires over 100 MB. After 5 iterations, the representation requires more than 2 GB, at which point it timed out. Interestingly, MWEM was able to run on the stroke dataset, which has the largest domain and greatest number of attributes. This is mainly because the workload did not contain as many queries involving common attributes. In general, MWEM's representation will not explode as long as the workload (and therefore its measurements) consist solely of queries defined over low-dimensional marginals that do not have common attributes. Unfortunately this imposes a serious restriction on the workloads MWEM can support.

Although the HDMM algorithm fails to run, for the purpose of comparison, we run a modified version of the algorithm (denoted HDMM+LLS) which uses local least squares independently over each measurement set instead of global least squares over the full data vector. While scalable, Figure 1d shows that this estimation is substantially worse than PGM estimation, especially on the titanic and loans dataset. Incorporating PGM estimation offers error reductions of 6.6×, 3.2×, 27×, and 6.3× on the four datasets. These improvements primarily stem from non-negativity and global consistency.

Varying epsilon. While ϵ is set to 1 in Figure 1, in Figure 2a we look at the impact of varying ϵ, for a fixed dataset and measurement set. We use the Adult dataset and the measurements selected by HDMM, (which do not depend on ϵ). The magnitude of the improvement offered by our PGM estimation algorithm increases as ϵ decreases. At $\epsilon = 0.3$ and below, the mechanism has virtually no utility without



(a) (b)

Figure 2: (a) Error of HDMM variants on Adult as a function of ϵ. (b) Scalability of estimation algorithms.

PGMs. At the highest ϵ of 10.0, HDMM+LLS actually offers slightly lower error than HDMM+PGM on the workload, although both have very low error in an absolute sense. The error of HDMM+PGM on the *measurements* is still better by more than a factor of three at this privacy level. This behavior has been observed before in the low-dimensional setting, where the ordinary least squares estimator generalizes better than the non-negative least squares estimator for workloads with range queries (Li et al., 2015).

5.2. The scalability of PGM estimation

We now evaluate the scalability of our approach compared with two other general-purpose estimation techniques: multiplicative weights (MW; Hardt et al., 2012) and iterative ordinary least squares (LSMR; Fong & Saunders, 2011, Zhang et al., 2018). We omit from comparison PrivBayes estimation and DualQuery estimation because they are special-purpose estimation methods that cannot handle arbitrary linear measurements. We use synthetic data so that we can systematically vary the domain size and the number of attributes. We measure the marginals for each triple of adjacent attributes — i.e., $\mathbf{Q}_C = \mathbf{I}$ for all $C = (i, i+1, i+2)$ where $1 \leq i \leq d-2$. In Figure 2b, we vary the number of attributes from 3 to 1000 (fixing the domain of each attribute, $|\mathcal{X}_i|$ at 10), and plot the time per iteration of each of these estimation algorithms. Both MW and LSMR fail to scale beyond datasets with 10 attributes, as they both require materializing $\mathbf{p}$ in vector form, while PGM easily scales to datasets with 1000 attributes.

The domain size is the primary factor that determines scalability of the baseline methods. However, the scalability of PGM primarily depends on the complexity of the measurements taken. In the experiment above, the measurements were chosen to highlight a case where PGM estimation scales very well. In general, when the graphical model implied by the measurements has high tree-width, our methods will have trouble scaling, as MARGINAL-ORACLE is computationally expensive. In these situations, MARGINAL-ORACLE may be replaced with an approximate marginal inference algorithm, like loopy belief propagation (Wainwright & Jordan, 2008).

6. Related Work

The release of linear query answers has been extensively studied by the privacy community (Zhang et al., 2017; Li et al., 2015; Zhang et al., 2014; Li et al., 2014; Gaboardi et al., 2014; Yaroslavtsev et al., 2013; Qardaji et al., 2013b; Nikolov et al., 2013; Thaler et al., 2012; Hardt et al., 2012; Cormode et al., 2012; Acs et al., 2012; Gupta et al., 2011; Ding et al., 2011; Xiao et al., 2010; Li et al., 2010; Hay et al., 2010; Hardt & Talwar, 2010; Barak et al., 2007; McKenna et al., 2018; Eugenio & Liu, 2018). Early work using inference includes (Barak et al., 2007; Hay et al., 2010; Williams & McSherry, 2010), motivated by consistency as well as potential accuracy improvements. Inference has since been widely used in techniques for answering linear queries (Lee et al., 2015). These mechanisms often contain custom specialized inference algorithms that exploit properties of the measurements taken, and can be replaced by our algorithms.

(Williams & McSherry, 2010) introduce the problem of finding posterior distributions over model parameters from the output of differentially private algorithms. Their problem formulation requires a known model parameterization and a prior distribution over the parameter space. Their approach requires approximating a high-dimensional integral, which they do either by Markov chain Monte Carlo, or by upper and lower bounds via the "factored exponential mechanism". In the discrete data case, these bounds require summing over the data domain, which is just as hard as materializing $\mathbf{p}$ and is not feasible for high-dimensional data.

(Bernstein et al., 2017) consider the task of privately learning the parameters of an undirected graphical model. They do so by releasing noisy sufficient statistics using the Laplace mechanism, and then using an expectation maximization algorithm to learn model parameters from the noisy sufficient statistics. Their work shares some technical similarities with ours, but the aims are different. They have the explicit goal of learning a graphical model whose structure is specified in advance and used to determine the measurements. Our goal is to find a compact representation of some data distribution that minimizes a loss function where the measurements are determined externally; the graphical model structure is a by-product of the measurements made and the maximum entropy criterion.

(Chen et al., 2015) consider the task of privately releasing synthetic data. Their mechanism is similar to PrivBayes, but it uses undirected graphical models instead of Bayesian networks. It finds a good model structure using a mutual information criteria, then measures the sufficient statistics of the model (which are marginals) and post-processes them to resolve inconsistencies. This post-processing is based on a technique developed by (Qardaji et al., 2014) that ensures all measured marginals are internally consistent, and may be improved with our methods.

References

Acs, G., Castelluccia, C., and Chen, R. Differentially private histogram publishing through lossy compression. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pp. 1–10, 2012.

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

Beck, A. and Teboulle, M. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Bernstein, G., McKenna, R., Sun, T., Sheldon, D., Hay, M., and Miklau, G. Differentially private learning of undirected graphical models using collective graphical models. In *ICML*, 2017.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

Chen, R., Xiao, Q., Zhang, Y., and Xu, J. Differentially private high-dimensional data publication via sampling-based inference. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 129–138. ACM, 2015.

Cormode, G., Procopiuc, M., Srivastava, D., Shen, E., and Yu, T. Differentially private spatial decompositions. ICDE, 2012.

Ding, B., Winslett, M., Han, J., and Li, Z. Differentially private data cubes: optimizing noise sources and consistency. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pp. 217–228. ACM, 2011.

Domke, J. Learning graphical model parameters with approximate marginal inference. *IEEE transactions on pattern analysis and machine intelligence*, 35(10):2454–2467, 2013.

Dwork, C. and Roth, A. *The Algorithmic Foundations of Differential Privacy*. Foundations and Trends in Theoretical Computer Science, 2014.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Third Theory of Cryptography Conference*, 2006.

Eaton, F. and Ghahramani, Z. Choosing a variable to clamp. In *Artificial Intelligence and Statistics*, pp. 145–152, 2009.

Eugenio, E. C. and Liu, F. Cipher: Construction of differentially private microdata from low-dimensional histograms via solving linear equations with tikhonov regularization. *arXiv preprint arXiv:1812.05671*, 2018.

Fong, D. C.-L. and Saunders, M. Lsmr: An iterative algorithm for sparse least-squares problems. *SIAM Journal on Scientific Computing*, 33(5):2950–2971, 2011.

Gaboardi, M., Arias, E. J. G., Hsu, J., Roth, A., and Wu, Z. S. Dual query: Practical private query release for high dimensional data. In *ICML*, 2014.

Gupta, A., Hardt, M., Roth, A., and Ullman, J. Privately releasing conjunctions and the statistical query barrier. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pp. 803–812, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0691-1. doi: 10.1145/1993636.1993742. URL http://doi.acm.org/10.1145/1993636.1993742.

Hardt, M. and Rothblum, G. N. A multiplicative weights mechanism for privacy-preserving data analysis. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pp. 61–70. IEEE, 2010.

Hardt, M. and Talwar, K. On the geometry of differential privacy. In *Symposium on Theory of computing (STOC)*, pp. 705–714, 2010.

Hardt, M., Ligett, K., and McSherry, F. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems*, pp. 2339–2347, 2012.

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

Koller, D. and Friedman, N. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Lee, J., Wang, Y., and Kifer, D. Maximum likelihood post-processing for differential privacy under consistency constraints. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 635–644. ACM, 2015.

Li, C., Hay, M., Rastogi, V., Miklau, G., and McGregor, A. Optimizing linear counting queries under differential privacy. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 123–134. ACM, 2010.

Li, C., Hay, M., Miklau, G., and Wang, Y. A data-and workload-aware algorithm for range queries under differential privacy. *Proceedings of the VLDB Endowment*, 7 (5):341–352, 2014.

Li, C., Miklau, G., Hay, M., McGregor, A., and Rastogi, V. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB journal*, 24 (6):757–781, 2015.

Maclaurin, D., Duvenaud, D., and Adams, R. P. Autograd: Effortless gradients in numpy. In *ICML 2015 AutoML Workshop*, 2015.

McKenna, R., Miklau, G., Hay, M., and Machanavajjhala, A. Optimizing error of high-dimensional statistical queries under differential privacy. *Proceedings of the VLDB Endowment*, 11(10):1206–1219, 2018.

McSherry, F. and Talwar, K. Mechanism design via differential privacy. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pp. 94–103. IEEE, 2007.

Nesterov, Y. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

Nikolov, A., Talwar, K., and Zhang, L. The geometry of differential privacy: the approximate and sparse cases. In *Symposium on Theory of Computing*, 2013.

Parikh, N., Boyd, S., et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

Proserpio, D., Goldberg, S., and McSherry, F. Calibrating Data to Sensitivity in Private Data Analysis. In *Conference on Very Large Data Bases (VLDB)*, 2014.

Qardaji, W., Yang, W., and Li, N. Differentially private grids for geospatial data. In *ICDE, to appear*, 2013a. URL http://arxiv.org/abs/1209.1322.

Qardaji, W., Yang, W., and Li, N. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment*, 6(14):1954–1965, 2013b.

Qardaji, W., Yang, W., and Li, N. Priview: Practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 1435–1446. ACM, 2014.

Thaler, J., Ullman, J., and Vadhan, S. Faster algorithms for privately releasing marginals. In *Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part I*, ICALP'12, pp. 810–821, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-31593-0. doi: 10.1007/978-3-642-31594-7_68. URL http://dx.doi.org/10.1007/978-3-642-31594-7_68.

Vilnis, L., Belanger, D., Sheldon, D., and McCallum, A. Bethe projections for non-local inference. *arXiv preprint arXiv:1503.01397*, 2015.

Wainwright, M. J. and Jordan, M. I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Williams, O. and McSherry, F. Probabilistic inference and differential privacy. In *Advances in Neural Information Processing Systems*, pp. 2451–2459, 2010.

Xiao, L. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11(Oct):2543–2596, 2010.

Xiao, X., Wang, G., and Gehrke, J. Differential privacy via wavelet transforms. In *International Conference on Data Engineering*, 2010.

Yaroslavtsev, G., Cormode, G., Procopiuc, C. M., and Srivastava, D. Accurate and efficient private release of datacubes and contingency tables. In *ICDE*, 2013.

Zhang, D., McKenna, R., Kotsogiannis, I., Hay, M., Machanavajjhala, A., and Miklau, G. Ektelo: A framework for defining differentially-private computations. In *Conference on Management of Data (SIGMOD)*, 2018.

Zhang, J., Cormode, G., Procopiuc, C. M., Srivastava, D., and Xiao, X. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)*, 42(4):25, 2017.

Zhang, X., Chen, R., Xu, J., Meng, X., and Xie, Y. Towards accurate histogram publication under differential privacy. In *SIAM International Conference on Data Mining (SDM)*. SIAM, 2014.

A. Estimation

Define $\boldsymbol{\mu}(\boldsymbol{\theta})$ to be the marginals of the graphical model with parameters $\boldsymbol{\theta}$, which may be computed with the MARGINAL-ORACLE.

A.1. Proximal Algorithm Derivation

Our goal is to solve the following optimization problem:

$$\hat{\boldsymbol{\mu}} = \operatorname*{argmin}_{\boldsymbol{\mu} \in \mathcal{M}} L(\boldsymbol{\mu})$$

where L is some convex function such as $\|\mathbf{Q}_{\mathcal{C}}\boldsymbol{\mu} - \mathbf{y}\|$.

Using the mirror descent algorithm (Beck & Teboulle, 2003), we can use the following update equation:

$$\boldsymbol{\mu}^{t+1} = \operatorname*{argmin}_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{\mu}^T \nabla L(\boldsymbol{\mu}^t) + \frac{1}{\eta_t} D(\boldsymbol{\mu}, \boldsymbol{\mu}^t)$$

Where D is a Bregman distance measure defined as

$$D(\boldsymbol{\mu}, \boldsymbol{\mu}^t) = \psi(\boldsymbol{\mu}) - \psi(\boldsymbol{\mu}^t) - (\boldsymbol{\mu} - \boldsymbol{\mu}^t)^T \nabla \psi(\boldsymbol{\mu}^t)$$

for some strongly convex and continuously differentiable function ψ. Using $\psi = -H$ to be the negative entropy, we arrive at the following update equation:

$$\begin{aligned}
\boldsymbol{\mu}^{t+1} &= \operatorname*{argmin}_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{\mu}^T \nabla L(\boldsymbol{\mu}^t) + \frac{1}{\eta_t} D(\boldsymbol{\mu}, \boldsymbol{\mu}^t) \\
&= \operatorname*{argmin}_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{\mu}^T \nabla L(\boldsymbol{\mu}^t) + \frac{1}{\eta_t} \Big(- H(\boldsymbol{\mu}) + \boldsymbol{\mu}^T \nabla H(\boldsymbol{\mu}^t) \Big) \\
&= \operatorname*{argmin}_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{\mu}^T \Big(\nabla L(\boldsymbol{\mu}^t) + \frac{1}{\eta_t} \nabla H(\boldsymbol{\mu}^t) \Big) - \frac{1}{\eta_t} H(\boldsymbol{\mu}) \\
&= \operatorname*{argmin}_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{\mu}^T \Big(\eta_t \nabla L(\boldsymbol{\mu}^t) + \nabla H(\boldsymbol{\mu}^t) \Big) - H(\boldsymbol{\mu}) \\
&= \operatorname*{argmin}_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{\mu}^T \Big(\eta_t \nabla L(\boldsymbol{\mu}^t) - \boldsymbol{\theta}^t \Big) - H(\boldsymbol{\mu}) \\
&= \boldsymbol{\mu}\big(\boldsymbol{\theta}^t - \eta_t \nabla L(\boldsymbol{\mu}^t) \big)
\end{aligned}$$

The first four steps are simple algebraic manipulation of the mirror descent update equation. The final two steps use the observation that $\nabla H(\boldsymbol{\mu}^t) = -\boldsymbol{\theta}^t$ and that marginal inference can be cast as the following optimization problem: (Wainwright & Jordan, 2008; Vilnis et al., 2015)

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = \operatorname*{argmin}_{\boldsymbol{\mu} \in \mathcal{M}} -\boldsymbol{\mu}^T \boldsymbol{\theta} - H(\boldsymbol{\mu})$$

Thus, optimization over the marginal polytope is reduced to computing the marginals of a graphical model with parameters $\boldsymbol{\theta}^t - \eta_t \nabla L(\boldsymbol{\mu}^t)$, which can be accomplished using belief propagation or some other MARGINAL-ORACLE.

A.2. Accelerated Proximal Algorithm Derivation

The derivation of the accelerated proximal algorithm is similar. It is based on Algorithm 3 from (Xiao, 2010). Applied to our setting, step 4 of that algorithm requires solving the following problem:

$$\begin{aligned}
\boldsymbol{\nu}^t &= \operatorname*{argmin}_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{\mu}^T \bar{\boldsymbol{g}} - \frac{4K}{t(t+1)} H(\boldsymbol{\mu}) \\
&= \operatorname*{argmin}_{\boldsymbol{\mu} \in \mathcal{M}} \frac{t(t+1)}{4K} \boldsymbol{\mu}^T \bar{\boldsymbol{g}} - H(\boldsymbol{\mu}) \\
&= \boldsymbol{\mu}\Big(- \frac{t(t+1)}{4L} \bar{\boldsymbol{g}} \Big)
\end{aligned}$$

which we solve by using the MARGINAL-ORACLE.

A.3. Direct Optimization

In preliminary experiments we also evaluated a direct method to solve the optimization problem. For the direct method, we estimate the parameters $\hat{\boldsymbol{\theta}}$ directly by reformulating the optimization problem and instead solving the unconstrained problem $\hat{\boldsymbol{\theta}} = \operatorname{argmin}_{\boldsymbol{\theta}} L\big(\boldsymbol{\mu}(\boldsymbol{\theta})\big)$.To evaluate the optimization objective, we use MARGINAL-ORACLE to compute $\boldsymbol{\mu}(\boldsymbol{\theta})$ and then compute the loss. For optimization, it has been observed that it is possible to backpropagate through marginal inference procedures (with or without automatic differentiation software) to compute their gradients (Eaton & Ghahramani, 2009; Domke, 2013). We apply automatic differentiation to the entire forward computation (Maclaurin et al., 2015), which includes MARGINAL-ORACLE, to compute the gradient of L.

Since this is now an unconstrained optimization problem and we can compute the gradient of L, many optimization methods apply. In our experiments, we use L_2 loss, which is smooth, and apply the L-BFGS algorithm for optimization (Byrd et al., 1995).

However, despite its simplicity, there is a significant drawback to the direct algorithm. It is not, in general, convex with respect to $\boldsymbol{\theta}$. This may seem surprising since the original problem is convex, i.e., $L(\boldsymbol{\mu})$ is convex with respect to $\boldsymbol{\mu}$ and $\mathcal{M}$ is convex. Also, the most well known problem of this form, maximum-likelihood estimation in graphical models, *is* convex with respect to $\boldsymbol{\theta}$ (Wainwright & Jordan, 2008); however, this relies on properties of exponential families that do not apply to other loss functions. One can verify for losses as simple as L_2 that the Hessian need not be positive definite. As a result, the direct algorithm is not guaranteed to converge to a global minimum of the original convex optimization problem $\min_{\boldsymbol{\mu} \in \mathcal{M}} L(\boldsymbol{\mu})$. We did not observe convergence problems in our experiments, but it was not better in practice than the proximal algorithms, which is why it is not included in the paper.

Algorithm 3 Inference for Factored Queries

Input: Parameters $\boldsymbol{\theta}$, factored query matrix $\mathbf{Q}$
Output: Query answers $\mathbf{Q}\,\mathbf{p}_{\boldsymbol{\theta}}$
$\psi = \{\exp(\boldsymbol{\theta}_C) \mid C \in \mathcal{C}\} \cup \{\mathbf{Q}_i \mid i \in [d]\}$
$Z =$MARGINAL-ORACLE($\boldsymbol{\theta}$)
return VARIABLE-ELIM($\psi, \mathcal{X}$)$/Z$

B. Inference

We now discuss how to exploit our compact factored representation of $\mathbf{p}_{\boldsymbol{\theta}}$ to answer new linear queries. We give an efficient algorithm for answering *factored linear queries*.

Definition 4 (Factored Query Matrix)**.** *A factored query matrix $\mathbf{Q}$ has columns that are indexed by $\mathbf{x}$ and rows that are indexed by vectors $\mathbf{z} \in [r_1] \times \cdots \times [r_d]$. The total number of rows (queries) is $r = \prod_{i=1}^{d} r_i$. The entries of $\mathbf{Q}$ are given by $\mathbf{Q}(\mathbf{z}, \mathbf{x}) = \prod_{i=1}^{d} \mathbf{Q}_i(\mathbf{z}_i, \mathbf{x}_i)$, where $\mathbf{Q}_i \in \mathbb{R}^{r_i \times n_i}$ is a specified factor for the ith attribute. The matrix $\mathbf{Q}$ can be expressed as $\mathbf{Q} = \mathbf{Q}_1 \otimes \cdots \otimes \mathbf{Q}_d$, where $\otimes$ is the Kronecker product.*

Factored query matrices are expressive enough to encode any conjunctive query (or a cartesian product of such queries), and more. There are a number of concrete examples that demonstrate the usefulness of answering queries of this form, including:

- Computing the marginal $\boldsymbol{\mu}_C$ for any $C \subseteq [d]$ (including unmeasured marginals).
- Computing the multivariate CDF of $\boldsymbol{\mu}_C$ for any $C \subseteq [d]$.
- Answering range queries.
- Compressing the distribution by transforming the domain.
- Computing the (unnormalized) expected value of one variable conditioned on other variables.

For the first two examples, we could have used standard variable elimination to eliminate all variables except those in C. Existing algorithms are not able to handle the other examples without materializing $\hat{\mathbf{p}}$ (or a marginal that supports the queries). Thus, our algorithm generalizes variable elimination. A more comprehensive set of examples, and details on how to construct these query matrices are given in section B.1

The procedure for answering these queries is given in Algorithm 3, which can be understood as follows. For a particular $\mathbf{z}$, write $f(\mathbf{z}, \mathbf{x}) = \mathbf{Q}(\mathbf{z}, \mathbf{x})\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}) = \prod_i \mathbf{Q}_i(\mathbf{z}_i, \mathbf{x}_i)\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x})$. This can be viewed as an augmented graphical model on the variables $\mathbf{z}$ and $\mathbf{x}$ where we have introduced new pairwise factors between each $(\mathbf{x}_i, \mathbf{z}_i)$ pair defined by the query matrix. Unlike a regular graphical model, the new factors can contain negative values. The query answers are obtained by multiplying $\mathbf{Q}$ and $\mathbf{p}$, which sums over $\mathbf{x}$. The $\mathbf{z}$th answer is given by:

$$
\begin{aligned}
(\mathbf{Q}\mathbf{p}_{\boldsymbol{\theta}})(\mathbf{z}) &= \sum_{\mathbf{x} \in \mathcal{X}} \mathbf{Q}(\mathbf{z}, \mathbf{x})\mathbf{p}_{\boldsymbol{\theta}}(\mathbf{x}) \\
&= \frac{1}{Z} \sum_{\mathbf{x} \in \mathcal{X}} \prod_{i=1}^{d} \mathbf{Q}_i(\mathbf{z}_i, \mathbf{x}_i) \prod_{C \in \mathcal{C}} \exp[\boldsymbol{\theta}_C(\mathbf{x}_C)]
\end{aligned}
$$

This can be understood as marginalizing over the $\mathbf{x}$ variables in the augmented model $f(\mathbf{z}, \mathbf{x})$. The VARIABLE-ELIM routine referenced in the algorithm is standard variable elimination to perform this marginalization; it can handle negative values with no modification. We stress that, in practice, factor matrices $\mathbf{Q}_i$ may have only one row ($r_i = 1$, e.g., for marginalization); hence the output size $r = \prod_{i=1}^{d} r_i$ is not necessarily exponential in d.

B.1. Factored Query Matrices

Table 2 gives some example "building block" factors that can be used to construct factored query matrices. This is by no means an exhaustive list of possible factors but it provides the reader with evidence that answering these types of queries efficiently is practically useful. The factored query matrix for computing the marginal $\boldsymbol{\mu}_C$ uses $\mathbf{Q}_i = \mathbf{I}$ for $i \in C$ and $\mathbf{Q}_i = \mathbf{1}$ for $i \notin C$. Similarly, the factored query matrix for computing the multivariate CDF of $\boldsymbol{\mu}_C$ would simply use $\mathbf{Q}_i = \mathbf{P}$ for $i \in C$. A query matrix for compressing a distribution could be characterized by functions $f_i : [n_i] \to [2]$ or equivalently binary matrices $\mathbf{Q}_i = \mathbf{R}_{f_i} \in \mathbb{R}^{2 \times n_i}$. The query matrix for computing the (unnormalized) expected value of variable i conditioned on variable j would use $\mathbf{Q}_i = \mathbf{E}$ and $\mathbf{Q}_j = \mathbf{I}$ (and $\mathbf{Q}_k = \mathbf{1}$ for all other k). These are only a few examples; these building blocks can be combined arbitrarily to construct a wide variety of interesting query matrices.

C. Loss Functions

C.1. L_1 and L_2 losses

The L_1 and L_2 loss functions have simple (sub)gradients.

$$
\begin{aligned}
\nabla L_1(\boldsymbol{\mu}) &= \mathbf{Q}_{\mathcal{C}}^T \text{sign}(\mathbf{Q}_{\mathcal{C}}\boldsymbol{\mu} - \mathbf{y}) \\
\nabla L_2(\boldsymbol{\mu}) &= \mathbf{Q}_{\mathcal{C}}^T (\mathbf{Q}_{\mathcal{C}}\boldsymbol{\mu} - \mathbf{y})
\end{aligned}
$$

C.2. Linear measurements with unequal noise

When the privacy budget is not distributed evenly to the measurements in the we have to appropriately modify the loss functions, which assume that the noisy answers all have equal variance. In order to do proper estimation and inference we have to account for this varying noise level in the loss function. In section 3.1 we claimed that $L(\mathbf{p}) = \|\mathbf{Q}\mathbf{p} - \mathbf{y}\|$ makes sense as a loss function when the noise introduced to $\mathbf{y}$ are iid. Luckily, even if this assumption is

$\mathbf{Q}_i$	Requirements	Size	Definition ($\forall a \in [n_i]$)	Description
$\mathbf{I}$		$n_i \times n_i$	$\mathbf{Q}_i(a,a) = 1$	keep variable in
$\mathbf{1}$		$1 \times n_i$	$\mathbf{Q}_i(1,a) = 1$	marginalize variable out
$\boldsymbol{e}_j$	$j \in [n_i]$	$1 \times n_i$	$\mathbf{Q}_i(1,j) = 1$	inject evidence
$\boldsymbol{e}_S$	$S \subseteq [n_i]$	$1 \times n_i$	$\mathbf{Q}_i(1,j) = 1\ \forall j \in S$	inject evidence (disjuncts)
$\mathbf{P}$		$n_i \times n_i$	$\mathbf{Q}_i(b,a) = 1\ \forall b \geq a$	transform into CDF
$\mathbf{R}_f$	$f : [n_i] \to [r_i]$	$r_i \times n_i$	$\mathbf{Q}_i(f(a),a) = 1$	compress domain
$\mathbf{E}$		$1 \times n_i$	$\mathbf{Q}_i(1,a) = a$	reduce to mean
$\mathbf{E}_k$	$k \geq 1$	$k \times n_i$	$\mathbf{Q}_i(b,a) = a^b\ \forall b \leq k$	reduce to first k moments

Table 2: Example factors in the factored query matrix

not satisfied it is easy to correct. Assume that $y_i = \mathbf{q}_i^T \mathbf{p} + \varepsilon_i$ where $\varepsilon_i \sim Lap(b_i)$. Then $\frac{1}{b_i} y_i = \frac{1}{b_i} \mathbf{q}_i^T \mathbf{p} + \frac{1}{b_i} \varepsilon_i$ and $\frac{1}{b_i} \varepsilon_i \sim Lap(1)$. Thus, we can replace the query matrix $\mathbf{Q} \leftarrow \mathbf{D}\mathbf{Q}$ and the answer vector $\mathbf{y} \leftarrow \mathbf{D}\mathbf{y}$ where $\mathbf{D}$ is the diagonal matrix defined by $\mathbf{D}_{ii} = \frac{1}{b_i}$. All the new query answers have the same effective noise scale, and so the standard loss functions may be used. This idea still applies if the noise on each query answer is sampled from a normal distribution as well (for (ϵ, δ)-differential privacy).

C.3. Dual Query Loss Function

Algorithm 4 shows DualQuery applied to workloads defined over the marginals of the data. There are five hyper-parameters, of which four must be specified and the remaining one can be determined from the others.

The first step of the algorithm computes the answers to the workload queries. Then for T time steps observations are made about the true data via samples from the distribution Q^t. These observations are used to find a record $\mathbf{x} \in \mathcal{X}$ to add to the synthetic database.

Algorithm 4 Dual Query for marginals workloads

Input: $\boldsymbol{X}$, the true data
Input: $\mathbf{W}_{\mathcal{C}}$, workload queries
Input: $(s, T, \eta, \epsilon, \delta)$, hyper-parameters
Output: synthetic database of T records
$\mathbf{y} = \mathbf{W}_C \boldsymbol{\mu}_{\boldsymbol{X}}$
$Q^1 = \text{uniform}(\mathbf{W})$
for $t = 1, \ldots, T$ **do**
  sample $\mathbf{q}_1^t, \ldots \mathbf{q}_s^t$ from Q^t
  $\mathbf{x}^t = \text{argmax}_{\mathbf{x} \in \mathcal{X}} \sum_{i=1}^s \mathbf{q}_i^t \boldsymbol{\mu} - \mathbf{q}_i^t \boldsymbol{\mu}_{\mathbf{x}}$
  $Q^{t+1} = Q^t \odot \exp\left(-\eta * (\mathbf{y} - \mathbf{W}_C \boldsymbol{\mu}_{\mathbf{x}^t})\right)$
  normalize Q^t
end for
return $(\mathbf{x}^1, \ldots, \mathbf{x}^T)$

Algorithm 5 shows a procedure for computing the negative log likelihood (our loss function) of observing the DualQuery output, given some marginals. Evaluating the log likelihood is fairly expensive, as it requires basically simulating the entire DualQuery algorithm. Fortunately we do not have to run the most computationally expensive step within the procedure, which is finding $\mathbf{x}^t$. We differentiate this loss function using automatic differentiation (Maclaurin et al., 2015) for use within our estimation algorithms.

Algorithm 5 Dual Query Loss Function

Input: $\boldsymbol{\mu}$, marginals of the data
Input: $\mathbf{W}_{\mathcal{C}}$, workload queries
Input: cache, all relevant output from DualQuery
> $\mathbf{q}_1^t, \ldots \mathbf{q}_s^t$ - sampled queries at each time step
> $\mathbf{x}^t$ - chosen record at each time step
Output: $L(\boldsymbol{\mu})$, the negative log likelihood
$\mathbf{y} = \mathbf{W}_C \boldsymbol{\mu}$
$Q^1 = \text{uniform}(\mathbf{W})$
loss $= 0$
for $t = 1, \ldots, T$ **do**
  loss $-= \sum_{i=1}^s \log\left(Q^t(\mathbf{q}_i^t)\right)$
  $Q^{t+1} = Q^t \odot \exp\left(-\eta * (\mathbf{y} - \mathbf{W}_C \boldsymbol{\mu}_{\mathbf{x}^t})\right)$
  normalize Q^t
end for
return loss

D. Additional Experiments

D.1. L_1 vs. L_2 Loss

In Section 3 we mentioned that minimizing L_1 loss is equivalent to maximizing likelihood for linear measurements with Laplace noise, but that L_2 loss is more commonly used in the literature. In this experiment we compare these two estimators side-by-side. Specifically, we consider the workload from Figure 1 and measurements chosen by HDMM with $\epsilon = 1.0$. As expected, performing L_1 minimization results in lower L_1 loss but higher L_2 loss, although the difference is quite small, especially for L_1 loss. The difference is larger for L_2 loss. Minimizing L_2 loss results in lower workload error, indicating that it generalizes better. This is somewhat surprising given that L_1 minimization is maximizing likelihood. Another interesting observation is

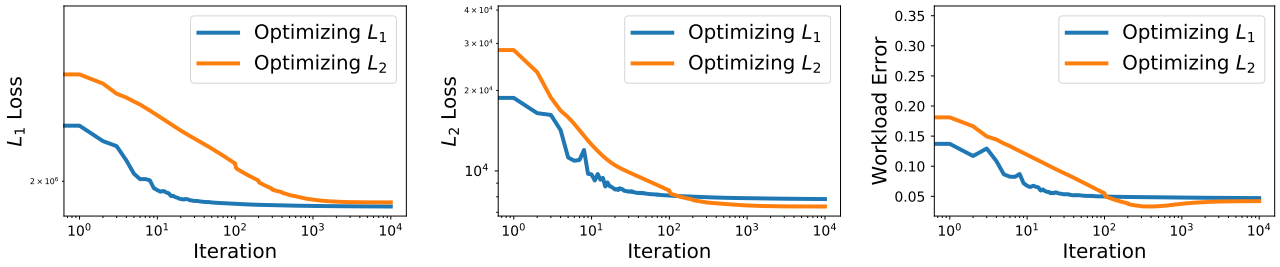


Figure 3: L_1 minimization vs. L_2 minimization, evaluated on L_1 loss, L_2 loss, and workload error

that the workload error actually starts going up after about 200 iterations, suggesting that some form of over-fitting is occurring. There minimum workload error achieved was 0.066 while the final workload error was 0.084 — a pretty meaningful difference. Of course, in practice we cannot stop iterating when workload error starts increasing because evaluating it requires looking at the true data.

E. Additional Details

E.1. Unknown Total

Our algorithms require m, the total number of records in the dataset is known or can be estimated. Under a slightly different privacy definition where $\text{nbrs}(\mathbf{X})$ is the set of databases where a single record is added or removed (instead of modified), this total is a sensitive quantity which cannot be released exactly (Dwork & Roth, 2014). Thus, the total is not known in this setting, but a good estimate can typically be obtained from the measurements taken, without spending additional privacy budget. First observe that $\mathbf{1}^T\boldsymbol{\mu}_C = m$ is the total for an unnormalized database. Now suppose we have measured $\mathbf{y}_C = \mathbf{Q}_C\boldsymbol{\mu}_C + \mathbf{z}_C$. Then as long as $\mathbf{1}^T$ is in the row-space of $\mathbf{Q}_C$, $m_C = \mathbf{1}^T\mathbf{Q}_C^+\mathbf{y}_C$ is an unbiased estimate for m with variance $Var(m_C) = Var(\mathbf{y}_C)\left\|\mathbf{1}^T\mathbf{Q}_C^+\right\|_2^2$. This is a direct consequence of Proposition 9 from (Li et al., 2015). We thus have multiple estimates for m which we can combine using inverse variance weighting, resulting in the final estimate of $\hat{m} = \frac{\sum_C m_C/Var(m_C)}{\sum_C 1/Var(m_C)}$, which we can use in place of m.

E.2. Multiplicative Weights vs Entropic Mirror Descent

Recall from Section 4 that the multiplicative weights update equation is:

$$\hat{\mathbf{p}} \leftarrow \hat{\mathbf{p}} \odot \exp\left(\mathbf{q}_i(\mathbf{q}_i^T\hat{\mathbf{p}} - y_i)\right)/2m/Z$$

and the update is applied (possibly cyclically) for $i = 1, \ldots, T$. Now imagine taking all of the measurements and organizing them into a $T \times n$ matrix $\mathbf{Q}$. Then we can apply all the updates at once, instead of sequentially, and we end up with the following update equation.

$$\hat{\mathbf{p}} \leftarrow \hat{\mathbf{p}} \odot \exp\left(\mathbf{Q}^T(\mathbf{Q}\hat{\mathbf{p}} - \mathbf{y})/2m\right)/Z$$

Observing that $\nabla L_2(\hat{\mathbf{p}}) = \mathbf{Q}^T(\mathbf{Q}\hat{\mathbf{p}} - \mathbf{y})$, this simplifies to:

$$\hat{\mathbf{p}} \leftarrow \hat{\mathbf{p}} \odot \exp\left(\nabla L_2(\hat{\mathbf{p}})/2m\right)/Z$$

which is precisely the update equation for entropic mirror descent for minimizing $L_2(\mathbf{p})$ over the probability simplex (Beck & Teboulle, 2003).