

Optimizing Fitness-For-Use of Differentially Private Linear Queries

Yingtai Xiao, Zeyu Ding, Yuxin Wang, Danfeng Zhang, Daniel Kifer
Pennsylvania State University
{yxx5224,dxd437,ykw5163,dbz5017,duk17}@psu.edu

ABSTRACT

In practice, differentially private data releases are designed to support a variety of applications. A data release is fit for use if it meets target accuracy requirements for each application. In this paper, we consider the problem of answering linear queries under differential privacy subject to per-query accuracy constraints. Existing practical frameworks like the matrix mechanism do not provide such fine-grained control (they optimize *total* error, which allows some query answers to be more accurate than necessary, at the expense of other queries that become no longer useful). Thus, we design a fitness-for-use strategy that adds privacy-preserving Gaussian noise to query answers. The covariance structure of the noise is optimized to meet the fine-grained accuracy requirements while minimizing the cost to privacy.

PVLDB Reference Format:

Yingtai Xiao, Zeyu Ding, Yuxin Wang, Danfeng Zhang, Daniel Kifer. Optimizing Fitness-For-Use of Differentially Private Linear Queries. PVLDB, 14(10): 1730 - 1742, 2021.
doi:10.14778/3467861.3467864

1 INTRODUCTION

Differential privacy gives data collectors the ability to publish information about sensitive datasets while protecting the confidentiality of the users who supplied the data. Real-world applications include OnTheMap [10, 36], Yahoo Password Frequency Lists [4], Facebook URLs Data [38], and the 2020 Decennial Census of Population and Housing [2]. Lessons learned from these early applications help identify deployment challenges that should serve as guides for future research. One of these challenges is supporting applications that end-users care about [21]. It is well-known that no dataset can support arbitrary applications while providing a meaningful degree of privacy [15]. Consequently, as shown in theory and in practice, privacy-preserving data releases must be carefully designed with intended use-cases in mind. Without such use-cases, a so-called “general-purpose” data release might not provide sufficient accuracy for any practical purpose.

Ensuring accuracy for pre-specified use-cases has strong precedents and was common practice even before the adoption of differential privacy. For example, in the 2010 Decennial Census, the U.S. Census Bureau released data products such as:

- PL 94-171 Summary Files [8] designed to support redistricting. The information was limited to the total number of people, various race/ethnicity combinations in each Census block, along with the number of such people with age 18 or higher.
- Advance Group Quarters Summary File [9] containing the number of people living in group quarters such as correctional institutions, university housing, and military quarters. One of its purposes is to help different states comply with their own redistricting laws regarding prisoners [26].
- Summary File 1 [7] provides a fixed set of tables that are commonly used to allocate federal funds and to support certain types of social science research.

Thus we consider a setting where a data publisher must release differentially private query answers to support a given set of N applications. Each application provides measures of accuracy that, if met, make the differentially private query answers fit for use. For example, one of the measures used by the American Community Survey (ACS) is *margin of error* [44], an estimated confidence interval that is a function of variance.

In this paper, we study the case where the workload consists of linear queries w_1, \dots, w_N . Differential privacy does not allow exact query answers to be released, so the data publisher must release noisy query answers instead. We consider the following fitness-for-use criteria: each workload query w_i must be answered with expected squared error $\leq c_i$ (where c_1, \dots, c_N are user-specified constants that serve as upper bounds on desired error).

Given these fitness-for-use constraints, the data publisher must determine whether they can be met under a given privacy budget and, if so, how to correlate the noise in the query answers in order to meet these constraints.¹ We note that earlier applied work on the matrix mechanism [34, 50, 51] optimized *total* error rather than per-query error, and could not guarantee that each query is fit for use. Theoretical work on the matrix mechanism [20, 40] studied expected worst-case, instead of per-query, error and, as we discuss later, it turns out that the same algorithm can optimize them.

We propose a mechanism that adds correlated Gaussian noise to the query answers. The correlation structure is designed to meet accuracy constraints while minimizing the privacy cost, and is obtained by solving an optimization problem. We analyze the solution space and show that although there potentially exist many such correlation structures, there is a unique solution that allows the maximal release of additional noisy query answers *for free* – that is, some queries that cannot be derived from the differentially private workload answers can be noisily answered without affecting the privacy parameters (this is related to personalized privacy [19]). Our contributions are:

¹If the desired accuracy cannot be met under a given privacy budget, the data collector could reinterpret the constants c_i to represent relative priorities, see Section 3.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 14, No. 10 ISSN 2150-8097.
doi:10.14778/3467861.3467864

Table 1: Table of Notation

\mathbf{x} :	Dataset vector
d :	Domain size of the data
\mathbf{W} :	Workload query matrix.
m :	Number of workload queries.
k :	Rank of \mathbf{W} .
\mathbf{y} :	True answers to workload queries ($\mathbf{y} = \mathbf{W}\mathbf{x}$).
\mathbf{B} :	Basis matrix with linearly independent rows.
\mathbf{b}_i :	The i^{th} column of \mathbf{B} .
\mathbf{L} :	Representation matrix. Note, $\mathbf{L}\mathbf{B} = \mathbf{W}$.
\mathbf{c} :	Vector of accuracy targets (c_1, \dots, c_m) .
\leq_R :	Refined privacy ordering (Definition 5).
$\text{prof}(\Sigma, \mathbf{B})$:	Privacy profile (Definition 4).
Δ_M :	Privacy cost (Corollary 1).

- A novel differentially private mechanism for releasing linear query answers subject to fitness-for-use constraints (to the best of our knowledge, this is the first such mechanism). It uses non-trivial algorithms for optimizing the covariance matrix of the Gaussian noise that is added to the query answers.
- We theoretically study the fitness-for-use problem. Although there are potentially infinitely many covariance matrices that can be used to minimize privacy cost while meeting the accuracy constraints, we show that there is a unique solution that allows the data publisher to noisily answer a maximal amount of additional queries at no extra privacy cost.
- We design experiments motivated by real-world use cases to show the efficacy of our approach.

The outline of this paper is as follows. In Section 2, we present notation and background material. We formalize the problem in Section 3. We discuss related work in Section 4. We present theoretical results of the solution space in Section 5. We present our optimization algorithms for the fitness-for-use problem in Section 6. We present experiments in Section 7 and conclusions in Section 8. All proofs can be found in the full version [48] of this paper.

2 NOTATION AND BACKGROUND

We denote vectors as bold lower-case letters (e.g., \mathbf{x}), matrices as bold upper-case (e.g., \mathbf{W}), scalars as non-bold lower-case (e.g., c).

Following earlier work on differentially private linear queries [34, 50, 51], we work with a table whose attributes are categorical (or have been discretized). As in prior work, we represent such a table as a vector \mathbf{x} of counts. That is, letting $\{t_0, t_1, \dots, t_{d-1}\}$ be the set of possible tuples, $\mathbf{x}[i]$ is the number of times tuple t_i appears in the table. For example, consider a table on two attributes, *adult* (yes/no) and *Hispanic* (yes/no). We set t_0 = “not adult, not Hispanic”, t_1 = “adult, not Hispanic”, t_2 = “not adult, Hispanic”, t_3 = “adult, Hispanic”. Then $\mathbf{x}[3]$ is the number of Hispanic adults in the dataset.

We refer to \mathbf{x} as a *dataset vector* and we say that two dataset vectors \mathbf{x} and \mathbf{x}' are *neighboring* (denoted as $\mathbf{x} \sim \mathbf{x}'$) if \mathbf{x} can be obtained from \mathbf{x}' by adding or subtracting 1 from some component of \mathbf{x}' (this means $\|\mathbf{x} - \mathbf{x}'\|_1 = 1$) – this is the same as adding or removing 1 person from the underlying table.

A single linear query \mathbf{w} is a vector, whose answer is $\mathbf{w} \cdot \mathbf{x}$. A set of m linear queries is represented by an $m \times d$ matrix \mathbf{W} , where each row corresponds to a single linear query. The answers to those queries are obtained by matrix multiplication: $\mathbf{W}\mathbf{x}$. For example, for the query matrix $W = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$, the first row is the query for number of adults in the table (since $0\mathbf{x}[0] + 1\mathbf{x}[1] + 0\mathbf{x}[2] + 1\mathbf{x}[3]$ sums up over the tuples corresponding to adults); the second row is the query for number of Hispanic individuals; and the last row is the query for the total number of people. We summarize notation in Table 1.

Our privacy mechanisms are compatible with many variations of differential privacy, including concentrated differential privacy [6] and Renyi differential privacy [39]. As these are complex definitions, for simplicity we focus on approximate differential privacy [16, 17], defined as follows.

DEFINITION 1 ((ϵ, δ) -Differential Privacy [16]). *Given privacy parameters $\epsilon > 0$ and $\delta \in (0, 1)$, a randomized algorithm M satisfies (ϵ, δ) -differential privacy if for all pairs of neighboring dataset vectors \mathbf{x} and \mathbf{x}' and all sets S , the following equations hold:*

$$P(M(\mathbf{x}) \in S) \leq e^\epsilon P(M(\mathbf{x}') \in S) + \delta$$

Intuitively, differential privacy guarantees that the output distribution of a randomized algorithm is barely affected by any person’s record being included in the data.

In the case of privacy-preserving linear queries, this version of differential privacy is commonly achieved by adding independent Gaussian noise to the query answers $\mathbf{W}\mathbf{x}$. The scale of the noise depends on the L_2 sensitivity of the queries.

DEFINITION 2 (L_2 sensitivity). *The L_2 sensitivity of a function f , denoted by $\Delta_2(f)$ is defined as $\sup_{\mathbf{x} \sim \mathbf{x}'} \|f(\mathbf{x}) - f(\mathbf{x}')\|_2$.*

If f computes linear query answers (i.e., $f(\mathbf{x}) = \mathbf{W}\mathbf{x}$) then we slightly abuse notation and denote the L_2 sensitivity as $\Delta_2(W)$. It follows [50] that $\Delta_2(W)$ is equal to $\max_i \|W[:, i]\|_2$, where $W[:, i]$ is the i^{th} column of W .

The Gaussian Mechanism adds independent noise with variance σ^2 to $f(\mathbf{x})$ and releases the resulting noisy query answers. Its differential privacy properties are provided by the following theorem.

THEOREM 1 (Exact Gaussian Mechanism [3]). *Let Φ be the cumulative distribution function of the standard Gaussian distribution. Let f be a (vector-valued) function. Let M be the algorithm that, on input \mathbf{x} , outputs $f(\mathbf{x}) + \mathbf{z}$, where \mathbf{z} is an m -dimensional vector of independent $N(0, \sigma^2)$ random variables. Then M satisfies (ϵ, δ) -differential privacy if and only if*

$$\delta \geq \Phi\left(\frac{\Delta_2(f)}{2\sigma} - \frac{\epsilon\sigma}{\Delta_2(f)}\right) - e^\epsilon \Phi\left(-\frac{\Delta_2(f)}{2\sigma} - \frac{\epsilon\sigma}{\Delta_2(f)}\right)$$

Furthermore, this is an increasing function of $\Delta_2(f)/\sigma$.

The importance of this result is that the privacy properties of the Gaussian mechanism are completely determined by $\Delta_2(f)/\sigma$ (in fact, this is true for (ϵ, δ) -differential privacy, Renyi differential privacy and concentrated differential privacy). In particular, decreasing $\Delta_2(f)/\sigma$ increases the amount of privacy [3].

Personalized differential privacy [19] refines differential privacy by letting each domain element have its own privacy parameter.

For example, a domain element r has privacy parameters (ϵ, δ) if $P(M(\mathbf{x}) \in S) \leq e^\epsilon P(M(\mathbf{x}') \in S) + \delta$ for all S and neighbors \mathbf{x} and \mathbf{x}' that differ by 1 in the count of record r .

3 THE FITNESS-FOR-USE PROBLEM

In this section, we start with an intuitive problem statement and then mathematically formalize it, while justifying our design choices along the way. We consider two problems: one that prioritizes accuracy and a second that prioritizes privacy. Later we will show that they are equivalent (the same method yields a solution to both).

When **prioritizing accuracy**, the workload consists of m linear queries $\mathbf{w}_1, \dots, \mathbf{w}_m$. Given a vector \mathbf{c} of accuracy targets, we seek to find a mechanism M that produces (ϵ, δ) -differentially private answers to these queries such that the expected squared error for each query \mathbf{w}_i is less than or equal to $c[i]$. The mechanism should maximize privacy subject to these accuracy constraints. Maximizing privacy turns out to be a surprisingly subtle and nontrivial concept involving (1) the minimization of the privacy parameters ϵ and δ and (2) enabling the data publisher to noisily answer extra queries at a later point in time at no additional privacy cost; it is discussed in detail in Section 3.2. When **prioritizing privacy**, again we have a workload of m linear queries $\mathbf{w}_1, \dots, \mathbf{w}_m$ and a vector \mathbf{c} . Given target privacy parameters ϵ and δ , the goal is to find a mechanism M that (1) satisfies (ϵ, δ) -differential privacy and (2) finds the smallest number $k > 0$ such that each query \mathbf{w}_i can be privately answered with accuracy at most $kc[i]$. Thus $c[i]$ represents the relative importance of query \mathbf{w}_i and the goal is to obtain the most accurate query answers while respecting the privacy constraints and relative accuracy preferences.

We next discuss additional accuracy-related desiderata (Section 3.1), formalize the maximization of privacy (Section 3.2), and then fully formalize the fitness-for-use problem (Section 3.3).

3.1 Additional Accuracy Desiderata

We first require that the differentially private mechanism M produces *unbiased* noisy query answers (i.e., the expected value of the noisy query answers equals the true value). Unbiased query answers allow end-users to compute the expected error of derived queries. This property is best explained with an example. Suppose a mechanism M provides noisy counts (with independent noise) for (1) \tilde{y}_1 : the number of adults with cancer, and (2) \tilde{y}_2 : the number of children with cancer. From these two queries, we can derive an estimate for the total number of cancer patients as $\tilde{y}_1 + \tilde{y}_2$. Suppose \tilde{y}_1 has expected squared error c_1 and \tilde{y}_2 has expected squared error c_2 . What can we say about the expected squared error of $\tilde{y}_1 + \tilde{y}_2$? If both \tilde{y}_1 and \tilde{y}_2 are unbiased, then the expected squared error of $\tilde{y}_1 + \tilde{y}_2$ equals $c_1 + c_2$. However, if \tilde{y}_1 and \tilde{y}_2 are biased, then the expected squared error of $\tilde{y}_1 + \tilde{y}_2$ cannot be accurately determined from the expected errors of \tilde{y}_1 and \tilde{y}_2 .

Since statistical end-users need to be able to estimate the errors of quantities they derive from noisy workload query answers, we thus require our mechanism to produce unbiased query answers.

We next require that the noise added to query answers should be correlated multivariate Gaussian noise $N(\mathbf{0}, \Sigma)$. Gaussian noise is familiar to statisticians and simplifies their calculations for tasks such as performing hypothesis tests and computing confidence

intervals [45]. Furthermore, Gaussian noise is often a preferred distribution for various versions of approximate differential privacy [1, 6, 18, 39].

For readers familiar with the matrix mechanism [34, 50, 51] (which optimizes total squared error, not per-query error), it is important to note that we add correlated noise in a different way. The matrix mechanism starts with a workload of linear queries \mathbf{W} and solves an optimization problem to get a different collection \mathbf{S} of linear queries, called the *strategy matrix*. Noisy strategy answers $\tilde{\mathbf{y}}_s$ are obtained by adding independent noise \mathbf{z} to the strategy queries ($\tilde{\mathbf{y}}_s = \mathbf{S}\mathbf{x} + \mathbf{z}$) and the workload query answers $\tilde{\mathbf{y}}_w$ are reconstructed as follows $\tilde{\mathbf{y}}_w = \mathbf{W}\mathbf{S}^+\tilde{\mathbf{y}}_s$ (where \mathbf{S}^+ is the Moore-Penrose pseudo-inverse of \mathbf{S} [25]). Instead of optimizing for a matrix \mathbf{S} and adding independent Gaussian noise, we fix the matrix and optimize the correlation structure of the noise (as in [40]). This turns out to be a more convenient formulation for our problem. Formally,²

DEFINITION 3 (Linear Query Mechanism [20]). *Let \mathbf{W} be a linear query workload matrix. Let \mathbf{B} and \mathbf{L} be matrices such that $\mathbf{W} = \mathbf{L}\mathbf{B}$ and \mathbf{B} has linearly independent rows with the same rank as \mathbf{W} (we call \mathbf{B} the basis matrix and \mathbf{L} the representation matrix). Let $N(\mathbf{0}, \Sigma)$ be the multivariate Gaussian distribution with covariance matrix Σ . Then the mechanism M , on input \mathbf{x} , outputs $\mathbf{L}(\mathbf{B}\mathbf{x} + N(\mathbf{0}, \Sigma))$.*

Note that $E[\mathbf{L}(\mathbf{B}\mathbf{x} + N(\mathbf{0}, \Sigma))] = \mathbf{L}\mathbf{B}\mathbf{x} = \mathbf{W}\mathbf{x}$ so the mechanism is indeed unbiased. The basis matrix \mathbf{B} is chosen to be any convenient matrix after the workload is known (but before the dataset is seen). It is there for computational convenience – a linear query mechanism can be represented using any basis matrix we want, as shown in Theorem 2. Hence we can choose the basis \mathbf{B} (and corresponding \mathbf{L}) for which we can speed up matrix operations in our optimization algorithms. In our work, we typically set \mathbf{B} to be the identity matrix or a linearly independent subset of the workload matrix.

THEOREM 2. *Given a workload \mathbf{W} and two possible decompositions: $\mathbf{W} = \mathbf{L}_1\mathbf{B}_1$ and $\mathbf{W} = \mathbf{L}_2\mathbf{B}_2$, where the rows of \mathbf{B}_1 are linearly independent with the same rank as \mathbf{W} (and same for \mathbf{B}_2). Let $M_1(\mathbf{x}) = \mathbf{L}_1(\mathbf{B}_1\mathbf{x} + N(\mathbf{0}, \Sigma_1))$. There exists a Σ_2 such that the mechanism $M_2(\mathbf{x}) = \mathbf{L}_2(\mathbf{B}_2\mathbf{x} + N(\mathbf{0}, \Sigma_2))$ has the same output distribution as M_1 for all \mathbf{x} .*

3.2 Maximizing Privacy

In this section we explain how to compare the privacy properties of the basic mechanisms from Definition 3, based on their privacy parameters and additional noisy query answers they can release for free. We first generalize Theorem 1 to allow correlated noise.

THEOREM 3 (Exact Correlated Gaussian Mechanism). *Let f be a function that returns a vector of query answers. Let M be the mechanism that, on input \mathbf{x} , outputs $f(\mathbf{x}) + N(\mathbf{0}, \Sigma)$. Define the quantity*

$$\Delta_\Sigma(f) = \max_{\mathbf{x}_1 \sim \mathbf{x}_2} \sqrt{|(f(\mathbf{x}_1) - f(\mathbf{x}_2))^T \Sigma^{-1} (f(\mathbf{x}_1) - f(\mathbf{x}_2))|} \quad (1)$$

where the max is over all pairs of neighboring datasets \mathbf{x}_1 and \mathbf{x}_2 . Then M satisfies (ϵ, δ) -differential privacy if and only if:

$$\delta \geq \Phi \left(\frac{\Delta_\Sigma(f)}{2} - \frac{\epsilon}{\Delta_\Sigma(f)} \right) - e^\epsilon \Phi \left(-\frac{\Delta_\Sigma(f)}{2} - \frac{\epsilon}{\Delta_\Sigma(f)} \right) \quad (2)$$

²In [20], this is called the factorization mechanism. For their metric, they propose setting $\Sigma = \mathbf{I}$ while optimizing \mathbf{B} and \mathbf{L} . In our work, it is easier to optimize Σ instead.

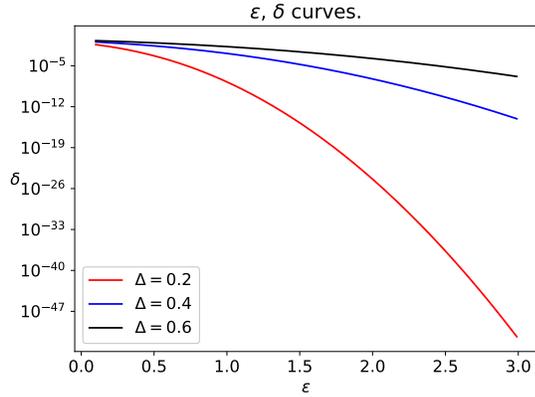


Figure 1: ϵ, δ curves for different values of Δ . A Gaussian Mechanism with $\Delta = 0.2$ satisfies (ϵ, δ) -differential privacy for all pairs (ϵ, δ) that lie on or above the red curve.

where Φ is the CDF of the standard Gaussian $N(0, 1)$. Furthermore, this quantity is an increasing function of $\Delta_{\Sigma}(f)$.

Applying Theorem 3 to the linear query mechanisms, we get:

COROLLARY 1. Let M be a linear query mechanism (Definition 3) with basis matrix \mathbf{B} and Gaussian covariance Σ . Let $\mathbf{b}_1, \dots, \mathbf{b}_d$ be the columns of \mathbf{B} and denote the privacy cost $\Delta_M = \max_{i=1, \dots, d} \sqrt{\mathbf{b}_i^T \Sigma^{-1} \mathbf{b}_i}$. Then M satisfies (ϵ, δ) -differential privacy if and only if $\delta \geq \Phi\left(\frac{\Delta_M}{2} - \frac{\epsilon}{\Delta_M}\right) - e^{\epsilon} \Phi\left(-\frac{\Delta_M}{2} - \frac{\epsilon}{\Delta_M}\right)$. Furthermore, this quantity is increasing in Δ_M .

Any given mechanism M typically satisfies (ϵ, δ) -differential privacy for infinitely many ϵ, δ pairs, which defines a curve in space [39] (see Figure 1). The importance of Corollary 1 is that this curve for a linear query mechanism M is completely determined by the single number Δ_M (as defined in Corollary 1). Furthermore, for any two linear query mechanisms $M_1(\mathbf{x}) = \mathbf{L}_1(\mathbf{B}_1 \mathbf{x} + N(\mathbf{0}, \Sigma_1))$ and $M_2(\mathbf{x}) = \mathbf{L}_2(\mathbf{B}_2 \mathbf{x} + N(\mathbf{0}, \Sigma_2))$, if $\Delta_{M_1} < \Delta_{M_2}$ then the (ϵ, δ) curve for M_1 is strictly below the curve for M_2 . This means that the set of pairs (ϵ, δ) for which M_1 satisfies differential privacy is a strict superset of the pairs for which M_2 satisfies differential privacy (while if $\Delta_{M_1} = \Delta_{M_2}$ then the (ϵ, δ) curves are exactly the same). For this reason, we call Δ_M the **privacy cost**.³

Thus one goal for maximizing privacy is to choose mechanisms M with as small Δ_M as possible as this will result in the mechanism satisfying differential privacy with the smallest choices ϵ and δ parameters.

3.2.1 Query answers for Free. Even for two mechanisms that satisfy differential privacy with exactly the same ϵ, δ parameters, it is still possible to say that one provides more privacy than the other by comparing them in terms of personalized differential privacy [19]. To see why, consider the matrices $\mathbf{B}_1 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$ and $\mathbf{B}_2 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$, and the two mechanisms $M_1(\mathbf{x}) = \mathbf{B}_1 \mathbf{x} + N(\mathbf{0}, \Sigma_1)$ and $M_2 = \mathbf{B}_2 \mathbf{x} + N(\mathbf{0}, \Sigma_2)$, where Σ_1 and Σ_2 are identity matrices. Note

³Readers familiar with ρ -zCDP [6] should note that privacy cost is the same as $\sqrt{2\rho}$.

that the difference between M_1 and M_2 is that M_2 answers the same queries as M_1 plus an additional query (corresponding to the third row of \mathbf{B}_2). However, from Corollary 1, $\Delta_{M_1} = \Delta_{M_2}$ which means that they satisfy differential privacy for exactly the same privacy parameters. But, in terms of personalized differential privacy, the personalized privacy cost of domain element $x[i]$ under mechanism M_1 is the square root of the i^{th} diagonal element of $\mathbf{B}_1^T \Sigma_1^{-1} \mathbf{B}_1$ (and similarly for M_2), while the overall non-personalized privacy cost is the max of these (as in Corollary 1). The personalized privacy costs of $x[0]$ and $x[2]$ under M_1 are smaller than under M_2 , while the privacy cost of $x[1]$ is the same under M_1 and M_2 . Thus it makes sense to say that M_2 is also less private.⁴ At the same time, we can say that M_1 provides more flexibility to the data publisher than M_2 . There are many possible choices of additional rows (queries) to add to \mathbf{B}_1 without affecting Δ_{M_1} . In fact, these queries can be determined (and noisily answered) at a later date after noisy answers to the first two queries in \mathbf{B}_1 are released.

This discussion leads to the concept of a privacy profile (which captures the personalized privacy costs of the domain elements) and refined privacy ordering for linear query mechanisms.

DEFINITION 4 (Privacy Profile). Given a Linear Query Mechanism $M(\mathbf{x}) = \mathbf{L}(\mathbf{B}\mathbf{x} + N(\mathbf{0}, \Sigma))$, the privacy profile of M is the d -dimensional vector $[\mathbf{b}_1^T \Sigma^{-1} \mathbf{b}_1, \dots, \mathbf{b}_d^T \Sigma^{-1} \mathbf{b}_d]$ (where \mathbf{b}_i is the i^{th} column of \mathbf{B}) and is denoted by $\text{prof}(\Sigma, \mathbf{B})$ or by $\text{prof}(M)$ (a slight abuse of notation).

For example, the privacy profile of M_1 above is $[1, 2, 1]$ while the profile for M_2 is $[2, 2, 2]$. Note that for any linear query mechanism M , Δ_M^2 is equal to the largest entry in the privacy profile of M .

The privacy profile is invariant to the choice of basis matrix \mathbf{B} as shown in Theorem 4. Thus the privacy profile is an intrinsic privacy property of a linear query mechanism rather than a specific parameterization.

THEOREM 4. Let $M_1(\mathbf{x}) = \mathbf{L}_1(\mathbf{B}_1 \mathbf{x} + N(\mathbf{0}, \Sigma_1))$ and $M_2(\mathbf{x}) = \mathbf{L}_2(\mathbf{B}_2 \mathbf{x} + N(\mathbf{0}, \Sigma_2))$ be two mechanisms that have the same output distribution for each \mathbf{x} (i.e., they add noise to different basis matrices but achieve the same results). Then $\text{prof}(M_1) = \text{prof}(M_2)$.

Given this invariance, we can use the privacy profile to define a more refined ordering that compares the privacy properties of linear query mechanisms.

DEFINITION 5 (Refined Privacy Ordering). Given two Linear Query Mechanisms M_1 (with basis \mathbf{B}_1 and covariance matrix Σ_1) and M_2 (with basis \mathbf{B}_2 and covariance matrix Σ_2), let p_1 be the privacy profile of M_1 when sorted in decreasing order and let p_2 be the sorted privacy profile of M_2 . We say that M_1 is at least as private as M_2 , denoted by $M_1 \leq_R M_2$ if p_1 is less than or equal to p_2 according to the dictionary (lexicographic) order. We also denote this as $(\Sigma_1, \mathbf{B}_1) \leq_R (\Sigma_2, \mathbf{B}_2)$. If the dictionary ordering inequality is strict, we use the notation $<_R$.

Note that if M has sorted privacy profile p , then the first element of p is equal to Δ_M^2 . Therefore if $\Delta_{M_1} < \Delta_{M_2}$ then also $M_1 <_R M_2$, but if $\Delta_{M_1} = \Delta_{M_2}$, it is still possible that $M_1 <_R M$ (hence $<_R$

⁴Note that with uncorrelated noise, the personalized cost of domain element $x[i]$ degenerates to the L_2 norm of the i^{th} column of the basis matrix. If different columns had different norms, prior work on the matrix mechanism added additional queries until the norms were the same (e.g., using M_2 instead of M_1 in our example above). We believe this practice should be re-examined – if M_1 satisfies the utility goals, why force the analyst to commit to an extra query and limiting their future options?

refines our privacy comparisons of mechanisms). Thus, our goal is to satisfy fitness-for-use constraints while finding the M that is minimal according to \leq_R . This is equivalent to finding an M that minimizes Δ_M (a continuous optimization) and then, among all the mechanisms that fit this criteria, we want to select a mechanism that is minimal according to \leq_R (a combinatorial optimization because of the sorting performed in the privacy profile).

We note that \leq is a weak ordering – that is, we can have two distinct mechanisms M_1 and M_2 such that $M_1 \leq_R M_2$ and $M_2 \leq_R M_1$ (i.e., they are two different mechanisms with the same privacy properties). Nevertheless, we will show in Section 5 that in fact, there exists a unique solution to the fitness-for-use problem.

3.3 The Formal Fitness-for-use Problem

Given a workload $\mathbf{W} = \mathbf{L}\mathbf{B}$ and a linear query mechanism $M(\mathbf{x}) = \mathbf{L}(\mathbf{B}\mathbf{x} + N(\mathbf{0}, \Sigma)) = \mathbf{W}\mathbf{x} + \mathbf{L}N(\mathbf{0}, \Sigma)$, the variance of the noisy answer to the i^{th} query of the workload is the i^{th} diagonal element of $\mathbf{L}\Sigma\mathbf{L}^T$. Thus, we can formalize the “prioritizing accuracy” and “prioritizing privacy” problems as follows.

PROBLEM 1 (Prioritizing Accuracy). *Let \mathbf{W} be an $m \times d$ workload matrix representing m linear queries. Let $\mathbf{c} = [c_1, \dots, c_m]$ be accuracy constraints (such that the noisy answer to the i^{th} query is required to be unbiased with variance at most c_i). Let \mathbf{B} and \mathbf{L} be matrices such that $\mathbf{W} = \mathbf{L}\mathbf{B}$ and \mathbf{B} has linearly independent rows. Select the covariance Σ of the Gaussian noise by solving the following constrained optimization problem:*

$$\begin{aligned} \alpha, \Sigma \leftarrow \arg \min_{\alpha, \Sigma} \alpha & \quad (3) \\ \text{s.t. } \mathbf{b}_i^T \Sigma^{-1} \mathbf{b}_i \leq \alpha & \text{ for all } i = 1, \dots, d \\ \text{and } (\text{diag}(\mathbf{L}\Sigma\mathbf{L}^T))[j] \leq c_j & \text{ for } j = 1, \dots, m \\ \text{and } \Sigma \text{ is symmetric positive definite.} & \end{aligned}$$

In case of multiple solutions to Equation 3, choose a Σ that is minimal under the privacy ordering \leq_R . Then release the output of the mechanism $M(\mathbf{x}) = \mathbf{L}(\mathbf{B}\mathbf{x} + N(\mathbf{0}, \Sigma))$.

Note that in Problem 1, α is going to equal the maximum of $\max_{i=1, \dots, d} \mathbf{b}_i^T \Sigma^{-1} \mathbf{b}_i$ and hence will equal the squared privacy cost, Δ_M^2 , for the resulting mechanism (and hence minimizing it will minimize the ϵ, δ privacy parameters).

We can also prioritize privacy, following the intuition at the beginning of Section 3, as finding the smallest k such that the variance of query i is at most kc_i given a target privacy level.

PROBLEM 2 (Prioritizing Privacy). *Under the same setting as Problem 1, given a target value α^* for Δ_M^2 (which uniquely defines the ϵ, δ curve), solve the following optimization for Σ :*

$$\begin{aligned} k, \Sigma \leftarrow \arg \min_{k, \Sigma} k & \quad (4) \\ \text{s.t. } \mathbf{b}_i^T \Sigma^{-1} \mathbf{b}_i \leq \alpha^* & \text{ for all } i = 1, \dots, d \\ \text{and } (\text{diag}(\mathbf{L}\Sigma\mathbf{L}^T))[j] \leq kc_j & \text{ for } j = 1, \dots, m \\ \text{and } \Sigma \text{ is symmetric positive definite.} & \end{aligned}$$

In case of multiple solutions to Equation 4, choose a Σ that is minimal under the refined privacy ordering \leq_R .

It is interesting to note that we arrived at Problem 2 by trying to find a mechanism that minimizes per-query error, which mathematically is the same as minimizing $\|E[(\mathbf{W}\mathbf{x} - M(\mathbf{x}))^2 ./ \mathbf{c}]\|_\infty$ (where $./$ is pointwise division of vectors). Edmonds et al. [20] and Nikolov [40] studied mechanisms that minimize outlier error (also known as joint error): $E[\|\mathbf{W}\mathbf{x} - M(\mathbf{x})\|^2 ./ \mathbf{c}]$. Their nearly-optimal algorithm for this metric can also be obtained by solving Equation 4 (which computes an ellipsoid infinity norm [40]) but they did not study the tie-breaking condition in Problem 2.

The following result shows that a solution to Problem 1 can be converted into a solution for Problem 2, so for the rest of this paper, we focus on solving the optimization defined in Problem 1.

THEOREM 5. *Let Σ be a solution to Problem 1. Define the quantities $\alpha = \max_{i=1, \dots, d} \mathbf{b}_i^T \Sigma^{-1} \mathbf{b}_i$ and $\Sigma_2 = \frac{\alpha}{\alpha^2} \Sigma$. Then Σ_2 is a solution to Problem 2.*

Our mechanism in Section 6 approximates Problem 1 with an optimization problem that has fewer constraints and avoids discrete optimization when breaking ties.

4 RELATED WORK

Developing differentially private algorithms under accuracy constraints is an underdeveloped area of research. Wu et al. [46] considered the problem of setting privacy parameters to achieve the desired level of accuracy in a certain machine learning task like logistic regression. Our work focuses on linear queries and has multiple (not just one) accuracy constraints. We consider this to be the difference between fitness-for-use (optimizing to support multiple applications within a desired accuracy level) vs. accuracy constrained differential privacy (optimizing for a single overall measure of quality).

The Matrix Mechanism [34, 37, 50] answers linear queries while trying to minimize the sum squared error of the queries (rather than per-query fitness for use error). Instead of answering the workload matrix directly, it solves an optimization problem to find a different set of queries, called the strategy matrix. It adds independent noise to query answers obtained from the strategy matrix and then recovers the answer to workload queries from the noisy strategy queries. The major challenge of Matrix Mechanism is that the optimization problem is hard to solve. Multiple mechanisms [33, 37, 51, 52] have been proposed to reformulate or approximate the Matrix Mechanism optimization problem. For pure differential privacy (i.e., $\delta = 0$), the solution is often sub-optimal because of non-convexity. Yuan et al. [50] propose a convex problem formulation for (ϵ, δ) -differential privacy and provided the first known optimal instance of the matrix mechanism. Although their work solves the total error minimization problem, the strategy may fail to satisfy accuracy constraints for every query. Prior to the matrix mechanism, the search for query strategies was done by hand, often using hierarchical/lattice-based queries (e.g., [14, 29, 43, 47]) and later by selecting an optimal strategy from a limited set of choices (e.g., [33, 43, 49]). The Matrix-Variate Gaussian (MVG) Mechanism [12] is an extension in a different direction that is used to answer matrix-valued queries, such as computing the covariance matrix of the data. It does not perform optimization to decide how to best answer a workload query.

Work related to the factorization mechanism [20, 40], is most closely related to ours. Starting from a different error metric, they arrived at a similar optimization to Problem 2. They focus on theoretical optimality properties, while we focus on special-purpose algorithms for the optimization problems, hence approximately optimizing our fitness-for-use and their joint error criteria.

The Laplace and Gaussian mechanisms [17, 18] are the most common building blocks for differential privacy algorithms, adding noise from the Laplace or Gaussian noise to provide $(\epsilon, 0)$ -differential privacy and (ϵ, δ) -differential privacy, respectively. Other distributions are also possible (e.g., [11, 22–24, 35]) and their usage depends on application requirements (specific privacy definition and measure of error).

Our work and the matrix/factorization mechanism work are examples of data-independent mechanisms – the queries and noise structure does not depend on the data. There are many other works that focus on data-dependent mechanisms [13, 27, 30–32, 42], where the queries receiving the noise depend on the data. These mechanisms reserve some privacy budget for estimating properties of the data that help choose which queries to ask. For example, the DAWA Algorithm [32] first privately learns a partitioning of the domain into buckets that suits the input data well and then privately estimates counts for each bucket. While these algorithm may perform well on certain dataset, they can often be outperformed on other datasets by data-independent mechanisms [28]. A significant disadvantage of data-dependent algorithms is that they cannot provide closed-form error estimates for query answers (in many cases, they cannot provide any accurate error estimates). Furthermore, data-dependent methods also often produce *biased* query answers, which can be undesirable for subsequent analysis, as discussed in Section 3.1.

5 THEORETICAL ANALYSIS

In this section, we theoretically analyze the solution to Problem 1. We prove uniqueness results for the solution and derive results that simplify the algorithm construction (for Section 6). We first show that optimizing per-query error targets is a fundamentally different problem than optimizing for total squared error, as was done in prior work (e.g., [34, 43, 50, 51]). Our results show that there are natural problems where algorithms that optimize for sum of squared errors can have maximum per-query errors that are $O(\sqrt{d})$ times larger than optimal (d is the domain size).

5.1 Analytical Case Study

Suppose the dataset is represented by an d -dimensional vector $\mathbf{x} = [x_1, \dots, x_d]^T$. Let the workload matrix consists of identity queries (i.e., for each i , what is the value of x_i) and the total sum query. Its matrix representation is:

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}$$

We compare *closed-form* solutions for sum-squared-error and fitness-for-use optimizations for the case where all per-query variance targets are set to γ . Solutions to both problems can be interpreted as adding *correlated* noise to \mathbf{x} as follows: $\mathbf{z} = \mathbf{x} + N(\mathbf{0}, \Sigma)$ and then releasing \mathbf{Wz} . By convexity of the sum-squared-error [50] and fitness-for-use (Section 5.2) problems, and since all $x[i]$ are treated symmetrically by \mathbf{W} , the covariance matrices for both problems should treat the domain elements symmetrically. That is, the correlation between the noise added to $x[i]$ and $x[j]$ (i.e., $\Sigma[i, j]$) should be the same as the correlation between the noise added to $x[i']$ and $x[j']$. Thus Σ (and consequently Σ^{-1}) should have the form:

$$\Sigma = a\mathbf{I} + b\mathbf{1}\mathbf{1}^T \quad \Sigma^{-1} = \frac{1}{a}\mathbf{I} - \frac{b}{a^2 + dab}\mathbf{1}\mathbf{1}^T$$

where a and b are scalars and $\mathbf{1}$ is the column vector of ones. The eigenvalues are $a + bd$ (with eigenvector $\mathbf{1}$) and a (for all vectors orthogonal to the vector $\mathbf{1}$). Hence positive definiteness requires $a > 0$ and $a + bd > 0$. The variance for each of the first d queries is $a + b$. For the sum query it is $\mathbf{1}^T \Sigma \mathbf{1} = da + d^2 b$. The squared privacy cost is $\Sigma_{i,i}^{-1} = \frac{a+(d-1)b}{a^2+dab}$.

5.1.1 Sum-squared-error Optimization. Thus, the optimization problem for sum squared error given a privacy cost β is:

$$\begin{aligned} \arg \min_{a,b} \quad & d(a+b) + (da + d^2 b) \\ \text{s.t.} \quad & \frac{a+(d-1)b}{a^2+dab} = \beta^2 \text{ and } a > 0 \text{ and } a+bd > 0 \end{aligned} \quad (5)$$

THEOREM 6. For $d \geq 5$, the solution to Equation 5 is:

$$\begin{aligned} a &= \frac{-3+d}{(-1+d) - \sqrt{1+d}} \frac{1}{\beta^2} \\ b &= \frac{(-3+d)(2 - \sqrt{1+d})}{\left((-1+d) - \sqrt{1+d}\right) \left(-1-d + (-1+d)\sqrt{1+d}\right)} \frac{1}{\beta^2} \end{aligned}$$

5.1.2 Fitness-for-use Optimization. Fitness-for-use optimization with target variance γ for all queries can be written as:

$$\begin{aligned} \arg \min_{a,b} \quad & \frac{a+(d-1)b}{a^2+dab} \\ \text{s.t.} \quad & a+b \leq \gamma \text{ and } ad+bd^2 \leq \gamma \text{ and } a > 0 \text{ and } a+bd > 0 \end{aligned} \quad (6)$$

THEOREM 7. When $d \geq 5$, the solution to Equation 6 is $a = \left(\frac{d+1}{d}\right)\gamma$, $b = -\left(\frac{1}{d}\right)\gamma$, with squared privacy cost $\frac{2d}{1+d} \frac{1}{\gamma}$.

5.1.3 Comparison. To compare the two mechanisms, we can make their privacy costs equal by setting $\beta^2 = \frac{2d}{1+d} \frac{1}{\gamma}$.

THEOREM 8. When the two mechanisms have the same privacy cost, the maximum ratio of query variance to its desired variance bound γ is $O(\sqrt{d})$ for sum-squared-error and 1 for fitness-for-use.

This result shows that sum-squared-error optimization and fitness-for-use optimization produce very different solutions, hence fitness-for-use optimization is needed for applications that demand per-query accuracy constraints.

5.2 Properties of the Problem and Solution

We next theoretically analyze the problem and its solution space. The first result is convexity of the optimization problem.

THEOREM 9. *The optimization problem in Equation 3 is convex.*

Some convex optimization problems have no solutions because they are not *closed*. As a simple example, consider $\arg \min x^3 + x$ s.t., $x > 0$. The value $x = 0$ is ruled out by the constraint and no positive value of x can be optimal (i.e. dividing a candidate solution by 2 always improves the objective function). A similar concern holds for positive definite matrices – the set of positive definite matrices is not closed, but the set of positive semi-definite matrices is closed. The next result shows that even if we allowed Σ to be positive semi-definite (hence guaranteeing an optimal solution exists), optimal solutions will still be positive definite.

THEOREM 10. *If the fitness-for-use variance targets c_i are all positive then optimization problem in Equation 3 (Problem 1) is feasible and all optimal solutions for Σ have smallest eigenvalue $\geq \chi$ for some fixed $\chi > 0$ (i.e., they are symmetric positive definite).*

Problem 1 asks us to solve the optimization problem in Equation 3 and if there are multiple optimal solutions, pick one such Σ^\dagger that is minimal under the refined privacy ordering $<_R$. In principle, since this is a weak ordering (i.e., two distinct mechanisms can have the same exact privacy properties), one can expect there to exist many such minimal solutions with equivalent privacy properties.

Surprisingly, it turns out that there is a unique solution. The main idea of the proof is to first show that all solutions to Equation 3 that are minimal under \leq_R have the same exact privacy profile (the refined privacy ordering guarantees that the *sorted* privacy profiles are the same, but we show that for solutions of Equation 3, the entire privacy profiles are exactly the same). This places at least k constraints on the covariance matrix Σ for such solutions. Note that Σ has $k(k+1)/2$ parameters in all, so, in general, two matrices with the same privacy profile do not have to be identical. However, our proof then shows that if two matrices have the same privacy profile *and* are solutions to Equation 3, then they must in fact be the same matrix.

THEOREM 11. *In Problem 1, there exists a unique minimal (under the privacy ordering \leq_R) solution.*

6 ALGORITHMS

In this section, we present an algorithm to solve the fitness-for-use problem (Problem 1). Since a major component of Problem 1 includes a constrained convex optimization problem, a reasonable choice is to implement an interior point method [41]. However, given the number of constraints involved, we found that such an algorithm would require a significant amount of complexity to ensure scalability, numerical stability, and a feasible initialization for the algorithm to start at. Instead, we closely approximate it with a series of unconstrained optimization problems that are much easier to implement and initialize. For low-dimensional problems, both the interior point method and our approximation returned nearly identical results.

In Section 6.1, we present our reformulation/approximation of the problem. Then, in Section 6.2, we provide a Newton-based algorithm with line search for solving it. We use a trick from the matrix

mechanism optimization [34] to avoid computing a large Hessian matrix. Then in Section 6.3 we present initialization strategies that help speed up convergence.

6.1 Problem Reformulation

Two of the main features of Problem 1 are optimizing the max of the privacy cost while dealing with the variance constraints. Our first step towards converting this to an unconstrained optimization problem is to bring the variance constraints into the objective function. We do this by noting that our variance constraints can be rephrased as a max constraint:

$$\forall j, \mathbf{v}_j \Sigma \mathbf{v}_j^T \leq c_j \iff \forall j, \frac{\mathbf{v}_j \Sigma \mathbf{v}_j^T}{c_j} \leq 1 \iff \max_j \frac{\mathbf{v}_j \Sigma \mathbf{v}_j^T}{c_j} \leq 1$$

where \mathbf{v}_j is the j^{th} row of the matrix \mathbf{L} (recall $\mathbf{W} = \mathbf{L}\mathbf{B}$) and $\mathbf{c} = [c_1, \dots, c_m]$ are the desired variance upper bounds. This observation raises the following question: instead of optimizing privacy cost subject to variance constraints, what if we optimize privacy cost *plus* max variance in the objective function (and then rescale Σ so that the accuracy constraints are met)? This results in the following problem, which we show is equivalent to Problem 1.

PROBLEM 3. *Under the same setting as Problem 1, let \mathbf{v}_j be the j^{th} row of the representation matrix \mathbf{L} . Solve the following optimization:*

$$\Sigma \leftarrow \arg \min_{\Sigma} \max_i (\mathbf{b}_i^T \Sigma^{-1} \mathbf{b}_i) + \max_j \left(\frac{\mathbf{v}_j \Sigma \mathbf{v}_j^T}{c_j} \right) \quad (7)$$

s.t. Σ is symmetric positive definite.

Among all optimal solutions, choose a Σ^ that is minimal according to the privacy ordering $<_R$, and then output Σ^* / γ , where $\gamma = \max_j \frac{\mathbf{v}_j \Sigma^* \mathbf{v}_j^T}{c_j}$.*

THEOREM 12. *The optimal solution to Problem 1 is the same as the optimal solution to Problem 3.*

Next, we can use a common trick [51] for continuous approximation of the max function, known as the *soft max*: $\text{sm}_t(a_1, \dots, a_m) = \frac{1}{t} \log(\sum_i e^{t a_i})$, where t is a smoothing parameter. As $t \rightarrow \infty$, the soft max converges to the max function. We can apply this trick to both maxes in Problem 3. Now, the nice thing about the soft max function is that it also approximates the refined privacy ordering. Specifically, if $\Sigma_1 <_R \Sigma_2$, then clearly when t is large enough,

$$\frac{1}{t} \log\left(\sum_i \exp(t * \mathbf{b}_i^T \Sigma_1^{-1} \mathbf{b}_i)\right) < \frac{1}{t} \log\left(\sum_i \exp(t * \mathbf{b}_i^T \Sigma_2^{-1} \mathbf{b}_i)\right)$$

Plugging the soft max function in place of the max in Problem 3, we finally arrive at our approximation problem:

PROBLEM 4. *Given parameters t_1 and t_2 , under the same setting as Problem 1, solve the following optimization problem:*

$$\Sigma^* \leftarrow \arg \min_{\Sigma} \frac{1}{t_1} \log\left(\sum_i \exp(t_1 * \mathbf{b}_i^T \Sigma^{-1} \mathbf{b}_i)\right) + \frac{1}{t_2} \log\left(\sum_j t_2 * \frac{\mathbf{v}_j \Sigma \mathbf{v}_j^T}{c_j}\right) \quad (8)$$

s.t. Σ is symmetric positive definite.

and then output Σ^*/γ , where $\gamma = \max_j \frac{\mathbf{v}_j \Sigma^* \mathbf{v}_j^T}{c_j}$.

Our algorithm solves Equation 8 as a sequence of optimization problems that gradually increase t_1 and t_2 .

THEOREM 13. *The optimization Problem 4 is convex.*

Although we focus on controlling per-query error, we note that some data publishers may wish to strike a balance between achieving the per-query error targets and minimizing total squared error. This is easy to achieve by adding a $+\lambda \sum_j \mathbf{v}_j \Sigma \mathbf{v}_j^T$ term to the objective function of Problem 4, where λ is a weight indicating relative importance of total squared error. This modification changes the objective function gradient by $+\lambda \sum_j \mathbf{v}_j \mathbf{v}_j^T$. This modification to the gradient computation is the only change needed by our algorithm (in Section 6.2) to handle this hybrid setting.

6.2 The Optimization Algorithm

We use Newton's Method with Conjugate Gradient approximation to the Hessian to solve Problem 4 (Algorithm 1). For a given value of t_1 and t_2 , we use an iterative algorithm to solve the optimization problem in Equation 8 to convergence to get an intermediate value Z_{t_1, t_2} . Then, starting from Z_{t_1, t_2} , we again iteratively solve Equation 8 but with a larger value of t_1 and t_2 . We repeat this process until convergence. Each sub-problem uses Conjugate Gradient [41] to find an update direction for Σ without materializing the large Hessian matrix. We then use backtracking line search to find a step size for this direction that ensures that the updated Σ improves over the previous iteration and is still positive definite.

Algorithm 1: Optimize(B, L, c)

Input: Basis matrix \mathbf{B} , reconstruction matrix \mathbf{L} , accuracy constraints \mathbf{c} ;

Output: Solution Σ ;

```

1 Initialize  $\Sigma = \Sigma_0$ ,  $t_1 = 1$ ,  $t_2 = 1$ ;
2 for  $iter = 1$  to  $MAXITER$  do
3    $\mathbf{s} = \text{ConjugateGradient}(\Sigma, \mathbf{B}, \mathbf{L}, \mathbf{c})$ ;
4    $\delta = \langle \mathbf{s}, \nabla F(\Sigma) \rangle$ ;
   // Stopping Criteria
5   if  $|\delta| < NTTOL$  then
6      $gap = (d + m)/t_1$ ;
7     if  $gap < TOL$  then
8        $\text{break}$ ;
9      $t_1 = MU * t_1$ ,  $t_2 = MU * t_2$ ;
10   $\alpha = \text{LineSearch}(\Sigma, \mathbf{v})$ ;
11   $\Sigma = \Sigma + \alpha * \mathbf{s}$ ;
12 Return  $\Sigma$ 

```

In Algorithm 1, MAXITER is the maximum number of iterations to use, while NTTOL, TOL are tolerance parameters. Larger tolerance values make the program stop faster at a slightly less accurate solution. MU is the factor we use to rescale t_1 and t_2 . Typical values we used are $MU \in \{2, 5, 10\}$. In Line 3, we get an approximate Newton direction and then we use Line 4 to see if this direction can provide sufficient decrease (here ∇F is the gradient of the objective

function). If not, the sub-problem is over and we update t_1 and t_2 (Line 9) to continue with the next sub-problem. Otherwise, we find a good step size for the search direction (Line 10) and then update Σ (Line 11).

6.2.1 Gradient and Hessian Computation. The gradient and Hessian of the objective function F can be derived in closed form. Let $g_i(\Sigma) = \mathbf{b}_i^T \Sigma^{-1} \mathbf{b}_i$, $G_i(t_1, \Sigma) = \exp\left(t_1 * \mathbf{b}_i^T \Sigma^{-1} \mathbf{b}_i\right)$, $h_j(\Sigma) = \frac{\mathbf{v}_j \Sigma \mathbf{v}_j^T}{c_j}$, $H_j(t_2, \Sigma) = \exp\left(t_2 * \frac{\mathbf{v}_j \Sigma \mathbf{v}_j^T}{c_j}\right)$. Noting that the matrix Σ is symmetric, we can calculate the gradient and Hessian of the functions f_i and g_i as follows:

$$\nabla g_i(\Sigma) = -\Sigma^{-1} \mathbf{b}_i \mathbf{b}_i^T \Sigma^{-1} \quad (9)$$

$$\nabla^2 g_i(\Sigma) = -\nabla g_i(\Sigma) \otimes \Sigma^{-1} - \Sigma^{-1} \otimes \nabla g_i(\Sigma) \quad (10)$$

$$\nabla h_j(\Sigma) = \frac{\mathbf{v}_j^T \mathbf{v}_j}{c_j} \quad (11)$$

$$\nabla^2 h_j(\Sigma) = \mathbf{0} \quad (12)$$

Then the gradient and Hessian of the objective function $F(\Sigma)$ can be calculated as follows.

$$\nabla F(\Sigma) = \frac{\sum_i (G_i(t_1, \Sigma) \nabla g_i(\Sigma))}{\sum_i G_i(t_1, \Sigma)} + \frac{\sum_j (H_j(t_2, \Sigma) \nabla h_j(\Sigma))}{\sum_j H_j(t_2, \Sigma)} \quad (13)$$

$$\begin{aligned} \nabla^2 F(\Sigma) = & \frac{t_1 \sum_i (G_i(t_1, \Sigma) (\nabla^2 g_i(\Sigma) + \nabla g_i(\Sigma) \otimes \nabla g_i(\Sigma)))}{\sum_i G_i(t_1, \Sigma)} \\ & - \frac{t_1 \sum_i (G_i(t_1, \Sigma) \nabla g_i(\Sigma)) \otimes \sum_i (G_i(t_1, \Sigma) \nabla g_i(\Sigma))}{(\sum_i G_i(t_1, \Sigma))^2} \\ & + \frac{t_2 \sum_j (H_j(t_2, \Sigma) (\nabla h_j(\Sigma) \otimes \nabla h_j(\Sigma)))}{\sum_j H_j(t_2, \Sigma)} \\ & - \frac{t_2 \sum_j (H_j(t_2, \Sigma) \nabla h_j(\Sigma)) \otimes \sum_j (H_j(t_2, \Sigma) \nabla h_j(\Sigma))}{(\sum_j H_j(t_2, \Sigma))^2} \end{aligned} \quad (14)$$

Here \otimes is the Kronecker product. Because of the Kronecker product, multiplication of the Hessian by a search direction can be done without materializing the Hessian itself. Specifically, we use the well-known property $(\mathbf{A} \otimes \mathbf{B}) \text{vec}(\mathbf{C}) = \text{vec}(\mathbf{BCA}^T)$ that is frequently exploited in the matrix mechanism literature [37, 52]. We let HessTimesVec denote the function that exploits this trick to efficiently compute the multiplication $\mathbf{H}\mathbf{p}$ of the Hessian (of objective function F) by a search direction without explicitly computing \mathbf{H} .

6.2.2 Conjugate Gradient. Algorithm 1 finds a search direction using the Conjugate Gradient algorithm, which is commonly used for large scale optimization [41]. The main idea is that Newton's method uses second-order Taylor expansion to approximate $F(\Sigma)$ and would like to compute the search direction by solving

$$\mathbf{s} = \arg \min_{\mathbf{s}} \frac{1}{2} \mathbf{s}^T \nabla^2 F(\Sigma) \mathbf{s} + \mathbf{s}^T \nabla F(\Sigma) \quad (15)$$

whose solution is $-\mathbf{H}^T \nabla F$, where $\mathbf{H} = \nabla^2 F$ is the Hessian. However, the size of Hessian matrix is $d^2 \times d^2$ (where d is the size of the domain of possible tuples in our case). This is intractably large, but fortunately, only approximate solutions to Equation 15 are

Algorithm 2: ConjugateGradient($\Sigma, \mathbf{B}, \mathbf{L}, \mathbf{c}$)

Input: Variable Σ , basis matrix \mathbf{B} , index matrix \mathbf{L} , accuracy constraints \mathbf{c} ;

Output: Search direction \mathbf{s} ;

```
1 Initialize  $\mathbf{s} = 0, \mathbf{r} = -\nabla F(\Sigma), \mathbf{p} = \mathbf{r}, rsold = \langle \mathbf{r}, \mathbf{r} \rangle$  ;
2 for  $i = 1$  to MAXCG do
3    $\mathbf{Hp} = \text{HessTimesVec}(\mathbf{p})$  ;
4    $a = \frac{rsold}{\mathbf{p}^T * \mathbf{Hp}}, \mathbf{s} = \mathbf{s} + a * \mathbf{p}, \mathbf{r} = \mathbf{r} - a * \mathbf{Hp}, rsnew = \langle \mathbf{r}, \mathbf{r} \rangle$  ;
5   if  $rsnew \leq \text{TOL2}$  then
6     break;
7    $b = \frac{rsnew}{rsold}, \mathbf{p} = \mathbf{r} + b * \mathbf{p}, rsold = rsnew$ ;
8 Return  $\mathbf{s}$ 
```

necessary for optimization, and this is what conjugate gradient does [41]. Algorithm 2 provides the pseudocode for our application (recall that HessTimesVec is the function that efficiently computes the product of the Hessian times a vector by taking advantage of the Kronecker products in Equation 14). We note that Yuan et al. [52] used 5 conjugate gradient iterations in their matrix mechanism application. Similarly we use 5 iterations (MAXCG = 5) in Line 2. We also terminate the loop early if there is very little change, checked in Line 5 (we use TOL2 = 10^{-10}).

6.2.3 Step Size. Once the conjugate gradient algorithm returns a search direction, we need to find a step size α to use to update Σ in Algorithm 1. For this we use the standard backtracking line search [41], whose application to our problem is shown in Algorithm 3. It makes sure that the step size is small enough (but not too small) so that it will (1) result in a positive definite matrix and (2) the objective function will decrease sufficiently. We use Cholesky decomposition to check for positive definiteness. The parameter σ determines how much the objective function need to decrease before breaking the iteration, a typical setting is $\sigma = 0.01$.

Algorithm 3: LineSearch(Σ, \mathbf{s})

Input: Variable Σ , search direction \mathbf{s} ;

Output: Step size α ;

```
1  $fcurr = F(\Sigma), flast = fcurr, \Sigma_{old} = \Sigma, j = 1, \beta = 0.5$ ;
2 while true do
3    $\alpha = \beta^{j-1}, j = j + 1, \Sigma_{new} = \Sigma_{old} + \alpha * \mathbf{s},$   
    $fcurr = F(\Sigma_{new})$  ;
4   if  $\Sigma_{new} \leq 0$  then
5     continue;
6   if  $fcurr \leq flast + \alpha * \sigma * \langle \mathbf{s}, \nabla F(\Sigma) \rangle$  then
7     break;
8 Return  $\alpha$ 
```

6.3 Initialization

Optimization algorithms need a good initialization in order to converge reasonably well. However, specifying a correlation structure for Gaussian noise is typically not an intuitive approach for data

publishers. Instead, data publishers may feel more comfortable specifying a query matrix \mathbf{Q} (with linearly independent rows) to which it may be reasonable (as a first approximation) to add independent noise. That would result in an initial suboptimal mechanism $M_0(\mathbf{x}) = \mathbf{W}\mathbf{Q}^+(\mathbf{Q}\mathbf{x} + N(0, \sigma^2\mathbf{I}))$ [34], where \mathbf{Q}^+ is the Moore-Penrose pseudo-inverse of \mathbf{Q} [25] – that is, this suggested mechanism would add independent noise to $\mathbf{Q}\mathbf{x}$ and then recover workload query answers by multiplying the result by $\mathbf{W}\mathbf{Q}^+$.

We do not run this mechanism, instead we derive a Gaussian correlation Σ from it. The covariance matrix for that mechanism would be $\sigma^2\mathbf{W}\mathbf{Q}^+(\mathbf{W}\mathbf{Q}^+)^T$. Noting that $\mathbf{W} = \mathbf{L}\mathbf{B}$, this can be written as $\sigma^2\mathbf{L}\mathbf{B}\mathbf{Q}^+(\mathbf{L}\mathbf{B}\mathbf{Q}^+)^T$. This is equivalent to adding $N(0, \sigma^2\mathbf{Q}^+(\mathbf{Q}^+)^T)$ noise to $\mathbf{B}\mathbf{x}$ and hence we can set the initial Σ to be $\sigma^2\mathbf{Q}^+(\mathbf{Q}^+)^T$.

Next we need to choose a small enough value for σ^2 so that the initial covariance matrix $\Sigma = \sigma^2\mathbf{Q}^+(\mathbf{Q}^+)^T$ would satisfy the fitness-for-use constraints. We do this by choosing

$$\sigma^2 = \min_{j=1 \dots d} \frac{c_j}{j^{\text{th element of } \text{diag}(\mathbf{L}\mathbf{Q}^+(\mathbf{Q}^+)^T\mathbf{L}^T)}} * 0.99$$

Our optimizer (Algorithm 1) starts with this Σ and iteratively improves it.

We can also offer guidance about the setting for \mathbf{Q} in cases where the data publisher does not have a good guess. A simple strategy is to set $\mathbf{Q} = \mathbf{I}$; in fact, this is the setting we use for our experiments (to show that our algorithms can succeed without specialized knowledge about the problem). It is also possible to set \mathbf{Q} to be the result of Matrix Mechanism algorithms such as [52].

7 EXPERIMENTS

In these experiments, our proposed algorithm is referred to as SM-II (for the two applications of soft max). We compare against the following algorithms. Input Perturbation (IP) adds independent Gaussian noise directly to the data vector. The optimal Matrix Mechanism Convex Optimization Algorithm (CA) [50] minimizes sum-squared error under (ϵ, δ) -differential privacy. Since queries can have different relative importance, we also consider two variations of CA. wCA-I weights each query by the inverse of its target variance (so queries that need low variance have higher weight in the objective function) and wCA-II weights each query by the inverse of the square root of the variance. We also consider the Hierarchical Mechanism (HM) [29, 43] as it is good for some workloads. Specifically, HM uses a strategy matrix \mathbf{H} which represents a tree structure with optimal branching factor [43]. We also considered the Gaussian Mechanism (GM), which adds independent Gaussian noise to the workload query answers. However, it generally performed worse than IP and always significantly worse than CA, so we dropped it from the tables to save space. We do not compare with alternative Matrix Mechanism algorithms like Low-Rank Mechanism [51], Adaptive Mechanism [33] and Exponential Smoothing Mechanism [51] because the CA baseline is optimal for Matrix Mechanism [50] under (ϵ, δ) -differential privacy.

We compare these algorithms on a variety of workloads based on their performance at the same privacy cost. Our algorithm SM-II finds the minimal privacy cost needed to satisfy the accuracy constraints. We then set the other algorithms to use this privacy

Table 2: Privacy cost comparison between SM-II and interior point method. Note that IP did not scale well, thus limiting the size of d .

d	2	4	8	16	64
Interior Point	1.33	1.76	2.28	2.91	4.46
SM-II	1.33	1.76	2.28	2.91	4.46

cost and we check by how much they exceed the pre-specified variance requirements (i.e., to what degree are they sub-optimal for the fitness-for-use problem). Note that these algorithms are all data-independent, so the variance can be computed in closed form and will be the same for any pair of datasets that have the same dimensionality and domain size. One can also compare privacy cost needed to reach the target variance as follows. If, given the same privacy cost, Method 1 exceeds its target variance by a factor of x_1 and Method 2 exceeds it by x_2 (which is what our experiments show), then if they were both given enough privacy cost to meet their target variance goals, the ratio of privacy cost for Method 1 to that of Method 2 is $\sqrt{x_1/x_2}$ (this follows from Theorem 5).

We follow Section 6.3 to initialize our algorithm by setting the initialization parameter $\mathbf{Q} = \mathbf{I}$ (the $d \times d$ identity matrix). For basis matrices \mathbf{B} , we use two choices: $\mathbf{B} = \mathbf{I}$ or $\mathbf{B} = \mathbf{U}$, where \mathbf{U} is an upper triangle matrix ($\mathbf{U}[i, j] = 1$ if $i \leq j$ and $\mathbf{U}[i, j] = 0$ otherwise).

All experiments are performed on a machine with an Intel i7-9750H 2.60GHz CPU and 16GBBytes RAM.

7.1 Evaluating Design Choices

We made two key design choices for our algorithm. The first was to approximate the constrained optimization in Problem 1 with the soft max optimization in Problem 4. Then we used a Newton-style method to solve it instead of something simpler like gradient descent. We evaluate the effect of these choices here.

7.1.1 The Soft Max Approximation. For small problems, we can directly solve Problem 1 by implementing an interior point method [5] and so we can compare the resulting privacy cost with the one returned by our soft max approximation SM-II. For this experiment, we considered prefix-range queries over an ordered domain of size d . These are one-sided range queries (i.e., the sum of the first 2 records, the sum of the first 3 records) from which all one-dimensional range query answers can be computed by subtraction of two one-sided range queries. Thus the workload \mathbf{W} is a lower triangular matrix where the lower triangle consists of ones. The result is shown in Table 2 and shows excellent agreement. Note that $d = 64$ is the limit for our interior point solver, while with SM-II we can easily scale to $d = 1024$ on a relatively weak machine (using a server-grade machine or a GPU can improve scalability even more). We note that extreme scalability, like in the high-dimensional matrix mechanism [37], is an area of future work and we believe that a combination of the ideas from [37] and our work can make this possible, by breaking a large optimization problem into smaller pieces and using our SM-II method to optimize each smaller piece.

7.1.2 SM-II vs. Gradient Descent. As far as optimizers go, SM-II is relatively simple, but not as simple as the very popular gradient descent (GD). To justify the added complexity, we compared these

Table 3: Convergence time for SM-II vs. Gradient Descent (GD) in seconds. Even at small scales, gradient descent is vastly inferior.

d	2	4	8	16
GD	7.02	44.90	198.52	264.36
SM-II	0.027	0.055	0.053	0.058

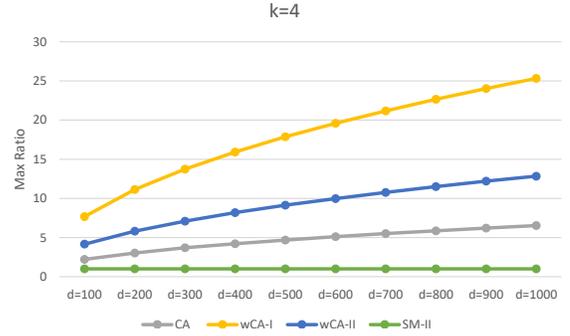


Figure 2: Comparison for different d when $k = 4$.

two optimizers for solving Problem 4. We used the same setup as in Section 7.1.1 and the results are shown in Table 3. Clearly the use of SM-II is justified as gradient descent does not converge fast enough even at small problem sizes.

7.2 The Identity-Sum Workload

In Section 5.1, we considered the workload consisting of the identity query and the sum query, with equal target variance. We further explore this example to illustrate further surprising behavior of algorithms that optimize for sum-squared-error (so for just this experiment we focus on CA, wCA-I, wCA-II). Specifically, it is believed that in sum-squared error optimization, weighting queries by the inverse of their desired variance helps improve their individual accuracy. We show that this is not always the case.

In these experiments, the identity query has target variance 1, but we will vary the sum query’s target variance, which we denote by the variable k .

In Figure 2, we set $k = 4$ and vary the domain size d . As expected, SM-II outperforms the others. What is surprising is that wCA-I and wCA-II perform worse than CA because this means that the typical recommended weighting strategy actually makes things worse (the sum query exceeds its target variance by a larger ratio). For reference, we derive the optimal analytical solutions for these methods on this workload in the appendix of the full version of our paper [48].

Figure 3 fixes d at 256 and varies k (starting from $k = 1$). Again we see the same qualitative effects as in Figure 2.

7.3 PL-94

PL94-171 [8] is a Census dataset used for redistricting. In 2010, it contained the following variables:

- **voting-age:** a binary variable where 0 indicates someone is 17 years old and under, while 1 indicates 18 or over.

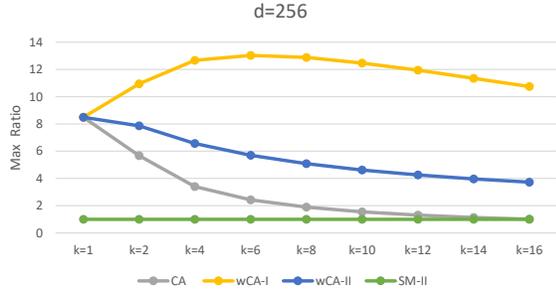


Figure 3: Comparison for different k when $d = 256$.

Table 4: Max variance ratio on PL94 with uniform targets.

Mechanism	IP	HM	CA	wCA-I	wCA-II	SM-II
PL94	36.56	13.93	3.99	3.99	3.99	1.00

- **ethnicity**: a binary variable used to indicate if someone is Hispanic (value 1) or not (value 0).
- **race**: this is a variable with 63 possible values. The OMB race categories in this file in 2010 were *White, Black or African American, American Indian and Alaska Native, Asian, Native Hawaiian and Other Pacific Islander, and Some Other Race*. An individual is able to select any non-empty subset of races, giving a total of 63 possible values.

These variables create a $(2, 2, 63)$ histogram which can be flattened into a 252-dimension vector.

For this experiment, we created a workload consisting of multiple marginals. These were a) voting-age marginal (i.e., number of voting-age people and number of non-voting-age people); b) ethnicity marginal; c) the number of people in each OMB category who selected only one race; d) for each combination of 2 or more races, the number of people who selected that combination; e) the identity query (i.e., for each demographic combination, how many people fit into it). As we are not aware of any public error targets, we will assume all queries are equally important and set all variance targets to 1. The basis matrix is $\mathbf{B} = \mathbf{U}$ (upper triangular matrix).

We found the minimal privacy cost needed to match these variance bounds using SM-II and set that as the privacy cost for all algorithms. Then we measured the maximum ratio of query variance to target variance for each method. The results are shown in Table 4. We see that input perturbation performs the worst. The matrix mechanism optimized for total error (CA) is much better, but it still has a maximum variance ratio of 3.99 times larger than optimal. This means that the matrix mechanism is more accurate than necessary for some queries at the expense of missing the target bound for other queries. Meanwhile, SM-II can match the desired bounds for each query, which is what it is designed to do. In terms of *sum squared error*, which is what CA is optimized for, the error for SM-II is 2.07 times that of CA.

7.4 Range Queries

We next consider one-dimensional range queries. Here we follow a similar setting to [51]. We vary d , the number of items in an

Table 5: Maximum ratio of achieved variance to target variance on Range Queries with uniform accuracy targets.

Mechanism	IP	HM	CA	wCA-I	wCA-II	SM-II
$d = 64$	10.63	2.84	1.27	1.27	1.27	1.00
$d = 128$	17.45	3.48	1.25	1.25	1.25	1.00
$d = 256$	28.43	3.90	1.29	1.29	1.29	1.00
$d = 512$	47.36	4.68	1.23	1.23	1.23	1.00
$d = 1024$	77.26	5.42	1.24	1.24	1.24	1.00

Table 6: Maximum ratio of achieved variance to target variance on Range Queries with random accuracy targets.

Mechanism	IP	HM	CA	wCA-I	wCA-II	SM-II
$d = 64$	16.04	4.63	2.53	1.62	1.98	1.00
$d = 128$	21.28	5.87	2.14	1.49	1.75	1.00
$d = 256$	41.76	6.05	1.91	1.45	1.69	1.00
$d = 512$	71.51	7.35	1.80	1.46	1.52	1.00
$d = 1024$	109.34	7.68	1.69	1.43	1.49	1.00

Table 7: Maximum achieved variance to target variance ratio on Random Queries with a Uniform target variance.

Mechanism	IP	HM	CA	wCA-I	wCA-II	SM-II
$d = 64$	1.14	2.29	1.14	1.14	1.14	1.00
$d = 128$	1.14	2.29	1.07	1.07	1.07	1.00
$d = 256$	1.14	3.39	1.06	1.06	1.06	1.00
$d = 512$	1.15	3.40	1.05	1.05	1.05	1.00
$d = 1024$	1.15	3.41	1.04	1.04	1.04	1.00

ordered domain. The workload is composed of $2 * d$ random range queries, where the end-points of each query are sampled uniformly at random. We compare two settings, (1) the uniform cases where the target variance bound is equal to 1 for each query, and (2) the random case where the target variance bound for each query is randomly sampled from a uniform(1, 10) distribution. In the experiments we vary the domain size $d \in \{64, 128, 256, 512, 1024\}$, and use the basis matrix $\mathbf{B} = \mathbf{U}$ (upper triangular matrix).

Again, we find the minimum privacy cost using SM-II, set each algorithm to use that privacy cost and then compute the maximum variance to target variance ratio for each method. The results for the uniform targets are shown in Table 5, with SM-II achieving $\approx 25\%$ improvement over the best competitor. The results for the random targets are shown in Table 6, with SM-II improving by at least 40% over competitors. In the uniform case, the *sum squared error* of SM-II ranges from 1.04 to 1.09 times that of CA, depending on the value of d .

7.5 Random Queries

We next consider a random query workload similar to the setting in [51]. Here for each entry in each d -dimensional query vector, we flip a coin with $P(\text{heads}) = 0.2$. If it lands heads, the entry is set to 1, otherwise it is set to -1 . Again we consider two scenarios. (1) the uniform cases where the target variance bound is equal to 1

Table 8: Maximum achieved variance to target variance ratio on Random Queries with a Random target variance.

Mechanism	IP	HM	CA	wCA-I	wCA-II	SM-II
d = 64	2.58	7.44	2.43	1.15	1.61	1.00
d = 128	2.82	8.28	2.58	1.18	1.72	1.00
d = 256	2.82	8.36	2.58	1.17	1.70	1.00
d = 512	2.79	8.27	2.51	1.16	1.68	1.00
d = 1024	2.76	8.17	2.49	1.16	1.67	1.00

Table 9: Maximum achieved variance to target variance ratio for Age pyramids with uniform accuracy targets.

Mechanism	IP	HM	CA	wCA-I	wCA-II	SM-II
AGE	32.49	6.2	1.58	1.58	1.58	1.00

for each query, and (2) the random case where the target variance bound for each query is randomly sampled from a uniform(1, 10) distribution. We vary $d \in \{64, 128, 256, 512, 1024\}$, and the number of queries is $m = 2 * d$. The basis matrix is $\mathbf{B} = \mathbf{I}$.

Table 7 lists the maximum variance ratio among all queries under the same privacy cost for different methods for the uniform scenario. Table 8 shows the corresponding results for random targets. In both cases, the best competitor is not much worse than SM-II. We speculate that this is because the queries do not have much structure that can be exploited to reduce privacy cost. In terms of *sum squared error*, for the case of uniform target variances, the total error of SM-II was at most 1.005 times that of CA.

7.6 Age Pyramids

Demographers often study the distribution of ages in a population by gender. We consider a dataset schema similar to the Census, where gender is a binary attribute and there are 116 ages (0-115). The range queries we consider are prefix queries (i.e., age $\in [0, x]$ for all x) and age $\in [18, 115]$ (the voting age population). The workload consists of (1) range queries for males, (2) range queries for females, (3) range queries for all. We set uniform target variance bounds of 1. We use the basis matrix $\mathbf{B} = \mathbf{U}$ (upper triangular). Table 9 shows the maximum ratio of achieved variance to target variance for each algorithm, with SM-II clearly outperforming the competitors by at least 58%. Meanwhile the sum squared error of SM-II is 1.07 times that of CA.

7.7 Marginals

We next consider histograms H on r variables, where each variable can take on one of t values, resulting in a domain size of $d = t^r$. We consider the workload consisting of all one-way and two-way marginals with uniform target variance of 1. For this set of experiments, we set $r = 3$ and varied the values of t . We used the basis matrix is $\mathbf{B} = \mathbf{I}$ for SM-II. Table 10 shows the maximum target variance ratio of different algorithms. SM-II outperforms the competitors, especially as t increases. The ratio of sum squared error of SM-II to CA ranged from 1.1 ($t = 2$) to 1.34 ($t = 16$).

Table 10: Maximum achieved variance to target variance ratio on Marginal Queries when $r = 3$

Mechanism	IP	HM	CA	wCA-I	wCA-II	SM-II
t = 2	1.82	3.02	1.14	1.14	1.14	1.00
t = 4	4.55	10.28	1.42	1.42	1.42	1.00
t = 6	8.6	21.38	1.66	1.66	1.66	1.00
t = 8	14.03	36.72	1.88	1.88	1.88	1.00
t = 10	20.84	56.23	2.08	2.08	2.08	1.00
t = 12	28.76	78.96	2.25	2.25	2.25	1.00
t = 14	38.17	106.22	2.41	2.41	2.41	1.00
t = 16	48.85	137.41	2.57	2.57	2.57	1.00

Table 11: Maximum achieved variance to target variance ratio on WRelated Queries with a Uniform target variance.

Mechanism	IP	HM	CA	wCA-I	wCA-II	SM-II
d = 64	3.51	9.59	1.31	1.31	1.31	1.00
d = 128	4.16	11.16	1.31	1.31	1.31	1.00
d = 256	4.28	12.60	1.19	1.19	1.19	1.00
d = 512	3.62	10.46	1.12	1.12	1.12	1.00
d = 1024	3.50	10.05	1.07	1.07	1.07	1.00

7.8 WRelated Queries

We next consider the one-dimensional WRelated workload of [51]. The workload matrix is $W = CA$. Here matrix C has size $m \times s$ and matrix A has size $s \times d$, each follows the Gaussian(0, 1) distribution. In the experiment we set $m = d/2, s = d/2$. Due to space constraints, we only show results for the uniform target variance bound of 1. We vary the domain size $d \in \{64, 128, 256, 512, 1024\}$, and use the basis matrix $\mathbf{B} = \mathbf{I}$. The results in Table 11 shows SM-II outperforming competitors by around 31% on the smaller domain sizes. The ratio of total squared error for SM-II to that of CA was at most 1.08

8 CONCLUSIONS AND FUTURE WORK

In this work we introduce the fitness-for-use problem, where the goal is to calculate minimal privacy cost under accuracy constraints for each query. After theoretical analysis of the problem, we proposed an algorithm named SM-II to solve it. While our algorithm used variance as the accuracy measure for each query, other applications may require their own specific accuracy measures. This consideration leads to two important directions for future work. The first is to create optimized differentially private algorithms that meet fitness-for-use goals under measures other than squared error. The second direction is to achieve the same kind of extreme scalability as the high-dimensional matrix mechanism [37] that provided a tradeoff between optimality and scalability.

ACKNOWLEDGMENTS

This work was supported by NSF Awards CNS-1702760 and CNS-1931686. We are grateful to Aleksandar Nikolov for discussions about factorization mechanisms.

REFERENCES

- [1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In CCS.
- [2] John Abowd. 2018. The U.S. Census Bureau Adopts Differential Privacy. KDD Invited Talk.
- [3] Borja Balle and Yu-Xiang Wang. 2018. Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising. In *International Conference on Machine Learning, ICML*.
- [4] Jeremiah Blocki, Anupam Datta, and Joseph Bonneau. 2016. Differentially Private Password Frequency Lists. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society.
- [5] Stephen Boyd, Stephen P Boyd, and Lieven Vandenbergh. 2004. *Convex optimization*. Cambridge university press.
- [6] Mark Bun and Thomas Steinke. 2016. Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds. In *Proceedings, Part I, of the 14th International Conference on Theory of Cryptography - Volume 9985*.
- [7] U.S. Census Bureau. [n.d.]. Decennial Census: 2010 Summary Files. https://www.census.gov/mp/www/cat/decennial_census_2010/.
- [8] U.S. Census Bureau. 2010. PL 94-171 Redistricting Data. <https://www.census.gov/programs-surveys/decennial-census/about/rdo/summary-files.html>.
- [9] U.S. Census Bureau. 2011. Advance Group Quarters Summary File. <https://www.census.gov/prod/cen2010/doc/gqsf.pdf>.
- [10] U. S. Census Bureau. [n.d.]. On The Map: Longitudinal Employer-Household Dynamics. https://lehd.ces.census.gov/applications/help/onthemap.html#!confidentiality_protection.
- [11] Clement L Canonne, Gautam Kamath, and Thomas Steinke. 2020. The Discrete Gaussian for Differential Privacy. In *NeurIPS*.
- [12] Thee Chanyaswad, Alex Dytso, H Vincent Poor, and Prateek Mittal. 2018. Myg mechanism: Differential privacy under matrix-valued query. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 230–246.
- [13] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. 2012. Differentially private spatial decompositions. In *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 20–31.
- [14] Bolin Ding, Marianne Winslett, Jiawei Han, and Zhenhui Li. 2011. Differentially Private Data Cubes: Optimizing Noise Sources and Consistency. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*.
- [15] Irit Dinur and Kobbi Nissim. 2003. Revealing information while preserving privacy. In *PODS*.
- [16] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy via Distributed Noise Generation. In *EUROCRYPT*. 486–503.
- [17] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*.
- [18] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3–4 (2014), 211–407. <https://doi.org/10.1561/04000000042>
- [19] Hamid Ebad, David Sands, and Gerardo Schneider. 2015. Differential Privacy: Now It's Getting Personal. In *POPL*.
- [20] Alexander Edmonds, Aleksandar Nikolov, and Jonathan Ullman. 2020. The power of factorization mechanisms in local and central differential privacy. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. 425–438.
- [21] Simson L. Garfinkel, John M. Abowd, and Sarah Powazek. 2018. Issues Encountered Deploying Differential Privacy. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*.
- [22] Quan Geng, Wei Ding, Ruiqi Guo, and Sanjiv Kumar. 2020. Tight analysis of privacy and utility tradeoff in approximate differential privacy. In *International Conference on Artificial Intelligence and Statistics*. 89–99.
- [23] Quan Geng and Pramod Viswanath. 2015. The optimal noise-adding mechanism in differential privacy. *IEEE Transactions on Information Theory* 62, 2 (2015), 925–951.
- [24] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. 2009. Universally Utility-Maximizing Privacy Mechanisms. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*.
- [25] Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations* (third ed.). The Johns Hopkins University Press.
- [26] Robert Groves. 2010. So, How do You Handle Prisons? <https://www.census.gov/newsroom/blogs/director/2010/03/so-how-do-you-handle-prisons.html>.
- [27] Moritz Hardt, Katrina Ligett, and Frank Mcsherry. 2012. A Simple and Practical Algorithm for Differentially Private Data Release. In *Advances in Neural Information Processing Systems*.
- [28] Michael Hay, Ashwin Machanavajjhala, Gerome Miklau, Yan Chen, Dan Zhang, and George Bissias. 2016. Exploring Privacy-Accuracy Tradeoffs Using DPComp. In *Proceedings of the 2016 International Conference on Management of Data*.
- [29] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2010. Boosting the Accuracy of Differentially Private Histograms Through Consistency. *Proceedings of the VLDB Endowment* 3, 1 (2010).
- [30] Georgios Kellaris and Stavros Papadopoulos. 2013. Practical differential privacy via grouping and smoothing. *Proceedings of the VLDB Endowment* 6, 5 (2013), 301–312.
- [31] Ios Kotsogiannis, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2017. Pythia: Data dependent differentially private algorithm selection. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1323–1337.
- [32] Chao Li, Michael Hay, Gerome Miklau, and Yue Wang. 2014. A Data-and Workload-Aware Algorithm for Range Queries Under Differential Privacy. *Proceedings of the VLDB Endowment* 7, 5 (2014).
- [33] Chao Li and Gerome Miklau. 2012. An Adaptive Mechanism for Accurate Query Answering under Differential Privacy. *Proceedings of the VLDB Endowment* 5, 6 (2012).
- [34] Chao Li, Gerome Miklau, Michael Hay, Andrew Mcgregor, and Vibhor Rastogi. 2015. The Matrix Mechanism: Optimizing Linear Counting Queries under Differential Privacy. *The VLDB Journal* 24, 6 (Dec. 2015), 757–781. <https://doi.org/10.1007/s00778-015-0398-x>
- [35] Fang Liu. 2018. Generalized gaussian mechanism for differential privacy. *IEEE Transactions on Knowledge and Data Engineering* 31, 4 (2018), 747–756.
- [36] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. 2008. Privacy: From Theory to Practice On the Map. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 277–286.
- [37] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2018. Optimizing error of high-dimensional statistical queries under differential privacy. *Proceedings of the VLDB Endowment* 11, 10 (2018).
- [38] Solomon Messing, Christina DeGregorio, Bennett Hillenbrand, Gary King, Saurav Mahanti, Zagreb Mukerjee, Chaya Nayak, Nate Persily, Bogdan State, and Arjun Wilkins. 2020. Facebook Privacy-Protected Full URLs Data Set. <https://doi.org/10.7910/DVN/TDOAPG>
- [39] Ilya Mironov. 2017. Rényi Differential Privacy. In *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*. 263–275.
- [40] Aleksandar Nikolov. 2014. *New computational aspects of discrepancy theory*. Ph.D. Dissertation. Rutgers University-Graduate School-New Brunswick.
- [41] Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization* (second ed.). Springer, New York, NY, USA.
- [42] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Differentially private grids for geospatial data. In *2013 IEEE 29th international conference on data engineering (ICDE)*. IEEE, 757–768.
- [43] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding Hierarchical Methods for Differentially Private Histograms. *Proc. VLDB Endow.* 6, 14 (2013).
- [44] U.S. Census Bureau. 2020. (Chapter 7) Understanding and Using American Community Survey Data: What All Data Users Need to Know. <https://www.census.gov/programs-surveys/acs/guidance/handbooks/general.html>.
- [45] Yue Wang, Daniel Kifer, and Jaewoo Lee. 2019. Differentially Private Confidence Intervals for Empirical Risk Minimization. *Journal of Privacy and Confidentiality* (2019).
- [46] S. Wu, A. Roth, K. Ligett, B. Waggoner, and S. Neel. 2019. Accuracy First: Selecting a Differential Privacy Level for Accuracy-Constrained ERM. 9, 2 (2019).
- [47] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2011. Differential Privacy via Wavelet Transforms. *IEEE Transactions on Knowledge and Data Engineering* 23, 8 (2011), 1200–1214.
- [48] Yingtai Xiao, Zeyu Ding, Yuxin Wang, Danfeng Zhang, and Daniel Kifer. 2020. Optimizing Fitness-for-Use of Differentially Private Linear Queries (full version). <https://arxiv.org/abs/2012.00135>.
- [49] Grigory Yaroslavtsev, Graham Cormode, Cecilia M Procopiuc, and Divesh Srivastava. 2013. Accurate and efficient private release of datacubes and contingency tables. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 745–756.
- [50] Ganzhao Yuan, Yin Yang, Zhenjie Zhang, and Zhifeng Hao. 2016. Convex Optimization for Linear Query Processing under Approximate Differential Privacy. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- [51] Ganzhao Yuan, Zhenjie Zhang, Marianne Winslett, Xiaokui Xiao, Yin Yang, and Zhifeng Hao. 2012. Low-Rank Mechanism: Optimizing Batch Queries under Differential Privacy. *Proc. VLDB Endow.* 5, 11 (July 2012), 1352–1363. <https://doi.org/10.14778/2350229.2350252>
- [52] Ganzhao Yuan, Zhenjie Zhang, Marianne Winslett, Xiaokui Xiao, Yin Yang, and Zhifeng Hao. 2015. Optimizing batch linear queries under exact and approximate differential privacy. *ACM Transactions on Database Systems (TODS)* 40, 2 (2015), 1–47.