9-2013

Optimizing Linear Queries Under Differential Privacy

Chao Li
University of Massachusetts Amherst, cornemuse@gmail.com

OPTIMIZING LINEAR QUERIES UNDER DIFFERENTIAL PRIVACY

A Dissertation Presented

by

CHAO LI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September, 2013

Computer Science

OPTIMIZING LINEAR QUERIES UNDER DIFFERENTIAL PRIVACY

A Dissertation Presented

by

CHAO LI

Approved as to style and content by:

Gerome Miklau, Chair

Andrew McGregor, Member

Don Towsley, Member

Michael Lavine, Member

Lori Clarke, Department Chair
Computer Science

ABSTRACT

OPTIMIZING LINEAR QUERIES UNDER DIFFERENTIAL PRIVACY

SEPTEMBER, 2013

CHAO LI

B.Sc., PEKING UNIVERSITY

M.Math, UNIVERSITY OF WATERLOO

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Gerome Miklau

Private data analysis on statistical data has been addressed by many recent literatures. The goal of such analysis is to measure statistical properties of a database without revealing information of individuals who participate in the database. Differential privacy is a rigorous privacy definition that protects individual information using output perturbation: a differentially private algorithm produces statistically indistinguishable outputs no matter whether the database contains a tuple corresponding to an individual or not.

It is straightforward to construct differentially private algorithms for many common tasks and there are published algorithms to support various tasks under differential privacy. However methods to design error-optimal algorithms for most non-trivial tasks are still unknown. In particular, we are interested in error-optimal algorithms for sets of linear queries. A linear query is a sum of counts of tuples that satisfy a certain condition, which covers the scope of many aggregation tasks including count, sum and histogram. We present the matrix mechanism, a novel mechanism for answering sets of linear queries under differential privacy. The matrix mechanism makes a clear distinction between a set of queries submitted by users, called the *query workload*, and an alternative set of queries to be answered under differential privacy, called the *query strategy*. The answer to the query workload can then be computed using the answer to the query strategy. Given a query workload, the query strategy determines the distribution of the output noise and the power of the matrix mechanism comes from adaptively choosing a query strategy that minimizes the output noise.

Our analyses also provide a theoretical measure to the quality of different strategies for a given workload. This measure is then used in accurate and approximate formulations to the optimization problem that outputs the error-optimal strategy. We present a lower bound of error to answer each workload under the matrix mechanism. The bound reveals that the hardness of a query workload is related to the spectral properties of the workload when it is represented in matrix form. In addition, we design an approximate algorithm, which generates strategies generated by our a out perform state-of-art mechanisms over (ϵ, δ)-differential privacy. Those strategies lead to more accurate data analysis while preserving a rigorous privacy guarantee. Moreover, we also combine the matrix mechanism with a novel data-dependent algorithm, which achieves differential privacy by adding noise that is adapted to the input data and to the given query workload.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Private data analysis on statistical databases

Statistical data is widely collected and analyzed in various fields such as statistics, computer science, economics, psychology and so on. Many statistical databases involve sensitive personal information which should not be revealed during data analyzing and publishing. The naive approach, which simply removes identifiers from the database, can not protect personal information and is vulnerable when partial information is publicly available [10, 55].

Sophisticated private query answering techniques have been developed to reduce potential privacy breaches. Though there are different privacy models that define the behavior of adversaries and basic privacy requirements, those techniques can be categorized into three groups [58]: local perturbation, in which the information has been modified before being submitted to the statistical database; data publishing, in which a synthetic database that is based on the original database is published; output perturbation, in which the query answers are modified before returned to users.

Local perturbation has been studied in [7, 28]. The problem of local perturbation is that there is no fine grained control on privacy and the noise from different individuals accumulates when the analysis relates to multiple individuals.

One famous data publishing approach in privacy data analysis is k-anonymity introduced by Sweeney et al. [63]. K-anonymity is a syntactically private mechanism that has been widely accepted by data publishers and analyzers. In k-anonymity, the attributes of a table are separated into two groups: quasi identifiers and sensitive attributes. The idea of k-anonymity is to group the values of quasi identifiers so that each group of quasi identifiers is associated with at least k different tuples. Several papers discussed and improved the theory of k-anonymity [50, 48, 66, 68]. However, there are still three major drawbacks in k-anonymity. First, in practice, there is no clear distinction between quasi identifiers and sensitive attributes. For example, the address attribute can be a quasi identifier in a database of patients but may be sensitive in a location tracking database. In addition, with k-anonymity, it is difficult to prevent an adversary from indicating whether a user participates in the database. Such information sometimes is highly sensitive (e.g. a database of certain diseases). Lastly, there is actually no theoretical guarantee on the effect of k-anonymity and later literatures [48, 50, 31] demonstrated several attacks under which k-anonymity and its variations are vulnerable.

Dwork et al. [25, 26] introduced differential privacy, which is a rigorous privacy definition that protects individual information using output perturbation: a differentially private algorithm produces statistically indistinguishable outputs no matter whether the database contains a tuple corresponding to an individual or not. Furthermore, differential privacy makes no assumption on the prior knowledge of adversaries and provides privacy guarantee even if the adversary knows all but one tuple in the table. It is straightforward to construct differentially private algorithms for many common tasks and there are published algorithms to support various tasks under differential privacy, as summarized in [22, 23, 24]. Systems that answer queries under differential privacy have also been designed, such as PINQ[53], Airavat[61] and GUPT[54]. However methods to design error-optimal algorithms for most non-trivial tasks are still unknown. In many cases, the optimal error to answer a set of queries with a certain privacy guarantee under differential privacy can be greatly impacted by the choices of the mechanisms to answer those queries.

1.2 Answering linear queries under differential privacy

One of the most widely studied categories of queries under differential privacy is linear queries. A linear query is a sum of counts of tuples that satisfy a certain condition, which covers the scope of many aggregation tasks including count, sum and histogram. To answer one single linear counting query under differential privacy, the Laplace mechanism has been proved [32] to be the mechanism that introduces the least amount of noise. However, the best mechanism that answers multiple linear counting queries simultaneously is remain unknown. Many have pointed out that using the Laplace mechanism to answer each query in a set independently introduces more noise than it is needed in many scenarios. It hence calls for more sophisticated mechanisms to answer set of linear counting queries under differential privacy with low noise.

Recently, a number of related approaches have been proposed which improve on the Laplace mechanism, sometimes allowing for low error where only unacceptably high error was possible before. They each embody a basic (but perhaps counter-intuitive) principle: better results are possible when you *don't ask for what you want.*

The earliest example of this approach focuses on workloads consisting of sets of k-way marginals, for which Barak et al. answer a set of Fourier basis queries using the Laplace mechanism, and then derive the desired marginals [9]. For workloads consisting of all range-count queries over an ordered domain, two approaches have been proposed. Xiao et al. [70] first answer a set of wavelet basis queries, while Hay et al. [40] use a hierarchical set of counting queries which recursively decompose the domain. For workloads consisting of sets of marginals, Ding et al. [19] propose a method for selecting an alternative set of marginals, from which the desired counts can be derived. However, those approaches only support one type of query sets and can not be generalized to arbitrary set of linear counting queries.

All approaches mentioned above, as well as the Laplace mechanism, share one important property: the queries they answer only depend on the domain properties and the input queries, and are independent of the concrete tuples in the database. Those approaches are hence called data-independent approaches. Meanwhile, another research trend in answering linear counting queries under differential privacy is data-aware approaches, which take the underlying database into consideration. Early works either synthesizes a database [14] or maintain samples of possible databases [60], which may not be applicable in practice. More practical algorithms are emerging most recently [38, 73, 6, 17, 72, 67, 37]. Compared with other data-independent works related with the matrix mechanism, the amount of noise introduced by those data-aware algorithms either largely depend on the underlying database: they can sometimes do much better than data-independent works while do much worse in other cases. or do not take significant advantage of the properties of underlying databases. Furthermore, many of those works just aim to generate a database and the input query sets are largely ignored.

1.3 Contributions

We present the matrix mechanism, a novel mechanism for answering sets of linear queries under differential privacy. Our mechanism works as an improvement to any differentially privacy mechanism. In general, the matrix mechanism builds an alternative set of queries and uses the answer to the alternative query set to derive the answer to the input query set. The matrix mechanism makes a clear distinction between a set of queries submitted by users, called the *query workload*, and an alternative set of queries to be answered under differential privacy, called the *query strategy*. The answer to the query workload can then be computed using the answer to the query strategy. The power of the mechanism then yields to the flexibility in the choice of query strategies that leads to low noisy answers to different query workloads. The matrix mechanism covers those approaches that improve the Laplace mechanism [9, 70, 40, 19], as well as mechanisms that work on other differentially private mechanisms [39]. To avoid the limitation of previous approaches, we present an efficient algorithm that creates a truly adaptive solution to answer any set of linear counting queries with low error, relieving the user of the burden of choosing among mechanisms or carefully analyzing their query workloads. The strategies generated by our approximate algorithm out perform state-of-art mechanisms [9, 70, 40, 19] over (ϵ, δ)-differential privacy.

In addition, we provide a thorough error analysis under the matrix mechanism, and formulate the noise of the matrix mechanism in a closed form. The analytic formula of noise leads to a much easier comparison among many algorithms: it is not necessary to run repeated experiments on concrete databases. It can also be used in accurate and approximate formulations to the optimization problem that outputs the error-optimal strategy. Furthermore, with the analyses in the matrix mechanism, we also characterize the "hardness" of a query workload by lower bounding the minimum noise of the workload under the matrix mechanism. Our bound is tight or almost tight on many commonly interested sets of queries and serves as a more comprehensive measure on the "hardness" of a query workload than the basic differential privacy measurement on the query workload [25, 26] or the information theoretical measurement of it [14].

To take the advantage of both the input query set and the underlying database, we further design a mechanism by combining the matrix mechanism with a novel data-dependent algorithm. To our knowledge, our mechanism is the first data-dependent mechanism that provides significant improvement on databases with easy-to-exploit properties yet does not break-down on databases with complex distributions.

CHAPTER 2
BACKGROUND

The content of this chapter serves as the foundation of all our discussions in the remaining chapters of this dissertation. In this chapter, we first formally define the concept of linear queries and workloads we are working on, as well as the vector or matrix representation of the query or query set. An introduction to differential privacy is also included in the chapter, consisting of basic definitions and mechanisms in differential privacy. We also cover the linear algebra fundamentals at the end of this chapter.

2.1 Background: linear queries and query workloads

The matrix mechanism is designed to answer a set of linear queries. A linear query is an aggregation query over a single relation that can be expressed as a linear combination of a set of database counts . In this section, we first describe the representation of a relational table as a vector of counts. We then describe linear queries, represented as a vector of coefficients, and a workload of linear queries, represented as a matrix. Lastly, we show that the matrix representation of a set of linear queries is not unique.

2.1.1 Data domain and cell lists

We consider a database instance I of a single-table relational schema $R(\mathbb{A})$ with attributes $\mathbb{A} = \{A_1, A_2, \ldots, A_m\}$. The domain $dom(A_i)$ of an attribute A_i may be discrete or continuous, finite or infinite, ordered or unordered. The set of all tuples that may exist in I is the cross-product of the domains of attributes in $\mathbb{A}$: $dom(\mathbb{A}) = dom(A_1) \times dom(A_2) \times \cdots \times dom(A_m)$. The database instance is encoded as a vector of cell counts, each counting the number of tuples included in a distinct subset of the domain.

Definition 2.1 (Cell and Cell List)**.** *A cell is a non-empty subset of* $dom(\mathbb{A})$*. A cell list* $\Phi = \phi_1, \phi_2 \dots \phi_n$ *is an ordered list of mutually-exclusive cells:* $\forall i, j \quad \phi_i \cap \phi_j = \varnothing$*.*

We do not require that the cells in a cell list cover $dom(\mathbb{A})$. For a specified cell list, a relational table can be represented in the form of a *data vector* consisting of a non-negative integer for each cell.

Definition 2.2 (Data vector)**.** *Given instance* I *and cell list* $\Phi = \phi_1, \phi_2 \dots \phi_n$*, the vector representation of* I *using* Φ*, denoted* $\mathbf{x}(I, \Phi)$*, is the length-n column vector consisting of a non-negative integer for each cell, i.e the* i^{th} *entry in* $\mathbf{x}(I, \Phi)$ *is* $|I \cap \phi_i|$*.*

When I and Φ are clear from the context, we denote the data vector simply by $\mathbf{x}$.

Example 2.1. *Consider a relational schema* $R = (name, gradyear, gender, gpa)$ *describing students. Fig. 2.1(a) shows a sample instance of this relation. Fig. 2.1(b) shows a cell list based on gender (Male or Female), and gradyear (*$2011, 2012, 2013$ *or* 2014*). Fig. 2.1(c) shows the data vector that results from the instance and the cell list. Note that the sum of the counts in the data vector does not equal the total number of tuples in the instance because the cells happen not to cover the entire active domain of gradyear.*

A common case is to define a cell list by partitioning $dom(\mathbb{A})$ according to a single ordered attribute. In this case, the data vector would describe a one-dimensional histogram. The main criterion for selecting a cell list for a given schema is that the cells can be used to express the queries of the intended workload. This can be done in multiple ways and we return to the choice of cell lists later in this section.

Name	Gradyear	Gender	Gpa
Alice	2012	F	3.8
Bob	2011	M	3.1
Charlie	2014	M	3.6
Dave	2014	M	3.3
Evelyn	2013	F	3.9
Frank	2011	M	3.2
Gary	2015	M	3.5

(a) Instance of relation R

ϕ_1 :- $R(*, 2011, M, *)$	x_1: 2
ϕ_2 :- $R(*, 2011, F, *)$	x_2: 0
ϕ_3 :- $R(*, 2012, M, *)$	x_3: 0
ϕ_4 :- $R(*, 2012, F, *)$	x_4: 1
ϕ_5 :- $R(*, 2013, M, *)$	x_5: 0
ϕ_6 :- $R(*, 2013, F, *)$	x_6: 1
ϕ_7 :- $R(*, 2014, M, *)$	x_7: 2
ϕ_8 :- $R(*, 2014, F, *)$	x_8: 0

(b) Cell list Φ (c) $\mathbf{x}$

Figure 2.1. For schema $R = (name, gradyear, gender, gpa)$ (a) shows a sample instance. A cell list consisting of 8 cells described in terms of the tuples that match conditions on *gradyear* and *gender* is shown in (b). The database vector, shown in (c), accordingly consists of 8 counts.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \end{bmatrix}$$

(a) Query matrix $\mathbf{W}$

$\mathbf{w}_1$: Students of any gender with $gradyear \in [2011, 2014]$
$\mathbf{w}_2$: Students with $gradyear \in [2011, 2012]$
$\mathbf{w}_3$: Female students with $gradyear \in [2011, 2012]$
$\mathbf{w}_4$: Male students with $gradyear \in [2011, 2012]$
$\mathbf{w}_5$: Difference between 2013 grads and 2014 grads

(b) Five linear queries

$$\begin{aligned} \mathbf{w}_1\mathbf{x} &= x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 &&= 6 \\ \mathbf{w}_2\mathbf{x} &= x_1 + x_2 + x_3 + x_4 &&= 3 \\ \mathbf{w}_3\mathbf{x} &= x_2 + x_4 &&= 1 \\ \mathbf{w}_4\mathbf{x} &= x_1 + x_3 &&= 2 \\ \mathbf{w}_5\mathbf{x} &= x_5 + x_6 - x_7 - x_8 &&= -1 \end{aligned}$$

(c) The evaluation of $\mathbf{Wx}$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 \end{bmatrix}$$

(d) Query matrix $\mathbf{W}'$

Figure 2.2. (a) A query matrix $\mathbf{W}$ consisting of five linear queries; (b) The description of the queries in $\mathbf{W}$ using the cell list Φ in Fig 2.1; (c) The evaluation of $\mathbf{W}$ on $\mathbf{x}$; (d) A semantically equivalent query matrix $\mathbf{W}'$ expressed w.r.t. a reduced cell list (columns 5 and 6 in $\mathbf{W}$ have been combined to get $\mathbf{W}'$).

2.1.2 Linear queries

A linear query computes a linear combination of the counts in the data vector $\mathbf{x}$.

Definition 2.3 (Linear query)**.** *A linear query is a length-n row vector* $\mathbf{w} = [w_1 \dots w_n]$ *with each* $w_i \in \mathbb{R}$*. The answer to a linear query* $\mathbf{w}$ *on* $\mathbf{x}$ *is the dot product* $\mathbf{wx} = w_1 x_1 + \cdots + w_n x_n$*.*

Linear queries can express a variety of common aggregation queries. We refer to a linear query whose coefficients are exclusively zero or one as a *predicate counting query*, since it computes the number of tuples satisfying a predicate defined by the disjunction of the cells corresponding to query coefficients of one. For an ordered attribute domain, a *range-count query* is a special case of a predicate counting query whose non-zero coefficients form a contiguous range. Range count queries have a natural extension to *multi-dimensional range count queries.* Multi-dimensional range count queries are a versatile class: histograms, data cubes, marginal queries, and group-by queries are all sets of one-dimensional or multi-dimensional range count queries.

Even so, we do not restrict our attention only to linear queries with zero or one coefficients. With other coefficients, linear queries can compute differences (e.g. query $\mathbf{w}_5$ in Fig. 2.2(b)) and can express aggregate queries that are not, strictly speaking, counting queries. For example, referring to the cell list in Fig. 2.1, the average graduation year of male students graduating between 2011 and 2014 can be computed as $(2011x_1 + 2012x_3 + 2013x_5 + 2014x_7)/4$.

We will consider query workloads that consist of sets of linear queries, organized into the rows of a *query matrix.*

Definition 2.4 (Query matrix)**.** *A query matrix is a collection of m linear queries, arranged by rows to form an* $m \times n$ *matrix.*

If $\mathbf{W}$ is an $m \times n$ query matrix, the evaluation of $\mathbf{W}$ results in a length-m column vector of query answers, which can be computed as the matrix product $\mathbf{Wx}$.

Example 2.2. *Fig. 2.2 shows a query matrix representing a workload of five linear queries, along with the meaning of the queries using the cell list in Fig. 2.1(b). The queries are evaluated by computing* $\mathbf{Wx}$*, as shown in Fig. 2.2(c).*

2.1.3 Representing query workloads in matrix form

Later in the paper we will assume that an analyst has decided on a workload of queries of interest, selected a cell list, and represented the workload as a query matrix. The workload query matrix is the main input to our algorithms. We describe next a few guidelines and subtleties involved in representing a query workload in matrix form.

The matrix mechanism can be seen as automatically optimizing the workload to reduce error. As a result, the analyst does not have to think carefully about the workings of the privacy mechanism when representing the workload. In particular, the analyst need not try to reduce the sensitivity of the workload or avoid redundancy of queries. In fact, the analyst should include in the workload *all* queries of interest, even if some queries could be computed from others in the workload. As a concrete example, in Fig. 2.2(b), $\mathbf{w}_4$ can be computed as $(\mathbf{w}_2 - \mathbf{w}_3)$, but it is nevertheless included in the workload. This reflects our assumption that we wish to simultaneously answer all given workload queries with minimum aggregate error, treating each equally. The analyst may, however, choose to scale individual rows of the workload by a positive scalar value. This has the effect of increasing the importance of the query and reducing the error of that query relative to total error of the workload.

After deciding on the workload queries, the next step is to select an appropriate cell list that can support the workload queries. If each attribute domain is finite, then it is possible to fully represent instance I by defining the (finite) vector $\mathbf{x}$ with one cell for every element of $dom(\mathbb{A})$. Then $\mathbf{x}$ is a bit vector of size $|dom(\mathbb{A})|$ with nonzero counts for each tuple present in I. This is also a vector representation of the full contingency table built from I. (Note that if the schema contains infinite attribute domains, they would typically be partitioned into finite regions of sufficient granularity to support the desired queries.)

Selecting the cell list in this manner allows a wide range of desired queries to be supported. But it is often inefficient, since the size $\mathbf{x}$ vector grows exponentially with the sizes of the attribute domains, and ineffective, since the base counts are typically too small to be estimated very accurately. Alternatively, it may be sufficient to partially represent I by the cell counts in $\mathbf{x}$, for example by focusing on a subset of the attributes of $\mathbb{A}$ that are relevant to a specialized set of queries and/or a subset of the attribute domains (as in Example 2.1).

When representing a workload as a matrix, the order of workload queries are determined by the order of rows of the matrix. However, semantically, a workload means a *set* of queries, in which there is no specific orders amount those queries. Therefore the order of rows in a matrix does not semantically change its corresponding workload. In addition, there will always be many feasible choices for the cell list supporting a given workload. We formalize this using a notion of semantically equivalent workloads.

Definition 2.5 (Workload semantic equivalence)**.** *Workload* $\mathbf{W}$ *over cell list* Φ *is semantically equivalent to workload* $\mathbf{W}'$ *over cell list* Φ'*, denoted* $(\mathbf{W}, \Phi) \equiv (\mathbf{W}', \Phi')$*, if there is a permutation matrix* $\mathbf{P}$ *such that for every instance* I*,* $\mathbf{W}\mathbf{x}(I, \Phi) = \mathbf{P}\mathbf{W}'\mathbf{x}(I, \Phi')$*.*

Example 2.3. *Observe in Fig. 2.2 that columns 5 and 6 of workload* $\mathbf{W}$ *are identical. With respect to the example workload, positions 5 and 6 of the data vector are either both ignored, or are summed together. It follows that cells* ϕ_5 *and* ϕ_6 *can be combined and the query matrix altered by dropping one of the columns and that these operations will not modify the semantics of the workload. More precisely,* $(\mathbf{W}, \Phi) \equiv (\mathbf{W}', \Phi')$

where Φ' is derived from Φ as follows. The first four cells in Φ' are equal to those of Φ, cell $\phi'_5 = \phi_5 \vee \phi_6$, $\phi'_6 = \phi_7$, and $\phi'_7 = \phi_8$. Observe that $\mathbf{W}'$ results from removing column 6 from $\mathbf{W}$.

The following proposition shows that semantic equivalence can be characterized by considering a small set of semantic-preserving operations over cell lists and workload matrices.

Proposition 2.1. *For workload $\mathbf{W}$ over cell list Φ and a workload $\mathbf{W}'$ over cell list Φ', $(\mathbf{W}, \Phi) \equiv (\mathbf{W}', \Phi')$ if and only if $\mathbf{W}'$ and Φ' result from a sequence of one or more of the following operations:*

1. Permutation: *apply permutation μ to the rows of $\mathbf{W}$, or the cells of Φ and the columns of $\mathbf{W}$.*

2. Cell union: *if $\mathbf{W}$ contains two columns with identical coefficients, form $\mathbf{W}'$ by removing one of the columns and replacing the cells by their union.*

3. Cell division: *for any column W_i of $\mathbf{W}$ and corresponding cell ϕ_i of Φ, construct Φ' by replacing condition ϕ_i with ϕ_{i_1} and ϕ_{i_2} where $\phi_{i_1} \cup \phi_{i_2} = \phi_i$ and $\phi_{i_1} \cap \phi_{i_2} = \varnothing$. Then associate cell ϕ_{i_1} and ϕ_{i_2} with the column of coefficients W_i (i.e., two copies of W_i will appear in $\mathbf{W}'$).*

4. Add irrelevant cells: *add a new cell to Φ and a corresponding column to $\mathbf{W}$ whose coefficients are all zeros.*

5. Remove irrelevant cells: *if $\mathbf{W}$ contains a column of zeros, remove it along with its associated cell in Φ.*

The definition below is introduced to prove Proposition 2.1.

Definition 2.6 (Minimized workload)**.** *Given a workload $\mathbf{W}$, the minimized workload of $\mathbf{W}$ is defined as the workload $\mathbf{W}'$ that combines all duplicate columns in $\mathbf{W}$ and removes all zero columns from $\mathbf{W}$.*

Proof. The adequacy of the conditions in Proposition 2.1 is easy to be verified. Here we prove the necessity of those conditions.

Given $\mathbf{W}_1$ over Φ_1 and $\mathbf{W}_2$ over Φ_2 such that $(\mathbf{W}_1, \Phi_1) \equiv (\mathbf{W}_2, \Phi_2)$. As it is defined in Definition 2.5, there exists a permutation matrix $\mathbf{P}$ such that $\mathbf{W}_1\mathbf{x}(I, \Phi_1) = \mathbf{P}\mathbf{W}_2\mathbf{x}(I, \Phi_2)$ for any instance I. Noticing the row permutation is a part of the Permutation operation in Proposition 2.1, it is sufficient to consider the case that $\mathbf{P} = \mathbf{I}$.

Consider the minimized workload $\mathbf{W}'_1$ of $\mathbf{W}_1$ and its corresponding cell condition Φ'_1 such that $(\mathbf{W}'_1, \Phi'_1) \equiv (\mathbf{W}_1, \Phi_1)$. Apply the same process to $\mathbf{W}_2$ to get its minimized workload $\mathbf{W}'_2$ such that $(\mathbf{W}'_2, \Phi'_2) \equiv (\mathbf{W}_2, \Phi_2)$. According to Definition 2.5, $(\mathbf{W}'_1, \Phi'_1)$ must be semantically equivalent to $(\mathbf{W}'_2, \Phi'_2)$.

First of all, $\bigvee_{\phi \in \Phi'_1} \phi = \bigvee_{\phi \in \Phi'_2} \phi$. Otherwise, without loss of generality, assume $\bigvee_{\phi \in \Phi'_1} \phi$ is not a subset of $\bigvee_{\phi \in \Phi'_2} \phi$ and let

$$I_0 = \{t | \phi(t) \wedge (\neg \phi'(t)) \text{ is True}, \forall \phi \in \Phi'_1, \forall \phi' \in \Phi'_2\}.$$

Then $I_0 \neq \varnothing$ and $\mathbf{W}'_1\mathbf{x}(I_0, \Phi'_1) \neq \mathbf{W}'_2\mathbf{x}(I_0, \Phi'_2) = \mathbf{0}$, which contradicts with the fact that $(\mathbf{W}'_1, \Phi'_1) \equiv (\mathbf{W}'_2, \Phi'_2)$.

In addition, for any i, j such that $\phi_i \in \Phi'_1$ and $\phi'_j \in \Phi'_2$ such that $\phi_i \wedge \phi'_j \neq \varnothing$. Let W_i be the column of $\mathbf{W}'_1$ corresponding to ϕ_i and W'_j be the column of $\mathbf{W}'_2$ corresponding to ϕ'_j. W_i must be equal to W'_j. Otherwise, let $I_1 = \{t | \phi_i(t) \wedge \phi'_j(t) \text{ is True}\}$ and $\mathbf{W}'_1\mathbf{x}(I_1, \Phi'_1) = |I_1| W_i \neq \mathbf{W}'_2\mathbf{x}(I_1, \Phi'_2) = |I_1| W'_j$, which leads to a contradiction. Moreover, since neither $\mathbf{W}'_1$ nor $\mathbf{W}'_2$ contains duplicate columns, any cell conditions in Φ'_1 other than ϕ_i is disjoint with ϕ'_j and any cell conditions in Φ'_2 other than ϕ'_j is disjoint with ϕ_i. Therefore $\phi_i = \phi'_j$, otherwise $\bigvee_{\phi \in \Phi'_1} \phi \neq \bigvee_{\phi \in \Phi'_2} \phi$.

Above all, we know there must exist a permutation μ to the cells of Φ'_1 and the columns of $\mathbf{W}'_1$ that gets us $(\mathbf{W}'_2, \Phi'_2)$. Thus $(\mathbf{W}_1, \Phi_1)$ can be transformed into $(\mathbf{W}_2, \Phi_2)$ with the operations in Prop 2.1. $\square$

We will show later in the paper that many aspects of the performance of our algorithms are independent of the cell list used and the particular query matrix that results. Most importantly, the optimal error achievable for a workload is the same for any semantically-equivalent workload matrix. However, in terms of efficiency, it is beneficial to represent a workload with the smallest possible set of cells. The number of cells in the cell list, n, (which is also the number of columns in the workload matrix) is a key parameter in the computational complexity of the algorithms to come. Fortunately, using Prop. 2.1, it is straightforward to create the smallest cell list for a given workload of interest. After starting with any feasible representation of the workload, we can repeatedly apply steps (2) and (5), in any order.

2.2 Differential privacy

Informally, a randomized algorithm is differentially private if it produces statistically close outputs whether or not any one individual's record is present in the database. Two instances I and I' are neighbors, denoted $nbrs(I, I')$ if they differ by at most one record, i.e., if $|(I - I') \cup (I' - I)| = 1$.

Definition 2.7 (Differential privacy). *A randomized algorithm $\mathcal{K}$ is (ϵ, δ)-differentially private if for any instances I, I' such that $nbrs(I, I')$, and any subset of outputs $S \subseteq Range(\mathcal{K})$, the following holds:*

$$Pr[\mathcal{K}(I) \in S] \leq \exp(\epsilon) \times Pr[\mathcal{K}(I') \in S] + \delta,$$

where the probability is taken over the randomness of the $\mathcal{K}$.

If an algorithm satisfies the definition above for $\delta = 0$, then it is ϵ-differentially private. When $\delta > 0$ the privacy standard is sometimes referred to as *approximate differential privacy.*

Both ϵ and (ϵ, δ)-differential privacy can be satisfied by algorithms that add random noise to query answers. The magnitude of the required noise is determined by the privacy parameters, ϵ and/or δ, and the *sensitivity* of the set of queries: the maximum change in a vector of query answers over any two neighboring databases. The two privacy definitions differ, however, in the measurement of sensitivity and in their noise distributions. Standard ϵ-differential privacy can be achieved by adding Laplace noise calibrated to the L_1 sensitivity of the queries [26]. Approximate (ϵ, δ)-differential privacy can be achieved by adding Gaussian noise calibrated to the L_2 sensitivity of the queries [25, 51].

Since our query workloads are represented as matrices, we describe the sensitivity of a workload matrix as a matrix norm. Recall that, for any cell list Φ, cells are always disjoint and $\mathbf{x}(I, \Phi)$ is the vector representation of I using Φ. Since neighboring databases I and I' differ in exactly one tuple, it follows that the corresponding vectors $\mathbf{x}(I, \Phi)$ and $\mathbf{x}(I', \Phi)$ differ in at most one component, by at most one.

In the propositions below, cols$(\mathbf{W})$ is the set of column vectors W_i of $\mathbf{W}$. For a query matrix $\mathbf{W}$, the L_1 sensitivity is the maximum L_1 norm of the columns of $\mathbf{W}$, which is defined as the sum of absolute values of entries in one column.

Proposition 2.2 (L_1 Query matrix sensitivity)**.** *For any cell list Φ, the L_1 sensitivity of a query matrix $\mathbf{W}$ using cell list Φ is denoted $\|\mathbf{W}\|_1$ and defined as:*

$$\bar{\Delta}_{\mathbf{W}} \overset{\text{def}}{=} \max_{I, I' \in nbrs(I, I')} \|\mathbf{W}\mathbf{x}(I, \Phi) - \mathbf{W}\mathbf{x}(I', \Phi)\|_1 = \max_{W_i \in cols(\mathbf{W})} \|W_i\|_1$$

Similarly, the L_2 sensitivity of $\mathbf{W}$ is equal to the maximum L_2 norm of the columns of $\mathbf{W}$, which is defined as the square root of sum of squares of entries in one column.

Proposition 2.3 (L_2 Query matrix sensitivity)**.** *For any cell list Φ, the L_2 sensitivity of a query matrix $\mathbf{W}$ using cell list Φ is denoted $||\mathbf{W}||_2$ and defined as:*

$$\bar{\bar{\Delta}}_{\mathbf{W}} \overset{\text{def}}{=} \max_{I,I' \in nbrs(I,I')} ||\mathbf{W}\mathbf{x}(I,\Phi) - \mathbf{W}\mathbf{x}(I',\Phi)||_2 = \max_{W_i \in cols(\mathbf{W})} ||W_i||_2$$

It is clear from the above propositions that the sensitivity of a query matrix is in fact independent of any cell list that accompanies it and our notation reflects this. Further, we occasionally use $\Delta_{\mathbf{W}}$ to represent the sensitivity when the context does not specify whether it is L_1 or L_2 sensitivity.

Example 2.4. *Figure 2.3 shows three query matrices, over an unspecified cell list of size four, which we use as a running example. $\mathbf{I}_4$ is the identity matrix of size four. This matrix consists of four queries, each asking for an individual element of the data vector $\mathbf{x}$. $\mathbf{H}_4$ contains seven queries, which represent a binary hierarchy of sums: the first row is the sum of the elements of $\mathbf{x}$, the second and third rows each sum one half of $\mathbf{x}$, and the last four rows return individual elements of $\mathbf{x}$. $\mathbf{Y}_4$ is the matrix of the Haar wavelet. It can also be seen as a hierarchical set of queries: the first row is the total sum, the second row computes the difference between sums in two halves of $\mathbf{x}$, and the last two rows return differences between smaller partitions of $\mathbf{x}$.*

The sensitivity of each of the query matrices in Figure 2.3 is: $\bar{\Delta}_{\mathbf{I}_4} = 1$ and $\bar{\Delta}_{\mathbf{H}_4} = \bar{\Delta}_{\mathbf{Y}_4} = 3$; $\bar{\bar{\Delta}}_{\mathbf{I}_4} = 1$ and $\bar{\bar{\Delta}}_{\mathbf{H}_4} = \bar{\bar{\Delta}}_{\mathbf{Y}_4} = \sqrt{3}$. A change by one in any component $\mathbf{x}_i$ will change the query answer $\mathbf{I}_4\mathbf{x}$ by exactly one under both L_1 and L_2, but will change $\mathbf{H}_4\mathbf{x}$ and $\mathbf{Y}_4\mathbf{x}$ by 3 under L_1 and $\sqrt{3}$ under L_2 since each x_i contributes to three predicate queries in both $\mathbf{H}_4$ and $\mathbf{Y}_4$.

The following propositions describe, in vector form, the standard mechanisms for answering a set of queries under ϵ-differential privacy and (ϵ,δ)-differential privacy. The Laplace mechanism [25, 22] achieves ϵ-differential privacy by adding Laplace noise calibrated to the L_1 sensitivity of the input queries. We use Laplace$(b)^m$ to

$$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} \quad \begin{bmatrix}1&1&1&1\\1&1&0&0\\0&0&1&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} \quad \begin{bmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&0&0\\0&0&1&-1\end{bmatrix}$$

$$\mathbf{I}_4 \qquad \mathbf{H}_4 \qquad \mathbf{Y}_4$$

Figure 2.3. Query matrices with $dom = \{1, 2, 3, 4\}$. Each is full rank. I_4 returns each unit count. H_4 computes seven sums, hierarchically partitioning the domain. W_4 is based on the Haar wavelet.

denote a column vector consisting of m independent samples drawn from a Laplace distribution with mean 0 and scale b.

Proposition 2.4 (Laplace mechanism). *Given an $m \times n$ query matrix $\mathbf{W}$, the randomized algorithm $\mathcal{L}$ that outputs the following vector is ϵ-differentially private:*

$$\mathcal{L}(\mathbf{W}, \mathbf{x}) = \mathbf{W}\mathbf{x} + Laplace(b)^m$$

where $b = \bar{\Delta}_{\mathbf{W}}/\epsilon$.

The Gaussian mechanism [51] achieves (ϵ, δ)-differential privacy by adding Gaussian noise calibrated to the L_2 sensitivity. We use Normal$(\sigma)^m$ to denote a column vector consisting of m independent samples drawn from a Gaussian distribution with mean 0 and scale σ.

Proposition 2.5. (Gaussian mechanism [25, 51]) *Given an $m \times n$ query matrix $\mathbf{W}$, the randomized algorithm $\mathcal{G}$ that outputs the following vector is (ϵ, δ)-differentially private:*

$$\mathcal{G}(\mathbf{W}, \mathbf{x}) = \mathbf{W}\mathbf{x} + Normal(\sigma)^m$$

where $\sigma = \bar{\bar{\Delta}}_{\mathbf{W}}\sqrt{2\ln(2/\delta)}/\epsilon$.

Recall that $\mathbf{Wx}$ is a vector consisting of the true answers to each query in $\mathbf{W}$. The algorithms above add independent Laplace noise (scaled by $\bar{\Delta}_{\mathbf{W}}$ and ϵ) or Gaussian noise (scaled by $\bar{\bar{\Delta}}_{\mathbf{W}}$, ϵ, and δ) to each query answer. Thus both $\mathcal{L}(\mathbf{W}, \mathbf{x})$ and $\mathcal{G}(\mathbf{W}, \mathbf{x})$ are length-m column vectors containing a noisy answer for each linear query in $\mathbf{W}$.

2.3 Linear algebra fundamentals

Most of our discussions and analyses base on linear algebra operations. In this section, we summarize the concepts and results in linear algebra that are used throughout the dissertation.

In the dissertation, we use the standard notation of linear algebra and employ standard techniques of matrix analysis. We use $diag(d_1, \dots d_n)$ to indicate the $n \times n$ diagonal matrix with scalars d_i on the diagonal and $\mathbf{0}^{m \times n}$ to indicate a matrix of zeroes with m rows and n columns. Recall that for a matrix $\mathbf{A}$, $\mathbf{A}^T$ is its transpose, $\mathbf{A}^{-1}$ is its inverse. We say $\mathbf{A}$ is symmetric if $\mathbf{A}^T = \mathbf{A}$ and orthogonal if $\mathbf{A}^T = \mathbf{A}^{-1}$. The rank of a matrix $\mathbf{A}$, $\text{rank}(\mathbf{A})$, is defined as the size of the largest set of linearly independent rows (or equivalently columns) of $\mathbf{A}$. We say a matrix is full row (column) rank if its rank is equal to the number of its rows (columns). In particular $\mathbf{A}^{-1}$ exists if and only if $\mathbf{A}$ is a square matrix with full rank.

If matrix $\mathbf{A}$ is a square matrix, the trace of $\mathbf{A}$, denoted as $\text{trace}(\mathbf{A})$, is the sum of entries on the main diagonal if $\mathbf{A}$. The trace of a matrix has a very important property: it is invariant under cyclic permutations, i.e, if matrix $\mathbf{A}_1$ has m columns and matrix $\mathbf{A}_3$ has m rows,

$$\text{trace}(\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3) = \text{trace}(\mathbf{A}_3\mathbf{A}_1\mathbf{A}_2).$$

Another concept that is related with trace is the Frobenius norm. The Frobenius norm of $\mathbf{A}$ is denoted as $||\mathbf{A}||_F$ and defined as $\sqrt{\text{trace}(\mathbf{A}^T\mathbf{A})}$, or, equivalently, the square root of the squared sum of all entries in $\mathbf{A}$.

As a powerful tool in matrix analysis, matrix decomposition is extensively used in the dissertation. We focus on two decompositions: eigenvalue decomposition and singular value decomposition. Given a matrix $\mathbf{A}$, the eigenvalue decomposition of $\mathbf{A}$ always exists when $\mathbf{A}$ is symmetric. It can be written as the form of $\mathbf{A} = \mathbf{QDQ}^T$ where $\mathbf{Q}$ is an orthogonal matrix whose columns are eigenvectors of $\mathbf{A}$ and $\mathbf{D}$ is a diagonal matrix whose diagonal entries are eigenvalues of $\mathbf{A}$. The singular value decomposition of $\mathbf{A}$ always exists and is in form of $\mathbf{A} = \mathbf{QDP}^T$ where $\mathbf{Q}$ and $\mathbf{P}$ are orthogonal matrices and $\mathbf{D}$ is a diagonal matrix padding with columns or rows of 0s.

We will also rely on the notion of the positive semidefinite matrix. A symmetric square matrix $\mathbf{A}$ is called positive semidefinite, denoted as $\mathbf{A} \geq 0$, if for any vector $\mathbf{x}$, $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$. In particular, for any matrix $\mathbf{A}$, $\mathbf{A}^T\mathbf{A} \geq 0$. Here we present two equivalent conditions to positive semidefinite.

Proposition 2.6. *Given an $n \times n$ symmetric matrix $\mathbf{A}$, both of the following conditions are equivalent with $\mathbf{A} \geq 0$.*

(i) All the eigenvalues of $\mathbf{A}$ are non-negative.

(ii) For any $1 \leq i_1 < \ldots < i_k \leq n$, the determinant of the matrix that consists of the intersection of the $i_1^{th}, \ldots, i_k^{th}$ rows and $i_1^{th}, \ldots, i_k^{th}$ columns of matrix $\mathbf{A}$ is non-negative.

In addition, we consider a generalization of matrix inverse, called the Moore-Penrose pseudoinverse, which is defined as following.

Definition 2.8. (Moore-Penrose Pseudoinverse [11]) *Given a $m \times n$ matrix $\mathbf{A}$, a matrix $\mathbf{A}^+$ is the Moore-Penrose pseudoinverse of $\mathbf{A}$ if it satisfies each of the following:*

$$\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A},\ \mathbf{A}^+\mathbf{A}\mathbf{A}^+ = \mathbf{A}^+,\ (\mathbf{A}\mathbf{A}^+)^T = \mathbf{A}\mathbf{A}^+,\ (\mathbf{A}^+\mathbf{A})^T = \mathbf{A}^+\mathbf{A}.$$

The Moore-Penrose pseudoinverse is unique and can be computed with the singular value decomposition of a matrix.

Proposition 2.7 ([11])**.** *Given an $n \times n$ diagonal matrix $\mathbf{D}_0$, $\mathbf{D}_0^+ = \{d'_{ij}\}$ is an $n \times n$ diagonal matrix such that*

$$d'_{ij} = \begin{cases} 0 & d_{ij} = 0 \\ \frac{1}{d_{ij}} & d_{ij} \neq 0 \end{cases}$$

For an $m \times n$ matrix $\mathbf{D}$ consists of a diagonal matrix $\mathbf{D}_0$ padding with columns (rows) of 0s, $\mathbf{D}^+$ is an $n \times m$ consists of the diagonal matrix $\mathbf{D}_0^+$ with rows (columns) of 0s. Given a matrix $\mathbf{A}$ with singular value decomposition $\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{P}^T$, $\mathbf{A}^+ = \mathbf{P}\mathbf{D}^+\mathbf{Q}^T$.

When $\mathbf{A}$ has full column rank, $\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$. We include some important properties of the Moore-Penrose pseudoinverse in the following proposition.

Proposition 2.8. ([11]) *The Moore-Penrose pseudoinverse satisfies the following properties:*

1. *Given any matrix $\mathbf{A}$, there exists a unique matrix that is the Moore-Penrose pseudoinverse of $\mathbf{A}$.*
2. *Given a vector $\mathbf{y}$, we have $||\mathbf{y} - \mathbf{A}\mathbf{x}||_2 \geq ||\mathbf{y} - \mathbf{A}\mathbf{A}^+\mathbf{y}||_2$ for any vector $\mathbf{x}$.*
3. *For any satisfiable linear system $\mathbf{B}\mathbf{A} = \mathbf{W}$, $\mathbf{W}\mathbf{A}^+$ is a solution to the linear system and $||\mathbf{W}\mathbf{A}^+||_F \leq ||\mathbf{B}||_F$ for any solution $\mathbf{B}$ to the linear system.*

CHAPTER 3
MATRIX MECHANISM

This chapter covers the matrix mechanism, a novel mechanism that answers a group of linear queries under differential privacy. Central to our approach is the idea that "what you ask is different from what you want", which relies on the distinction between a query strategy and a query workload, both of which are sets of linear queries. A query workload is a set of queries that is originally submitted to the mechanism. Though a query workload can be answered directly with Laplace or Gaussian mechanism, the noise required may be more than necessary due to the linear dependency among the queries in the query workload. The matrix mechanism, instead, submits an alternative set of linear queries, called query strategy, whose answer can later be used to derive the answer to the query workload with linear combinations.

The matrix mechanism is a general framework that can applied to any differentially private mechanism. It is particularly powerful when the underlying differentially private mechanisms add i.i.d noise that is independent of the input database. Many works in differential privacy before or parallel with the matrix mechanism can actually be viewed as special instances of the matrix mechanism [19, 40, 70, 39, 9].

In this chapter, we formally define the matrix mechanism, and follow with a thorough error analysis that provides analytic formula of error under the matrix mechanism. Our error formula theoretically explains the experimental results in many of works above and show that many those experimental results can be acquired theoretically independent of any concrete database. As an example, a case study that analyzes algorithms in [40, 70] using the matrix mechanism is also included in this chapter. Furthermore, we formulate the corresponding optimization problems to find the strategy that minimize the error of a given workload. The matrix mechanism with non-negativity constraint is briefly discussed at the end of this chapter.

3.1 The matrix mechanism

A linear query $\mathbf{w}$ can be answered directly with the Laplace or Gaussian mechanism. Under ϵ-differential privacy, it has also been proven [32] that the amount of noise added by the Laplace mechanism is optimal. However, when answering a batch of linear queries simultaneously, the noise required by either the Laplace or Gaussian mechanism may be more than necessary due to the linear dependency among the queries in the query workload. Alternatively, previous works derive the answer for the workload from the noisy answer of a selected subset [40, 19]. In addition, other works [9, 70] apply manually designed linear transformations to the data vector, and add Laplace or Gaussian noise on the transformed domain. An estimated data vector can be generated by the inverse transformation and the workload queries can be answered by the estimated data vector.

The idea of the matrix mechanism is more general compared with past works. The matrix mechanism submits an alternative set of linear queries, called the query strategy, whose answer can later be used to derive the answer to the query workload using linear combinations. Such a query strategy can be a subset of the query workload, queries of linear transformation, or an arbitrary set of linear queries that can represent the query workload using linear combinations.

In this section we present the formal basis for the derivation process. We define the set of queries whose estimates can be derived and we provide optimal mechanisms for deriving estimates. In the remainder of this dissertation, we use $\mathbf{W}$ and $\mathbf{A}$ to denote the query workload and query strategy as well as their representation matrices, respectively.

Given a query strategy $\mathbf{A}$ and its noisy answer from any differentially private algorithm, in order to answer a query workload $\mathbf{W}$ with the answer to $\mathbf{A}$, each query $\mathbf{w}$ in $\mathbf{W}$ should be expressible as a linear combination of queries in $\mathbf{A}$:

Definition 3.1 (A workload supported by a strategy)**.** *Given a query workload* $\mathbf{W}$ *and a query strategy* $\mathbf{A}$*, we say* $\mathbf{A}$ *supports* $\mathbf{W}$ *if each query in* $\mathbf{W}$ *can be expressed as a linear combination of queries in* $\mathbf{A}$*. In other words, there exists a solution matrix* $\mathbf{X}$ *to the linear system* $\mathbf{W} = \mathbf{X}\mathbf{A}$*.*

To derive the answer to $\mathbf{W}$, one needs to solve linear system $\mathbf{W} = \mathbf{X}\mathbf{A}$. Noticing that there maybe multiple solutions to the linear system, we take the advantage of the uniqueness of the Moore-penrose pseudoinverse of matrix $\mathbf{A}$ and express the answer to $\mathbf{W}$ as following:

Definition 3.2 (Estimate the answer of $\mathbf{W}$ using $\mathbf{A}$)**.** *Let* $\mathbf{A}$ *be a query strategy that supports* $\mathbf{W}$ *and* $\hat{\mathbf{y}}$ *be the noise answers to* $\mathbf{A}$*. Then the noisy answer to* $\mathbf{W}$ *is defined as* $\mathbf{W}\mathbf{A}^{+}\hat{\mathbf{y}}$*, where* $\mathbf{A}^{+}$ *is the Moore-penrose pseudoinverse of matrix* $\mathbf{A}$*.*

Example 3.1. *Recall the cell conditions and queries in Figure 2.1. Let the query workload be* $\mathbf{W}_1 = \{\mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4\}$*. Then query strategy* $\mathbf{A}_1 = \{\mathbf{q}_1, \mathbf{q}_5\}$ *does not support* $\mathbf{W}$ *since it can not represent* $\mathbf{q}_2$*,* $\mathbf{q}_3$ *or* $\mathbf{q}_4$*.* $\mathbf{A}_2 = \{\mathbf{q}_3, \mathbf{q}_4\}$ *supports* $\mathbf{W}_1$ *and*

$$\mathbf{W}\mathbf{A}_2^{+} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

In such case, the answer to $\mathbf{W}_1$ *can be uniquely computed from the answer to* $\mathbf{A}_2$*, without any further assumption to the data vector* $\mathbf{x}$*.*

Now we introduce the matrix mechanism. Given any differentially private algorithm $\mathcal{K}$ that answers linear queries, the matrix mechanism can be considered as an

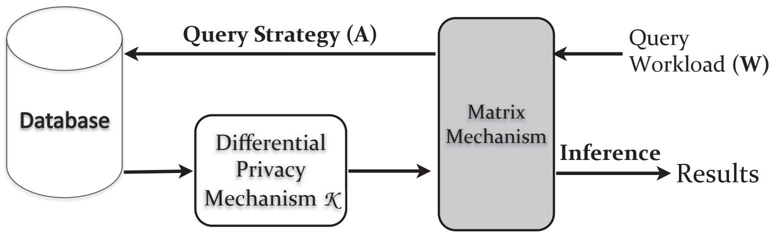


Figure 3.1. Query answering using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$.

enhancement mechanism to $\mathcal{K}$. With the supporting query strategy $\mathbf{A}$, the matrix mechanism is denoted as $\mathcal{M}_{\mathcal{K},\mathbf{A}}$.

Definition 3.3 (Matrix Mechanism)**.** *Given an $m \times n$ workload matrix $\mathbf{W}$, a $p \times n$ strategy matrix $\mathbf{A}$ that supports $\mathbf{W}$ and a differentially private algorithm $\mathcal{K}(\mathbf{A}, \mathbf{x})$ that answers $\mathbf{A}$ with a given database instance $\mathbf{x}$. The matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ outputs the following vector:*

$$\mathcal{M}_{\mathcal{K},\mathbf{A}}(\mathbf{W}, \mathbf{x}) = \mathbf{W}\mathbf{A}^{+}\mathcal{K}(\mathbf{A}, \mathbf{x}). \tag{3.1}$$

Figure 3.1 illustrates the process of query answering using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$. When it comes a query workload $\mathbf{W}$, the matrix mechanism chooses a query strategy $\mathbf{A}$ that supports $\mathbf{W}$, answers $\mathbf{A}$ with the differentially private algorithm $\mathcal{K}$ and outputs the derived answer to $\mathbf{W}$ using the answer to $\mathbf{A}$. The power of the matrix mechanism comes from the potential that the query strategy $\mathbf{A}$ can be more carefully designed to be answered under differential privacy. In addition, the matrix mechanism inherits the privacy and unbiased property of $\mathcal{K}$.

Proposition 3.1. *The matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ shares the same privacy guarantee with $\mathcal{K}$ and is unbiased if $\mathcal{K}$ is unbiased.*

Proof. According to Eqn (3.1), the matrix mechanism can be considered as a post process on $\mathcal{K}(\mathbf{A}, \mathbf{x})$ and hence share the same privacy guarantee with $\mathcal{K}(\mathbf{A}, \mathbf{x})$. In addition, noticing $\mathbf{W}\mathbf{A}^{+}\mathbf{A} = \mathbf{W}$,

$$\mathbb{E}[\mathcal{M}_{\mathcal{K},\mathbf{A}}(\mathbf{A}, \mathbf{x})] = \mathbb{E}[\mathbf{W}\mathbf{A}^{+}\mathcal{K}(\mathbf{A}, \mathbf{x})] = \mathbf{W}\mathbf{A}^{+}\mathbb{E}[\mathcal{K}(\mathbf{A}, \mathbf{x})] = \mathbf{W}\mathbf{A}^{+}\mathbf{A}\mathbf{x} = \mathbf{W}\mathbf{x}.$$

□

In general, Eqn. (3.1) is valid with any differentially private mechanism $\mathcal{K}$. The choice of $\mathcal{K}$ impatcs the hardness of error analysis and the complexity of find a query strategy $\mathbf{A}$ to minimize error on a given workload $\mathbf{W}$. Here we are interested in a differentially private algorithm $\mathcal{K}$ that satisfies three properties: 1) $\mathcal{K}(\mathbf{W}, \mathbf{x})$ achieves differential privacy by adding noise to $\mathbf{Wx}$; 2) the distribution of noise added is independent with $\mathbf{W}$ and $\mathbf{x}$; 3) the standard deviation of noise added is linearly scaled up with $\Delta_{\mathbf{W}}$ and is independent with $\mathbf{x}$. Analytically, such mechanism $\mathcal{K}$ can be represented into the following form:

$$\mathcal{K}(\mathbf{W}, \mathbf{x}) = \mathbf{Wx} + \Delta_{\mathbf{W}}\tilde{\mathbf{b}}, \tag{3.2}$$

where $\tilde{\mathbf{b}}$ is a vector of i.i.d random variables that does not depend on $\mathbf{W}$ or $\mathbf{x}$. Many data independent differentially private mechanisms based on adding noise can be represented in the form of Eqn. (3.2), such as the Laplace mechanism [25, 26], Gaussian mechanism[51, 26], the Geometric mechanism [32], and the K-Norm mechanism [39].

Proposition 3.2. *When $\mathcal{K}$ has the form of Eqn (3.2), the matrix mechanism can be presented as:.*

$$\mathcal{M}_{\mathcal{K},\mathbf{A}}(\mathbf{W}, \mathbf{x}) = \mathbf{Wx} + \mathbf{WA}^{+}\Delta_{\mathbf{A}}\tilde{\mathbf{b}}. \tag{3.3}$$

In the rest of this dissertation, we focus on the matrix mechanism with the form of Eqn. (3.3). In particular, we use ϵ-matrix mechanism to denote the case in which $\mathcal{K}$ is the Laplace mechanism, and (ϵ, δ)-matrix mechanism to denote the case in which $\mathcal{K}$ is the Gaussian mechanism.

According to Proposition 2.8, since entries of $\Delta_{\mathbf{A}}\tilde{\mathbf{b}}$ are generated from i.i.d random distributions, $\mathbf{WA}^{+}$ is the min-variance estimation to the noisy answer of $\mathbf{A}$.

Proposition 3.3. *When $\mathcal{K}$ has the form of Eqn (3.2), the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ produces the min-variance estimator to $\mathbf{Wx}$ given $\mathcal{K}(\mathbf{A}, \mathbf{x})$.*

Given the noisy answer to $\mathbf{A}$, there are multiple ways to estimate the answer to $\mathbf{W}$. [76] applies matrix decomposition on $\mathbf{W}$ and claims to solve a matrix $\mathbf{B}$ to minimize the square error of answering $\mathbf{W}$. As Prop. 3.3 indicates, their solution matrix $\mathbf{B}$ must be exactly the same as $\mathbf{W}\mathbf{A}^{+}$. On the other hand, in [75], a fixed "recovery" matrix $\mathbf{R}$ is used regardless of the noisy distribution of $\tilde{\mathbf{b}}_{\mathbf{A}}$, which, according to Prop. 3.3, is suboptimal.

3.2 Analyzing the error of the matrix mechanism

The error introduced using the matrix mechanism is impacted by two factors: the noise from the differentially private mechanism $\mathcal{K}$ and the linear combinations that generate the answer to the query workload $\mathbf{W}$ from the answer to the query strategy $\mathbf{A}$. We analyze the error of the matrix mechanism in this section and further derive a closed form expression with given $\mathcal{K}$, $\mathbf{W}$ and $\mathbf{A}$. We also studied the equivalent query workloads and query strategies under the matrix mechanism.

3.2.1 Error of the matrix mechanism

Given a query $\mathbf{w}$ and a query strategy $\mathbf{A}$ that supports $\mathbf{w}$, the error of answering $\mathbf{w}$ using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ is defined as the mean square error (variance) of the estimated answer to $\mathbf{w}$.

Definition 3.4 (Error of a single query). *Let $\mathbf{x}$ be the database instance and $\mathbf{A}$ be a query strategy. Given a single query $\mathbf{w}$ that $\mathbf{A}$ supports, the error of answer $\mathbf{w}$ using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ is:*

$$\textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{w}) = \mathbb{E}[(\mathbf{w}\mathbf{x} - \mathbf{w}\mathcal{K}(\mathbf{A},\mathbf{x}))^2].$$

For a query workload $\mathbf{W}$ that $\mathbf{A}$ supports, the total error of answering $\mathbf{W}$ using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ is:

$$\textsc{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W}) = \sum_{\mathbf{w}\in\mathbf{W}} \mathbb{E}[(\mathbf{w}\mathbf{x} - \mathbf{w}\mathbf{A}^{+}\mathcal{K}(\mathbf{A},\mathbf{x}))^2].$$

With a query strategy $\mathbf{A}$, the following proposition describes how to compute the error of answering a query $\mathbf{w}$ or a query workload $\mathbf{W}$ that is supported by $\mathbf{A}$ using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$.

Proposition 3.4. *Let $\mathbf{A}$ be a query strategy. Given a query $\mathbf{w}$ that $\mathbf{A}$ supports, the error of answering $\mathbf{w}$ using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ is:*

$$\textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{w}) = P(\mathcal{K})\Delta_{\mathbf{A}}^2 \|\mathbf{w}\mathbf{A}^{+}\|_F^2. \tag{3.4}$$

For a query workload $\mathbf{W}$ that $\mathbf{A}$ supports, the total error of answering $\mathbf{W}$ using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ is:

$$\textsc{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W}) = P(\mathcal{K})\Delta_{\mathbf{A}}^2 \|\mathbf{W}\mathbf{A}^{+}\|_F^2. \tag{3.5}$$

Here $P(\mathcal{K})$ is a constant determined by $\mathcal{K}$ and independent with $\mathbf{W}$, $\mathbf{A}$ and $\mathbf{x}$.

Proof. Recall that $\mathcal{K}(\mathbf{A},\mathbf{x}) = \mathbf{A}\mathbf{x} + \Delta_{\mathbf{A}}\tilde{\mathbf{b}}$ and the entires of $\tilde{\mathbf{b}}$ are i.i.d random variables. Let $\tilde{\mathbf{b}} = (b_1, \ldots, b_n)$. According to Definition 3.4, the error of answer $\mathbf{w}$ using the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ is:

$$\begin{aligned}
\textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{w}) &= \mathbb{E}[(\mathbf{w}\mathbf{x} - \mathbf{w}\mathbf{A}^{+}\mathcal{K}(\mathbf{A},\mathbf{x}))^2] = \mathbb{E}[(\mathbf{w}\mathbf{x} - \mathbf{w}\mathbf{A}^{+}(\mathbf{A}\mathbf{x} + \Delta_{\mathbf{A}}\tilde{\mathbf{b}}))^2] \\
&= \mathbb{E}[(\mathbf{w}\mathbf{A}^{+}\Delta_{\mathbf{A}}\tilde{\mathbf{b}})^2] \\
&= \mathrm{Var}(\mathbf{w}\mathbf{A}^{+}\Delta_{\mathbf{A}}\tilde{\mathbf{b}}) \\
&= \mathrm{Var}(b_1)\Delta_{\mathbf{A}}^2 \|\mathbf{w}\mathbf{A}^{+}\|_F^2.
\end{aligned}$$

Therefore, for a given query workload $\mathbf{W}$,

$$\begin{aligned}
\textsc{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W}) &= \sum_{\mathbf{w}\in\mathbf{W}} \mathbb{E}[(\mathbf{wx} - \mathbf{wA}^{+}\mathcal{K}(\mathbf{A},\mathbf{x}))^2 \\
&= \mathrm{Var}(b_1) \sum_{\mathbf{w}\in\mathbf{W}} \Delta_{\mathbf{A}}^2 \|\mathbf{wA}^{+}\|_2^2 \\
&= \mathrm{Var}(b_1)\Delta_{\mathbf{A}}^2 \|\mathbf{WA}^{+}\|_F^2.
\end{aligned}$$

Let $P(\mathcal{K}) = \mathrm{Var}(b_1)$, and recall that $\tilde{\mathbf{b}}$ only depends on $\mathcal{K}$. $P(\mathcal{K})$ is dependent with $\mathbf{W}$, $\mathbf{A}$ and $\mathbf{x}$. □

Since we only consider the matrix mechanism based on the data independent differentially private algorithm, the results in Proposition 3.4 do not contain the database instance $\mathbf{x}$ as well. Recall the parameter $P(\mathcal{K})$ is a constant that determined by the specific private algorithm $\mathcal{K}$. In particular, $P(\mathcal{K}) = 2/\epsilon^2$ and $P(\mathcal{K}) = 2\log(1/\delta)/\epsilon^2$ when $\mathcal{K}$ is Laplace mechanism and Gaussian mechanism, respectively. Moreover, the computation of $\Delta_{\mathbf{A}}$ is also determined by the choice of $\mathcal{K}$: it can either be the maximum L_1 or L_2 norm of the columns of $\mathbf{A}$ depending on whether $\mathcal{K}$ satisfies ϵ- or (ϵ,δ)-differential privacy, respectively.

Notice that a query strategy $\mathbf{A}$ impacts both Eqn. (3.4) and (3.5) in two ways: through $\Delta_{\mathbf{A}}$ and $\|\mathbf{WA}^{+}\|_F$. The former determines the cost of querying $\mathbf{A}$ using $\mathcal{K}$ and the later reflects the difficulty of computing the answer to $\mathbf{W}$ from the answer to $\mathbf{A}$. To lower the error, an ideal query strategy $\mathbf{A}$ should have low sensitivity while being as similar to $\mathbf{W}$ as possible. Here let us consider two extreme cases to those requirements. The first case is $\mathbf{A} = \mathbf{I}$, in which the sensitivity is as low as 1. However, if the queries in $\mathbf{W}$ are biased towards some cells or some of their combinations, the value of $\|\mathbf{WA}^{+}\|_F$ can be very large and hence lead to large noise. The other case is $\mathbf{A} = \mathbf{W}$, in which $\mathbf{A}$ and $\mathbf{W}$ are exactly the same. In such case $\|\mathbf{WA}^{+}\|_F = \mathrm{rank}(\mathbf{W})$, which is small, but the strategy performs badly if $\Delta_{\mathbf{W}}$ is high. In many practical cases, the best strategy is the one that achieves a proper balance between the sensitivity and the similarity to $\mathbf{W}$.

3.2.2 Total error equivalent workloads

For two distinct queries $\mathbf{w}_1$ and $\mathbf{w}_2$, one can verify that the query strategy $\mathbf{A} = \left[\begin{smallmatrix}\mathbf{w}_1 \\ 2\mathbf{w}_2\end{smallmatrix}\right]$ supports $\mathbf{w}_1$ and $\mathbf{w}_2$ and guarantees $\textsc{Error}_{\mathbf{A}}(\mathbf{w}_1) \neq \textsc{Error}_{\mathbf{A}}(\mathbf{w}_2)$. Therefore, there are no two queries that have the same error on all query strategies that support both of them. However, there exist pairs of query workloads whose total error are the same for all of their commonly supporting query strategies. Such workloads are defined as total error equivalent workloads.

Definition 3.5. *Two query workloads $\mathbf{W}_1$ and $\mathbf{W}_2$ are called total error equivalent, if for any query strategy $\mathbf{A}$ that supports both $\mathbf{W}_1$ and $\mathbf{W}_2$,$\textsc{TotalError}_{\mathbf{A}}(\mathbf{W}_1) = \textsc{TotalError}_{\mathbf{A}}(\mathbf{W}_2)$.*

Analysing Eqn. (3.5) leads to the following condition of total error equivalent.

Proposition 3.5. *Given two query workloads $\mathbf{W}_1$ and $\mathbf{W}_2$ where $\mathbf{W}_1$ has at least as naby queries as $\mathbf{W}_1$. $\mathbf{W}_1$ and $\mathbf{W}_2$ are total error equivalent, if and only if there exists an orthogonal matrix $\mathbf{Q}$ such that $\mathbf{W}_1 = \mathbf{Q}\mathbf{W}_2$ or $\mathbf{W}_1 = \mathbf{Q}\left[\begin{smallmatrix}\mathbf{W}_2 \\ \mathbf{0}\end{smallmatrix}\right]$ if $\mathbf{W}_1$ has more queries than $\mathbf{W}_2$.*

Proof. ($\Leftarrow$): When $\mathbf{W}_1 = \mathbf{Q}\mathbf{W}_2$ or $\mathbf{W}_1 = \mathbf{Q}\left[\begin{smallmatrix}\mathbf{W}_2 \\ \mathbf{0}\end{smallmatrix}\right]$ if $\mathbf{W}_1$ has more queries than $\mathbf{W}_2$, we have $\mathbf{W}_1^T\mathbf{W}_1 = \mathbf{W}_2^T\mathbf{W}_2$. Notice that

$$||\mathbf{W}\mathbf{A}^+||_F^2 = \text{trace}(\mathbf{W}^T(\mathbf{A}^T\mathbf{A})^+\mathbf{W}) = \text{trace}(\mathbf{W}^T\mathbf{W}(\mathbf{A}^T\mathbf{A})^+),$$

for any query strategy $\mathbf{A}$ that supports both $\mathbf{W}_1$ and $\mathbf{W}_2$, $\textsc{TotalError}_{\mathbf{A}}(\mathbf{W}_1) = \textsc{TotalError}_{\mathbf{A}}(\mathbf{W}_2)$.

($\Rightarrow$): If $\mathbf{W}_1^T\mathbf{W}_1 \neq \mathbf{W}_2^T\mathbf{W}_2$, consider the eigenvalue decomposition of $\mathbf{W}_1^T\mathbf{W}_1 - \mathbf{W}_2^T\mathbf{W}_2 =$

$\mathbf{QDQ}^T$ and $d_1, \ldots, d_n$ be the diagonal entries of $\mathbf{D}$. Without loss of generality, assume $d_1 \neq 0$ and let $\mathbf{D}' = \text{diag}(d_1', \ldots, d_n')$ where $d_1' = \sqrt{|d_1|}/\sqrt{|d_2| + \ldots + |d_n| + 1}$ and $d_2' = \ldots = d_n' = 1$. Let query strategy $\mathbf{A} = \mathbf{D}'\mathbf{Q}^T$. $\mathbf{A}$ supports $\mathbf{W}_1$ and $\mathbf{W}_2$ since it is full rank. Moreover,

$$\begin{aligned}
\|\mathbf{W}_1\mathbf{A}^+\|_F^2 - \|\mathbf{W}_2\mathbf{A}^+\|_F^2 &= \text{trace}(\mathbf{W}_1^T(\mathbf{A}^T\mathbf{A})^+\mathbf{W}_1) - \text{trace}(\mathbf{W}_2^T(\mathbf{A}^T\mathbf{A})^+\mathbf{W}_2) \\
&= \text{trace}(\mathbf{W}_1^T\mathbf{W}_1(\mathbf{A}^T\mathbf{A})^+) - \text{trace}(\mathbf{W}_2^T\mathbf{W}_2(\mathbf{A}^T\mathbf{A})^+) \\
&= \text{trace}((\mathbf{W}_1^T\mathbf{W}_1 - \mathbf{W}_2^T\mathbf{W}_2)(\mathbf{A}^T\mathbf{A})^+) \\
&= \frac{d_1}{|d_1|}(|d_2| + \ldots + |d_n| + 1) + d_2 + \ldots + d_n \neq 0.
\end{aligned}$$

When $\mathbf{W}_1^T\mathbf{W}_1 = \mathbf{W}_2^T\mathbf{W}_2$, there exists singular value decompositions $\mathbf{W}_1 = \mathbf{Q}_1\mathbf{D}_1\mathbf{P}^T$ and $\mathbf{Q}_2\mathbf{D}_2\mathbf{P}^T$, where the non-zero entries of $\mathbf{D}_1$ and $\mathbf{D}_2$ are the same. Thus, let $\mathbf{Q}_0 = \mathbf{Q}_1\mathbf{Q}_2^T$ or $\mathbf{Q}_0 = \mathbf{Q}_1\left[\begin{smallmatrix}\mathbf{Q}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}\end{smallmatrix}\right]$ if $\mathbf{W}_1$ has more queries than $\mathbf{W}_2$, and $\mathbf{W}_1 = \mathbf{Q}_0\mathbf{W}_2$. □

Noticing that any query strategy that supports $\mathbf{W}$ will support $\mathbf{QW}$ for any matrix $\mathbf{Q}$, the conclusion of Proposition 3.5 also indicates that the total error equivalent workloads share the same set of supporting strategies.

3.2.3 Equivalence between query strategies

In both Eqn. (3.4) and (3.5), the error is computed by $\Delta_{\mathbf{A}}$ and a Frobenius norm term $\|\mathbf{wA}^+\|_F^2$ and $\|\mathbf{WA}^+\|_F^2$, respectively. According to the definition of the Frobenius norm,

$$\|\mathbf{wA}^+\|_F^2 = \text{trace}(\mathbf{wA}^+(\mathbf{wA}^+)^T) = \text{trace}(\mathbf{w}(\mathbf{A}^T\mathbf{A})^+\mathbf{w}),$$
$$\|\mathbf{WA}^+\|_F^2 = \text{trace}(\mathbf{WA}^+(\mathbf{WA}^+)^T) = \text{trace}(\mathbf{W}(\mathbf{A}^T\mathbf{A})^+\mathbf{W}^T).$$

The righthand side of both equations above share a common term $(\mathbf{A}^T\mathbf{A})^+$, which we call an error profile.

Definition 3.6. *Given a query strategy* $\mathbf{A}$*, the matrix* $(\mathbf{A}^T\mathbf{A})^+$ *is called the error profile of the query strategy* $\mathbf{A}$*.*

When the matrix mechanism is instantiated with $\mathbf{A}$, its error profile characterizes the distribution of the error of answering queries under the matrix mechanism: the diagonal entries contains the variance of error for each cell and the off-diagonal entries encodes the covariance of error between cells. We hence define the equivalence between query strategies according to their error profiles and the error of answering different workloads using those strategies.

Definition 3.7 (Profile Equivalence between query strategies)**.** *Given two query strategies* $\mathbf{A}_1$ *and* $\mathbf{A}_2$*. We say that* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *are profile equivalent if there exists a non-zero constant* c *such that*

$$(\mathbf{A}_1^T\mathbf{A}_1)^+ = c(\mathbf{A}_2^T\mathbf{A}_2)^+.$$

The profile equivalent strategies are independent og the choice of $\mathcal{K}$. We also consider strategies equivalence, which depends on whether $\mathcal{K}$ satisfies ϵ- or (ϵ,δ)-differential privacy.

Definition 3.8 (Equivalence between strategies)**.** *Given two query strategies* $\mathbf{A}_1$ *and* $\mathbf{A}_2$*, We say that* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *are equivalent under* $\mathcal{K}$ *if they support the same sets of queries and for any query* $\mathbf{w}$ *that their support,* $\textsc{Error}_{\mathcal{K},\mathbf{A}_1}(\mathbf{w}) = \textsc{Error}_{\mathcal{K},\mathbf{A}_2}(\mathbf{w})$*.*

Example 3.2. *Figure 3.2 contains three query strategies* $\mathbf{H}_4'$*,* $\mathbf{H}_4''$ *and* $\mathbf{Y}_4$ *that are profile equivalent. In particular, under* ϵ*-differentially private mechanisms (e.g. the Laplace mechanism),* $\mathbf{H}_4'$ *and* $\mathbf{Y}_4$ *are equivalent but not equivalent with* $\mathbf{H}_4''$*.*

Besides the definition, there are other equivalent conditions for the profile equivalence, based on strategy matrices and their transformations.

$$\begin{bmatrix}1&1&0&0\\0&0&1&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} \quad \begin{bmatrix}1&1&0&0\\0&0&1&1\\\sqrt{2}&0&0&0\\0&\sqrt{2}&0&0\\0&0&\sqrt{2}&0\\0&0&0&\sqrt{2}\end{bmatrix} \quad \begin{bmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&0&0\\0&0&1&-1\end{bmatrix}$$

$$\mathbf{H}'_4 \qquad\qquad \mathbf{H}''_4 \qquad\qquad \mathbf{Y}_4$$

Figure 3.2. Profile equivalent strategies with $dom = \{1,2,3,4\}$.

Proposition 3.6. *Given two query strategies $\mathbf{A}_1$ and $\mathbf{A}_2$, where $\mathbf{A}_1$ has at least as many rows as $\mathbf{A}_2$. All of the following conditions are equivalent:*

(i) $\mathbf{A}_1$ and $\mathbf{A}_2$ are profile equivalent;

(ii) There exists a non-zero constant c such that $\mathbf{A}_1^T\mathbf{A}_1 = c \cdot \mathbf{A}_2^T\mathbf{A}_2$;

(iii) There exists a non-zero constant c and an orthogonal matrix $\mathbf{Q}$ such that $\mathbf{A}_1 = c \cdot \mathbf{Q}\mathbf{A}_2$ or $\mathbf{A}_1 = c \cdot \mathbf{Q}\left[\begin{smallmatrix}\mathbf{A}_2\\\mathbf{0}\end{smallmatrix}\right]$ if $\mathbf{A}_1$ has more rows than $\mathbf{A}_2$;

Proof. (i) $\Leftrightarrow$ (ii): According to the definition of the profile equivalent, $(\mathbf{A}_1^T\mathbf{A}_1)^+ = c \cdot \mathbf{A}_2^T\mathbf{A}_2)^+$. Take the Moore-Penrose pseudoinverse to both sides of the equation and we have $\mathbf{A}_1^T\mathbf{A}_1 = \frac{1}{c} \cdot \mathbf{A}_2^T\mathbf{A}_2$.
(ii) $\Rightarrow$ (iii): Since $\mathbf{A}_1^T\mathbf{A}_1 = c \cdot \mathbf{A}_2^T\mathbf{A}_2$, there exists singular value decompositions of $\mathbf{A}_1 = \mathbf{Q}_1\mathbf{D}_1\mathbf{P}_1^T$ and $\mathbf{A}_2 = \mathbf{Q}_2\mathbf{D}_2\mathbf{P}_2^T$ such that $\mathbf{P}_1 = \mathbf{P}_2$ and the diagonal entries of $\mathbf{D}_1$ is equal to $\sqrt{c}$ times the diagonal entries of $\mathbf{D}_2$. If $\mathbf{A}_1$ has more rows than $\mathbf{A}_2$, the matrix $\mathbf{Q} = \mathbf{Q}_1\left[\begin{smallmatrix}\mathbf{Q}_2^T & \mathbf{0}\\ \mathbf{0} & I\end{smallmatrix}\right]$ is the orthogonal matrix such that $\mathbf{A}_1 = \sqrt{c} \cdot \mathbf{Q}\left[\begin{smallmatrix}\mathbf{A}_2\\\mathbf{0}\end{smallmatrix}\right]$.
(iii) $\Rightarrow$ (ii): $\mathbf{A}_1^T\mathbf{A}_1 = c^2 \cdot \mathbf{A}_2^T\mathbf{Q}^T\mathbf{Q}\mathbf{A}_2 = c^2 \cdot \mathbf{A}_2^T\mathbf{A}_2$. □

The conditions in Proposition 3.6 imply that the profile equivalent strategies support the same set of queries. In addition, for each query they support, the ratio between the error introduced by those strategies is consistent.

Corollary 3.1. *Given two query strategies* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *that are profile equivalent. For any query* $\mathbf{W}$*,* $\mathbf{A}_1$ *supports* $\mathbf{W}$ *if and only if* $\mathbf{A}_2$ *supports* $\mathbf{W}$*. Furthermore, there exists a non-zero constant* c *such that given a differentially private algorithm* $\mathcal{K}$*, for any workload query* $\mathbf{W}$ *that* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *support,* $\textsc{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}) = c \cdot \textsc{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W})$*.*

Proof. Given a query workload $\mathbf{W}$, if $\mathbf{A}_1$ supports $\mathbf{W}$, there exists a matrix $\mathbf{X}$ such that $\mathbf{W} = \mathbf{X}\mathbf{A}_1$. According to Proposition 3.6(iii), $\mathbf{A}_1$ and $\mathbf{A}_2$ are profile equivalent if and only if there exists a non-zero constant c and a orthogonal matrix $\mathbf{Q}$ such that $\mathbf{A}_1 = c \cdot \mathbf{Q}\mathbf{A}_2$. Then $c \cdot \mathbf{X}\mathbf{Q}$ satisfies $\mathbf{W} = c \cdot \mathbf{X}\mathbf{Q}\mathbf{A}_2$ and therefore $\mathbf{A}_2$ supports $\mathbf{W}$ as well.

The definition of profile equivalent indicates that there is a constant c' such that $(\mathbf{A}_1^T\mathbf{A}_1)^+ = c' \cdot (\mathbf{A}_2^T\mathbf{A}_2)^+$. Thus for any query workload that $\mathbf{A}_1$ supports:

$$\begin{aligned}\frac{\textsc{TotalError}_{\mathcal{K},A_1}(W)}{\textsc{TotalError}_{\mathcal{K},A_2}(W)} &= \frac{\Delta^2_{\mathbf{A}_1}\|\mathbf{W}\mathbf{A}_1^+\|_F^2}{\Delta^2_{\mathbf{A}_2}\|\mathbf{W}\mathbf{A}_2^+\|_F^2} \\ &= \frac{\Delta^2_{\mathbf{A}_1}\mathrm{trace}(\mathbf{W}(\mathbf{A}_1^T\mathbf{A}_1)^+\mathbf{W}^T)}{\Delta^2_{\mathbf{A}_2}\mathrm{trace}(\mathbf{W}(\mathbf{A}_2^T\mathbf{A}_2)^+\mathbf{W}^T)} = c'\frac{\Delta^2_{\mathbf{A}_1}}{\Delta^2_{\mathbf{A}_2}},\end{aligned}$$

where the ratio is a value that is independent with $\mathbf{W}$. □

The following proposition reveals that the strategy equivalence is a special case of profile equivalence with an extra constraint.

Proposition 3.7. *Two query strategies* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *are equivalent if they are profile equivalent and*

$$\Delta^2_{\mathbf{A}_1}(\mathbf{A}_1^T\mathbf{A}_1)^+ = \Delta^2_{\mathbf{A}_2}(\mathbf{A}_2^T\mathbf{A}_2)^+,$$

In particular, $\mathbf{A}_1$ *and* $c \cdot \mathbf{A}_1$ *are equivalent for any non-zero scalar* c*.*

Proof. (⇐): If $\mathbf{A}_1$ and $\mathbf{A}_2$ are profile equivalent, Corollary 3.1 indicates that they support the same set of query workloads. Furthermore, one can verify that for any workload query that $\mathbf{A}_1$ and $\mathbf{A}_2$ support,

$$\begin{aligned}
\Delta^2_{\mathbf{A}_1}||\mathbf{W}\mathbf{A}_1^+||_F^2 &= \Delta^2_{\mathbf{A}_1}\text{trace}(\mathbf{W}^T(\mathbf{A}_1^T\mathbf{A}_1)^+\mathbf{W}) \\
&= \Delta^2_{\mathbf{A}_2}\text{trace}(\mathbf{W}^T(\mathbf{A}_2^T\mathbf{A}_2)^+\mathbf{W}) \\
&= \Delta^2_{\mathbf{A}_2}||\mathbf{W}\mathbf{A}_2^+||_F^2.
\end{aligned}$$

(⇒): First we prove that $\mathbf{A}_1^T\mathbf{A}_1$ and $\mathbf{A}_2^T\mathbf{A}_2$ have same eigenvectors. Otherwise, let $\mathbf{Q}_0$ be the matrix whose rows are orthogonal eigenvectors that are shared by $\mathbf{A}_1$ and $\mathbf{A}_2$, $\mathbf{Q}_1$ be the matrix whose rows are orthogonal eigenvectors of $\mathbf{A}_1$ that are supported by $\mathbf{A}_1$ and not eigenvectors of $\mathbf{A}_2$ and $\mathbf{Q}_2$ be the matrix whose rows are orthogonal eigenvectors of $\mathbf{A}_2$ that are supported by $\mathbf{A}_2$ and are not eigenvectors of $\mathbf{A}_1$. In addition, let $\mathbf{D}_1$ be the diagonal matrix whose diagonal entries are the eigenvalues of $\mathbf{A}_1^T\mathbf{A}_1$ corresponding to the rows of $\mathbf{Q}_1$ and let $\mathbf{D}_2$ be the diagonal matrix whose diagonal entries are the eigenvalues of $\mathbf{A}_2^T\mathbf{A}_2$ corresponding to the rows of $\mathbf{Q}_2$.

Noticing that the spanning space of $\mathbf{Q}_1$ contains all vectors that are supported by $\mathbf{A}_1$ and are orthogonal to all vectors in $\mathbf{Q}_0$ and so does the spanning space of $\mathbf{Q}_2$. Recall the equivalent query strategies support the same set of queries, the rows in $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are actually two orthogonal basis to the same subspace. There hence exists an orthogonal matrix $\mathbf{Q}$ such that $\mathbf{Q}_1 = \mathbf{Q}\mathbf{Q}_2$. For any vector $\mathbf{v}$, $\mathbf{v}\mathbf{Q}_1$ is a query that $\mathbf{A}_1$ supports and

$$\begin{aligned}
\textsc{Error}_{\mathcal{K},\mathbf{A}_1}(\mathbf{v}\mathbf{Q}_1) &= P(\mathcal{K})\Delta^2_{\mathbf{A}_1}\|\mathbf{v}\mathbf{Q}_1\mathbf{A}_1^+\|_F^2 \\
&= P(\mathcal{K})\Delta^2_{\mathbf{A}_1}\mathrm{trace}(\mathbf{v}\mathbf{Q}_1(\mathbf{A}_1^T\mathbf{A}_1)^+\mathbf{Q}_1^T\mathbf{v}^T) \\
&= P(\mathcal{K})\Delta^2_{\mathbf{A}_1}\mathbf{v}\mathbf{D}_1^{-1}\mathbf{v}^T, \\
\textsc{Error}_{\mathcal{K},\mathbf{A}_2}(\mathbf{v}\mathbf{Q}_1) &= P(\mathcal{K})\Delta^2_{\mathbf{A}_2}\|\mathbf{v}\mathbf{Q}_1\mathbf{A}_2^+\|_F^2 \\
&= P(\mathcal{K})\Delta^2_{\mathbf{A}_2}\mathrm{trace}(\mathbf{v}\mathbf{Q}\mathbf{Q}_2(\mathbf{A}_2^T\mathbf{A}_2)^+\mathbf{Q}_2^T\mathbf{Q}^T\mathbf{v}^T) \\
&= P(\mathcal{K})\Delta^2_{\mathbf{A}_2}\mathbf{v}\mathbf{Q}\mathbf{D}_2^{-1}\mathbf{Q}^T\mathbf{v}^T.
\end{aligned}$$

Since $\mathbf{A}_1$ and $\mathbf{A}_2$ are equivalent, $\textsc{Error}_{\mathcal{K},\mathbf{A}_1}(\mathbf{v}\mathbf{Q}_1) = \textsc{Error}_{\mathcal{K},\mathbf{A}_2}(\mathbf{v}\mathbf{Q}_1)$ for any $\mathbf{v}$, which is equivalent to for any $\mathbf{v}$,

$$\textsc{Error}_{\mathcal{K},\mathbf{A}_1}(\mathbf{v}\mathbf{Q}_1) - \textsc{Error}_{\mathcal{K},\mathbf{A}_2}(\mathbf{v}\mathbf{Q}_1) = P(\mathcal{K})\mathbf{v}(\Delta^2_{\mathbf{A}_1}\mathbf{D}_1^{-1} - \Delta^2_{\mathbf{A}_2}\mathbf{Q}\mathbf{D}_2^{-1}\mathbf{Q}^T)\mathbf{v}^T = 0.$$

Thus $\Delta^2_{\mathbf{A}_1}\mathbf{D}_1^{-1} = \Delta^2_{\mathbf{A}_2}\mathbf{Q}\mathbf{D}_2^{-1}\mathbf{Q}^T$ and we can consider $\mathbf{Q}(\Delta^2_{\mathbf{A}_2}\mathbf{D}_2^{-1})\mathbf{Q}^T$ is an eigenvalue decomposition of matrix $\Delta^2_{\mathbf{A}_1}\mathbf{D}_1^{-1}$. Recall $\mathbf{Q}_1 = \mathbf{Q}\mathbf{Q}_2$ and none of the rows of $\mathbf{Q}_1$ belongs to $\mathbf{Q}_2$. Therefore there is no columns in $\mathbf{Q}$ that consists one entry equal to 1 and all other entries equal to 0, which indicates all diagonal entries of $\mathbf{D}_2^{-1}$ should be equal. However, in such case, the rows of $\mathbf{Q}_1$ will be eigenvectors of $\mathbf{A}_2^T\mathbf{A}_2$, which leads to a contradiction and we known $\mathbf{A}_1^T\mathbf{A}_1$ and $\mathbf{A}_2^T\mathbf{A}_2$ must have same eigenvectors.

In addition, given an eigenvector $\mathbf{u}$ of $\mathbf{A}_1^T\mathbf{A}_1$ and $\mathbf{A}_2^T\mathbf{A}_2$ that $\mathbf{A}_1$ and $\mathbf{A}_2$ support. Let $\mathbf{A}_1^T\mathbf{A}_1\mathbf{u}^T = \xi_1\mathbf{u}^T$ and $\mathbf{A}_2^T\mathbf{A}_2\mathbf{u}^T = \xi_2\mathbf{u}^T$. Since $\mathbf{A}_1$ and $\mathbf{A}_2$ support $\mathbf{u}$, $\xi_1 \neq 0$ and $\xi_2 \neq 0$. Furthermore,

$$\begin{aligned}
\textsc{Error}_{\mathbf{A}_1}(\mathbf{u}) &= P(\mathcal{K})\Delta^2_{\mathbf{A}_1}||\mathbf{u}\mathbf{A}_1^+||_F^2 \\
&= \Delta^2_{\mathbf{A}_1}\mathrm{trace}(\mathbf{u}(\mathbf{A}_1^T\mathbf{A}_1)^+\mathbf{u}^T) = \frac{\Delta^2_{\mathbf{A}_1}||\mathbf{u}||_2^2}{\xi_1}, \\
\textsc{Error}_{\mathbf{A}_2}(\mathbf{u}) &= P(\mathcal{K})\Delta^2_{\mathbf{A}_2}||\mathbf{u}\mathbf{A}_2^+||_F^2 \\
&= \Delta^2_{\mathbf{A}_2}\mathrm{trace}(\mathbf{u}(\mathbf{A}_2^T\mathbf{A}_2)^+\mathbf{u}^T) = \frac{\Delta^2_{\mathbf{A}_2}||\mathbf{u}||_2^2}{\xi_2},
\end{aligned}$$

Since $\textsc{Error}_{\mathbf{A}_1}(\mathbf{u}) = \textsc{Error}_{\mathbf{A}_2}(\mathbf{u})$, $\Delta^2_{\mathbf{A}_1}\xi_2 = \Delta^2_{\mathbf{A}_2}\xi_1$. Noticing it is true for all pairs of corresponding eigenvalues of $\mathbf{A}_1^T\mathbf{A}_1$ and $\mathbf{A}_2^T\mathbf{A}_2$, we have

$$\Delta^2_{\mathbf{A}_2}(\mathbf{A}_1^T\mathbf{A}_1) = \Delta^2_{\mathbf{A}_1}(\mathbf{A}_2^T\mathbf{A}_2).$$

According to Corollary 3.1, $\mathbf{A}_1$ and $\mathbf{A}_2$ are profile equivalent and

$$\Delta^2_{\mathbf{A}_1}(\mathbf{A}_1^T\mathbf{A}_1)^+ = \Delta^2_{\mathbf{A}_2}(\mathbf{A}_2^T\mathbf{A}_2)^+.$$

□

Proposition 3.8. *Given a differentially private algorithm $\mathcal{K}$, If $\Delta_{\mathbf{A}} = \bar{\bar{\Delta}}_{\mathbf{A}}$, all profile equivalent query strategies are equivalent.*

Proof. Noticing that $\bar{\bar{\Delta}}^2_{\mathbf{A}}$ is equal to the largest diagonal entry of matrix $\mathbf{A}^T\mathbf{A}$, given two query strategies $\mathbf{A}_1$ and $\mathbf{A}_2$ that are profile equivalent, by definition there exists a constant c such that $(\mathbf{A}_1^T\mathbf{A}_1)^+ = c\cdot(\mathbf{A}_2^T\mathbf{A}_2)^+$. Then $c\cdot\mathbf{A}_1^T\mathbf{A}_1 = \mathbf{A}_2^T\mathbf{A}_2$ and $c\cdot\bar{\bar{\Delta}}^2_{\mathbf{A}_1} = \bar{\bar{\Delta}}^2_{\mathbf{A}_2}$. Substitute those values into Eqn. (3.5) and we know $\textsc{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}) = \textsc{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W})$ for any query workload $\mathbf{W}$ that $\mathbf{A}_1$ and $\mathbf{A}_2$ support. □

3.3 Application: analyzing $\mathbf{H}_n$ and $\mathbf{Y}_n$ using the matrix mechanism

In this section we use our techniques to analyze and improve existing approaches. We analyze two techniques proposed recently [70, 40]. Both strategies can be seen as instances of the matrix mechanism, each using different query strategies designed to support a workload consisting of all range queries. Although both techniques can support multidimensional range queries, we focus our analysis on one dimensional range queries, i.e. interval queries with respect to a total order over $dom(\mathbb{B})$.

We will show that the seemingly distinct approaches have remarkably similar behavior: they have low (but not minimal) sensitivity, and they are highly accurate for range queries but much worse for other types of queries. We describe these techniques briefly and how they can each be represented in matrix form.

In the *hierarchical* scheme proposed in [40], the query strategy can be envisioned as a recursive partitioning of the domain. We consider the simple case of a binary partitioning, although higher branching factors were considered in [40]. First we ask for the total sum over the whole domain, and then ask for the count of each half of the domain, and so on, terminating with counts of individual elements of the domain. For a domain of size n (assumed for simplicity to be a power of 2), this results in a query strategy consisting of $2n-1$ rows. We represent this strategy as matrix $\mathbf{H}_n$, and $\mathbf{H}_4$ in Fig. 2.3 is a small instance of it.

In the *wavelet* scheme, proposed in [70], query strategies are based on the Haar wavelet. For one dimensional range queries, the technique can also be envisioned as a hierarchical scheme, asking the total query, then asking for the difference between the left half and right half of the domain, continuing to recurse, asking for the difference in counts between each binary partition of the domain at each step. Though presented differently in [70], we prove later in this section the equivalence of that construction with our formulation $\mathbf{Y}_n$. This results in n queries—fewer than the hierarchical scheme of [40]. The matrix corresponding to this strategy is the matrix of the Haar wavelet transform, denoted $\mathbf{Y}_n$, and $\mathbf{Y}_4$ in Fig. 2.3 is a small instance of it.

Thus $\mathbf{H}_n$ is a rectangular $(2n-1)\times n$ strategy, with answers derived using the linear regression technique, and $\mathbf{Y}_n$ is an $n \times n$ strategy with answers derived by inverting the strategy matrix. As suggested by the examples in earlier sections, these seemingly different techniques exhibit similar behavior. We analyze them in detail below, proving new bounds on the error for each technique, and proving new results about their relationship to one another. We also include $\mathbf{I}_n$ in the analysis, which is the strategy represented by the dimension n identity matrix, which asks for each individual count.

3.3.1 Representing the Haar wavelet technique

The representation $\mathbf{Y}_n$ is different from the original presentation in Xiao et al. [70]. The following theorem shows the equivalence of both representations.

Proposition 3.9 (Equivalence of Haar wavelet representations)**.** *Let $\hat{\mathbf{x}}_{Haar}$ denote the estimate derived from the Haar wavelet approach of Xiao et al. [70]. Let $\hat{\mathbf{x}}_{\mathbf{Y}_n}$ denote the estimate from asking query $\mathbf{W}_n$. Then $\hat{\mathbf{x}}_{Haar}$ and $\hat{\mathbf{x}}_{\mathbf{Y}_n}$ are equal in distribution, i.e., $Pr[\hat{\mathbf{x}}_{Haar} \leq x] = Pr[\hat{\mathbf{x}}_{\mathbf{Y}_n} \leq x]$ for any vector x.*

Proof. Given vector $\mathbf{x}$, the Haar wavelet is defined in terms of a binary tree over $\mathbf{x}$ such that the leaves of the tree are $\mathbf{x}$.

Each node in the tree is associated with a coefficient. Coefficient c_i is defined as $c_i = (a_L - a_R)/2$ where a_L (a_R) is the average of the leaves in the left (right) subtree of c_i. Each c_i is associated with a weight $\mathcal{W}(c_i)$ which is equal to the number of leaves in subtree rooted at c_i. (In addition, there is a coefficient c_0 that is the equal to the average of $\mathbf{x}$ and $\mathcal{W}(c_0) = n$).

An equivalent definition for c_i is $c_i = \sum_{j=1}^{n} x_j z_i(j)$ where for $i > 0$,

$$z_i(j) \;=\; \begin{cases} 1/\mathcal{W}(c_i), & \text{if } j \text{ is in the left subtree of } c_i \\ -1/\mathcal{W}(c_i), & \text{if } j \text{ is in the right subtree of } c_i \\ 0, & \text{otherwise} \end{cases}$$

For $i = 0$, then $z_i(j)$ is equal to $1/\mathcal{W}(c_0)$ for all j.

Let $\mathbf{A}$ be a matrix where $a_{ij} = z_i(j)$. The i^{th} row of $\mathbf{A}$ corresponds to coefficient c_i. Since there are n coefficients, $\mathbf{A}$ is an $n \times n$ matrix.

The approach of [70] computes the following $\mathbf{y}_{Haar} = \mathbf{A}\mathbf{x} + \mathbb{E}$ where $\mathbb{E}$ is an $n \times 1$ vector such that each $\mathbb{E}_i$ is an independent sample from a Laplace distribution with scale $b_i = \frac{1+\log n}{\epsilon \mathcal{W}(c_i)}$. Observe that $\mathbb{E}$ can be equivalently represented as:

$$\mathbb{E} = \mathbf{R}^{-1}\left(\frac{1+\log n}{\epsilon}\right)\tilde{\mathbf{b}}$$

where $\mathbf{R}$ is an $n \times n$ diagonal matrix with $r_{ii} = \mathcal{W}(c_i)$. The estimate for $\mathbf{x}$ is then equal to:

$$\begin{aligned} \hat{\mathbf{x}}_{Haar} &= \mathbf{A}^{-1}\mathbf{y}_{Haar} = \mathbf{x} + \mathbf{A}^{-1}\mathbb{E} \\ &= \mathbf{x} + \mathbf{A}^{-1}\mathbf{R}^{-1}\left(\frac{1+\log n}{\epsilon}\right)\tilde{\mathbf{b}} \\ &= \mathbf{x} + (\mathbf{R}\mathbf{A})^{-1}\left(\frac{1+\log n}{\epsilon}\right)\tilde{\mathbf{b}} \end{aligned}$$

We now describe an equivalent approach based on the matrix $\mathbf{Y}_n$. Observe that $\mathbf{Y}_n = \mathbf{R}\mathbf{A}$. The sensitivity of $\mathbf{Y}_n$ is $\Delta_{\mathbf{Y}_n} = 1 + \log n$. Using the matrix mechanism, the estimate $\hat{\mathbf{x}}_{\mathbf{Y}_n}$ is:

$$\begin{aligned}\hat{\mathbf{x}}_{\mathbf{Y}_n} &= {\mathbf{Y}_n}^{-1}\left(\mathbf{Y}_n\mathbf{x} + (\frac{\Delta_{\mathbf{Y}_n}}{\epsilon})\tilde{\mathbf{b}})\right) \\ &= \mathbf{x} + {\mathbf{Y}_n}^{-1}\frac{\Delta_{\mathbf{Y}_n}}{\epsilon}\tilde{\mathbf{b}} \\ &= \mathbf{x} + (\mathbf{RA})^{-1}\left(\frac{1+\log n}{\epsilon}\right)\tilde{\mathbf{b}}\end{aligned}$$

□

3.3.2 Similarity between $\mathbf{H}_n$ and $\mathbf{Y}_n$

Though represented differently, $\mathbf{H}_n$ and $\mathbf{Y}_n$ are actually very similar strategies under the matrix mechanism. In particular, a strategy matrix that is equivalent to $\mathbf{Y}_n$ can be achieved by removing the query of total sum and adding identity queries on each cell.

Theorem 3.1. *Let n be a power of 2, denoted as $n = 2^k$. Let $\mathbf{H}'_n$ be the matrix that remove the row of all 1s from matrix $\left[\begin{smallmatrix}\mathbf{H}_n \\ \mathbf{I}_n\end{smallmatrix}\right]$. Then $\mathbf{H}'_n$ and $\mathbf{W}$ are equivalent strategies under both ϵ- and (ϵ,δ)-differential privacy.*

Proof. Noticing $\mathbf{H}'_n$ has the same L_1 and L_2 sensitivity with $\mathbf{Y}_n$, it is sufficient to proof ${\mathbf{H}'_n}^T\mathbf{H}'_n = \mathbf{Y}_n^T\mathbf{Y}_n$, which is equivalent to prove $\mathbf{H}_n^T\mathbf{H}_n = \mathbf{Y}_n^T\mathbf{Y}_n + \mathbf{1}_{n\times n} - \mathbf{I}_n$.

Recall $n = 2^k$, and we will prove the conclusion by induction on k. When $k = 1$,

$$\mathbf{H}_2^T\mathbf{H}_2 = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix},$$

$$\mathbf{Y}_2^T\mathbf{Y}_2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}.$$

Assume the conclusion is correct for $k-1$. Since

$$\begin{bmatrix}1&1&1&1\\1&1&0&0\\0&0&1&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

(a) $\mathbf{H}_4$

$$\begin{bmatrix}3&2&1&1\\2&3&1&1\\1&1&3&2\\1&1&2&3\end{bmatrix}$$

(b) $\mathbf{H}_4^T\mathbf{H}_4$

$$\begin{bmatrix}1&1&0&0\\0&0&1&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

(c) $\mathbf{H}'_4$

$$\begin{bmatrix}1&1&1&1\\1&1&-1&-1\\1&-1&0&0\\0&0&1&-1\end{bmatrix}$$

(d) $\mathbf{Y}_4$

$$\begin{bmatrix}3&1&0&0\\1&3&0&0\\0&0&3&1\\0&0&1&3\end{bmatrix}$$

(e) $\mathbf{H}_4'^T\mathbf{H}'_4$, $\mathbf{Y}_4^T\mathbf{Y}_4$

Figure 3.3. The strategy matrices $\mathbf{H}_4$, $\mathbf{H}'_4$ and $\mathbf{Y}_4$.

$$\mathbf{H}_{2^k} = \begin{bmatrix}\mathbf{1}_{1\times 2^{k-1}} & \mathbf{1}_{1\times 2^{k-1}}\\ \mathbf{H}_{2^{k-1}} & 0\\ 0 & \mathbf{H}_{2^{k-1}}\end{bmatrix},$$

$$\mathbf{Y}_{2^k} = \begin{bmatrix}\mathbf{Y}_{2^{k-1}} & \begin{matrix}\mathbf{1}_{1\times 2^{k-1}}\\ 0\end{matrix}\\ \begin{matrix}-\mathbf{1}_{1\times 2^{k-1}}\\ 0\end{matrix} & \mathbf{Y}_{2^{k-1}}\end{bmatrix},$$

one can verify that $\mathbf{H}_{2^k}^T\mathbf{H}_{2^k} = \mathbf{Y}_{2^k}^T\mathbf{Y}_{2^k} + \mathbf{1}_{2^k\times 2^k} - \mathbf{I}_{2^k}$. □

Example 3.3. *Figure 3.3 contains the strategy matrices $\mathbf{H}_4$, $\mathbf{H}'_4$ and $\mathbf{Y}_4$, which reveals the relationship among those matrices. Adding $\mathbf{I}_4$ to $\mathbf{H}_4$ and removing the row of all 1s yields $\mathbf{H}'_4$. In addition, we can see that $\mathbf{H}'_4$ and $\mathbf{Y}_4$ are equivalent strategies under both ϵ- and (ϵ, δ)- differentially private mechanisms.*

It follows from the similarity of $\mathbf{H}_n$ and $\mathbf{Y}_n$ that the error profiles are asymptotically equivalent to one another. We thus prove a close equivalence between the error of the two techniques:

Corollary 3.2. *For any linear counting query* $\mathbf{w}$ *and differentially private mechanism* $\mathcal{K}$*,*

$$\frac{1}{2}\textsc{Error}_{\mathcal{K},\mathbf{Y}}(\mathbf{w}) \leq \textsc{Error}_{\mathcal{K},\mathbf{H}}(\mathbf{w}) \leq 2\textsc{Error}_{\mathcal{K},\mathbf{Y}}(\mathbf{w}).$$

Proof. According to Theorem 3.1, let $\mathbf{H}'_n$ be the matrix that remove the row of all 1s from matrix $\left[\begin{smallmatrix}\mathbf{H}_n\\ \mathbf{I}_n\end{smallmatrix}\right]$. Since $\mathbf{H}'_n$ and $\mathbf{Y}_n$ are equivalent strategies under both ϵ- and (ϵ,δ)- differentially private mechanisms, it is sufficient to prove that for any linear counting query $\mathbf{w}$,

$$\frac{1}{2}\textsc{Error}_{\mathcal{K},\mathbf{H}'_n}(\mathbf{w}) \leq \textsc{Error}_{\mathcal{K},\mathbf{H}_n}(\mathbf{w}) \leq 2\textsc{Error}_{\mathcal{K},\mathbf{H}'_n}(\mathbf{w}).$$

Let $\mathbf{v} = \mathbf{w}\mathbf{H}_n^{+}$, and $\mathbf{v}'$ be a vector such that

$$v'_i = \begin{cases} v_1 & 1 \leq i \leq 2 \\ v_{i-1} & 3 \leq i \leq 2n \\ 0 & 2n+1 \leq i \leq 3n \end{cases}.$$

One can verify that $\mathbf{v}'\mathbf{H}'_n = \mathbf{v}\mathbf{H}_n = \mathbf{w}$. Since

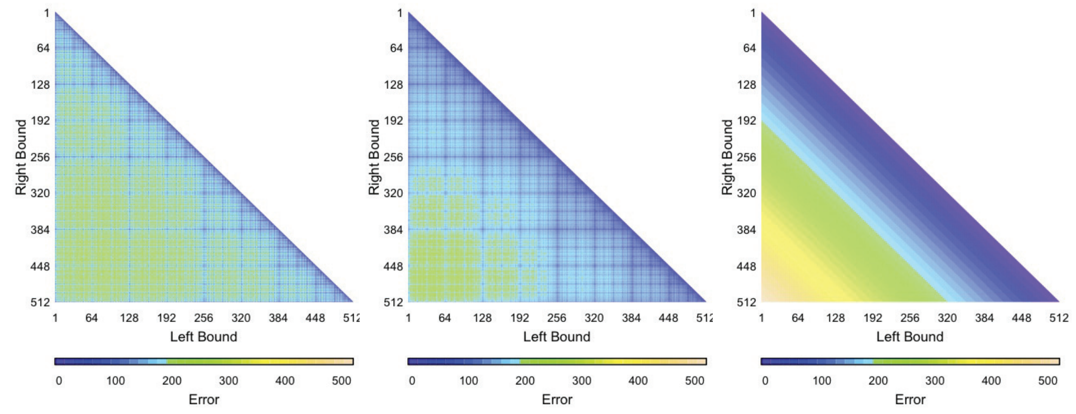
$$||\mathbf{w}\mathbf{H}'^{+}_n||_F \leq ||\mathbf{v}'||_F \leq 2||\mathbf{v}||_F = ||\mathbf{w}\mathbf{H}_n^{+}||_F,$$

noticing that $\bar{\Delta}_{\mathbf{H}_n} = \bar{\Delta}_{\mathbf{H}'_n}$ and $\bar{\bar{\Delta}}_{\mathbf{H}_n} = \bar{\bar{\Delta}}_{\mathbf{H}'_n}$, $\textsc{Error}_{\mathcal{K},\mathbf{H}'_n}(\mathbf{w}) \leq 2\textsc{Error}_{\mathcal{K},\mathbf{H}_n}(\mathbf{w})$.

On the other hand, $\mathbf{H}'_n$ contains two copies of queries $\mathbf{I}_n$, which is equivalent to reduce the error on those queries by a factor of 2. Noticing all other queries in $\mathbf{H}'_n$ are contained in $\mathbf{H}_n$, we have $\textsc{Error}_{\mathcal{K},\mathbf{H}_n}(\mathbf{w}) \leq 2\textsc{Error}_{\mathcal{K},\mathbf{H}'_n}(\mathbf{w})$. □

3.3.3 Error analysis for $\mathbf{I}_n$,$\mathbf{H}_n$ and $\mathbf{Y}_n$

In this part we analyze the error for two specific workloads of interest. We focus on two typical workloads: $\mathbf{W}_R$, the set of all range queries, and $\mathbf{W}_{01}$, which includes



(a) Use $\mathbf{H}_n$ as the strategy matrix (b) Use $\mathbf{Y}_n$ as the strategy matrix (c) Use $\mathbf{I}_n$ as the strategy matrix

Figure 3.4. Error of answering queries in $\mathbf{W}_R$ under the Laplace mechanism with $n = 512$, $\epsilon = 1$.

arbitrary predicate queries, since it consists of all linear 0-1 queries. Note that attempting to use either of these workloads as strategies leads to high sensitivity: the sensitivity of $\mathbf{W}_R$ is $O(n^2)$ while the sensitivity of $\mathbf{W}_{01}$ is $O(2^n)$. Here we consider the total error as well as the maximum error under the matrix mechanism, and the later one is defined as the worst case error of a single query and denoted as MaxError.

In the original papers describing $\mathbf{H}_n$ and $\mathbf{Y}_n$ [40, 70], both techniques are shown to have worst case error bounded under ϵ-differential privacy by $O(\log^3 n)$ on $\mathbf{W}_R$. Both papers resort to experimental analysis to understand the distribution of total error across the class of range queries. We note that our results allow the error for any query to be analyzed analytically.

Example 3.4. *Figure 3.4 demonstrates error of answering queries in $\mathbf{W}_R$ under the Laplace mechanism with $n = 512$ and $\epsilon = 1$ using strategy matrices $\mathbf{W}_n$, $\mathbf{Y}_n$ and $\mathbf{I}_n$, respectively.*

Next we summarize the total error and the maximum error for these strategies. The following results tighten known bounds for $\mathbf{W}_R$, and establish new bounds for $\mathbf{W}_{01}$ with the ϵ-differential privacy.

Theorem 3.2 (Error and Maximum Error)**.** *The total error and the maximum error on workloads* $\mathbf{W}_R$ *and* $\mathbf{W}_{01}$ *using strategies* $\mathbf{H}_n, \mathbf{Y}_n$*, and* $\mathbf{I}_n$ *under the* ϵ*-matrix mechanism is given by:*

TotalError	$\mathbf{H}_n$	$\mathbf{Y}_n$	$\mathbf{I}_n$
$\mathbf{W}_R$	$\Theta(n^2 \log^3 n/\epsilon^2)$	$\Theta(n^2 \log^3 n/\epsilon^2)$	$\Theta(n^3/\epsilon^2)$
$\mathbf{W}_{01}$	$\Theta(n2^n \log^2 n/\epsilon^2)$	$\Theta(n2^n \log^2 n/\epsilon^2)$	$\Theta(n2^n/\epsilon^2)$

MaxError	$\mathbf{H}_n$	$\mathbf{Y}_n$	$\mathbf{I}_n$
$\mathbf{W}_R$	$\Theta(\log^3 n/\epsilon^2)$	$\Theta(\log^3 n/\epsilon^2)$	$\Theta(n/\epsilon^2)$
$\mathbf{W}_{01}$	$\Theta(n \log^2 n/\epsilon^2)$	$\Theta(n \log^2 n/\epsilon^2)$	$\Theta(n/\epsilon^2)$

Proof. Since $\mathbf{W}_n$ and $\mathbf{H}_n$ are asymptotically equivalent, we can derive the error bounds for either. We analyze the error of $\mathbf{W}_n$. Let $n = 2^{k+1}$, consider the range query $[2^k - \frac{1}{3}(4^{\lfloor \frac{k-1}{2} \rfloor + 1} - 1), 2^k + \frac{1}{3}(4^{\lfloor \frac{k-1}{2} \rfloor + 1} - 1)]$. The error of this query is $\Theta(\log^3 n)$, which follows from algebraic manipulation of Equation 3.5, facilitated by knowing the eigen decomposition of $(\mathbf{W}_n^T \mathbf{W}_n)^+$. Since Xiao et al. [70] have already shown that the worst case error of $\mathbf{W}_n$ is $O(\log^3 n)$, we know the maximum error of answering any query in $\mathbf{W}_R$ is $\Theta(\log^3 n)$.

Moreover, it follows from algebraic manipulation that the error of answering any query $\mathbf{w}$ where the number of non-zero entries is 1 is $O(\log^2 n)$. Therefore the error of any 0-1 query is $O(n \log^2 n)$. Consider the query $(0, 1, 0, 1, \ldots, 0, 1)$: it can can be shown to have error $\Theta(n \log^2 n)$. Therefore the maximum error of answering any query in $\mathbf{W}_{01}$ is $\Theta(n \log^2 n)$.

Recall that when $\mathcal{K}$ is the Laplace mechanism,

$$\text{TotalError}_{\mathbf{A}}(\mathbf{W}) = \frac{2}{\epsilon^2} \bar{\Delta}_{\mathbf{A}}^2 ||\mathbf{W}\mathbf{A}^+||_F^2.$$

Total error of workloads $\mathbf{W}_R$, $\mathbf{W}_{01}$ can be computed by applying the equation above to strategies $\mathbf{H}_n, \mathbf{W}_n$ and $\mathbf{I}_n$. $\square$

Given a strategy matrix $\mathbf{A}$, the only differences in computing the an error of answer any query $\mathbf{w}$ with $\mathbf{A}$ under the ϵ- and (ϵ, δ)-matrix mechanism are the sensitivity and the constant $P(\mathcal{K})$. Therefore the total error and the maximum error of the strategies above under the (ϵ, δ)-differential privacycan be proved in exactly the same way as Theorem 3.2.

Theorem 3.3 (Error and Maximum Error)**.** *The total error and the maximum error on workloads $\mathbf{W}_R$ and $\mathbf{W}_{01}$ using strategies $\mathbf{H}_n, \mathbf{Y}_n,$ and $\mathbf{I}_n$ under the (ϵ, δ)-matrix mechanism is given by:*

TotalError	$\mathbf{H}_n$	$\mathbf{Y}_n$	$\mathbf{I}_n$
$\mathbf{W}_R$	$\Theta(n^2 \log^2 n \log(1/\delta)/\epsilon^2)$	$\Theta(n^2 \log^2 n \log(1/\delta)/\epsilon^2)$	$\Theta(n^3 \log(1/\delta)/\epsilon^2)$
$\mathbf{W}_{01}$	$\Theta(n 2^n \log n \log(1/\delta)/\epsilon^2)$	$\Theta(n 2^n \log n \log(1/\delta)/\epsilon^2)$	$\Theta(n 2^n \log(1/\delta)/\epsilon^2)$

MaxError	$\mathbf{H}_n$	$\mathbf{Y}_n$	$\mathbf{I}_n$
$\mathbf{W}_R$	$\Theta(\log^2 n \log(1/\delta)/\epsilon^2)$	$\Theta(\log^2 n \log(1/\delta)/\epsilon^2)$	$\Theta(n \log(1/\delta)/\epsilon^2)$
$\mathbf{W}_{01}$	$\Theta(n \log n \log(1/\delta)/\epsilon^2)$	$\Theta(n \log n \log(1/\delta)/\epsilon^2)$	$\Theta(n \log(1/\delta)/\epsilon^2)$

3.4 Optimization

As it is mentioned above, the matrix mechanism enhances the differentially private algorithm $\mathcal{K}$ by choosing a fine tuned query strategy $\mathbf{A}$. The core to the matrix mechanism is to determine an appropriate query strategy $\mathbf{A}$ for a given query workload $\mathbf{W}$. In this section, we present techniques that generate optimal or approximate strategies for given workloads under the matrix mechanism as well as a heuristic that enhances existing strategies. We first demonstrate our main problem as following.

Program 3.4.1 Minimizing the Total Error under ϵ-Differential Privacy

$$\begin{aligned}
\text{Given: } & \mathbf{W} \in \mathbb{R}^{m \times n} \\
\text{Minimize: } & u_1 + u_2 + \ldots + u_m \\
\text{Subject to: } & \text{For } i \in [m] : \mathbf{w}_i \text{ is the } i\text{-th row of } \mathbf{W}. \\
& \begin{bmatrix} \mathbf{X} & \mathbf{w}_i^T \\ \mathbf{w}_i & u_i \end{bmatrix} \geq 0 && (3.6) \\
& \bar{\Delta}_{\mathbf{A}} \leq 1 && (3.7) \\
& \operatorname{rank}\left(\begin{bmatrix} \mathbf{I}_{m'} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{X} \end{bmatrix} \right) = m' && (3.8)
\end{aligned}$$

Problem 3.1 (MinError). *Given a query workload* $\mathbf{W}$ *and a differentially private algorithm* $\mathcal{K}$*, find a query strategy* $\mathbf{A}$ *that supports* $\mathbf{W}$ *and minimizes* $\text{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W})$.

Noticing that both $\mathbf{W}$ and $-\mathbf{W}$ support $\mathbf{W}$ but $\mathbf{0} = \mathbf{W} + (-\mathbf{W})$ does not, the MinError problem is a non-convex problem. In this section, we will formulate the MinError problem as a semidefinite program with rank constraint, which is a non-convex variation of the semidefinite program. We then discuss the problem in two cases corresponding to the differential private guarantee of $\mathcal{K}$. A general technique that can be used to improve a query strategy is also provided in the later section. We also show that two semantically equivalent workloads yield the same minimum total error at the end of this section.

3.4.1 Formulating the MinError Problem

Here we show that MinError problem under ϵ-differential privacy can be expressed as a semidefinite program with rank constraints. While rank constraints make the semidefinite program non-convex, there are algorithms that can solve such problems by iteratively solving a pair of related semidefinite programs.

Theorem 3.4. *Given an* $m \times n$ *workload* $\mathbf{W}$*, Program 3.4.1 is a semidefinite program with rank constraint whose solution is the tuple* $(\mathbf{A}, \mathbf{u}, \mathbf{X})$ *and the* $m' \times n$ *strategy* $\mathbf{A}$ *minimizes* $\text{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W})$ *among all* $m' \times n$ *strategies.*

Proof. To prove that the output strategy $\mathbf{A}$ of Program 3.4.1 is an optimal $m' \times n$ strategy to the MinError problem, one need to show that the solution of Program 3.4.1 supports $\mathbf{W}$ and the optimization goal of Program 3.4.1 is equivalent with the MinError problem.

The semidefinite condition in (3.6) is important, which guarantees that there exists a matrix $\mathbf{A}'$ such that $\mathbf{A}'^T\mathbf{A}' = \mathbf{X}$, $\mathbf{A}'$ supports $\mathbf{w}_i$ and $u_i \geq ||\mathbf{w}_i\mathbf{A}'^+||_F^2$. According to the properties of positive semidefinite matrices, its a symmetric matrix with non-negative eigenvalues. Let $\mathbf{X} = \mathbf{P}\boldsymbol{\Sigma}\mathbf{P}^T$ be an eigenvalue decomposition of matrix $\mathbf{X}$. Consider matrix

$$\mathbf{Y} = \begin{bmatrix} \mathbf{P}^T & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X} & \mathbf{w}_i^T \\ \mathbf{w}_i & u_i \end{bmatrix} \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma} & (\mathbf{w}_i\mathbf{P})^T \\ \mathbf{w}_i\mathbf{P} & u_i \end{bmatrix},$$

(3.6) holds if and only if $\mathbf{Y} \geq 0$. Since $\boldsymbol{\Sigma}$ is a diagonal matrix, if its j^{th} diagonal entry is 0, the j^{th} entry of $\mathbf{w}_i\mathbf{P}$ must be 0 as well. Otherwise, $\mathbf{Y}$ can not be positive semidefinite no matter what value u_i is. Recall the diagonal entries of $\boldsymbol{\Sigma}$ are eigenvalues of $\mathbf{X}$ and hence be non-negative. Let $\mathbf{D}$ be the diagonal matrix whose diagonal entries are the square roots of diagonal entries of $\boldsymbol{\Sigma}$. We know $\mathbf{D}$ supports $\mathbf{w}_i\mathbf{P}$. Then $\mathbf{A}' = \mathbf{D}\mathbf{P}^T$ supports $\mathbf{w}_i$ and $\mathbf{A}'^T\mathbf{A}' = \mathbf{X}$. In addition, let $\mathbf{Y}'$ be the matrix that is constructed by removing all 0 columns and rows from $\mathbf{Y}$. For any $\mathbf{w}_i$ that is supported by $\mathbf{A}'$, $\mathbf{Y} \geq 0$ is equivalent to $|\mathbf{Y}'| \geq 0$. The expansion of $|\mathbf{Y}'|$ implies that the determinant non-negative if and only if $u_i \geq ||\mathbf{w}_i\mathbf{A}'^+||_F^2$. Since the goal of the optimization problem is to minimize the sum of u_i, when the optimal case is achieved, we must have $u_i = ||\mathbf{w}_i\mathbf{A}'^+||_F^2 = \mathbf{w}_i(\mathbf{A}'^T\mathbf{A}')^+\mathbf{w}_i^T = \mathbf{w}_i\mathbf{X}^+\mathbf{w}_i^T$. Furthermore, (3.8) guarantees that $\mathbf{X} = \mathbf{A}^T\mathbf{A}$, and hence $u_i = ||\mathbf{w}_i\mathbf{A}^+||_F^2$.

According to Proposition 3.7, apply a non-zero scalar c to a query strategy $\mathbf{A}$ leads to its equivalent strategy. Therefore the condition (3.7) does not limit the scope of query strategies to be considered since any query strategy has equivalent strategies with sensitivity no more than 1. This constraint is as well a convex constraint since the sensitivity of $\mathbf{A}$ is convex for both ϵ- and (ϵ, δ)-differential privacy. Noticing $||\mathbf{w}_i(c \cdot \mathbf{A})^+||_F = ||\mathbf{w}_i\mathbf{A}^+||_F/c < ||\mathbf{w}_i\mathbf{A}^+||_F$ for any $c > 1$, $\bar{\Delta}_{\mathbf{A}}$ must be 1 in the optimal case.

Above all, any solution to Program 3.4.1 supports $\mathbf{W}$. When the optimal case is achieved, $\bar{\Delta}_{\mathbf{A}} = 1$ and $u_i = ||\mathbf{w}_i\mathbf{A}^+||_F^2 = \bar{\Delta}_{\mathbf{A}}^2||\mathbf{w}_i\mathbf{A}^+||_F^2 = P(\mathcal{K})\textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{w}_i)$. Therefore the goal of the optimization, minimizing $\sum_{i=1}^m u_i$, is equivalent to minimizing $\sum_{i=1}^m \textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{w}_i) = \textsc{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W})$. □

Theorem 3.4 provides the best strategy to the MinError problem with at most m' queries. If the optimal strategy has $m'' < m'$ queries, then Program 3.4.1 will return an $m' \times n$ matrix with $m' - m''$ rows of 0s. In addition, if the workload only contains queries with coefficients in $\{-1, 0, 1\}$, we can show that n^2 is upper bound on the number of queries in the optimal strategy [45].

In addition, since Program 3.4.1 encodes the error of each query $\mathbf{w}_i$ in query workload $\mathbf{W}$, we can actually use other convex function of $u_1, \ldots, u_m$ to take the place of $u_1 + \ldots + u_m$ in the optimization goal. One variation to the optimization goal is $\max_i u_i$, under which the result from Program 3.4.1 becomes the query strategy that minimizes the maximum error of all queries in $\mathbf{W}$.

Dattorro [18] shows that solving a semidefinite program with rank constraints can be converted into solving two semidefinite programs iteratively. The convergence follows the widely used trace heuristic for rank minimization. We are not aware of results that quantify the number of iterations that are required for convergence. However, notice it takes $O(n^4)$ time to solve a semidefinite program with an $n \times n$ semidefinite constraint matrix and in Program 3.4.1, there are m semidefinite constraint matrices with size $m + n$, which can be represented as a semidefinite constraint matrix with size $m(m + n)$. Thus, the complexity of solving our semidefinite program with rank constraints is at least $O(m^4(m + n)^4)$.

Program 3.4.2 Minimizing the Total Error with $\mathcal{K}$ under (ϵ, δ)-differential privacy

$$\begin{aligned}
\text{Given: } & \mathbf{W} \in \mathbb{R}^{m \times n}. \\
\text{Minimize: } & u_1 + u_2 + \ldots + u_m. \\
\text{Subject to: } & \text{For } i \in [m] : \mathbf{w}_i \text{ is the } i\text{-th row of } \mathbf{W}. \\
& \begin{bmatrix} \mathbf{X} & \mathbf{w}_i^T \\ \mathbf{w}_i & u_i \end{bmatrix} \geq 0 \\
& \mathbf{X}_{ii} \leq 1, \quad i \in [n].
\end{aligned}$$

The difficulty of Program 3.4.1 comes from the rank constraint (3.8), which is used to connect $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}$ since we need $\mathbf{A}^T\mathbf{A}$ to compute $\|\mathbf{w}_i\mathbf{A}^+\|_F^2$ and $\mathbf{A}$ to compute $\bar{\Delta}_{\mathbf{A}}$. However, when $\mathcal{K}$ bases on (ϵ, δ)-differential privacy, $\bar{\bar{\Delta}}_{\mathbf{A}}$ can be computed directly from $\mathbf{A}^T\mathbf{A}$. In such case, $\mathbf{A}$ is not necessary in the optimization problem and the rank constraint can be removed. The optimization problem can then be reduced to a semidefinite program, which can be solved in polynomial time.

Theorem 3.5. *Given an $m \times n$ workload $\mathbf{W}$, Program 3.4.2 is a semidefinite program whose solution is the tuple $(\mathbf{X}, \mathbf{u})$ and any $m' \times n$ strategy $\mathbf{A}$ such that $\mathbf{X} = \mathbf{A}^T\mathbf{A}$ minimizes* $\text{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W})$ *among all strategies.*

3.4.2 Approximation to the MinError Problem under ϵ-Differential Privacy

Though the MinError problem can be formulated as Program 3.4.1. Solving the optimization problem under ϵ-differential privacy is intractable. Hence, we demonstrate two heuristics that can be used to approximate the solution to the MinError problem. Both of the approaches give a bound to the sensitivity so that the rank constraint can then be removed from the optimization formulation.

3.4.2.1 L_2 approximation

The first idea is straightforward. Recall that under (ϵ, δ)-differential privacy, sensitivity can be computed from $\mathbf{A}^T\mathbf{A}$ directly, which leads to a simpler optimization problem, Program 3.4.2. If we can bound $\bar{\Delta}_{\mathbf{A}}$ with $\bar{\bar{\Delta}}_{\mathbf{A}}$, we can take advantage of the semidefinite programming under (ϵ, δ)-differential privacy.

According to the basic property of L norms, for any vector $\mathbf{u}$ of dimension n, $\|\mathbf{u}\|_2 \le \|\mathbf{u}\|_1 \le \sqrt{n}\|\mathbf{u}\|_2$. Therefore we can bound the approximation rate of using Program 3.4.2.

Theorem 3.6. *Given a workload* $\mathbf{W}$*, let* $\mathbf{A}$ *be the optimal solution given by Program 3.4.1 and* $\mathbf{A}'$ *be an optimal solution given by Program 3.4.2. Then with any differentially private algorithm* $\mathcal{K}$ *under the* ϵ*-matrix mechanism,*

$$\textsc{TotalError}_{\mathcal{K},\mathbf{A}'}(\mathbf{W}) \le n\textsc{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W}).$$

3.4.2.2 Singular value bound approximation

We can also bound $\bar{\Delta}_{\mathbf{A}}$ using its geometric properties. Recall that the matrix $\mathbf{A}$ can be reperesented by its singular value decomposition $\mathbf{A} = \mathbf{Q}_{\mathbf{A}}\mathbf{D}_{\mathbf{A}}\mathbf{P}_{\mathbf{A}}^T$. Let us consider the geometry explanation of $\bar{\Delta}_{\mathbf{A}}$, which is the radius of the minimum L_1 ball that cover all column vectors of $\mathbf{A}$. In such case, the column vectors of $\mathbf{A}$ lay on ellipsoid

$$\psi_{\mathbf{A}} : \mathbf{x}^T(\mathbf{A}\mathbf{A}^T)^+\mathbf{x} = 1.$$

Let $\bar{\Delta}_{\psi_{\mathbf{A}}}$ denote radius of the minimum L_1 ball that covers the ellipsoid $\psi_{\mathbf{A}}$. Observe that all column vectors of $\mathbf{A}$ are contained in $\psi_{\mathbf{A}}$, which indicates $\bar{\Delta}_{\mathbf{A}} \le \bar{\Delta}_{\psi_{\mathbf{A}}}$. The minimum sensitivity that can be achieved by the strategies that are profile equivalent to $\mathbf{A}$ can be bounded by:

$$\min_{\mathbf{B}:\mathbf{B}^T\mathbf{B}=\mathbf{A}^T\mathbf{A}} \bar{\Delta}_{\mathbf{B}} \le \min_{\mathbf{B}:\mathbf{B}^T\mathbf{B}=\mathbf{A}^T\mathbf{A}} \bar{\Delta}_{\psi_{\mathbf{B}}}.$$

The query strategy $\mathbf{B}$ that is profile equivalent to $\mathbf{A}$ and has the minimum $\bar{\Delta}_{\psi_{\mathbf{B}}}$ is given by the theorem below.

Theorem 3.7. *Given a strategy matrix* $\mathbf{A}$ *with singular value decomposition* $\mathbf{A} = \mathbf{Q}_{\mathbf{A}}\mathbf{D}_{\mathbf{A}}\mathbf{P}_{\mathbf{A}}^T$.

$$\begin{aligned} \operatorname*{argmin}_{\mathbf{B}:\mathbf{B}^T\mathbf{B}=\mathbf{A}^T\mathbf{A}} \bar{\Delta}_{\psi_{\mathbf{B}}} &= \mathbf{D}_{\mathbf{A}}\mathbf{P}_{\mathbf{A}}^T, \\ \min_{\mathbf{B}:\mathbf{B}^t\mathbf{B}=\mathbf{A}^t\mathbf{A}} \bar{\Delta}_{\psi_{\mathbf{B}}} &= \|\mathbf{A}\|_F \le \sqrt{n}\bar{\Delta}_{\mathbf{A}}. \end{aligned} \tag{3.9}$$

To prove of Theorem 3.7, tangent hyperplanes of $\psi_{\mathbf{A}}$ need to be formulated, as it is in the following lemma.

Lemma 1. *Given an ellipsoid defined by* $\mathbf{x}^T\mathbf{\Psi}\mathbf{x} = 1$ *and a vector* $\mathbf{u}$, $\mathbf{u}^T\mathbf{w} = \sqrt{\mathbf{u}^T\mathbf{\Psi}^+\mathbf{u}}$ *is a tangent hyperplane of the ellipsoid.*

Proof. For any point $\mathbf{y}$ on the ellipsoid, the tangent hyperplane of the ellipsoid on $\mathbf{y}$ is $\mathbf{y}^T\mathbf{\Psi}\mathbf{x} = 1$. Consider a tangent hyperplane of the ellipsoid: $\mathbf{u}^T\mathbf{x} = c$, where c is an unknown constant, and there exists a point $\mathbf{x}_0$ on the ellipsoid such that $\mathbf{x}_0^T\mathbf{\Psi} = \frac{\mathbf{u}^T}{c}$. Therefore $\mathbf{x}_0 = \frac{\mathbf{\Psi}^+\mathbf{u}}{c}$. Noticing $\mathbf{x}_0^T\mathbf{\Psi}\mathbf{x}_0 = 1$, we know

$$1 = \mathbf{x}_0^T\mathbf{\Psi}\mathbf{x}_0 = (\frac{1}{c}\mathbf{u}^T\mathbf{\Psi}^+)\mathbf{\Psi}(\frac{1}{c}\mathbf{\Psi}^+\mathbf{u}) = \frac{1}{c^2}\mathbf{u}^T\mathbf{\Psi}^+\mathbf{u}.$$

Therefore $c = \sqrt{\mathbf{u}^T\mathbf{\Psi}^+\mathbf{u}}$. □

Proof. to Theorem. 3.7: According to the definition of sensitivity under ϵ-differential privacy, for any strategy $\mathbf{B}$, the ellipsoid $\psi_{\mathbf{B}}$ must tangent with the diamond $\|\mathbf{x}\|_1 = \bar{\Delta}_{\psi_{\mathbf{B}}}$. With out lose of generality, let us assume it is tangent to the hyperplane $(1,\ldots,1)\mathbf{x} = \bar{\Delta}_{\psi_{\mathbf{B}}}$ and $(a_1,\ldots,a_n)\mathbf{x} \le \bar{\Delta}_{\psi_{\mathbf{B}}}$, here $a_i = 1,-1$. Let $\mathbf{B} = \mathbf{Q}_{\mathbf{B}}\mathbf{D}_{\mathbf{A}}\mathbf{P}_{\mathbf{A}}^T$ be the singular value decomposition of $\mathbf{B}$ and let $\mathbf{\Psi'} = (\mathbf{B}\mathbf{B}^T)^+ = \{\psi'_{ij}\}$ to simplify the notation. According to Lemma 1,

$$(1,\ldots,1)\mathbf{\Psi'}(1,\ldots,1)^T \geq (a_1,\ldots,a_n)\mathbf{\Psi'}(a_1,\ldots,a_n)^T.$$

In particular,

$$(1,\ldots,1)\mathbf{\Psi'}(1,\ldots,1)^T \geq (-1,1,1,\ldots,1)\mathbf{\Psi'}(-1,1,1,\ldots,1)^T,$$

which means $\psi'_{12} + \psi'_{13} + \ldots + \psi'_{1n} = \sum_{i=1}^n \psi'_{1i} - \psi'_{11} \geq 0$. Similarly, we can show for any j we have $\sum_{i=1}^n \psi'_{ji} - \psi'_{jj} \geq 0$. Therefore

$$\begin{aligned}(1,\ldots,1)\mathbf{\Psi'}(1,\ldots,1)^T &= \sum_i\sum_j \psi'_{ij} = \sum_j \psi'_{jj} + \sum_j(\sum_i \psi'_{ji} - \psi'_{jj})\\ \geq \sum_j \psi'_{jj} &= \text{trace}(\mathbf{\Psi'}) = \|(\mathbf{B}\mathbf{B}^T)^+\|_F^2 = \|(\mathbf{B}^T\mathbf{B})^+\|_F^2 = \|(\mathbf{D}_\mathbf{A}^T\mathbf{D}_A)^+\|_F^2\end{aligned}$$

The equal sign can be achieved when $\psi'_{ij} = 0$ for all $i \neq j$, which means $\mathbf{\Psi'}$ is a diagonal matrix and then $\mathbf{Q}_\mathbf{B} = \mathbf{I}$. Therefore,

$$\begin{aligned}\mathbf{B} &= \mathbf{D}_\mathbf{A}\mathbf{P}_\mathbf{A}^T\\ \bar{\Delta}_{\psi_\mathbf{B}} &= \sqrt{(1,\ldots,1)\mathbf{\Psi'}^+(1,\ldots,1)^T} = \sqrt{(1,\ldots,1)\mathbf{D}_\mathbf{A}\mathbf{D}_\mathbf{A}^T(1,\ldots,1)^T}\\ &= \sqrt{\text{trace}(\mathbf{A}^T\mathbf{A})} = \|\mathbf{A}\|_F \leq \sqrt{n}\bar{\bar{\Delta}}_\mathbf{A} \leq \sqrt{n}\bar{\Delta}_\mathbf{A}\end{aligned}$$

□

Using the singular value bound in Theorem 3.7 to substitute for the L_1 sensitivity, the minError problem can be converted to the following approximation problem.

Problem 3.2 (Singular value bound approximation). *Given a workload matrix* $\mathbf{W}$*, find the strategy* $\mathbf{A}$ *that minimizes* $\|\mathbf{A}\|_F^2\|\mathbf{W}\mathbf{A}^+\|_F^2$.

The singular value bound approximation has a closed-form solution.

Theorem 3.8. *Let* $\mathbf{W}$ *be the workload matrix with singular value decomposition* $\mathbf{W} = \mathbf{Q_W D_W P_W^T}$ *and* $\xi_1, \dots, \xi_n$ *be its singular values. The optimal solution to the singular value approximation* $\mathbf{A} = c \cdot \mathbf{Q_A D_A P_A}$ *satisfies* $\mathbf{P_A} = \mathbf{P_W}$ *and* $\mathbf{D_A} = diag(\sqrt{\xi_1}, \dots, \sqrt{\xi_n})$.

Proof. Recall the optimization goal of Problem 3.2:

$$||\mathbf{A}||_F^2 ||\mathbf{WA}^+||_F^2 = ||\mathbf{D_A}||_F^2 ||\mathbf{D_W P_W P_A^T D_A^+}||_F^2 \geq \sum_{i=1}^{n} ||\mathbf{Dp}_i||_2^2 \geq ||\mathbf{D}||_F^2,$$

Here $\mathbf{p}_i$ is the i^{th} column of matrix $\mathbf{P_W P_A^T}$ and two inequality signs base on Cauchy-Schwardz inequality. In order to have the equal sign satisfied in the first the inequalities, we need the diagonal entries $\mathbf{D_A^2}$ to be equal to the diagonal entries of $c \cdot \mathbf{D_W P_W P_A^T}$ for a constant c. In addition, $\mathbf{P_W P_A^T}$ must be $\mathbf{I}$ to make the equal sign of the second inequality. Since $\mathbf{P_W}$ is an orthogonal matrix, $\mathbf{P_A} = \mathbf{P_W}$. Then we have $\mathbf{D_A^2} = \mathbf{D_W}$ and $\mathbf{D_A} = diag(\sqrt{\xi_1}, \dots, \sqrt{\xi_n})$, where $\xi_1, \dots, \xi_n$ are singular values of $\mathbf{W}$. □

The solution of $\mathbf{A}$ in Theorem 3.8 is very similar to the idea of matching the shape of $\mathbf{W}$, which is equivalent to have $\mathbf{P_A}$ to $\mathbf{P_W}$ and $\mathbf{D_A}$ be $\mathbf{D_W}$. Here we use a slightly different $\mathbf{D_A}$ so as to find a balance between the shape of $\mathbf{A}$ and $\bar{\Delta}_\mathbf{A}$, which also provides an guaranteed error bound based on Theorem 3.7.

Theorem 3.9. *Given a differentially private algorithm* $\mathcal{K}$ *over* ϵ*-differential privacy and a workload* $\mathbf{W}$*, let* $\mathbf{A}$ *be the optimal solution to the minError problem over* $\mathcal{K}$ *and* $\mathbf{A}'$ *be the optimal solution to the singular value bound approximation. Then*

$$\textsc{TotalError}_{\mathcal{K},\mathbf{A}'}(\mathbf{W}) \leq n \textsc{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W}).$$

3.4.2.3 Minimizing sensitivity under ϵ-differential privacy

Above we presented two approximation approaches to produce heuristic query strategies under ϵ-differential privacy. Noticing both of the approaches bound $\bar{\Delta}_{\mathbf{A}}$ with some properties that can be determined by $\mathbf{A}^T\mathbf{A}$: the L_2 approximation relies on $\bar{\bar{\Delta}}_{\mathbf{A}}$, which is the square root of the largest diagonal entry of $\mathbf{A}^T\mathbf{A}$, and the singular value bound uses the squares of singular values of $\mathbf{A}$, which are the eigenvalues of $\mathbf{A}^T\mathbf{A}$. Therefore, each of those approximations only determine an error profile and any query strategies with this error profile satisfies the approximation condition. A further refinement to those approximations is to find the query strategy with minimized sensitivity among all query strategies that have the given error profile.

Problem 3.3. *Given a error profile* $\mathbf{M}$*, find the query matrix* $\mathbf{A}$ *whose error profile is* $\mathbf{M}$ *and has the minimum sensitivity under* ϵ*-differential privacy.*

Besides refining the result of the approximations as above, Problem 3.3 is also important in case that the user has certain error distribution in mind that specifies an error profile. Unfortunately, similar to the MinError problem, Problem 3.3 is non-convex as well. Problem 3.3 can also be formulated as a semidefinite program with rank constraint, as stated below.

Theorem 3.10. *Given an error profile* $\mathbf{M}$*, Program 3.4.3 is a semidefinite program with rank constraint that outputs an* $m \times n$ *matrix* $\mathbf{A}$ *such that* $(\mathbf{A}^T\mathbf{A})^+ = \mathbf{M}$ *with* $\bar{\Delta}_{\mathbf{A}}$ *minimized.*

3.4.3 Augmentation Heuristic

We formalize below the following intuition that applies to the matrix mechanism: as far as the error profile is concerned, additional noisy query answers can never detract from query accuracy as they must have some information content useful to one or more queries. Therefore if $\mathbf{A}'$ is a query strategy obtained by augmenting the query strategy $\mathbf{A}$ with additional rows, $||\mathbf{W}\mathbf{A}'^+||_F \leq ||\mathbf{W}\mathbf{A}'^+||_F$.

Program 3.4.3 Minimizing the sensitivity under ϵ-differential privacy

$$\begin{aligned} \text{Given: } & \mathbf{M} \in \mathbb{R}^{n \times n}. \\ \text{Minimize: } & r. \\ \text{Subject to: } & \bar{\Delta}_{\mathbf{A}} \le r; \\ & \text{rank}\left(\begin{bmatrix} \mathbf{I}_n & \mathbf{A} \\ \mathbf{A}^T & \mathbf{M}^+ \end{bmatrix} \right) = n. \end{aligned}$$

Theorem 3.11. (Augmenting a strategy) *Let* $\mathbf{A}$ *be a query strategy and consider a new strategy* $\mathbf{A}'$ *obtained from* $\mathbf{A}$ *by adding the additional rows of strategy* $\mathbf{B}$*, so that* $\mathbf{A}' = \left[\begin{smallmatrix} \mathbf{A} \\ \mathbf{B} \end{smallmatrix} \right]$*. For any workload* $\mathbf{W}$ *that* $\mathbf{A}$ *supports,* $\mathbf{A}'$ *supports* $\mathbf{W}$ *and*

$$||\mathbf{W}\mathbf{A}'^+||_F \le ||\mathbf{W}\mathbf{A}^+||_F.$$

Further, $||\mathbf{W}\mathbf{A}'^+||_F = ||\mathbf{W}\mathbf{A}^+||_F$ *if and only if* $\mathbf{W}\mathbf{A}'^+ = \left[\begin{smallmatrix} \mathbf{W}\mathbf{A}^+ \\ \mathbf{0} \end{smallmatrix} \right]$.

Proof. Since $\mathbf{A}$ supports $\mathbf{W}$, $\mathbf{A}'$ supports $\mathbf{W}$ as well. Noticing padding $\mathbf{W}\mathbf{A}^+$ with some 0s gives a solution to equation $\mathbf{X}\mathbf{A}' = \mathbf{W}$, according to Proposition 2.8, $||\mathbf{W}\mathbf{A}'^+||_F \le ||\mathbf{W}\mathbf{A}^+||_F$.

Let $\mathbf{w}_1, \dots, \mathbf{w}_m$ be rows of $\mathbf{W}$. Noticing that

$$\begin{aligned} ||\mathbf{W}\mathbf{A}^+||_F &= \sum_{i=1}^m ||\mathbf{w}_i\mathbf{A}^+||_F; \\ ||\mathbf{W}\mathbf{A}'^+||_F &= \sum_{i=1}^m ||\mathbf{w}_i\mathbf{A}'^+||_F; \\ ||\mathbf{w}_i\mathbf{A}'^+||_F &\le ||\mathbf{w}_i\mathbf{A}^+||_F, \quad i = 1, \dots, m. \end{aligned}$$

Therefore $||\mathbf{W}\mathbf{A}'^+||_F = ||\mathbf{W}\mathbf{A}^+||_F$ if and only if $||\mathbf{w}_i\mathbf{A}'^+||_F = ||\mathbf{w}_i\mathbf{A}^+||_F$ for all $i = 1, \dots, m$. Thus it is sufficient to consider the condition that $||\mathbf{w}\mathbf{A}'^+||_F = ||\mathbf{w}\mathbf{A}^+||_F$ for a single query $\mathbf{w}$ that $\mathbf{A}$ supports.

Given two distinct solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ to equation $\mathbf{x}\mathbf{A}' = \mathbf{w}$. If $||\mathbf{x}_1||_F = ||\mathbf{x}_2||_F$, noticing Frobenius norm is convex, we have $||(\mathbf{x}_1 + \mathbf{x}_2)/2||_F < ||\mathbf{x}_1||_F = ||\mathbf{x}_2||_F$. Since

$(\mathbf{x}_1 + \mathbf{x}_2)/2$ is also a solution to equation $\mathbf{x}\mathbf{A}' = \mathbf{w}$. Therefore the solution of $\mathbf{x}\mathbf{A}' = \mathbf{w}$ with minimized Frobenius norm is unique and $||\mathbf{w}\mathbf{A}'^{+}||_F = ||\mathbf{w}\mathbf{A}^{+}||_F$ if and only if $\mathbf{w}\mathbf{A}'^{+}$ is equal to $\mathbf{w}\mathbf{A}^{+}$ padding with 0s. □

This improvement in the error profile may have a cost—namely, augmenting $\mathbf{A}$ with strategy $\mathbf{B}$ may lead to a strategy $\mathbf{A}'$ with greater sensitivity than $\mathbf{A}$. A heuristic that follows from Theorem 3.11 is to augment strategy $\mathbf{A}$ only by completing deficient columns, that is, by adding rows with non-zero entries only in columns whose absolute column sums are less the sensitivity of $\mathbf{A}$. In this case the augmentation does not increase sensitivity and is guaranteed to strictly improve accuracy for any query with a non-zero coefficient in an augmented column.

Our techniques could also be used to reason formally about augmentations that do incur a sensitivity cost. We leave this as future work, as it is relevant primarily to an interactive differentially private mechanism which is not our focus here.

3.4.4 The MinError Problem over Semantic Equivalent Workloads

Intuitively, since semantic equivalent workloads only differ in their representations, answering them should introduce exactly the same amount of error. Here we formally prove that this intuition also holds under the matrix mechanism.

Theorem 3.12. *Given a workload $\mathbf{W}_1$ over a list of cell conditions Φ_1 and a workload $\mathbf{W}_2$ and a list of cell conditions Φ_2 such that $(\mathbf{W}_1, \Phi_1) = (\mathbf{W}_2, \Phi_2)$, $\min_{\mathbf{A}} \textsc{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W}_1) = \min_{\mathbf{A}} \textsc{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W}_2)$ for any differentially private algorithm $\mathcal{K}$.*

Proof. By symmetry, it is sufficient to prove that for any strategy $\mathbf{A}_1$ that supports $\mathbf{W}_1$, there exists a strategy $\mathbf{A}_2$ such that $\mathbf{A}_2$ supports $\mathbf{W}_2$ and $\textsc{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1) \geq \textsc{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2)$. As it is described in Proposition 2.1, there are five operations to generate semantic equivalent workloads. Then we can prove that such $\mathbf{A}_2$ exists for each of the five operations.

If $\mathbf{W}_2$ can be get by permute the column of $\mathbf{W}_1$, there exists permutation matrices $\mathbf{P}$ and $\mathbf{Q}$ such that $\mathbf{W}_2 = \mathbf{P}\mathbf{W}_1\mathbf{Q}$. Then $\mathbf{A}_1\mathbf{Q}$ supports $\mathbf{W}_2$ and $\textsc{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1) = \textsc{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2)$ for any $\mathcal{K}$.

If $\mathbf{W}_2$ can be get by split one of the columns of $\mathbf{W}_1$, without loss of generality, we assume that $\mathbf{W}_2$ is generated by split the last column of $\mathbf{W}_1$ in to two columns. Let $\mathbf{A}_2$ be the matrix that is generated by applying the same split to the strategy matrix $\mathbf{A}_1$. Then it is clear that $\mathbf{W}_1\mathbf{A}_1^+\mathbf{A}_2 = \mathbf{W}_2$.Therefore $\mathbf{A}_2$ supports $\mathbf{W}_2$ and $\textsc{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1) = \textsc{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2)$ for any $\mathcal{K}$.

If $\mathbf{W}_2$ can be get by combine two of the columns of $\mathbf{W}_1$ with the same entries, we assume that the last two columns of $\mathbf{W}_1$ have the same entries and removing one of them gives us $\mathbf{W}_2$. Let $\mathbf{A}_2$ be the matrix that is generated by removing the last column of $\mathbf{A}_2$. Then $\mathbf{W}_1\mathbf{A}_1^+\mathbf{A}_2 = \mathbf{W}_2$ and $\mathbf{A}_2$ hence supports $\mathbf{W}_2$. Noticing that $\Delta_{\mathbf{A}_2} \leq \Delta_{\mathbf{A}_1}$ for any $\mathcal{K}$, $\textsc{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1) \geq \textsc{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2)$ for any $\mathcal{K}$.

If $\mathbf{W}_2$ can be get by add columns of 0s to $\mathbf{W}_1$, let $\mathbf{A}_2$ be the matrix that is generated by adding corresponding columns of 0s to $\mathbf{A}_1$. Then $\mathbf{W}_1\mathbf{A}_1^+\mathbf{A}_2 = \mathbf{W}_2$. Thus $\mathbf{A}_2$ supports $\mathbf{W}_2$ and $\textsc{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1) = \textsc{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2)$ for any $\mathcal{K}$.

If $\mathbf{W}_2$ can be get by remove columns of 0s to $\mathbf{W}_1$, let $\mathbf{A}_2$ be the matrix that is generated by removing corresponding columns of 0s to $\mathbf{A}_1$. Then $\mathbf{W}_1\mathbf{A}_1^+\mathbf{A}_2 = \mathbf{W}_2$ and $\mathbf{A}_2$ hence supports $\mathbf{W}_2$. Noticing that $\Delta_{\mathbf{A}_2} \leq \Delta_{\mathbf{A}_1}$ for any $\mathcal{K}$, $\textsc{TotalError}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1) \geq \textsc{TotalError}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2)$ for any $\mathcal{K}$. □

Noticing that we actually do not consider cell conditions during the proof of Theorem 3.12, two workloads $\mathbf{W}_1$ and $\mathbf{W}_2$ have the minimum total error if they can be converted to each other with the matrix operations mentioned in Proposition 2.1.

3.5 The matrix mechanism with non-negativity constraints

As stated in Section 3.1, we only consider the matrix mechanism $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ with data independent $\mathcal{K}$ so that $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ is independent to the data vector $\mathbf{x}$ as well. Consequently, $\mathcal{M}_{\mathcal{K},\mathbf{A}}$ has no constraints on its estimated answer to any workload $\mathbf{W}$. Since the entries of data vector $\mathbf{x}$ count the number of tuples that satisfies certain cell conditions and hence cannot be negative, an answer to $\mathbf{W}$ is valid only if there exists a non-negative data vector $\hat{\mathbf{x}}$ such that $\mathbf{W}\hat{\mathbf{x}}$ agrees with the given answer.

Using the matrix mechanism directly cannot guarantee the non-negative constraint on $\hat{\mathbf{x}}$. In this section, we discuss enhancements to the matrix mechanism to cope with non-negative constraints. We consider the approach of rounding entries up to 0, non-negative least square estimation as well as a hybrid approach that takes the advantage of both methods. In addition, we also include some experimental comparison between those method over real world databases.

3.5.1 The Rounding Up Approach

Given a query strategy $\mathbf{A}$ with answer $\hat{\mathbf{y}}$, recall that the derived solution to a workload $\mathbf{W}$ supported by $\mathbf{A}$ is $\mathbf{W}\mathbf{A}^{+}\hat{\mathbf{y}}$. This answer further implies that there is an estimate $\hat{\mathbf{x}}$ to the underlying data vector such that $\mathbf{W}\hat{\mathbf{x}} = \mathbf{W}\mathbf{A}^{+}\hat{\mathbf{y}}$. It is clear that $\mathbf{A}^{+}\hat{\mathbf{y}}$ is one possible solution to $\hat{\mathbf{x}}$.

With an estimate of the underlying data set $\hat{\mathbf{x}}$, the most straightforward method to guarantee the non-negativity is to round up the negative results to 0, which can be applied to either $\mathbf{W}\mathbf{A}^{+}\hat{\mathbf{y}}$ or $\mathbf{A}^{+}\hat{\mathbf{y}}$. Rounding up the negative entries in $\mathbf{A}^{+}\hat{\mathbf{y}}$ actually gives an non-negative estimate to the underlying data vector, which can then be used to compute the answer to the query workload $\mathbf{W}$. However, by rounding up $\mathbf{A}^{+}\hat{\mathbf{y}}$ we lose the information contained in the negative estimates of positive query answers. For example, if an noisy estimate to a non-negative count x_1 is -10, there is a higher probability for x_1 to be 0 then the another non-negative count x_2 with a noisy estimate to 0. Such distinction has been lost if both x_1 and x_2 are rounded up to 0.

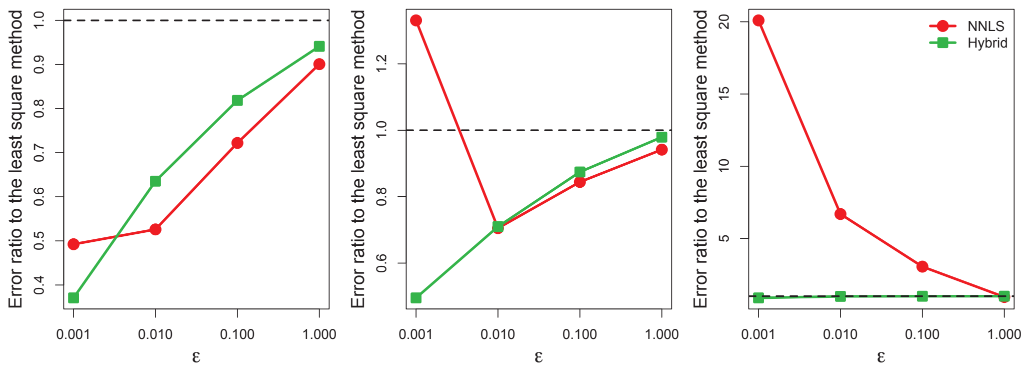
Since rounding leads to information loss, to ensure that most information is preserved, we should round up the result at the very last step, which means to round up $\mathbf{W}\mathbf{A}^{+}\hat{\mathbf{y}}$. The limitation to this approach is that it only works on queries whose entries are either all non-negative or non-positive. In addition, even if we round the answers to those queries to 0, there is no guarantee that there exists a non-negative data vector $\hat{\mathbf{x}}$ such that $\mathbf{W}\hat{\mathbf{x}}$ is equal our answer to $\mathbf{W}$. In order to deal this problem, we can publish the α-confidence intervals to each query in $\mathbf{W}$ along with their answers. For queries with all non-negative or non-positive queries, their confidence interval can also be rounded up. Then with a probability of at least α, the true answer to $\mathbf{W}$ is contained in those confidence intervals.

3.5.2 Non-negative Least Square and a Hybrid Approach

Another approach to satisfy the non-negativity constraint is to add the constraint into the process of deriving the answer to $\mathbf{W}$ from the answer to $\mathbf{A}$. Such a constraint can be achieved by considering non-negative least square estimator, a special constrained form of least square estimator to compute a non-negative estimate $\hat{\mathbf{x}}$ to the data vector and use it to generate the answer to $\mathbf{W}$, whose optimization formulation is as following.

$$\begin{aligned} \text{Given: } & \mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{y} \in \mathbb{R}^{m}. \\ \text{Minimize: } & ||\mathbf{A}\mathbf{x} - \mathbf{y}||_2^2. \\ \text{Subject to: } & \mathbf{x} \geq 0. \end{aligned}$$

Though intuitively the non-negative least square estimator should provide a better estimate towards the true answer to $\mathbf{W}$, it still works poorly when the query strategy $\mathbf{A}$ contains non-overlapping queries. The extreme case is that when $\mathbf{A} = \mathbf{I}_n$, using non-negative least square estimator is equivalent to rounding up $\hat{\mathbf{x}}$, which, as stated above, loss information and leads to a bad estimator to answer to $\mathbf{W}$. Furthermore,



(a) Use $\mathbf{H}_n$ as the strategy matrix (b) Use $\mathbf{Y}_n$ as the strategy matrix (c) Use $\mathbf{I}_n$ as the strategy matrix

Figure 3.5. Error comparison between the non-negative least square estimator and the hybrid estimator.

computing the α-confidence interval for a non-negative least square estimator is not as straightforward as it is for a least square estimator. [64] demonstrated an algorithm to compute the α-confidence interval for a non-negative least square estimator, but it is computationally more complicated and will produce larger confidence interval than for a least square estimator.

In order to address the weakness of the non-negative least square estimator, recall that we defer the round up before the output and we can apply the same method to the non-negative least square estimator. Here we proposed a hybrid approach that combines the least square and non-negative least square method. The major idea that instead of applying non-negative least square to estimate an $\hat{\mathbf{x}}$ from $\hat{\mathbf{y}}$, we first estimate the answer to $\mathbf{W}$ by $\mathbf{W}\mathbf{A}^{+}\hat{\mathbf{y}}$, the least square method and then solve a non-negative least square on $\mathbf{W}\mathbf{A}^{+}\hat{\mathbf{y}}$ to solve an $\hat{\mathbf{x}}$. The answer to $\mathbf{W}$ can then be computed with $\hat{\mathbf{x}}$. $\mathbf{W}\hat{\mathbf{x}}$ and the α-confidence interval of the least square estimator can be send back to user for further analysis.

3.5.3 Experimental Results

Here we compare the effect of different estimator experimentally. The experiment includes the least square estimator, the non-negative least squares (NNLS) estimator and the hybrid estimator and uses the Adult[8] dataset aggregated to 512 cells. Three different strategies are considered: the hierarchical ($\mathbf{H}_n$), the wavelet ($\mathbf{Y}_n$) and the identity ($\mathbf{I}_n$) strategies. The results are shown in Figure 3.5, where the error is reported as the ratio to the total squared error using the least square estimator. The $\mathcal{K}$ is set to be the Laplace mechanism and ϵ is varied from 0.001 to 1.

The figure demonstrates that, the hybrid estimator has comparable error as the non-negative least square estimator when the strategy is $\mathbf{H}_n$ or $\mathbf{Y}_n$, and is much better than the least square estimator for $\epsilon < 1$. When the strategy is $\mathbf{I}_n$ and the non-negative least square performs badly, the hybrid estimator can be as good as the least square estimator.

CHAPTER 4

BOUNDING THE ERROR IN THE MATRIX MECHANISM

As it is analyzed in the previous chapter, it is computationally difficult to find a strategy that minimize the error, both for ϵ- and (ϵ, δ)-differential privacy. Therefore the goal of this chapter is to develop tools that can explain what we informally term the error complexity of a given workload, which should measure, for fixed privacy parameters, the accuracy with which we can simultaneously answer all queries in the workload.

Such tools can help us to answer a number of natural questions that arise in the context of private synthetic data generation. Why is it possible to answer one set of queries more accurately than another? What properties of the queries, or of their relationship to one another, influence this? Can lower error be achieved by specializing the query set more closely to the task at hand? Does the combination of multiple users' workloads severely impact the accuracy possible for the combined workload?

Naive approaches to understanding the "hardness" of a query workload are unsatisfying. For example, one may naturally expect that the larger the number of queries in the workload, the larger the error in simultaneously answering them. Yet the number of queries in a workload is usually an inadequate measure of its hardness. Query workload sensitivity [26] is another natural approach. Sensitivity measures the maximum change in all query answers due to an insertion or deletion of a single database record. Basic differentially private mechanisms (e.g. the Laplace mechanism) add noise to each query in proportion to sensitivity, and in such cases sensitivity does in fact determine error rates. But the matrix mechanism can reduce error when answering multiple queries (with no cost to privacy), so that sensitivity alone fails to be a reliable measure.

In this chapter we seek a better understanding of workload error complexity by reasoning formally about the minimum error achievable for a workload, regardless of the underlying database. We pursue this goal in the context of a class differentially private algorithms: namely those that are instances of the matrix mechanism. In particular, to measure the hardness of a query workload, we present a lower bound on the error of answering this workload under the matrix mechanism. We primarily focus on the lower bound and its analyses under (ϵ, δ)-differential privacy. We demonstrate that our bound is tight in theory or almost tight empirically on many commonly considered workloads. The extended analysis on this bound connects the error of the matrix mechanism with the other error bounds on database-dependent mechanisms. At the end of this chapter, we also present how our bound will change along with operations of query workloads.

4.1 Equivalence and containment for workloads

First we develop a notion of equivalence and containment of workloads with respect to error. We will verify that the error bounds presented in the next section satisfy these relationships in most cases.

The special form of the expression for total error in Prop. 3.4 means that there are many workloads that are equivalent from the standpoint of error. For two workloads $\mathbf{W}_1$ and $\mathbf{W}_2$, if $\mathbf{W}_1^T\mathbf{W}_1 = \mathbf{W}_2^T\mathbf{W}_2$, then any strategy $\mathbf{A}$ that can represent the queries of $\mathbf{W}_1$ can also represent the queries of $\mathbf{W}_2$, and vice versa. In addition, $\mathbf{W}_1^T\mathbf{W}_1 = \mathbf{W}_2^T\mathbf{W}_2$ implies $||\mathbf{W}_1\mathbf{A}^+||_F^2 = ||\mathbf{W}_2\mathbf{A}^+||_F^2$ for any strategy $\mathbf{A}$. We therefore define the following notion of *equivalence* of two workloads:

Definition 4.1 (Workload Equivalence)**.** *An $m_1 \times n_1$ workload $\mathbf{W}_1$ and an $m_2 \times n$ workload $\mathbf{W}_2$ are* equivalent*, denoted $\mathbf{W}_1 \equiv \mathbf{W}_2$, if $\mathbf{W}_1^T\mathbf{W}_1 = \mathbf{W}_2^T\mathbf{W}_2$.*

Note that the concept of equivalent workloads is different from the concept semantic equivalent workloads. The semantic equivalent compares pairs $(\mathbf{W}, \Phi)$ of a given workloads and their associated cell conditions and two semantically equivalent pairs are means that those pairs are different representations of the same set of queries. Equivalent workloads, however, are not necessarily the same set of queries. We call them equivalent since they have the same set of supporting strategies and the same error for any give strategy.

The following conditions on pairs of workloads imply that they have equivalent minimum error:

Proposition 4.1 (Equivalence Conditions)**.** *Given an $m_1 \times n_1$ workload $\mathbf{W}_1$ and an $m_2 \times n_2$ workload $\mathbf{W}_2$, each of the following conditions implies that* $\textsc{MinError}_{\mathcal{K}}(\mathbf{W}_1) = \textsc{MinError}_{\mathcal{K}}(\mathbf{W}_2)$*:*

(i) $\mathbf{W}_1 \equiv \mathbf{W}_2$

(ii) $\mathbf{W}_1 = \mathbf{Q}\mathbf{W}_2$ *for some orthogonal matrix* $\mathbf{Q}$.

(iii) $\mathbf{W}_2$ *results from permuting the rows of* $\mathbf{W}_1$.

(iv) $\mathbf{W}_2$ *results from permuting the columns of* $\mathbf{W}_1$.

(v) $\mathbf{W}_2$ *results from the column projection of* $\mathbf{W}_1$ *on all of its nonzero columns.*

Proof. (i): If $\mathbf{W}_1 \equiv \mathbf{W}_2$, for any strategy $\mathbf{A}$,

$$\begin{aligned}||\mathbf{W}_1\mathbf{A}^+||_F^2 &= \operatorname{trace}((\mathbf{A}^+)^T\mathbf{W}_1^T\mathbf{W}_1\mathbf{A}^+)\\ &= \operatorname{trace}((\mathbf{A}^+)^T\mathbf{W}_2^T\mathbf{W}_2\mathbf{A}^+) = ||\mathbf{W}_2\mathbf{A}^+||_F^2.\end{aligned}$$

Therefore $\textsc{MinError}_{\mathcal{K}}(\mathbf{W}_1) = \textsc{MinError}_{\mathcal{K}}(\mathbf{W}_2)$.

(ii): It is equivalent with (i).

(iii): It is a special case of (ii) where $\mathbf{Q}$ is a permutation matrix.

(iv): Let $\mathbf{P}$ be the permutation matrix such that $\mathbf{W}_1\mathbf{P} = \mathbf{W}_2$. For any strategy $\mathbf{A}$ on $\mathbf{W}_1$, $\mathbf{AP}$ is a strategy of $\mathbf{W}_2$ and $\textsc{TotalError}_{\mathcal{K},\mathbf{A}}(\mathbf{W}_1) = \textsc{TotalError}_{\mathcal{K},\mathbf{AP}}(\mathbf{W}_2)$.

(v): Since (iv) is true, we can assume $\mathbf{W}_1 = [\mathbf{W}_2, \mathbf{0}]$. For any strategy matrix $\mathbf{A}_2$ on $\mathbf{W}_2$, $\mathbf{A}_1 = [\mathbf{A}_2, \mathbf{0}]$ is a strategy on $\mathbf{W}_1$ and

$$\begin{aligned}
||\mathbf{W}_1\mathbf{A}_1^+||_F^2 &= \text{trace}(\mathbf{A}_1^+(\mathbf{A}_1^+)^T\mathbf{W}_1^T\mathbf{W}_1) \\
&= \text{trace}([\mathbf{A}_2^+, \mathbf{0}]^T[\mathbf{A}_2^+, \mathbf{0}][\mathbf{W}_2, \mathbf{0}]^T[\mathbf{W}_2, \mathbf{0}]) \\
&= \text{trace}((\mathbf{A}_2^+)^T\mathbf{W}_2^T\mathbf{W}_2\mathbf{A}_2^+) = ||\mathbf{W}_2\mathbf{A}_2^+||_F^2.
\end{aligned}$$

For any strategy $\mathbf{A}_1$ on $\mathbf{W}_1$ there is a strategy on $\mathbf{W}_2$ with equal or smaller error formed by deleting corresponding columns from $\mathbf{A}_2$. □

It follows from this proposition that $\textsc{MinError}$ is row and column representation independent, and behaves well under the projection of extraneous columns.

Defining a notion of containment for workload matrices is more complex than simple inclusion of rows. Even if the rows of $\mathbf{W}_1$ are not present in $\mathbf{W}_2$, it could be that $\mathbf{W}_1$ is in fact contained in $\mathbf{W}_2$ when expressed using an alternate basis. The following definition considers this possibility:

Definition 4.2 (Workload Containment)**.** *An $m_1 \times n$ workload $\mathbf{W}_1$ is contained in an $m_2 \times n$ workload $\mathbf{W}_2$, denoted $\mathbf{W}_1 \subseteq \mathbf{W}_2$, if there exists a $\mathbf{W}_2' \equiv \mathbf{W}_2$ where the rows of $\mathbf{W}_1$ are contained in $\mathbf{W}_2'$.*

The following proposition shows two conditions that imply inequality of error among workloads:

Proposition 4.2 (Error inequality)**.** *Given an $m_1 \times n_1$ workload $\mathbf{W}_1$ and an $m_2 \times n_2$ workload $\mathbf{W}_2$, each of the following conditions implies that*

$$\textsc{MinError}_{\mathcal{K}}(\mathbf{W}_1) \leq \textsc{MinError}_{\mathcal{K}}(\mathbf{W}_2):$$

(i) $\mathbf{W}_1 \subseteq \mathbf{W}_2$

(ii) $\mathbf{W}_1$ *is a column projection of* $\mathbf{W}_2$*.*

Proof. For (i), let $\mathbf{W}_2' \equiv \mathbf{W}_2$ such that $\mathbf{W}_2'$ contains all rows of $\mathbf{W}_1$. According to Prop. 4.1 (iii), we can assume $\mathbf{W}_2' = \left[\begin{smallmatrix}\mathbf{W}_1\\ \mathbf{W}_3\end{smallmatrix}\right]$. For any strategy $\mathbf{A}$ on $\mathbf{W}_2$, since $\mathbf{A}$ is also a strategy on $\mathbf{W}_2'$, $\mathbf{A}$ can represent all queries in $\mathbf{W}_1$ as well. Thus $\mathbf{A}$ is a strategy on $\mathbf{W}_1$. In addition,

$$\begin{aligned}
&\textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{W}_2)\\
=\ & P(\mathcal{K})\Delta_{\mathbf{A}}^2 ||\mathbf{W}_2\mathbf{A}^+||_F^2\\
=\ & P(\mathcal{K})\Delta_{\mathbf{A}}^2 ||\mathbf{W}_2'\mathbf{A}^+||_F^2\\
=\ & P(\mathcal{K})\Delta_{\mathbf{A}}^2 (||\mathbf{W}_1\mathbf{A}^+||_F^2 + ||\mathbf{W}_3\mathbf{A}^+||_F^2)\\
=\ & \textsc{Error}_{\mathbf{A}}(\mathbf{W}_1) + \textsc{Error}_{\mathbf{A}}(\mathbf{W}_3) \geq \textsc{Error}_{\mathbf{A}}(\mathbf{W}_1).
\end{aligned}$$

Therefore, $\textsc{MinError}_{\mathcal{K}}(\mathbf{W}_1) \leq \textsc{MinError}_{\mathcal{K}}(\mathbf{W}_2)$.

For (ii), given a strategy $\mathbf{A}_2$ on $\mathbf{W}_2$, let $\mathbf{A}_1$ be a column projection of $\mathbf{A}_2$ using the same projection that generates $\mathbf{W}_1$ from $\mathbf{W}_2$. According to the construction of $\mathbf{A}_1$ and $\mathbf{W}_1$, since $\mathbf{W}_2 = \mathbf{W}_2\mathbf{A}_2^+\mathbf{A}_2$, we have $\mathbf{W}_2\mathbf{A}_2^+\mathbf{A}_1 = \mathbf{W}_1$. Therefore according to Prop. 2.8, $||\mathbf{W}_2\mathbf{A}_2^+||_F \geq ||\mathbf{W}_1\mathbf{A}_1^+||_F$. Furthermore, since $\Delta_{\mathbf{A}_2} \geq \Delta_{\mathbf{A}_1}$, we know $\textsc{Error}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2) \geq \textsc{Error}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1)$. □

4.2 The singular value bound

In this section we state and prove our main result: a lower bound on $\textsc{MinError}(\mathbf{W})$, the optimal error of a workload $\mathbf{W}$ under the matrix mechanism with $\mathcal{K}$ satisfies (ϵ, δ)-differential privacy. The bound shows that the hardness of a workload is a function of its eigenvalues. We describe the measure and its properties in Section 4.2.1 and prove that it is a lower bound in Section 4.2.2. In Section 4.2.3 we briefly discuss the challenge of adapting this bound to ϵ-differential privacy.

4.2.1 The singular value bound

We first present the simplest form of our bound, which is based on computing the square of the sum of eigenvalues of the workload matrix:

Definition 4.3 (Singular Value Bound). *Given an $m \times n$ workload $\mathbf{W}$, its singular value bound, denoted* $\textsc{svdb}(\mathbf{W})$*, is:*

$$\textsc{svdb}(\mathbf{W}) = \frac{1}{n}(\lambda_1 + \ldots + \lambda_n)^2,$$

where $\lambda_1, \ldots, \lambda_n$ are the singular values of $\mathbf{W}$.

The following theorem guarantees that the singular value bound is a valid lower bound to the minimal error of a workload. The proof is presented in detail in Sec. 4.2.2.

Theorem 4.1. *Given an $m \times n$ workload $\mathbf{W}$,*

$$\textsc{MinError}(\mathbf{W}) \geq P(\mathcal{K})\textsc{svdb}(\mathbf{W}).$$

In the rest of dissertation, we refer to $\textsc{svdb}(\mathbf{W})$ as the "SVD bound". For any workload $\mathbf{W}$, the SVD bound is determined by $\mathbf{W}^T\mathbf{W}$ and can be computed directly from it (which can be more efficient):

Proposition 4.3. *Given $n \times n$ matrix $\mathbf{W}^T\mathbf{W}$.*

$$\textsc{svdb}(\mathbf{W}) = \frac{1}{n}(\sum_{i=1}^{n} \sqrt{d_i})^2.$$

where $d_1, \ldots, d_n$ are the eigenvalues of $\mathbf{W}^T\mathbf{W}$.

The SVD bound satisfies equivalence properties analogous to (i), (ii), (iii), and (iv) in Prop. 4.1 and inequality (i) in Prop. 4.2. However, it does not satisfy properties related to column projection, as shown in the following counter-example.

Example 4.1. *Consider a $2 \times n$ workload $\mathbf{W}$ consisting of queries $[1, 0, \ldots, 0]$ and $[t, t, \ldots, t]$. Let μ be the column projection w.r.t. the first cell condition of $\mathbf{W}$. When $n > 8$ and $t < 1/8$, $\textsc{svdb}(\mathbf{W}) < \textsc{svdb}(\mu(\mathbf{W}))$.*

According to Prop 4.2, column projections reduce the minimum error. Therefore, the SVD bound on any column projection of $\mathbf{W}$ also constitutes a lower bound for the minimum error of $\mathbf{W}$. Because of this we extend the simple SVD bound in the following way. Recall that $\mathcal{U}_n$ is the set of all column projections.

Definition 4.4. *Given an $m \times n$ workload $\mathbf{W}$ and $U \subseteq \mathcal{U}_n$. The singular value bound of $\mathbf{W}$ w.r.t. U, denoted by $\textsc{svdb}_U(\mathbf{W})$ is defined as*

$$\textsc{svdb}_U(\mathbf{W}) = \max_{\mu \in U} \textsc{svdb}(\mu(\mathbf{W})).$$

In particular, if $U = \mathcal{U}_n$, we call this bound the supreme singular value bound*, denoted $\overline{\textsc{svdb}}(\mathbf{W})$.*

According to Prop. 4.2 and Thm. 4.1, for any $U \subseteq \mathcal{U}_n$, $\textsc{svdb}_U(\mathbf{W})$ provides a lower bound on $\textsc{MinError}(\mathbf{W})$.

Corollary 4.1. *Given an $m \times n$ workload $\mathbf{W}$, and for any $U \subseteq \mathcal{U}_n$*

$$\begin{aligned}\textsc{MinError}_{\mathcal{K}}(\mathbf{W}) &\geq \max_{\mu \in U} \textsc{MinError}_{\mathcal{K}}(\mu(\mathbf{W})) \\ &\geq P(\mathcal{K})\textsc{svdb}_U(\mathbf{W}).\end{aligned}$$

The supreme SVD bound satisfies all of the error equivalence and containment properties, analogous to those of Prop. 4.1 and Prop. 4.2, as stated below.

Theorem 4.2. *Given an $m_1 \times n_1$ workload $\mathbf{W}_1$ and an $m_2 \times n_2$ workload $\mathbf{W}_2$, the following conditions imply that $\overline{\textsc{svdb}}(\mathbf{W}_1) = \overline{\textsc{svdb}}(\mathbf{W}_2)$:*

(i) $\mathbf{W}_1 \equiv \mathbf{W}_2$

(ii) $\mathbf{W}_1 = \mathbf{Q}\mathbf{W}_2$ *for some orthogonal matrix* $\mathbf{Q}$.

(iii) $\mathbf{W}_2$ *results from permuting the rows of* $\mathbf{W}_1$.

(iv) $\mathbf{W}_2$ *results from permuting the columns of* $\mathbf{W}_1$.

(v) $\mathbf{W}_2$ *results from column projection of* $\mathbf{W}_1$ *on all of its nonzero columns.*

Proof. (i) (ii) (iii): Since any one of those conditions leads to $\mathbf{W}_1^T\mathbf{W}_1 = \mathbf{W}_2^T\mathbf{W}_2$, according to Prop. 4.3, $\overline{\text{SVDB}}(\mathbf{W}_1) = \overline{\text{SVDB}}(\mathbf{W}_2)$.

(iv): Given a workload $\mathbf{W}_1$, it is sufficient to prove that the singular values of $\mathbf{W}_1$ are column representation independent. Let $\mathbf{W}_2$ be a matrix resulting from a permutation of the columns of $\mathbf{W}_1$ and $\mathbf{P}$ be the permutation matrix such that $\mathbf{W}_1\mathbf{P} = \mathbf{W}_2$. If a singular value decomposition of $\mathbf{W}_1$ is $\mathbf{W}_1 = \mathbf{Q}_\mathbf{W}\mathbf{\Lambda}_\mathbf{W}\mathbf{P}_\mathbf{W}$, then the decomposition of $\mathbf{W}_2$ is $\mathbf{W}_2 = \mathbf{Q}_\mathbf{W}\mathbf{\Lambda}_\mathbf{W}\mathbf{P}_\mathbf{W}\mathbf{P}$. Since $\mathbf{P}_\mathbf{W}\mathbf{P}$ is still an orthogonal matrix, $\mathbf{W}_2 = \mathbf{Q}_\mathbf{W}\mathbf{\Lambda}_\mathbf{W}\mathbf{P}_\mathbf{W}\mathbf{P}$ is a singular value decomposition of $\mathbf{W}_2$. Therefore the singular values of $\mathbf{W}_2$ are exactly the same as the singular values of $\mathbf{W}_1$.

(v): Since $\mathbf{W}_2$ is a column projection of $\mathbf{W}_1$, $\overline{\text{SVDB}}(\mathbf{W}_1) \geq \overline{\text{SVDB}}(\mathbf{W}_2)$ by definition. In addition, for any matrix with columns of zeroes, removing thse columns will not impact the non-zero singular values of the matrix. Therefore projecting those columns out will reduce the total number of singular values but not their sum. Therefore projecting out all zero columns from $\mathbf{W}_1$ will not decrease $\overline{\text{SVDB}}(\mathbf{W}_1)$, which indicates $\overline{\text{SVDB}}(\mathbf{W}_1) = \overline{\text{SVDB}}(\mathbf{W}_2)$. □

Theorem 4.3. *Given an $m_1 \times n_1$ workload $\mathbf{W}_1$ and an $m_2 \times n_2$ workload $\mathbf{W}_2$, the following conditions imply that $\overline{\text{SVDB}}(\mathbf{W}_1) \leq \overline{\text{SVDB}}(\mathbf{W}_2)$:*

(i) $\mathbf{W}_1 \subseteq \mathbf{W}_2$

(ii) $\mathbf{W}_1$ *is a column projection of* $\mathbf{W}_2$.

To prove Thm. 4.3, the following property of matrices is needed.

Lemma 2. *Let* $\mathbf{D}$ *be a diagonal matrix with non-negative diagonal entries and* $\mathbf{P}$ *be an orthogonal matrix whose columns are* $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$.

$$trace(\mathbf{D}) \le \sum_{i=1}^{n} \|\mathbf{D}\mathbf{p}_i\|_2.$$

Proof. Use d_i to denote the diagonal entries of $\mathbf{D}$ and p_{ij} to denote the entries in $\mathbf{P}$. Noticing that $\sum_{j=1}^{n} p_{ji}^2 = 1$, we have

$$\|\mathbf{D}\mathbf{p}_i\|_2 = \sqrt{\sum_{j=1}^{n} p_{ji}^2 d_j^2} \ge \sum_{j=1}^{n} p_{ji}^2 d_j.$$

Therefore, since $\sum_{j=1}^{n} p_{ij}^2 = 1$,

$$\sum_{i=1}^{n} \|\mathbf{D}\mathbf{p}_i\|_2 \ge \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ji}^2 d_j = \sum_{j=1}^{n}(\sum_{i=1}^{n} p_{ji}^2) d_j = \text{trace}(\mathbf{D}).$$

□

Proof of Thm. 4.3. Since (ii) is naturally satisfied according to the definition of $\overline{\text{SVDB}}(\mathbf{W})$, it is sufficient to prove (i). Here we prove this is true even for the SVD bound so that it is also true for the supreme SVD bound.

Given $\mathbf{W}_1 \subseteq \mathbf{W}_2$, according to the definition, there exists a workload $\mathbf{W}_2'$ such that $\mathbf{W}_2' \equiv \mathbf{W}_2$ and $\mathbf{W}_2'$ contains all the queries of $\mathbf{W}_1$. Then $\mathbf{W}_2'$ has the following form:

$$\mathbf{W}_2' = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_3 \end{bmatrix}.$$

Then

$$\begin{aligned} \mathbf{W}_2^T\mathbf{W}_2 - \mathbf{W}_1^T\mathbf{W}_1 &= \mathbf{W}_2'^T\mathbf{W}_2' - \mathbf{W}_1^T\mathbf{W}_1 \\ &= \mathbf{W}_3^T\mathbf{W}_3 \ge 0 \end{aligned}$$

Let $\mathbf{W}_1 = \mathbf{Q}_1\mathbf{\Lambda}_1\mathbf{P}_1$ and $\mathbf{W}_2 = \mathbf{Q}_2\mathbf{\Lambda}_2\mathbf{P}_2$ be the singular value decomposition of $\mathbf{W}_1$ and $\mathbf{W}_2$, respectively. Then

$$\begin{aligned}
\mathbf{W}_2^T\mathbf{W}_2 - \mathbf{W}_1^T\mathbf{W}_1 \geq 0 \quad &\Leftrightarrow \quad \mathbf{P}_2^T\mathbf{\Lambda}_2^2\mathbf{P}_2 - \mathbf{P}_1^T\mathbf{\Lambda}_1^2\mathbf{P}_1 \geq 0 \\
&\Leftrightarrow \quad \mathbf{\Lambda}_2^2 - \mathbf{P}_2\mathbf{P}_1^T\mathbf{\Lambda}_1^2\mathbf{P}_1\mathbf{P}_2^T \geq 0 \\
&\Rightarrow \quad \forall\, i,\ \lambda_i \geq \|\mathbf{\Lambda}_1\mathbf{p}_i\|_2,
\end{aligned}$$

where λ_i is the i-th diagonal entry of $\mathbf{\Lambda}_2$ and $\mathbf{p}_i$ is the i-th column vector of $\mathbf{P}_1\mathbf{P}_2^T$. The inequality in the last row based on the property that the diagonal entries of any positive semidefinite matrix are non-negative. Therefore, according to Lemma 2,

$$\text{trace}(\mathbf{\Lambda}_2) = \sum_{i=1}^{n}\lambda_i \geq \sum_{i=1}^{n}\|\mathbf{\Lambda}_1\mathbf{p}_i\|_2 \geq \text{trace}(\mathbf{\Lambda}_1). \tag{4.1}$$

□

While Theorems 4.2 and 4.3 show that $\overline{\text{SVDB}}(\mathbf{W})$ matches all the properties of MinError($\mathbf{W}$), we often wish to avoid considering all possible column projections as required in the computation of $\overline{\text{SVDB}}(\mathbf{W})$. In many cases, using SVDB($\mathbf{W}$) as our lower bound provides good results. In other cases, we can choose an appropriate set of column projections to get a good approximation to the supreme SVD bound. We provide empirical evidence for this in the following example, along with an application of our bound to range and predicate workloads which have been studied in prior work. The bound allows us to evaluate, for the first time, how well existing solutions approximate the minimum achievable error under (ϵ, δ)-differential privacy.

Example 4.2. *In Table 4.1 we consider three workloads, each consisting of all multi-dimensional range queries for a different dimension set, along with a workload of all predicate queries. We report* SVDB($\mathbf{W}$) *and its ratio with* $\text{SVDB}_U(\mathbf{W})$ *where* U

Example Workload, $\mathbf{W}$	$\text{SVDB}(\mathbf{W})$	$\text{SVDB}_U(\mathbf{W})$	Error, as ratio to $P(\mathcal{K})\text{SVDB}(\mathbf{W})$			
			Identity	Hierarchical	Wavelet	Eigen Design
AllRange(2048)	3.034×10^7	1.001	47.25	1.776	1.545	1.028
AllRange$(64, 32)$	2.261×10^7	1.000	12.11	2.996	1.899	1.107
AllRange$(2,2,2,2,2,2,2,2,2,2)$	5.242×10^5	1.000	2.000	2.000	2.000	1.000
AllPredicate(1024)	4.885×10^{156}	1.000	1.884	3.464	6.292	1.000

Table 4.1. Four example workloads, their singular value bounds, and their error rates under common strategies and strategies proposed in prior work using Gaussian mechanism.

contains projections onto all possible ranges over the domain, showing that they are virtually indistinguishable.

We also compute the actual error introduced by several well-known strategies: the identity strategy, the hierarchical strategy [40], and the wavelet strategy [70], as well as a strategy generated by the Eigen-design mechanism [46]. These results reveal the quality of these approaches by their ratio to $\text{SVDB}(\mathbf{W})$. *For example, from the table we conclude that the Eigen-design mechanism and wavelet strategies have error at most* 1.5 *to* 3 *times the optimal for range workloads, but perform worse on the predicate queries. The identity strategy is far from optimal on low dimensional range queries, but better on high dimensional range queries and predicate queries.*

4.2.2 Proof of the SVD bound

We now describe the proof of Theorem 4.1. The key to the proof is an important property of the optimal strategy for the (ϵ, δ)-matrix mechanism. As shown in Lemma 3, among the optimal strategies for a workload $\mathbf{W}$, there is always a strategy $\mathbf{A}$ that has the same sensitivity for every cell condition (i.e. in every column). We use $\mathcal{A}_{\mathbf{W}}$ to denote the set that contains all strategies that satisfy $\mathbf{W}\mathbf{A}^{+}\mathbf{A} = \mathbf{W}$ and have the same sensitivity for every cell condition.

Recall that the L_2 sensitivity of strategy $\mathbf{A}$ (Prop. 2.3) is the maximum L_2 column norm of $\mathbf{A}$. The square of the sensitivity is also equal to the maximum diagonal entry of $\mathbf{A}^T\mathbf{A}$. By using Lemma 3, the sensitivity of $\mathbf{A}$ can instead be computed in terms of the trace of the matrix $\mathbf{A}^T\mathbf{A}$ and minimizing the error of $\mathbf{W}$ with this alternative expression of the sensitivity leads to the SVD bounds. Ultimately, to achieve the SVD bounds, a strategy $\mathbf{A}$ must simultaneously (i) minimize the error of $\mathbf{W}$ with the sensitivity computed in terms of the trace$(\mathbf{A}^T\mathbf{A})$, and (ii) have $\mathbf{A} \in \mathcal{A}_{\mathbf{W}}$. Such a strategy may not exist for every possible $\mathbf{W}$ and therefore the SVD bounds only serve as lower bounds to the minimal error of $\mathbf{W}$.

Lemma 3. *Given a workload* $\mathbf{W}$*, there exists a strategy* $\mathbf{A} \in \mathcal{A}_{\mathbf{W}}$ *such that* $\textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{W}) = \textsc{MinError}_{\mathcal{K}}(\mathbf{W})$.

Proof. For any workload $\mathbf{W}$, the problem of finding a strategy that minimizes the total error of $\mathbf{W}$ can be formulated as a SDP problem [45]. Therefore the optimal strategy that minimizes the total error of $\mathbf{W}$ always exists. Let $\mathbf{A}'$ be an optimal strategy on workload $\mathbf{W}$. We now construct a matrix $\mathbf{A}$ from $\mathbf{A}'$ such that $\mathbf{A} \in \mathcal{A}_{\mathbf{W}}$. Let $d_1, \ldots, d_n$ denote the diagonal entries of matrix $\Delta^2_{\mathbf{A}}\mathbf{I} - \mathbf{A}'^T\mathbf{A}'$, i.e. $d_1, \ldots, d_n$ is the difference between each diagonal entry of $\mathbf{A}'^T\mathbf{A}'$ and the maximal diagonal entry of $\mathbf{A}'^T\mathbf{A}'$. Since $d_1, \ldots, d_n \geq 0$, let $\mathbf{D}$ be the diagonal matrix whose diagonal entries are $\sqrt{d_1}, \ldots, \sqrt{d_n}$, and $\mathbf{A} = \left[\begin{smallmatrix}\mathbf{A}'\\ \mathbf{D}\end{smallmatrix}\right]$. Then $\mathbf{A}$ is a strategy matrix such that the diagonal entries of $\mathbf{A}^T\mathbf{A}$ are all the same. Let $\mathbf{B} = [\mathbf{A}'^{+}, \mathbf{0}]$. Then $\mathbf{WBA} = \mathbf{WA}'^{+}\mathbf{A}' = \mathbf{W}$. According to Prop. 2.8, $||\mathbf{WA}^{+}||_F \leq ||\mathbf{WB}||_F$. Recall $\bar{\bar{\Delta}}_{\mathbf{A}} = \bar{\bar{\Delta}}_{\mathbf{A}'}$, we have,

$$\begin{aligned}
\textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{W}) &= P(\mathcal{K})\bar{\bar{\Delta}}^2_{\mathbf{A}}||\mathbf{WA}^{+}||^2_F \\
&\leq P(\mathcal{K})\bar{\bar{\Delta}}^2_{\mathbf{A}}||\mathbf{WB}||^2_F \\
&= P(\mathcal{K})\bar{\bar{\Delta}}^2_{\mathbf{A}'}||\mathbf{WA}'^{+}||^2_F \\
&= \textsc{Error}_{\mathcal{K},\mathbf{A}'}(\mathbf{W}) = \textsc{MinError}_{\mathcal{K}}(\mathbf{W}).
\end{aligned}$$

Therefore $\textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{W}) = \textsc{MinError}_{\mathcal{K}}(\mathbf{W})$ and $\mathbf{A}$ is an optimal strategy for workload $\mathbf{W}$. □

Theorem 4.1 can hence be proved using the Lem. 2 and 3.

Proof. For a given workload $\mathbf{W}$, according to Lemma 3, it has an optimal strategy matrix $\mathbf{A} \in \mathcal{A}_{\mathbf{W}}$, whose sensitivity can then be computed as $\bar{\bar{\Delta}}_{\mathbf{A}}^2 = \frac{1}{n}\|\mathbf{A}\|_F^2$.

Let $\mathbf{W} = \mathbf{Q}_{\mathbf{W}}\mathbf{\Lambda}_{\mathbf{W}}\mathbf{P}_{\mathbf{W}}$ and $\mathbf{A} = \mathbf{Q}_{\mathbf{A}}\mathbf{\Lambda}_{\mathbf{A}}\mathbf{P}_{\mathbf{A}}$ be the singular decomposition of $\mathbf{W}$ and $\mathbf{A}$, respectively. We have:

$$\begin{aligned}
& \min_{\mathbf{A}:\, \mathbf{W}\mathbf{A}^+\mathbf{A}=\mathbf{W}} \bar{\bar{\Delta}}_{\mathbf{A}}^2 \|\mathbf{W}\mathbf{A}^+\|_F^2 \\
= \;& \min_{\mathbf{A}\in\mathcal{A}_{\mathbf{W}}} \frac{1}{n}\|\mathbf{A}\|_F^2\|\mathbf{W}\mathbf{A}^+\|_F^2 \\
= \;& \frac{1}{n}\min_{(\mathbf{\Lambda}_{\mathbf{A}}\mathbf{P}_{\mathbf{A}})\in\mathcal{A}_{\mathbf{W}}} \|\mathbf{\Lambda}_{\mathbf{A}}\|_F^2\|\mathbf{\Lambda}_{\mathbf{W}}\mathbf{P}_{\mathbf{W}}\mathbf{P}_{\mathbf{A}}^T\mathbf{\Lambda}_{\mathbf{A}}^+\|_F^2 \\
\geq \;& \frac{1}{n}\min_{\substack{\mathbf{\Lambda}_{\mathbf{A}},\mathbf{P}_{\mathbf{A}} \\ \mathbf{\Lambda}_{\mathbf{W}}\mathbf{\Lambda}_{\mathbf{A}}^+\mathbf{\Lambda}_{\mathbf{A}}=\mathbf{\Lambda}_{\mathbf{W}}}} \|\mathbf{\Lambda}_{\mathbf{A}}\|_F^2\|\mathbf{\Lambda}_{\mathbf{W}}\mathbf{P}_{\mathbf{W}}\mathbf{P}_{\mathbf{A}}^T\mathbf{\Lambda}_{\mathbf{A}}^+\|_F^2 && (4.2) \\
\geq \;& \frac{1}{n}\min_{\mathbf{P}_{\mathbf{A}}}(\sum_{i=1}^{n}\|\mathbf{\Lambda}_{\mathbf{W}}\mathbf{p}_i\|_2)^2 && (4.3) \\
\geq \;& \frac{1}{n}(\sum_{i=1}^{n}\lambda_i)^2, && (4.4)
\end{aligned}$$

where $\mathbf{p}_i$ is the i-th column of matrix $\mathbf{P}_{\mathbf{W}}\mathbf{P}_{\mathbf{A}}^T$, the inequality in (4.3) is based on the Cauchy-Schwarz inequality and the inequality in (4.4) comes from Lemma 2.

The equal sign in (4.3) is satisfied if and only if $\mathbf{\Lambda}_{\mathbf{A}} \propto \sqrt{\mathbf{\Lambda}_{\mathbf{W}}}$. Therefore to achieve equality in (4.3) and (4.4) simultaneously, we need $\mathbf{A} \propto \mathbf{Q}\sqrt{\mathbf{\Lambda}_{\mathbf{W}}}\mathbf{P}_{\mathbf{W}}$ for any orthogonal matrix $\mathbf{Q}$. Moreover, (4.2) is true if and only if $\mathbf{A} \in \mathcal{A}_{\mathbf{W}}$, which may not be satisfied when $\mathbf{A} \propto \mathbf{Q}\sqrt{\mathbf{\Lambda}_{\mathbf{W}}}\mathbf{P}_{\mathbf{W}}$, therefore the SVD bound only gives an lower bound to the minimum total error. □

Intuitively, the SVD bound is based on the assumption that the error can be evenly distributed to all the cells, which may not be achievable in all the cases. The supreme SVD bound considers only the case that the error can be evenly distributed to some of the cells and therefore may be tighter than the SVD bound.

4.2.3 Bounding $\textsc{MinError}_{\mathcal{K}}(\mathbf{W})$ under ϵ-differential privacy

The SVD bound is defined for $\mathcal{K}$ under (ϵ, δ)-differential privacy, so it is natural to consider extending these results to ϵ-differential privacy. When $\mathcal{K}$ bases on ϵ-differential privacy, the sensitivity of $\mathbf{A}$ as the largest L_1 norm of the columns of $\mathbf{A}$. For any vector, its L_1 norm is always greater than or equal to its L_2 norm. Given a workload $\mathbf{W}$ and a strategy matrix $\mathbf{A}$, $P(\mathcal{K})\bar{\bar{\Delta}}_{\mathbf{A}}^2 \|\mathbf{W}\mathbf{A}^{+}\|_F^2$ provides a lower bound to $\textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{W})$ under the ϵ-differential privacy. Therefore, error under the ϵ-differential privacy is also bounded below by $\textsc{svdb}(\mathbf{W})$.

When the number of queries in a workload is no more than the domain size, Bhaskara et al. [12] presented the following lower bound of error for any data-independent ϵ-differential privacy mechanism.

Theorem 4.4 ([12])**.** *Given an $m \times n$ workload $\mathbf{W}$ with $m \leq n$, let convex body $\mathbf{K} = \mathbf{W}\mathbf{B}_1^n$, where $\mathbf{B}_1^m$ is the m-dimensional L_1 ball. Let $\mathbf{P}_1, \ldots, \mathbf{P}_t$ be projection operators to a collection of t mutually orthogonal subspaces of $\mathbb{R}^m$ of dimension $m_1, \ldots, m_t$ respectively. Then the error of answering $\mathbf{W}$ under any data-independent ϵ-differentially private mechanism must be at least*

$$\max_{P_1,\ldots,P_t} \Omega\left(\sum_i \frac{m_i^3}{\epsilon^2} Vol_{m_i}(\mathbf{P}_i\mathbf{K})^{2/m_i}\right),$$

where $Vol_{m_i}(\mathbf{P}_i\mathbf{K})$ is the volume of the convex body $\mathbf{P}_i\mathbf{K}$ in m_i dimensional space.

In particular, when $\mathbf{P}_i$ are the projections to the singular vectors of $\mathbf{W}$, we can formulate the bound above using singular values of $\mathbf{W}$.

Corollary 4.2. *Given an $m \times n$ workload $\mathbf{W}$ with $m \leq n$, the error of answering $\mathbf{W}$ under any data-independent ϵ-differentially private mechanism must be at least*

$$\Omega\left(\sum_i^n \frac{\lambda_i^2}{\epsilon^2}\right),$$

where $\lambda_1, \ldots, \lambda_n$ are singular values of $\mathbf{W}$.

When $m \leq n$, we can compare the lower bound in Corollary 4.2 with the SVD bound under the ϵ-differential privacy. It is clear that the bound in Corollary 4.2 is tighter unless all singular values of $\mathbf{W}$ are equal. When $m > n$, the quality of the SVD bound under the ϵ-differential privacy is not yet known. The discussion on the tightness and looseness of the SVD bound in the next section is based on the (ϵ, δ)-matrix mechanism and cannot be directly extended to the ϵ-differential privacy.

4.3 Analysis of the SVD bound

In this section, we analyze the accuracy of the SVD bound as an approximation of the minimum error for a workload. We study the sufficient and necessary conditions under which the SVD bound is tight. In addition, we show the minimum error is equal to the bound over a specific class of workloads called variable-agnostic workloads and then generalize the result to the widely-studied class of data cube workloads. For both classes, strategies that achieve the minimum error can be constructed, as a by-product of the proof of the SVD bound.

We then show that the bound may be loose, underestimating the minimal error for some workloads. The worst case of looseness of the SVD bound is presented in Section 4.3.2, along with a formal estimate of the quality of the bound. We conclude this section with an example demonstrating empirically that error rates close to the lower bound can be achieved for workloads consisting of multi-dimensional range queries.

4.3.1 The tightness of the SVD bound

The circumstances under which the SVD bound is tight arise directly from inspection of the proof presented in Sec. 4.2.2. In particular, we noted the conditions that make the inequalities in equations (4.2), (4.3) and (4.4) actually equal. Those conditions are equivalent to a straightforward property of $\mathbf{W}^T\mathbf{W}$:

Theorem 4.5. *Given workload* $\mathbf{W}$, $\textsc{svdb}(\mathbf{W})$ *is tight if and only if the diagonal entries of* $\sqrt{\mathbf{W}^T\mathbf{W}}$ *are all equal.*

Proof. Recall that the SVD bound is tight if and only if (4.2), (4.3) and (4.4) takes equal sign simultaneously. The conditions that make all three inequalities to have equal sign is $\mathbf{A} \propto \mathbf{Q}\sqrt{\mathbf{\Lambda_W}}\mathbf{P_W}$ and $\mathbf{A} \in \mathcal{A}_\mathbf{W}$, which is equivalent to the case that the diagonal entries of $\mathbf{P}_\mathbf{W}^T\mathbf{\Lambda_W}\mathbf{P_W}$ are all the same. In addition, $\mathbf{P}_\mathbf{W}^T\mathbf{\Lambda_W}\mathbf{P_W} = \sqrt{\mathbf{W}^T\mathbf{W}}$ and we have the theorem proved. □

The condition in Thm 4.5 can be satisfied by many matrices. In particular, given a matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$ with singular value decomposition on *complex* domain $\mathbf{W} = \mathbf{Q_W}\mathbf{\Lambda_W}\mathbf{P_W}$ where $\mathbf{P_W}$ is the matrix of discrete Fourier transformation. The SVD bound is tight on $\mathbf{W}$.

Definition 4.5. *The discrete Fourier transformation (DFT) on a domain with size* n *can be represented as the following matrix:*

$$\mathbf{\Omega}_n = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega & \dots & \omega^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \dots & \omega^{(n-1)^2} \end{bmatrix},$$

where ω *is the primitive* n^{th} *root of unity.*

Theorem 4.6. *Given a matrix* $\mathbf{W} \in \mathbb{R}^{m\times n}$. *If* $\mathbf{W} = \mathbf{Q_W}\mathbf{\Lambda_W}\mathbf{\Omega}_n$ *is a singular value decomposition of* $\mathbf{W}$ *in the complex domain. The SVD bound is tight on* $\mathbf{W}$.

Proof. Since $\mathbf{W} \in \mathbb{R}^{m\times n}$, $\mathbf{W}^T\mathbf{W} = \mathbf{W}^H\mathbf{W}$, where ^{H} denotes the conjugate transpose of a matrix in the complex domain. Hence

$$\mathbf{W}^T\mathbf{W} = \mathbf{W}^H\mathbf{W} = \mathbf{\Omega}_n^H(\mathbf{\Lambda}_\mathbf{W}^H\mathbf{\Lambda_W})\mathbf{\Omega}_n$$

is an eigenvalue decomposition of $\mathbf{W}^T\mathbf{W}$ in the complex domain. Since $\mathbf{W}^T\mathbf{W}$ is also diagonalizable in the real domain, and according to the uniqueness of the set of the eigenvalues, we have

$$\mathbf{W}^T\mathbf{W} = \mathbf{P}_{\mathbf{W}}^T(\mathbf{\Lambda}_{\mathbf{W}}^H\mathbf{\Lambda}_{\mathbf{W}})\mathbf{P}_{\mathbf{W}}$$

as an eigenvalue decomposition of $\mathbf{W}^T\mathbf{W}$ in the real domain. Let $\mathbf{P}'$ be a the unitary matrix in the complex domain such that $\mathbf{P}'\mathbf{P}_{\mathbf{W}} = \mathbf{\Omega}_n$. Then

$$\mathbf{P}'^H(\mathbf{\Lambda}_{\mathbf{W}}^H\mathbf{\Lambda}_{\mathbf{W}})\mathbf{P}' = \mathbf{\Lambda}_{\mathbf{W}}^H\mathbf{\Lambda}_{\mathbf{W}}. \tag{4.5}$$

Represent $\mathbf{\Lambda}_{\mathbf{W}}^H\mathbf{\Lambda}_{\mathbf{W}}$ as

$$\mathbf{\Lambda}_{\mathbf{W}}^H\mathbf{\Lambda}_{\mathbf{W}} = \begin{bmatrix} \mu_1\mathbf{I} & & & \\ & \mu_2\mathbf{I} & & \\ & & \ddots & \\ & & & \mu_k\mathbf{I} \end{bmatrix},$$

where $\mu_1 \neq \mu_2 \neq \ldots \neq \mu_k$. (4.5) holds if and only if $\mathbf{P}'$ is a block diagonal matrix

$$\mathbf{P}' = \begin{bmatrix} \mathbf{P}'_1 & & & \\ & \mathbf{P}'_2 & & \\ & & \ddots & \\ & & & \mathbf{P}'_k \end{bmatrix},$$

where the shape of block $\mathbf{P}'_i$ and the diagonal block corresponding to μ_i is the same. In addition, noticing that whether (4.5) does not depend on the concrete values of $\mu_1, \ldots, \mu_k$, (4.5) also implies $\mathbf{P}'^H\sqrt{\mathbf{\Lambda}_{\mathbf{W}}^H\mathbf{\Lambda}_{\mathbf{W}}}\mathbf{P}' = \sqrt{\mathbf{\Lambda}_{\mathbf{W}}^H\mathbf{\Lambda}_{\mathbf{W}}}$. Therefore, consider the eigenvalue decomposition of $\sqrt{\mathbf{W}^T\mathbf{W}}$:

$$\sqrt{\mathbf{W}^T\mathbf{W}} = \mathbf{P}_{\mathbf{W}}^T\sqrt{\mathbf{\Lambda}_{\mathbf{W}}^H\mathbf{\Lambda}_{\mathbf{W}}}\mathbf{P}_{\mathbf{W}} = \mathbf{P}_{\mathbf{W}}^T\mathbf{P}'^H\sqrt{\mathbf{\Lambda}_{\mathbf{W}}^H\mathbf{\Lambda}_{\mathbf{W}}}\mathbf{P}'\mathbf{P}_{\mathbf{W}} = \mathbf{\Omega}_n^H\sqrt{\mathbf{\Lambda}_{\mathbf{W}}^H\mathbf{\Lambda}_{\mathbf{W}}}\mathbf{\Omega}_n.$$

Let $\lambda_1, \ldots, \lambda_n$ be the diagonal entries of the matrix $\sqrt{\mathbf{\Lambda}_{\mathbf{W}}^H \mathbf{\Lambda}_{\mathbf{W}}}$, then the i^{th} diagonal entry of the matrix $\sqrt{\mathbf{W}^T\mathbf{W}}$ is:

$$\frac{1}{n}\begin{bmatrix} 1 & \bar{\omega}^{i-1} & \ldots & \bar{\omega}^{(i-1)(n-1)} \end{bmatrix} \sqrt{\mathbf{\Lambda}_{\mathbf{W}}^H \mathbf{\Lambda}_{\mathbf{W}}} \begin{bmatrix} 1 \\ \omega^{i-1} \\ \vdots \\ \omega^{(i-1)(n-1)} \end{bmatrix} = \frac{1}{n}\sum_{j=0}^{n-1} \lambda_j \bar{\omega}^{(i-1)j}\omega^{(i-1)j} = \frac{1}{n}\sum_{j=0}^{n-1}\lambda_j.$$

Hence all diagonal entries of $\sqrt{\mathbf{W}^T\mathbf{W}}$ are the same and the SVD bound is tight on $\mathbf{W}$ according to Thm. 4.5. □

Workloads of convolution queries is a class of commonly interested workloads whose are $\mathbf{\Omega}_n$. Supporting such kind of queries and its applications under differential privacy has been extensively discussed in [29].

Definition 4.6 ([29])**.** *The matrix of circular convolution queries is*

$$\begin{bmatrix} h_0 & h_1 & \ldots & h_{n-1} \\ h_{n-1} & h_0 & \ldots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \ldots & h_0 \end{bmatrix}.$$

.

The decomposition of the circular convolution matrix has been given in [34], which guarantees the tightness of the SVD bound on workloads of circular convolutions.

Theorem 4.7 ([34])**.** *Any circular convolution matrix has an eigenvalue decomposition in the complex domain and the corresponding matrix of eigenvectors is* $\mathbf{\Omega}_n$*.*

Corollary 4.3. *The SVD bound is tight for any circular convolution workload* $\mathbf{W}$*.*

Another special class of workloads, called *variable-agnostic workloads*, in which the queries on each cell are fully symmetric and swapping any two cells does not change $\mathbf{W}^T\mathbf{W}$.

Definition 4.7 (Variable-agnostic workload)**.** *A workload* $\mathbf{W}$ *is* variable-agnostic *if* $\mathbf{W}^T\mathbf{W}$ *is unchanged when we swap any two columns of* $\mathbf{W}$*.*

For any variable-agnostic workload $\mathbf{W}$, $\mathbf{W}^T\mathbf{W}$ has the following special form: for some constants a and b such that $a > b$, all diagonal entries of $\mathbf{W}^T\mathbf{W}$ are equal to a and the remaining entries of $\mathbf{W}^T\mathbf{W}$ are equal to b.

The following theorem shows that any variable-agnostic workload $\mathbf{W}$ satisfies the condition in Thm 4.5. Furthermore, we also demonstrate the closed form expression of the SVD bound in case that n is a power of 2.

Theorem 4.8. *The SVD bound is tight for any variable-agnostic workload* $\mathbf{W}$*. In addition, when* $n = 2^k$ *for any nonnegative integer* k*,* $\textsc{svdb}(\mathbf{W}) = \frac{1}{n}(\sqrt{a+(n-1)b} + (n-1)\sqrt{a-b})^2$*, where* a *is the value of diagonal entries of* $\mathbf{W}^T\mathbf{W}$ *and* b *is the value of off-diagonal entries of* $\mathbf{W}^T\mathbf{W}$*.*

Proof. Noticing that $\mathbf{W}^T\mathbf{W}$ is a circular convolution matrix so that $\mathbf{\Omega}_n$ is a matrix of eigenvectors of $\mathbf{W}^T\mathbf{W}$, and hence it is a matrix of singular vectors of $\mathbf{W}$. Thus, according to Thm. 4.6, the SVD bound is tight on $\mathbf{W}$. Furthermore, $\mathbf{W}^T\mathbf{W}$ has the following special form:

$$\mathbf{W}^T\mathbf{W} = \begin{bmatrix} a & b & \dots & b \\ b & a & \dots & b \\ \vdots & \vdots & \ddots & \vdots \\ b & b & \dots & a \end{bmatrix},$$

where $a > b$. One can verify that $a + (n-1)b$ is an eigenvalue of $\mathbf{W}^T\mathbf{W}$ with order 1 and $a - b$ is an eigenvalue of $\mathbf{W}^T\mathbf{W}$ with order $n - 1$. $\square$

As a concrete example, the workload $\textsc{AllPredicate}(n)$ is variable-agnostic, and therefore we can construct its optimal strategy and compute the error rate directly.

Corollary 4.4. *The SVD bound is tight for the workload* $\textsc{AllPredicate}(n)$. *In addition, when* $n = 2^k$ *for any nonnegative integer* k, $\textsc{svdb}(\textsc{AllPredicate}(n)) = \frac{2^{n-2}}{n}(n-1+\sqrt{n+1})^2$.

For variable-agnostic workloads, using a naive strategy like the identity matrix or the workload itself results in total error equal to na and the ratio by which the error is reduced using the strategy in Thm. 4.8 is approximately $1 - \frac{b}{a}$. In the case of $\textsc{AllPredicate}(n)$, the ratio is at least as low as 0.5, which occurs when n is very large.

Another family of workloads for which the SVD bound is tight are those consisting of sets of data cube queries [33]. A data cube workload consists of one or more cuboids, each of which contains all aggregation queries on all possible values of the cross-product of a set of attributes. It has already been shown in [29] that the workload of one cuboid is a convolution workload and hence the SVD bound is tight. Here we consider the case that the data cube contains more than one cuboids and each cuboid can have its own weight, so that higher weighted queries will be estimated more accurately than lower weighted ones.

Theorem 4.9. *The SVD bound is tight for any weighted data cube workload* $\mathbf{W}$.

Proof. Let us induct on the number of attributes d in the database. When $d = 1$, there are only two cuboids, the cuboid asks for the sum of all the cells and the cuboid asks for all the individual cells. Consider the workload $\mathbf{W}$ that weight the first cuboid w_1 and the second cuboid w_2, one can compute that

$$\mathbf{W}^T\mathbf{W} = \begin{bmatrix} w_1^2 + w_2^2 & w_1^2 & \dots & w_1^2 \\ w_1^2 & w_1^2 + w_2^2 & \dots & w_1^2 \\ \vdots & \vdots & \ddots & \vdots \\ w_1^2 & w_1^2 & \dots & w_1^2 + w_2^2 \end{bmatrix},$$

which is a variable agnostic workload. Therefore, according to Thm 4.8, $\textsc{svdb}(\mathbf{W})$ is tight.

If the SVD bound is tight when $d = d_0$, consider the case that $d = d_0 + 1$. Given a data cube workload $\mathbf{W}$. The cuboids in the data cube can be separated into two groups: the first group is the cuboids that aggregate on the last attribute; the second group is the cuboids that do not aggregate on the last attribute. Let $\mathbf{W}_1$ be the projection of the cuboids in the first group on the first d_0 attributes and $\mathbf{W}_2$ be the projection of the cuboids in the first group on the first d_0 attributes. We can represent $\mathbf{W}$ using $\mathbf{W}_1$ and $\mathbf{W}_2$:

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_1 & \mathbf{W}_1 & \dots & \mathbf{W}_1 \\ \mathbf{W}_2 & 0 & \dots & 0 \\ 0 & \mathbf{W}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{W}_2 \end{bmatrix},$$

where the number of $\mathbf{W}_1$ blocks and $\mathbf{W}_2$ blocks are the number of values in the last attribute, denoted as n_0. Let $\mathbf{Q}_1$, $\mathbf{Q}_2$ be the orthogonal matrices such that $\mathbf{Q}_1\mathbf{W}_1 = \sqrt{\mathbf{W}_1^T\mathbf{W}_1}$ and $\mathbf{Q}_2\mathbf{W}_1 = \sqrt{\mathbf{W}_2^T\mathbf{W}_2}$. Let

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 & 0 & \dots & 0 \\ 0 & \mathbf{Q}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & \mathbf{Q}_2 \end{bmatrix},$$

and then

$$\mathbf{QW} = \begin{bmatrix} \sqrt{\mathbf{W}_1^T\mathbf{W}_1} & \sqrt{\mathbf{W}_1^T\mathbf{W}_1} & \dots & \sqrt{\mathbf{W}_1^T\mathbf{W}_1} \\ \sqrt{\mathbf{W}_2^T\mathbf{W}_2} & 0 & \dots & 0 \\ 0 & \sqrt{\mathbf{W}_2^T\mathbf{W}_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sqrt{\mathbf{W}_2^T\mathbf{W}_2} \end{bmatrix}.$$

One can verify that

$$\sqrt{\mathbf{W}^T\mathbf{W}} = \sqrt{\mathbf{W}^T\mathbf{Q}^T\mathbf{QW}} = \begin{bmatrix} \mathbf{W}_3 & \mathbf{W}_4 & \dots & \mathbf{W}_4 \\ \mathbf{W}_4 & \mathbf{W}_3 & \dots & \mathbf{W}_4 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{W}_4 & \mathbf{W}_4 & \dots & \mathbf{W}_3 \end{bmatrix},$$

where

$$\mathbf{W}_3 = \frac{1}{n_0}((n_0 - 1)\sqrt{\mathbf{W}_1^T\mathbf{W}_1} + \sqrt{\mathbf{W}_1^T\mathbf{W}_1 + n_0\mathbf{W}_2^T\mathbf{W}_2}),$$
$$\mathbf{W}_4 = \frac{1}{n_0}(-\sqrt{\mathbf{W}_1^T\mathbf{W}_1} + \sqrt{\mathbf{W}_1^T\mathbf{W}_1 + n_0\mathbf{W}_2^T\mathbf{W}_2}).$$

Noticing that both $\mathbf{W}_1$ and $\left[\begin{smallmatrix}\mathbf{W}_1 \\ \sqrt{n_0}\mathbf{W}_2\end{smallmatrix}\right]$ are data cube workloads on d_0 attributes, according to the induction assumptions, both $\sqrt{\mathbf{W}_1^T\mathbf{W}_1}$ and $\sqrt{\mathbf{W}_1^T\mathbf{W}_1 + n_0\mathbf{W}_2^T\mathbf{W}_2}$ are symmetric matrices whose diagonal entries are all the same, respectively. Thus $\mathbf{W}_3$ and $\mathbf{W}_4$ are also symmetric matrices whose diagonal entries are all the same, respectively. Then $\sqrt{\mathbf{W}^T\mathbf{W}}$ is a symmetric matrices whose diagonal entries are all the same and then the SVD bound is tight on $\mathbf{W}$. $\square$

Data cube workloads (a special case of marginal workloads) have been studied by the differential privacy community in both theory and practice [9, 19, 42]. Barak et al. [9] use the Fourier basis as a strategy for workloads consisting of marginals while Ding et al. [19] proposed an approximation algorithm for data cube workloads. Thm. 4.9 shows that under (ϵ, δ)-differential privacy we can now directly compute the optimal strategy, obviating the need to use an approximation algorithm or blindly relying on the Fourier basis for workloads of this type. The result in [42], however, involves data-dependent techniques and the comparison between [42] to the SVD bound relies on a thorough analysis of the spectral properties of data cube workloads, which is a direction of future work.

4.3.2 The looseness of the SVD bound

The SVD bound can also underestimate the minimum error when the workload is highly skewed. For example, the SVD bound does not work well when the sensitivity of one column in the workload is overwhelmingly larger than others. Recall the workload in Example 4.1, when $t \to 0$, the SVD bound will underestimate the total error by a factor of n. This is caused by the underestimate of the sensitivity of $\mathbf{A}$ considered in equation (4.2) in the proof of Thm. 4.1.

Since the proof of Thm. 4.1 constructs a concrete strategy, one way to measure the looseness of the SVD bound is to estimate its ratio to the actual error introduced by this strategy. Note that the sensitivity of the strategy is the only part of the SVD bound that is underestimated. The square of the sensitivity is the maximum diagonal entry of matrix $\mathbf{A}^T\mathbf{A}$, rather than the estimate given by trace$(\mathbf{A}^T\mathbf{A})/n$. The ratio between the actual sensitivity and the estimated sensitivity bounds the looseness of the SVD bound, as shown by the following theorem.

Theorem 4.10. *Given an $m \times n$ workload $\mathbf{W}$. Let d_0 be the maximum diagonal entry of $\sqrt{\mathbf{W}^T\mathbf{W}}$.*

$$\textsc{MinError}_{\mathcal{K}}(\mathbf{W}) \leq \frac{nd_0 P(\mathcal{K})\textsc{svdb}(\mathbf{W})}{\mathit{trace}(\sqrt{\mathbf{W}^T\mathbf{W}})}.$$

Proof.

$$\begin{aligned}\textsc{MinError}_{\mathcal{K}}(\mathbf{W}) &\leq \textsc{Error}_{\mathcal{K},\mathbf{A}}(\mathbf{W}) \\ &= \frac{nd_0 P(\mathcal{K})\textsc{svdb}(\mathbf{W})}{\mathrm{Trace}(\sqrt{\mathbf{W}^T\mathbf{W}})}.\end{aligned}$$

□

According to Thm. 4.10, the approximate ratio of the SVD bound corresponds to the ratio between d_0, the largest diagonal entry of $\sqrt{\mathbf{W}^T\mathbf{W}}$ and the trace of $\sqrt{\mathbf{W}^T\mathbf{W}}$, which is equal to the sum of all singular values of $\mathbf{W}$. This ratio, although upper-bounded by the ratio between the largest singular value of $\mathbf{W}$ and the sum of all singular values of $\mathbf{W}$, is much closer to 1 than the ratio between singular values. As a consequence, the skewness in singular values does not always lead to a bad approximation ratio for the SVD bound. For example, for variable-agnostic workloads, the largest singular value can be arbitrarily larger than the rest of the singular values, while the SVD bound is tight. Instead, the cases where the SVD bound has high approximation ratio, such as the one in Example 4.1, are due to the skewness of singular value of $\mathbf{W}$ and the particular distribution of singular vectors. The supreme SVD bound can help us to avoid some of these worst cases, but there is no guarantee of the quality of the bound with more sophisticated cases.

Nevertheless, for many common workloads, empirical evidence suggests that the SVD bound is quite close to the minimal error. The following example provides a comparison between the SVD bound and achievable error for a few common workloads.

Example 4.3. *Returning to Table 4.1, we observe empirical evidence that for range and predicate workloads, there are strategies that come quite close to the SVD bound. The last column of Table 4.1 lists the error for the Eigen-design mechanism [46], which attempts to find approximately optimal strategies for any given workload by computing optimal weights for the eigenvectors of the workload. This algorithm is able to find a strategy whose error is within a factor of* 1.028 *and* 1.107 *of optimal for* AllRange(2048) *and* AllRange(64, 32)*, respectively.*

4.4 Comparison of mechanisms

The matrix mechanism is a data-independent mechanism: the noise distribution (and therefore error) depends only on the workload and not on the particular input data. This makes it possible to process the workload once and apply the mechanism efficiently to any dataset. On the other hand, data-independent mechanisms lack the flexibility to exploit specific properties of individual datasets. In this section, we use the SVD bound to compare the error bounds of the matrix mechanism with error bounds of other mechanisms that are data-dependent.

4.4.1 Asymptotic estimation of the SVD bound

Before the comparison, we first convert the SVD bound into an error measure that can be directly related to other bounds in the literature. We assume all queries in the workload have sensitivity at most one and estimate the SVD bound as a function of the domain size n and the number of queries m. Recall that the error in previous sections is defined as the total mean squared error of the queries. We introduce a new measure of error which bounds the maximum absolute error of the workload queries by α with high probability (controlled by β).

Definition 4.8 ((α, β)-Accurate [35])**.** *Given a workload* $\mathbf{W}$*, an algorithm* $\mathcal{K}$ *is* (α, β)*-accurate if, for any uniformly drawn data vector* $\mathbf{x}$*, with a probability of at least* $1-\beta$*,* $\max_{\mathbf{q} \in \mathbf{W}} |\mathcal{K}(\mathbf{q}, \mathbf{x}) - \mathbf{q}\mathbf{x}| \leq \alpha$*.*

Since the SVD bound measures total error (rather than max error), here we modify the (α, β)-accuracy by bounding the root mean squared error of the workload.

Definition 4.9 (RMS-(α, β)-Accurate)**.** *Given a workload* $\mathbf{W}$*, an algorithm* $\mathcal{K}$ *is RMS-*(α, β)*-accurate if, for any uniformly drawn data vector* $\mathbf{x}$*, with a probability of at least* $1-\beta$*,* $\sqrt{\sum_{\mathbf{q} \in \mathbf{W}} ||\mathcal{K}(\mathbf{q}, \mathbf{x}) - \mathbf{q}\mathbf{x}||^2 / |\mathbf{W}|} \leq \alpha$*.*

Theorem 4.11. *Given an* $m \times n$ *workload* $\mathbf{W}$*, if the* SVDB$(\mathbf{W})$ *is asymptotically tight, then there exists a strategy under which the matrix mechanism is RMS-*(α, β)*-accurate, where*

$$\alpha = O\left(\frac{\sqrt{\min(m,n)}\sqrt{\log(2/\delta)\log(\sqrt{\pi/2}/\beta)}}{\epsilon} \right).$$

Proof. Given a workload $\mathbf{W}$. Let $\lambda_1, \ldots, \lambda_{\min(m,n)}$ be the non-zero singular values of $\mathbf{W}$.

$$\begin{aligned} (\lambda_1 + \ldots + \lambda_{\min(m,n)})^2 &\leq \min(m,n)(\lambda_1^2 + \ldots + \lambda_{\min(m,n)}^2) \\ &= \min(m,n)||\mathbf{W}||_F^2 \\ &\leq \min(m,n)mn \end{aligned}$$

Therefore

$$\text{SVDB}(\mathbf{W}) \leq \min(m,n)m.$$

Noticing SVDB($\mathbf{W}$) is estimating the L_2 error of m queries and the error is Gaussian random noise. Take the average of the SVDB($\mathbf{W}$), consider the error estimator for Gaussian random variable with mean m and standard deviation σ:

$$\mathbf{P}(|X-m|>t\sigma) \leq \frac{\sqrt{2}}{\sqrt{\pi}t}\exp(-\frac{t^2}{2}),$$

and we have the bound proved. □

Recall the discussion in Sec. 4.3.1 indicates that the SVD bound is tight or almost tight for many common workloads. Thus, it is reasonable to compare the asymptotic estimate of the SVD bound to the error introduced by other mechanisms.

4.4.2 Comparison of error bounds

Here we compare our SVD bound with other error bounds from data-dependent mechanisms. We include four competitors each representing fundamentally different mechanisms. The median mechanism [59] discards candidate data vectors that are inconsistent with historical query answers. The multiplicative weights mechanism (MW) [37] and the iterative database construction method (IDC) [35] repeatedly update an estimated data vector according to query answers. The boosting method [27] maintains a distribution of queries according to the quality of their answers and repeatedly samples queries from the distribution so as to improve their answers. The (α,β)-accuracy under (ϵ,δ)-differential privacy for the median and the multiplicative weight mechanism follows the result in [35].

Table 4.2 summarizes error bounds of different data dependent approaches. In particular, the comparison is over $(\epsilon,\exp(-t))$-differential privacy and $(\alpha,\exp(-t))$-accuracy.

The workload $\mathbf{W}$ we considered contains m queries with sensitivity no larger than 1. The database is of size N, which means the sum of all x_i's in the data vector is N.

Mechanism		α
1	Median [59]	$O\left(\frac{\sqrt{N}(\log n \log m)^{1/4}\sqrt{t(\log m+t)}}{\sqrt{\epsilon}}\right)$
2	MW [37]	$O\left(\frac{\sqrt{N}(\log n)^{1/4}\sqrt{t(\log m+t)}}{\sqrt{\epsilon}}\right)$
3	IDC [35]	$O\left(\frac{(nN)^{1/4}\sqrt{t(\log m+t)}}{\sqrt{\epsilon}}\right)$
4	Boosting [27]	$\tilde{O}\left(\frac{\sqrt{N\log n}\cdot t^{3/2}\log^{3/2} m}{\epsilon}\right)$
5	SVDB	$O\left(\frac{\sqrt{\min(m,n)}\cdot t}{\epsilon}\right)$

Table 4.2. For $t \geq 2$, bounds on the α required to achieve $(\epsilon, \exp(-t))$-differential privacy and accuracy measures of: $(\alpha, \exp(-t))$-accuracy (mechanisms 1-4); RMS-$(\alpha, \exp(-t))$-accuracy (mechanism 5).

Observing the values of α in Table 4.2, the matrix mechanism has a greater dependence on ϵ compared with the median, the multiplicative weights and the iterative database construction methods. In addition, since the matrix mechanism is data-independent, it cannot take advantage of the input dataset so that it always assumes $n = N$. However, when N is sufficiently large ($\Theta(n)$) and $m = O(n)$, the SVD bound is smaller than the error of the Boosting method and can outperform other competitors when $m = \Omega(\exp(t/\epsilon))$.

4.4.3 Data-dependency and the matrix mechanism

Although the techniques of the matrix mechanism are data-independent, they can be deployed in a data-dependent way, blurring the distinction between mechanism types. The differentially private domain compression technique [49] may be applied to reduce the domain size n to $\Theta(N)$ with an additional $O(\log n)$ noise, which suggests a method for improving the error dependency of the matrix mechanism on n.

Further, the optimal strategy matrix used in the matrix mechanism represents the fundamental building blocks of the workload and the matrix mechanism reduces error by using the strategy queries as differentially private observations, instead of the workload queries. Recent data-dependent approaches can benefit from the same approach.

In fact, [38] selects Fourier basis vectors adaptively in a data dependent manner, but could benefit from selecting from a more efficient strategy matrix. Therefore, the SVDB bound can serve as a baseline accuracy measure, which may be improved by data-dependent query selection.

4.5 Complexity of random workloads

The tightness of the SVD bound on different workloads has already been demonstrated experimentally in Tab. 4.1 and theoretically in Sec. 4.3.1. Most of those analyses focus on *all* queries from one certain category, such as all range queries or all marginal queries on some attributes. However, many realistic workloads may not contain all but only a subset of queries from one category and it is hence important to discuss the complexity and the quality of the SVD bound on a sampled subset of a workload. In this section, we focus on the complexity sets of random sampled queries of a given workload. In particular, we compare the SVD bound and the error of strategies generated by the Eigen Design algorithm[46] on different subsets of all 1-dimensional range queries and marginal queries.

4.5.1 The cell simplified SVD bound

Recall the workload matrix in Example 4.1

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

In this query matrix, the second to the last cell of the domain are in the same query with the same weight. Now let us consider an alternative group of cells which contains two cells: x_1 and $x_2 + \dots + x_n$ and an alternative workload

$$\mathbf{W}' = \begin{bmatrix} 1 & 0 \\ t & t \end{bmatrix}.$$

Answers to the workload $\mathbf{W}'$ on the new group of cells are exactly the same as answers to the workload $\mathbf{W}$ on the original group of cells. Furthermore, it is clear that add cells that are not participated in any queries will not impact the answers to the workload as well. Hence we propose the following enhancement of the SVD bound.

Definition 4.10 (Cell Simplified SVD Bound)**.** *Given a workload* $\mathbf{W}$*, let the workload matrix* $\mathbf{W}'$ *be its minimized workload. The cell simplified SVD bound of* $\mathbf{W}$*, denoted as* $\textsc{svdb}_+(\mathbf{W})$*, is computed by*

$$\textsc{svdb}_+(\mathbf{W}) = \textsc{svdb}(\mathbf{W}').$$

The cell simplified SVD bound is not guaranteed to be larger than the SVD bound, as the following example.

Example 4.4. *Given the workload*

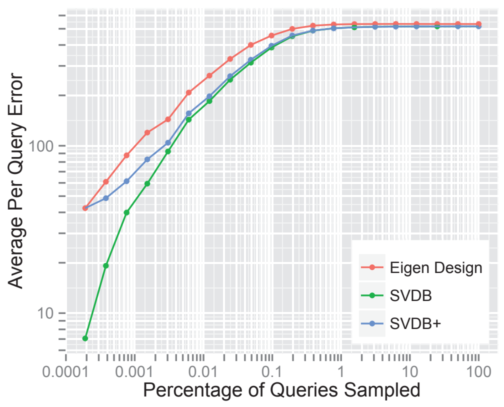
$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

$\textsc{svdb}(\mathbf{W}) = 2.61$ *and* $\textsc{svdb}_+(\mathbf{W}) = 2.5$*.*

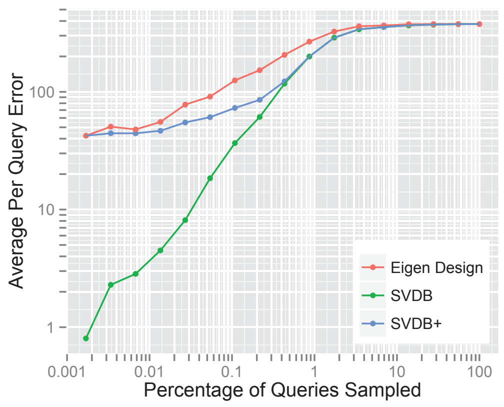
However, the cell simplified SVD bound can still benefit when the workload consists of small number of randomly sampled range queries or marginal queries, which will be shown in the experiments in the next section.

4.5.2 An empirical study on the complexity of random workloads

In this section, we use some experimental results to present the complexity of random workloads with different sizes. We generate workloads of random range queries on 1-dimensional domain of size 1024 (524800 different queries) and random marginal queries on 10-dimensional domain $2 \times 2 \times \ldots \times 2$ (59049 different queries). For each



(a) Per query error on random range workloads



(b) Per query error on random marginal workloads

Figure 4.1. Per query error on random range workloads on 1d domain [1024] and random marginal workloads on 10d domain $[2, \ldots, 2]$

sampled workload, we compare the error of the Eigen Design algorithm, the SVD bound and the cell simplified SVD bound. Since the number of queries are different among different sampled workloads, we record the average per query error to draw a fair comparison among different workloads.

Fig. 4.1 contains the experimental results on both random range queries and random marginal queries. The x-axis is the percentage of the queries that are sampled and the y-axis is the average per query error, both of which are in logarithm scale. One observes that the average per query error converges very fast in both cases. Furthermore, estimated error from the SVD bound is almost identical to the error from the Eigen Design algorithm when the average per query error converges. Therefore, it indicates that a relatively small subset of queries (1% for range queries and 10% for marginal queries) has almost the same complexity as the entire query set.

In addition, the cell simplified SVD bound estimates error better than the SVD bound on both random range queries and random marginal workloads when the number of queries is small. On workloads of random range queries (Fig. 5.1(a)), the simplified SVD bound is significantly better than the SVD bound up to about 100 queries. On workloads of random marginal queries (Fig. 5.2(a)), the simplified SVD bound is significantly better than the SVD bound up to about 1000 queries. Since most of the marginal queries cover small number of cells, there are lots of 0 columns in the workload matrix that the cell simplified SVD bound can take the advantage of. In general, the cell simplified SVD bound provides error bounds that is close to the error from the Eigen Design algorithm for most cases.

4.6 An algebra for workloads

In this section we briefly discuss the relationship between workload operations and the SVD bound. We define basic operators of negation-free relational algebra, union and crossproduct, on workloads and show how our error measure behaves in the presence of these operators. Many common workloads are the result of combining simpler workloads using these operators. Thus, the following results can be used to save computation of the SVD bound. In particular, for the crossproduct operation, the computation time for the SVD bound of the crossproduct of two workloads with size $m_1 \times n_1$ and $m_2 \times n_2$ can be reduced from $O(\min(m_1m_2, n_1n_2)m_1m_2n_1n_2)$ to $O(\min(m_1, n_1)m_1n_1 + \min(m_2, n_2)m_2n_2)$.

4.6.1 Union

The union operation on workloads has the standard meaning for rows of the workload matrix:

Definition 4.11 (Union). *Given an $m_1 \times n$ workload $\mathbf{W}_1$ and an $m_2 \times n$ workload $\mathbf{W}_2$ over the same n cell conditions. $\mathbf{W}_1 \cup \mathbf{W}_2$ is the union of $\mathbf{W}_1$ and $\mathbf{W}_2$, the workload consisting of the rows of both $\mathbf{W}_1$ and $\mathbf{W}_2$, without duplicates.*

The relationship between the SVD bounds of workloads and their unions can be bounded:

Theorem 4.12. *Given an $m_1 \times n$ workload $\mathbf{W}_1$ and an $m_2 \times n$ workload $\mathbf{W}_2$ on the same set of n cell conditions.*

$$\sqrt{\text{SVDB}(\mathbf{W}_1)} + \sqrt{\text{SVDB}(\mathbf{W}_2)} \geq \sqrt{\text{SVDB}(\mathbf{W}_1 \cup \mathbf{W}_2)};$$
$$\sqrt{\overline{\text{SVDB}}(\mathbf{W}_1)} + \sqrt{\overline{\text{SVDB}}(\mathbf{W}_2)} \geq \sqrt{\overline{\text{SVDB}}(\mathbf{W}_1 \cup \mathbf{W}_2)}.$$

The proofs of union is related to the relationship between singular values of matrices and their sum, as stated in the proposition below.

Proposition 4.4 ([30]). *Given two $n \times n$ matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ with singular values $\mu_1, \mu_2, \ldots, \mu_n$ and $\lambda_1, \lambda_2, \ldots, \lambda_n$ respectively. Let $\phi_1, \phi_2, \ldots, \phi_n$ be the singular values of $\mathbf{W}_1 + \mathbf{W}_2$, then*

$$\sum_{i=1}^{n} \mu_i + \sum_{i=1}^{n} \lambda_i \geq \sum_{i=1}^{n} \phi_i.$$

For any $m \times n$ matrix $\mathbf{W}$, there always exists an $n \times n$ matrix $\mathbf{W}'$ such that the nonzero singular values of $\mathbf{W}$ and $\mathbf{W}'$ are all the same. Theorem 4.4 holds even if both $\mathbf{W}_1$ and $\mathbf{W}_2$ are $m \times n$ matrices, which leads to the relationship between the SVD bounds of two workloads and their sum.

Proof. Let $\mathbf{W} = \mathbf{W}_1 \cup \mathbf{W}_2$. Expand $\mathbf{W}_1$, $\mathbf{W}_2$ to two $(m_1 + m_2) \times n$ matrices as follows:

$$\mathbf{W}'_1 = \begin{bmatrix} \mathbf{W}_1 \\ 0 \end{bmatrix}, \quad \mathbf{W}'_2 = \begin{bmatrix} 0 \\ \mathbf{W}_2 \end{bmatrix}.$$

Since $\mathbf{W}'_1$ and $\mathbf{W}'_2$ have the same singular values as $\mathbf{W}_1$ and $\mathbf{W}_2$, respectively, $\text{SVDB}(\mathbf{W}'_1) = \text{SVDB}(\mathbf{W}_1)$, $\text{SVDB}(\mathbf{W}'_2) = \text{SVDB}(\mathbf{W}_2)$. Furthermore, since

$$\mathbf{W}'_1 + \mathbf{W}'_2 = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \end{bmatrix} \supseteq \mathbf{W},$$

	$\textsc{svdb}$	$\overline{\textsc{svdb}}$
$\mathbf{W}_1 \cup \mathbf{W}_2$	$\sqrt{\textsc{svdb}(\mathbf{W}_1)} + \sqrt{\textsc{svdb}(\mathbf{W}_2)} \geq \sqrt{\textsc{svdb}(\mathbf{W}_1 \cup \mathbf{W}_2)}$	$\sqrt{\overline{\textsc{svdb}}(\mathbf{W}_1)} + \sqrt{\overline{\textsc{svdb}}(\mathbf{W}_2)} \geq \sqrt{\overline{\textsc{svdb}}(\mathbf{W}_1 \cup \mathbf{W}_2)}$
$\mathbf{W}_1 \times \mathbf{W}_2$	$\textsc{svdb}(\mathbf{W}_1)\textsc{svdb}(\mathbf{W}_2) = \textsc{svdb}(\mathbf{W}_1 \times \mathbf{W}_2)$	$\overline{\textsc{svdb}}(\mathbf{W}_1)\overline{\textsc{svdb}}(\mathbf{W}_2) \leq \overline{\textsc{svdb}}(\mathbf{W}_1 \times \mathbf{W}_2)$
Predicate Workloads		
$\mathbf{W}_1 \wedge \mathbf{W}_2$	$\textsc{svdb}(\mathbf{W}_1)\textsc{svdb}(\mathbf{W}_2) = \textsc{svdb}(\mathbf{W}_1 \wedge \mathbf{W}_2)$	$\overline{\textsc{svdb}}(\mathbf{W}_1)\overline{\textsc{svdb}}(\mathbf{W}_2) \leq \overline{\textsc{svdb}}(\mathbf{W}_1 \wedge \mathbf{W}_2)$

Table 4.3. Algebra operators and relations for the simple and supreme singular value bounds.

according to Prop. 4.4, the sum of the singular values of $\mathbf{W}_1'$ and $\mathbf{W}_2'$ is larger than or equal to the sum of singular values $\mathbf{W}$. Therefore, with Thm. 4.3 and Prop. 4.4,

$$\begin{aligned} & \sqrt{\textsc{svdb}(\mathbf{W}_1)} + \sqrt{\textsc{svdb}(\mathbf{W}_2)} \\ = \quad & \sqrt{\textsc{svdb}(\mathbf{W}_1')} + \sqrt{\textsc{svdb}(\mathbf{W}_2')} \geq \sqrt{\textsc{svdb}(\mathbf{W})}. \end{aligned}$$

For the case of $\overline{\textsc{svdb}}$, notice that for any projection μ,

$$\sqrt{\textsc{svdb}(\mu(\mathbf{W}_1))} + \sqrt{\textsc{svdb}(\mu(\mathbf{W}_2))} \geq \sqrt{\mu(\textsc{svdb}(\mathbf{W}))}.$$

Consider all projections and we have the result proved. □

4.6.2 Workload combination

Given two workloads over distinct sets of cell conditions, we can combine them to form a workload over the crossproduct of the individual cell conditions. This is most commonly used to combine workloads defined over distinct sets of attributes $\mathbb{B}_1$ and $\mathbb{B}_2$ to get a workload defined over $\mathbb{B}_1 \cup \mathbb{B}_2$. When we pair individual predicate queries, it is equivalent to pair them conjunctively.

Definition 4.12 (Workload combination)**.** *Given an $m_1 \times n_1$ workload $\mathbf{W}_1$ defined by cell conditions $\Phi = \phi_1 \dots \phi_{n_1}$ and an $m_2 \times n_2$ workload $\mathbf{W}_2$ defined by distinct cell conditions $\Psi = \psi_1 \dots \psi_{n_2}$, a new combined workload $\mathbf{W}$ is defined over cell conditions $\{\phi_i \wedge \psi_j \mid \phi_i \in \Phi, \psi_j \in \Psi\}$. For each $\mathbf{w}_1 = (w_{1,1}, \dots, w_{n_1,1}) \in \mathbf{W}_1$ and $\mathbf{w}_2 = (w_{1,2}, \dots, w_{n_2,2}) \in \mathbf{W}_2$, there is a query $\mathbf{w} \in \mathbf{W}$ accordingly:*

- (Crossproduct) *If the entry of* $\mathbf{w}$ *related to each cell condition* $\phi_i \wedge \psi_j$ *is* $w_{1,i} \cdot w_{2,j}$, $\mathbf{W}$ *is called the* crossproduct *of* $\mathbf{W}_1$ *and* $\mathbf{W}_2$, *denoted as* $\mathbf{W}_1 \times \mathbf{W}_2$.

- (Conjunction) *If both* $\mathbf{W}_1$ *and* $\mathbf{W}_2$ *consist of predicate queries and the entry of* $\mathbf{w}$ *related to each cell condition* $\phi_i \wedge \psi_j$ *is* $w_{1,i} \wedge w_{2,j}$, *then* $\mathbf{W}$ *is called the* conjunction *of* $\mathbf{W}_1$ *and* $\mathbf{W}_2$, *denoted as* $\mathbf{W}_1 \wedge \mathbf{W}_2$.

The next theorem describes the singular value bound for the crossproduct of workloads:

Theorem 4.13. *Given an* $m_1 \times n_1$ *workload* $\mathbf{W}_1$ *and an* $m_2 \times n_2$ *workload* $\mathbf{W}_2$ *defined on two distinct sets of cell conditions:*

$$\textsc{svdb}(\mathbf{W}_1 \times \mathbf{W}_2) = \textsc{svdb}(\mathbf{W}_1)\textsc{svdb}(\mathbf{W}_2)$$
$$\overline{\textsc{svdb}}(\mathbf{W}_1 \times \mathbf{W}_2) \geq \overline{\textsc{svdb}}(\mathbf{W}_1)\overline{\textsc{svdb}}(\mathbf{W}_2)$$

Proof. The property of crossproduct can be proved by constructing a proper representation to the resulting workload.

Let $\mathbf{w}_1$ and $\mathbf{w}_2$ be queries in $\mathbf{W}_1$ and $\mathbf{W}_2$, respectively and $\mathbf{W} = \mathbf{W}_1 \times \mathbf{W}_2$. Consider the vector representation of $\mathbf{w}_1$ and $\mathbf{w}_2$: $\mathbf{w}_1 = [w_{11}, w_{12}, \ldots, w_{1n_1}]^T$, $\mathbf{w}_2 = [w_{21}, w_{22}, \ldots, w_{2n_2}]^T$. The crossproduct of $\mathbf{w}_1$ and $\mathbf{w}_2$, denoted as $\mathbf{w}$ can be represented as an n_1 by n_2 matrix, whose (i,j) entry is equal to $w_{1i}w_{2j}$. In another word,

$$\mathbf{w} = \mathbf{w}_1\mathbf{w}_2^T.$$

We can the represent $\mathbf{w}$ as a vector, denoted as $\mathbf{w}'$, which is a $1 \times n_1n_2$ vector that contains entries in $\mathbf{w}$ row by row. Therefore,

$$\mathbf{w}' = [w_{11}\mathbf{w}_2, w_{12}\mathbf{w}_2, \ldots, w_{1n}\mathbf{w}_2]^T.$$

More generally, using $\mathbf{W}_{ij}^1$ to denote the (i,j) entry in $\mathbf{W}_1$, $\mathbf{W}$ can be represented as the following matrix:

$$\mathbf{W} = \begin{bmatrix} w_{11}^1\mathbf{W}_2 & w_{12}^1\mathbf{W}_2 & \dots & w_{1n_1}^1\mathbf{W}_2 \\ w_{21}^1\mathbf{W}_2 & w_{22}^1\mathbf{W}_2 & \dots & w_{2n_1}^1\mathbf{W}_2 \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1}^1\mathbf{W}_2 & w_{n2}^1\mathbf{W}_2 & \dots & w_{m_1n_1}^1\mathbf{W}_2 \end{bmatrix}.$$

Let $\mathbf{v}_1$, $\mathbf{v}_2$ be the eigenvectors of $\mathbf{W}_1^T\mathbf{W}_1$, $\mathbf{W}_2^T\mathbf{W}_2$ with eigenvalues λ_1, λ_2, respectively. Let the vector representation of $\mathbf{v}_1$ be $\mathbf{v}_1 = [v_{1n}, v_{2n}, \dots, v_{1n}]^T$. Consider the following vector

$$\mathbf{v} = [v_{11}\mathbf{v}_2, v_{12}\mathbf{v}_2, \dots, v_{1n}\mathbf{v}_2]^T,$$

According to block matrix multiplication,

$$\mathbf{W}^T\mathbf{W}\mathbf{v} = \mathbf{W}^T \begin{bmatrix} \sum_{i=1}^n w_{1i}^1 v_{1i}\mathbf{W}_2\mathbf{v}_2 \\ \sum_{i=1}^n w_{2i}^1 v_{1i}\mathbf{W}_2\mathbf{v}_2 \\ \vdots \\ \sum_{i=1}^n w_{m_1i}^1 v_{1i}\mathbf{W}_2\mathbf{v}_2 \end{bmatrix}$$

$$= \begin{bmatrix} \lambda_1 v_{11}\lambda_2\mathbf{v}_2 \\ \lambda_1 v_{12}\lambda_2\mathbf{v}_2 \\ \vdots \\ \lambda_1 v_{1n}\lambda_2\mathbf{v}_2 \end{bmatrix} = \lambda_1\lambda_2\mathbf{v}.$$

Thus $\mathbf{v}$ is an eigenvector of $\mathbf{W}^T\mathbf{W}$ with eigenvalue $\lambda_1\lambda_2$. Since $\mathbf{W}_1^T\mathbf{W}_1$ and $\mathbf{W}_2^T\mathbf{W}$ have n_1 and n_2 orthogonal eigenvectors, respectively, we can find n_1n_2 orthogonal eigenvectors with this method. Noticing $\mathbf{W}^T\mathbf{W}$ only has n_1n_2 eigenvalues, the eigenvalues of those $n_1 n_2$ eigenvectors are all the eigenvalues of $\mathbf{W}^T\mathbf{W}$. Let $\lambda_{11}, \ldots, \lambda_{1n}$ be the eigenvalues of $\mathbf{W}_1^T\mathbf{W}_1$ and $\lambda_{21}, \ldots, \lambda_{2n}$ be the eigenvalues of $\mathbf{W}_2^T\mathbf{W}_2$.

$$\begin{aligned}
\textsc{svdb}(\mathbf{W}) &= \frac{1}{n_1 n_2}\Big(\sum_{1\le i\le n_1, 1\le j\le n_2} \sqrt{\lambda_{1i}\lambda_{2j}}\Big)^2 \\
&= \frac{1}{n_1}\Big(\sum_{i=1}^{n_1}\sqrt{\lambda_{1i}}\Big)^2 \cdot \frac{1}{n_2}\Big(\sum_{i=1}^{n_2}\sqrt{\lambda_{2i}}\Big)^2 \\
&= \textsc{svdb}(\mathbf{W}_1)\textsc{svdb}(\mathbf{W}_2).
\end{aligned}$$

Though we use a specific rule above to represent the query cross products as query vectors, according to Theorem 4.2, the SVD bound is independent of the rule of representation. Thus we have the theorem proved in arbitrary cases.

For the case of $\overline{\textsc{svdb}}$, since for any projection μ_1 on $\mathbf{W}_1$ and μ_2 on $\mathbf{W}_2$, $\mu_1 \times \mu_2$ is a projection on $\mathbf{W}$. On the another hand, there are projections on $\mathbf{W}$ that can not be represented as a crossproduct of a projection on $\mathbf{W}_1$ and a projection on $\mathbf{W}_2$. Therefore

$$\begin{aligned}
\overline{\textsc{svdb}}(\mathbf{W}_1)\overline{\textsc{svdb}}(\mathbf{W}_2) &= \max_{\mu_1}\textsc{svdb}(\mu_1(\mathbf{W}_1))\max_{\mu_2}\textsc{svdb}(\mu_2(\mathbf{W}_2)) \\
&= \max_{\mu_1,\mu_2}\textsc{svdb}((\mu_1\times\mu_2)(\mathbf{W})) \\
&\le \max_{\mu}\textsc{svdb}(\mu(\mathbf{W})) = \overline{\textsc{svdb}}(\mathbf{W}).
\end{aligned}$$

□

The conjunction of predicate queries is a special case of crossproduct, Thus, applying Theorem 4.13 to predicate workloads we have:

Corollary 4.5. *Given an $m_1 \times n_1$ workload $\mathbf{W}_1$ and an $m_2 \times n_2$ workload $\mathbf{W}_2$, both of which consist of predicate queries.*

$$\begin{aligned}
\textsc{svdb}(\mathbf{W}_1 \wedge \mathbf{W}_2) &= \textsc{svdb}(\mathbf{W}_1)\textsc{svdb}(\mathbf{W}_2); \\
\overline{\textsc{svdb}}(\mathbf{W}_1 \wedge \mathbf{W}_2) &\ge \overline{\textsc{svdb}}(\mathbf{W}_1)\overline{\textsc{svdb}}(\mathbf{W}_2).
\end{aligned}$$

CHAPTER 5
AN EFFICIENT ALGORITHM UNDER THE MATRIX MECHANISM

The matrix mechanism makes clear that nearly any set of strategy queries can be used in this manner to answer a workload. Effective strategies have lower sensitivity than the workload, and are such that the workload queries can be concisely represented in terms of the strategy queries. In this chapter, we continue this line of work in order to create a truly adaptive mechanism that can answer a wide range of workloads with low error. The key to such a mechanism is *strategy selection*: the problem of computing the set of strategy queries that minimizes error for a given workload. Unfortunately, as it is discussed in Chapter 3, exact solutions to the strategy selection problem are infeasible in practice. One of our main contributions is an approximation algorithm capable of efficiently computing a nearly optimal strategy in $O(n^4)$ time (where n is the number of individual counting queries required to express the workload). The result is a mechanism that adapts the noise distribution to the set of queries of interest, relieving the user of the burden of choosing among mechanisms or carefully analyzing their workload.

Our main algorithm focuses on (ϵ, δ)-differential privacy[1] and is inspired by the statistical problem of optimal experimental design [15, 57], we formulate the strategy selection problem as a convex optimization problem which chooses n coefficients to serve as weights for a fixed set of *design queries.* Moreover, we show that the

[1]Our algorithm can also be adapted to ϵ-differential privacy, but it is less efficient, appears to be less effective, and is significantly harder to analyze. (Please see Sec. 5.1.5.)

eigenvectors of the workload (when represented in matrix form) capture the essential building blocks required for near-optimal strategies and are therefore a very effective choice for the design queries underlying the above optimization problem.

Our mechanism is also significantly more general than prior work. It can be applied to any workload of linear counting queries: a much larger class of queries than marginals or range queries. In addition, the algorithm avoids a subtle limitation of some previous approaches [40, 70, 19] in which achieving promised error rates depends on finding a proper representation for the workload.

5.1 An algorithm for efficient strategy selection

In this section we present an approximation algorithm for the strategy selection problem, prove its approximation rate and other properties, and discuss adapting the algorithm to ϵ-differential privacy. Below, we denote the optimal strategy under the matrix mechanism for a given workload $\mathbf{W}$ as OptStrat($\mathbf{W}$).

5.1.1 Optimal query weighting

The main difficulty in solving OptStrat($\mathbf{W}$) is computing (subject to complex constraints) all n^2 entries of a strategy matrix. To simplify the problem, we take inspiration from the related problem of *optimal experimental design* [57].

Consider a scientist who wishes to estimate the value of n unknown variables as accurately as possible. The variables cannot be observed directly, but only by running one or more of a fixed set of feasible experiments, each of which returns a linear combination of the variables. The experiments suffer from observational error, but those errors are assumed independent, and it follows that the least square method can be used to estimate the unknown variables once the results of the experiments are collected. Each experiment has an associated cost (which may represent time, effort, or financial expense) and the scientist has a fixed budget. The optimal experimental design is the subset (or weighted subset) of feasible experiments offering the best estimate of the unknown variables and with a cost less than the budget constraint.

There is an immediate analogy to the problem of strategy selection: our strategy queries are like experiments that provide partial information about the unknown data vector $\mathbf{x}$, and the final result will be computed using the least square method. However, in our setting, we are permitted to ask any query, with a cost (arising from the increase in sensitivity) which impacts the added noise. In addition, our goal is to minimize the sum of variances of the given workload queries, while experimental design always minimizes the error of the individual variables (i.e. the error metric in experimental design is equivalent to our problem only if $\mathbf{W}$ is the identity matrix).

Despite these important differences, we adopt from experimental design the idea to limit the selection of our strategy to weighted combinations of a set of *design queries* that are fixed ahead of time. Naturally, design queries with a weight of zero are omitted. For a set of design queries $\mathcal{Q}$, the following problem, denoted $\textsc{OptStrat}_{\mathcal{Q}}(\mathbf{W})$, selects the set of weights which minimizes the total error for $\mathbf{W}$.

Problem 5.1 (Approximate Strategy Selection). *Let* $\mathbf{W}$ *be a workload and* $\mathcal{Q} = \{\mathbf{q}_1, \dots \mathbf{q}_k\}$ *the design queries. For weights* $\mathbf{\Lambda} = (\lambda_1 \dots \lambda_k) \in \mathbb{R}^k$, *let matrix* $\mathbf{A}_{\mathbf{\Lambda},\mathcal{Q}} = [\lambda_1\mathbf{q}_, \dots, \lambda_k\mathbf{q}_k]^T$. *Choose weights* $\mathbf{\Lambda}_0 \in \mathbb{R}^k$ *such that:*

$$\textsc{Error}_{\mathcal{K},\mathbf{A}_{\mathbf{\Lambda}_0,\mathcal{Q}}}(\mathbf{W}) = \min_{\mathbf{\Lambda}\in\mathbb{R}^k}\textsc{Error}_{\mathcal{K},\mathbf{A}_{\mathbf{\Lambda},\mathcal{Q}}}(\mathbf{W}). \tag{5.1}$$

The solution to this problem only approximates the truly optimal strategy since it is limited to selecting a strategy that is a weighted combination of the design queries. But $\textsc{OptStrat}_{\mathcal{Q}}(\mathbf{W})$ can be computed much more efficiently than $\textsc{OptStrat}(\mathbf{W})$. To do so, we describe $\textsc{OptStrat}_{\mathcal{Q}}(\mathbf{W})$ as a semi-definite program [15], a special form of convex optimization in which a linear objective function is minimized over the cone of positive semidefinite matrices. Below, $\circ$ is the Hadamard (entry-wise) product of two matrices, and for symmetric matrix $\mathbf{Q}$, $\mathbf{Q} \geq 0$ denotes that $\mathbf{Q}$ is positive semidefinite, which means $\mathbf{x}^T\mathbf{Q}\mathbf{x} \geq 0$ for any vector $\mathbf{x}$.

Program 5.1.1 Optimal Query Weighting

$$\begin{aligned}
\textbf{Given:}\ & c_1, \ldots, c_n,\ \mathcal{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_n]. \\
\textbf{Choose:}\ & u_1, \ldots, u_n,\ v_1, \ldots, v_n. \\
\textbf{Mimimize:}\ & c_1 v_1 + \ldots + c_n v_n. \\
\textbf{Subject to:}\ & \begin{bmatrix} u_i & 1 \\ 1 & v_i \end{bmatrix} \geq 0, \quad i = 1, \ldots, n. \\
& (\mathcal{Q} \circ \mathcal{Q})^T \mathbf{u} \leq \mathbf{1}.
\end{aligned}$$

Theorem 5.1. *Given a workload* $\mathbf{W}$ *and a set of design queries* $\mathcal{Q} = \{\mathbf{q}_1, \ldots \mathbf{q}_n\}$*, let* $c_1, \ldots, c_n$ *be the squared* L_2 *norms of the columns of matrix* $\mathbf{W}\mathcal{Q}^+$*. If the output of Program 5.1.1 is* $u_1, \ldots, u_n$ *then setting* $\mathbf{\Lambda} = \{\sqrt{u_1} \ldots \sqrt{u_n}\}$ *achieves* $\textsc{OptStrat}_{\mathcal{Q}}(\mathbf{W})$.

Proof. (Sketch) To solve Problem 5.1, notice that applying a scalar to $\lambda_1, \ldots, \lambda_n$ will not change the value of $\textsc{Error}_{\mathcal{K},\mathbf{A}(\mathbf{\Lambda},\mathbf{Q})}(\mathbf{W})$. Thus we can constrain the sensitivity of the strategy to be 1. Then the problem is equivalent to minimizing $c_1/\lambda_1^2 + \ldots + c_n/\lambda_n^2$ with the constraint that the sensitivity of the strategy is 1. In Program 5.1.1, $u_i v_i \geq 1$ and the smaller v_i leads to smaller minimization goal. Thus the semidefinite constraints guarantee that $v_i = 1/u_i$ and the inequality constraints require the sensitivity to be 1 for any optimal solution. □

Algorithms for efficiently solving semidefinite programs have received considerable attention recently [15]. Using standard algorithms, Program 5.1.1 can be solved in $O(n|\mathcal{Q}|^3)$ time. Recall that the complexity of computing $\textsc{OptStrat}(\mathbf{W})$ is $O(n^8)$. Thus, Program 5.1.1 offers an efficiency improvement as long as $|\mathcal{Q}| = O(n^2)$. This provides a target size for selecting the design set, which we turn to next.

5.1.2 Choosing the design queries

The potential of the above approach depends on finding a set of design queries, $\mathcal{Q}$, that is concise (containing no more than n^2, and preferably n, queries) and also expressive (so that near-optimal solutions can be expressed as weighted combinations of its elements).

One straightforward idea is to adopt as the design queries one of the proposed strategy matrices from prior work. These are good strategy matrices for specific workloads such as the set of all range queries (wavelet or hierarchical strategy) or sets of low order marginals (the Fourier strategy). Choosing one of these for $\mathcal{Q}$ would guarantee that $\textsc{OptStrat}_{\mathcal{Q}}(\mathbf{W})$ produces a solution that improves upon the error of using that strategy. Unfortunately these strategies are not sufficiently expressive for workloads very different from their target workloads.

Another possibility is to use the workload itself as the set of design queries, but there are two difficulties with this. First, there is no guarantee that a workload includes within it the components from which a high quality strategy may be formed, especially if the workload only contains a small set of queries. The workloads of all range and all predicate queries are in fact sufficiently expressive (e.g. both the hierarchical strategy and a strategy equivalent to wavelet can be constructed by applying weights to the set of all range queries). But this leads to the second issue: these workloads, and others that serve important applications, are too large and fail to meet our conciseness requirement.

To avoid these pitfalls, we will derive the design set from the given workload $\mathbf{W}$ by applying tools of spectral analysis. Intuitively this is a good choice because the eigenvectors of a matrix often capture its most important properties. We will also show in the next section that this choice aids in the theoretical analysis of the approximation ratio because it allows us to relate the output of $\textsc{OptStrat}_{\mathcal{Q}}(\mathbf{W})$ to a lower bound on error that is a function of the workload eigenvalues.

Recall that the key part of the expression for Eqn. (3.5) in Prop. 3.4 is $||\mathbf{W}\mathbf{A}^{+}||_F^2$, which can also be represented as

$$||\mathbf{W}\mathbf{A}^{+}||_F^2 = \mathrm{trace}(\mathbf{W}^T\mathbf{W}(\mathbf{A}^T\mathbf{A})^{+}).$$

Notice that the workload occurs only in the form of $\mathbf{W}^T\mathbf{W}$. It follows that there are many workloads with equivalent total error because it is easy to construct a matrix $\mathbf{W}_0$ such that $\mathbf{W}_0^T\mathbf{W}_0 = \mathbf{W}^T\mathbf{W}$ by letting $\mathbf{W}_0 = \mathbf{Q}\mathbf{W}$ for any orthogonal matrix $\mathbf{Q}$. This suggests that, as far as total error under the matrix mechanism is concerned, the essential properties of the workload are reflected by $\mathbf{W}^T\mathbf{W}$. This motivates the following definition of eigen-queries of a workload, which we will use as our design set.

Definition 5.1 (Eigen-queries of a workload)**.** *Given a workload* $\mathbf{W}$*, consider the eigen-decomposition of* $\mathbf{W}^T\mathbf{W}$ *into* $\mathbf{W}^T\mathbf{W} = \mathbf{Q}^T\mathbf{D}\mathbf{Q}$*, where* $\mathbf{Q}$ *is an orthogonal matrix and* $\mathbf{D}$ *is a diagonal matrix. The* ***eigen-queries*** *of* $\mathbf{W}$ *are the rows of* $\mathbf{Q}$ *(i.e. the eigenvectors of* $\mathbf{W}^T\mathbf{W}$*).*

Choosing the eigen-queries of $\mathbf{W}$ as the design set meets our conciseness requirement because there are never more than n eigen-queries. Thus Program 5.1.1, $\textsc{OptStrat}_{\mathcal{Q}}(\mathbf{W})$, has complexity $O(n^4)$, which is $O(n^4)$ times faster than solving $\textsc{OptStrat}(\mathbf{W})$. We also find that the eigen-queries meet our expressiveness objective. We will show this next by proving a bound on the approximation ratio. In Sec. 5.2 we propose techniques that exploit the fact that using subsets of the eigen-queries retain much of the expressiveness and increase efficiency. And in Section 5.3, we show experimentally that weighted eigen-queries allow for near-optimal strategies, and also that the eigen-queries outperform other natural alternatives for the design set.

5.1.3 The Eigen-Design algorithm

It remains to define the complete Eigen-Design algorithm, which is Program 5.1.2:

Program 5.1.2 The Eigen-Design Algorithm

```
Input: Workload matrix $\mathbf{W}$.
Output: Strategy matrix $\mathbf{A}$.
1: Compute the eigenvalue decomposition of $\mathbf{W}^T\mathbf{W} = \mathbf{Q}^T\mathbf{D}\mathbf{Q}$, where $\mathbf{D} = diag(\sigma_1, \ldots, \sigma_n)$ and set $\mathcal{Q} = \mathbf{Q}$.
2: Compute weights $\lambda_1, \ldots, \lambda_n$ by solving Program 5.1.1 for above $\mathcal{Q}$ and with $c_i = \sigma_i$, $i \in [1..n]$.
3: Construct matrix $\mathbf{A}' = \mathbf{\Lambda}\mathbf{Q}$ where $\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_n)$.
4: Let $m_{11}, \ldots, m_{nn}$ be the $L_2$ norm of columns of $\mathbf{A}'$ and define $\mathbf{D}' = diag(\max_i\{\sqrt{m_{ii}^2 - m_{11}^2}\}, \ldots, \max_i\{\sqrt{m_{ii}^2 - m_{nn}^2}\})$.
5: return $\mathbf{A} = \left[\begin{smallmatrix}\mathbf{A}'\\ \mathbf{D}'\end{smallmatrix}\right]$.
```

The algorithm performs the decomposition of $\mathbf{W}^T\mathbf{W}$ to derive the design queries (Step 1), and solves $\textsc{OptStrat}_{\mathcal{Q}}(\mathbf{W})$ using the eigen-queries as the design set (Step 2). The matrix $\mathbf{A}'$ that is constructed in Step 3 is a candidate strategy but may have one or more columns whose norm is less than the sensitivity. In this case, it is possible to add queries, completing columns, without raising the sensitivity (Step 4 and 5). These additional queries can only provide more information about the database, and hence reduce error.

5.1.4 Analysis of the Eigen-Design algorithm

We now consider the accuracy and generality of the eigen-design algorithm, showing a bound on the worst-case approximation rate and that the accuracy of the algorithm is robust with respect to the representation of the input workload.

Approximation Rate

To bound the approximation rate, we rely on the error bound presented in the previous chapter. The existence of this bound does not imply an algorithm for achieving it, but it is a useful tool for understanding theoretically and experimentally the quality of the strategies produced by $\textsc{OptStrat}(\mathbf{W})$ using the eigenvalues of $\mathbf{W}$.

Recall how the singular value bound is proved. Let $\mathbf{A}_l$ be the strategy that is defined by weighting the eigen queries of $\mathbf{W}$ by $\sqrt{\sigma_1}, \ldots, \sqrt{\sigma_n}$. The singular value bound comes from underestimating the sensitivity of $\mathbf{A}_l$ using $\sqrt{\text{trace}(\mathbf{A}_l^T\mathbf{A}_l)/n}$. In practice, though the singular value bound may not be achieved since there is a gap between the sensitivity of $\mathbf{A}_l$ and $\sqrt{\text{trace}(\mathbf{A}_l^T\mathbf{A}_l)/n}$, the idea of weighting the eigen queries can be combined with the experimental design method to find good strategies to $\mathbf{W}$.

Notice the strategy $\mathbf{A}_l$ is contained in the possible solutions of Program 5.1.2. Thus the approximation ratio of Program 5.1.2 can be estimated by using the tightness and looseness results of the singular value bound.

Theorem 5.2. *Program 5.1.2 achieves the optimal solution whenever the singular value bound is tight. In addition, the strategy given by Program 5.1.2 approximates* $\textsc{MinError}_{\mathcal{K}}(\mathbf{W})$ *within a ratio of* $nd_0/\mathit{trace}(\sqrt{\mathbf{W}^T\mathbf{W}})$, *where* d_0 *is the largest diagonal entry of the matrix* $\sqrt{\mathbf{W}^T\mathbf{W}}$.

This theorem shows that the approximation ratio of applying Program 5.1.2 to a workload $\mathbf{W}$ can be bounded by analyzing the eigenvalues of matrix $\mathbf{W}^T\mathbf{W}$.

In practice, the ratio between the error of the eigen strategies and the optimal error is much smaller for a wide range of common workloads. In the experiments in Sec. 5.3, the largest ratio is at most 1.6 and in a number of cases the ratio is essentially equal to 1, modulo numerical imprecision.

Representation Independence

We say that the Eigen-Design algorithm is representation independent because its output is invariant for semantically equivalent workloads and error equivalent workloads. Recall that the logical semantics of a workload matrix $\mathbf{W}$ depends on its cell conditions and the semantic equivalent workloads are defined as Definition 2.5.

Naturally, we hope for a mechanism with equal error for any two semantically-equivalent representations of a workload. Some prior approaches do not have this property. For example, the wavelet and hierarchical strategies exploit the locality present in the canonical representation of range queries. An alternative matrix representation of the range queries may result in significantly larger error.

As it is pointed out in Proposition 2.1, semantic equivalent workloads can be generated by different operations, including splitting a cell condition, and merging cell conditions with same queries. Those two operations will impact the performance of the Eigen-Design algorithm. However, if we run the Eigen-Design algorithm on the minimized workload of the input workload $\mathbf{W}$, it does not suffer from this pitfall:

Proposition 5.1 (Semantic equivalence)**.** *Let $\mathbf{W}_1$ and $\mathbf{W}_2$ be two semantically-equivalent workloads whose minimized workloads are $\mathbf{W}'_1$ and $\mathbf{W}'_2$, respectively. Suppose Prog. 5.1.2 computes strategy $\mathbf{A}_1$ on workload $\mathbf{W}'_1$ and $\mathbf{A}_2$ on workload $\mathbf{W}'_2$. Then $\textsc{Error}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}'_1) = \textsc{Error}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}'_2)$.*

Proof. For any two semantically-equivalent workload matrices $\mathbf{W}_1$ and $\mathbf{W}_2$, there exist transformation matrices $\mathbf{T}_1$ and $\mathbf{T}_2$ such that $\mathbf{W}'_1 = \mathbf{T}_1\mathbf{W}'_2\mathbf{T}_2$ where $\mathbf{T}_1$ performs row swaps and $\mathbf{T}_2$ performs a sequence of column swaps, column duplications, or duplicate column elimination. Because $\mathbf{T}_1$ is actually an orthogonal matrix, ${\mathbf{W}'}_2^T\mathbf{W}'_2 = (\mathbf{T}_1\mathbf{W}'_2)^T(\mathbf{T}_1\mathbf{W}'_2)$. In addition, the operations on $\mathbf{T}_2$ do not change the nonzero of eigenvalues of ${\mathbf{W}'}_2^T\mathbf{W}$ and using $\mathcal{Q}\mathbf{T}_2$ instead of $\mathcal{Q}$ in Program 5.1.1 does not change the inequality constraint w.r.t. those x_i that have non-zero eigenvalues. Therefore, Program 5.1.1 computes semantically-equivalent strategies $\mathbf{A}\mathbf{T}_2$ and $\mathbf{A}$ for $\mathbf{W}'_1$ and $\mathbf{W}'_2$, respectively, and the final step in Program 5.1.2 will leave the strategies semantically-equivalent as well. Thus $\mathbf{A}_1 = \mathbf{A}_2\mathbf{T}_2$ and $\textsc{Error}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}'_1) = \textsc{Error}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}'_2)$. □

A related issue arises for two workloads that may be semantically different, but can be shown to have equivalent error. Since $\mathbf{W}$ appears as $\mathbf{W}^T\mathbf{W}$ in the expression for total error of a workload, it follows that, for any orthogonal matrix $\mathbf{Q}$, workload $\mathbf{QW}$ has error equal to $\mathbf{W}$ under any strategy. And in particular, any two such workloads have equal minimum error. The Eigen-Design algorithm always finds the same strategies for any two error-equivalent workloads:

Proposition 5.2 (Error equivalence)**.** *Let* $\mathbf{W}_1$ *and* $\mathbf{W}_2$ *be two error-equivalent workloads (i.e.* $\mathbf{W}_1 = \mathbf{Q}\mathbf{W}_2$ *for some orthogonal* $\mathbf{Q}$*) and suppose Program 5.1.2 computes strategy* $\mathbf{A}_1$ *on workload* $\mathbf{W}_1$ *and* $\mathbf{A}_2$ *on workload* $\mathbf{W}_2$*. Then* $\textsc{Error}_{\mathcal{K},\mathbf{A}_1}(\mathbf{W}_1) = \textsc{Error}_{\mathcal{K},\mathbf{A}_2}(\mathbf{W}_2)$

This result follows from the fact that the input to Program 5.1.1 uses the eigenvectors of $\mathbf{W}^T\mathbf{W}$, and therefore operates identically on equivalent workloads.

Optimizing for Relative Error

The discussion above is about workload error, an *absolute* measure of error. Our adaptive approach can also be used to find strategies offering low *relative* error. However, these are two fundamentally different optimization objectives and a single strategy matrix will not, in general, satisfy both.

One major difference between computing absolute error and relative error is the impact of the L_2 norm of a query vector. According to Proposition 3.4, the query error of $\mathbf{w}$ under strategy $\mathbf{A}$ is proportional to the L_2 norm of $\mathbf{w}$. Therefore a scaled query $k\mathbf{w}$ has k times larger query error compared with $\mathbf{w}$, and thus a query with higher L_2 norm contributes more to workload error. But because the relative error does not change with the L_2 norm of the query, using strategies optimized for workload error will not lead to optimal relative error.

Because the matrix mechanism is a data-independent mechanism, it is not possible to optimize for relative error directly. If the distribution of the target dataset were known, we could scale each query by its weighted L_2 norm, where the weight on each cell is proportional to the inverse of its probability. This scaling will optimize towards relative error by neutralizing the fact that the designed strategies are biased towards high norm queries. Since the underlying distribution is typically unknown, we introduce a heuristic scaling, prior to applying the Eigen-Design algorithm, in which each query is normalized to make its L_2 norm 1. This is equivalent to assuming a uniform distribution over the cells. In Sec 5.3, we show that, for two real datasets, this approach results in significantly lower relative error than competing techniques.

5.1.5 Application to ϵ-differential privacy

There are a number of challenges to applying the optimally weighted design approach under ϵ-differential privacy. Recall, once again, the formula for total error from Prop. 3.4: $P(\mathcal{K})\Delta_{\mathbf{A}}^2 \|\mathbf{W}\mathbf{A}^+\|_F^2$. To move to ϵ-differential privacy, only the sensitivity term changes, from L_2 to L_1: $P(\mathcal{K})\bar{\Delta}_{\mathbf{A}}^2 \|\mathbf{W}\mathbf{A}^+\|_F^2$. In the former case, the sensitivity term $\bar{\bar{\Delta}}_{\mathbf{A}}$ is uniquely determined by $\mathbf{A}^T\mathbf{A}$. But in the latter case, computing a near-optimal $\mathbf{A}^T\mathbf{A}$ is not enough, because $\bar{\Delta}_{\mathbf{A}}$ remains undetermined and is itself hard to optimize. As a result, it is more challenging to represent the optimal query weighting as a convex optimization problem. Below we present its formal encoding in a special case that $\mathcal{Q}$ contains no more than n queries, but note that the resulting problem is also less efficient because we can no longer rely on second order cone programming.

Program 5.1.3 Optimal Query Weighting under ϵ-Differential Privacy

$$\begin{aligned}
\textbf{Given: } & c_1, \ldots, c_n,\ \mathcal{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_n]. \\
\textbf{Choose: } & u_1, \ldots, u_n,\ v_1, \ldots, v_n. \\
\textbf{Mimimize: } & c_1 v_1 + \ldots + c_n v_n. \\
\textbf{Subject to: } & -\log(\mathbf{u}_i^2 \mathbf{v}_i) \le 0, \quad i = 1, \ldots, n. \\
& (\mathcal{Q} \circ \mathcal{Q})^T \mathbf{u} \le \mathbf{1}.
\end{aligned}$$

Theorem 5.3. *Given a workload* $\mathbf{W}$ *and a set of design queries* $\mathcal{Q} = \{\mathbf{q}_1, \ldots \mathbf{q}_n\}$, *let* $c_1, \ldots, c_n$ *be the squared* L_2 *norms of the columns of matrix* $\mathbf{W}\mathcal{Q}^+$. *If the output of Program 5.1.3 is* $u_1, \ldots, u_n$ *then setting* $\mathbf{\Lambda} = \{u_1 \ldots u_n\}$ *achieves* $\textsc{OptStrat}_{\mathcal{Q}}(\mathbf{W})$.

Theorem 5.3 can be proved in exactly the same way as Theorem 5.1.

Furthermore, there does not seem to be a universally good design set: the eigenqueries do not outperform other bases, in general, because they characterize only the properties of $\mathbf{W}^T\mathbf{W}$ but do not account for the L_1 sensitivity. We can nevertheless still use our algorithm to improve existing strategies. For example, using the Wavelet basis in the algorithm can improve its performance on all range and random range queries by a factor of 1.2 and 2.3, respectively; using the Fourier basis can improve its performance on low order marginals by a factor of 2.7.

Lastly, we do not know of an analogue of Theorem 4.1 providing a guaranteed error bound for the ϵ-differential privacy to verify the quality of the output.

These challenges motivate our choice to focus on (ϵ, δ)-differential privacy. While the two privacy guarantees are strictly-speaking incomparable, for conservative settings of δ, a user may be indifferent between the two. It is then possible to show that the asymptotic error rates for many workloads are roughly comparable between the two models.

5.2 Complexity and optimizations

We focus next on methods to further reduce the complexity of approximate strategy selection. We first analyze the complexity of the strategy selection algorithm and show that it can be solved more efficiently for low rank workloads, with no impact on the quality of the solution. Then we propose two approaches which can significantly speed up strategy selection by reducing the size of the input to Program 5.1.2. Intuitively, both approaches perform strategy selection over a summary of the workload that is constructed from its most significant eigenvectors, potentially sacrificing fidelity of the solution. We evaluate the latter two techniques in Sec 5.3.4.

5.2.1 Complexity analysis

The rank of workload matrix $\mathbf{W}$, denoted by $\text{rank}(\mathbf{W})$, is the size of the largest linearly-independent subset of the rows (or, equivalently, columns). When $\text{rank}(\mathbf{W})$ equals its maximum value, n, we say that $\mathbf{W}$ has full rank, which implies that accurate answers to the workload queries in $\mathbf{W}$ uniquely determine every cell count in $\mathbf{x}$. The complexity of the strategy selection algorithm can be broken into three parts: computing the eigenvectors and eigenvalues of matrix $\mathbf{W}^T\mathbf{W}$, solving the optimization problem, and constructing the strategy. If an eigenvalue is equal to zero, the eigenvalue and its corresponding eigenvectors are not actually involved the optimization and strategy construction, so they can be omitted in practice. Since the number of nonzero eigenvalues of $\mathbf{W}^T\mathbf{W}$ is equal to $\text{rank}(\mathbf{W})$, the complexity of Programs 5.1.2 is $O(nm\,\text{rank}(\mathbf{W}) + n\,\text{rank}(\mathbf{W})^3)$.

The complexity analysis above indicates that its efficiency can be significantly improved when $\text{rank}(\mathbf{W}) \ll n$. For example, the rank of low order marginal workloads can be bounded by the number of queries in the workload. Suppose a low-order marginal workload is defined on a k-dimensional space of cell conditions, each of which has size d. If the workload only contains one-way marginals, the complexity of solving Program 5.1.2 over this workload is bounded by $O(k^3d^{3+k})$. If the workload consists of one and two-way marginals the complexity is $O(k^6d^{k+6})$. Both of these bounds are much smaller than $O(d^{4k})$.

5.2.2 Workload reduction approaches

Next we propose two approaches which allow us to reduce the number of variables in the optimization problem. Both are inspired by principal component analysis (PCA), in which a matrix is characterized by the so-called principal eigenvectors, which are the eigenvectors associated with the largest eigenvalues.

In our case, recall that we cannot ignore the non-principal eigenvectors since the rank of the strategy matrix **A** cannot be lower than the workload matrix **W**. Instead, we either compute separately the weights for the principal and remaining eigenvectors, or we choose the same weights for all the remaining eigenvectors.

5.2.2.1 Eigen-Query separation

In *eigen-query separation*, we partition the eigen-queries into groups of a specified size according to their corresponding eigenvalues. Treating one group at a time, Program 5.1.1 is executed to determine the optimal weights just for the eigenvectors of that group. After the individual group optimizations are finished, another optimization can be used to calculate the best factor to be applied to all queries in each group. If the group size is large, all of the principal eigenvectors may be contained in one group, in which case the most important weights will be computed precisely.

The complexity of eigen-query separation depends on the group division. Notice that during the optimization of each group, the convex optimization problem is equivalent to setting all eigenvalues of excluded eigenvectors to zero. Analogous to the discussion of low rank workloads, letting the size of group be n_g, the complexity of solving the optimization problem over each group is $O(nn_g^3)$. Similarly, the time complexity to combine all the groups is $O(n(n/n_g)^3)$, and therefore $O(n^2n_g^3 + n(n/n_g)^3)$ in total. Asymptotically, the complexity of eigen-query separation is minimized when $n_g = O(n^{1/3})$. Then the complexity of the entire process is $O(n^3)$, the same as the cost of standard matrix multiplication.

5.2.2.2 Principal vector optimization

In the *principal vector optimization* we use a subset of the k most important eigenvectors as the design set, computing the optimal weights as usual. Instead of ignoring the less important eigenvectors (as is typical in PCA) we simply use a single common weight for each of the excluded vectors that have non-zero eigenvalues.

The number of variables in the convex optimization is reduced to $k + 1$ so that the time complexity is reduced to $O(nk^3)$. Experimentally we find that good results are possible with as few as 10% of the eigenvectors.

In Sec. 5.3.4 we show that both of the above approaches can improve execution time by two orders of magnitude with modest impact on solution quality. Extending our theoretical bound on the approximation rate to these approaches is an interesting direction for future work.

5.3 Experimental evaluation

The empirical evaluation of our mechanism has three objectives: $(i.)$ to measure solution quality of the Eigen-Design algorithm using both absolute and relative error; $(ii.)$ to measure the trade-off between speed-up and solution quality of our two performance optimizations; and $(iii.)$ to measure the effectiveness of using the eigen-queries as the design set. Experimental conclusions are presented in Sec. 5.3.6.

5.3.1 Experimental setup

Recall that total error is an absolute error measure based on root mean square error. Total error can be analytically computed using Prop. 3.4, and this is precisely the error that will be witnessed when running repeated trials and computing the mean deviation. Further, total error is independent of the true counts in data vector $\mathbf{x}$. That is, it is independent of the input data. These facts hold for all instances of the matrix mechanism, and therefore for each of the competing techniques we consider below. Therefore, when evaluating this absolute error measure, we do not perform repeated trials with samples of random noise nor do we use any datasets. In addition, all measures of workload error include the same factor $P(\epsilon, \delta)$, so that changing the privacy parameters impacts each method with the same factor, leaving the ratio of their error the same. Consequently, for total error, we simply fix $\epsilon = 0.5$ and $\delta = 0.0001$.

For total error, all error measurements are purely a function of the workload, reflecting the hardness of simultaneously answering a set of queries under differential privacy. In addition, these error rates can be compared directly with the lower bound as Theorem 4.1, reflecting a bound on the approximation rate. (This lower bound is not known to be achievable for all workloads, but nevertheless informs the quality of the eigen-strategy and its competitors.)

We also evaluate the relative error rates achievable using our algorithm by computing the strategy that minimizes absolute error on a scaled workload, as described in Sec. 5.1.4. Of course, the relative error rates reported in experiments are always for the original input workload. In these experiments we vary the value of ϵ, for a fixed $\delta = 0.0001$, and consider two real datasets. The first dataset is the US individual census data in the past five years[62], which are aggregated on age, occupation and income. The second is the Adult dataset[8], in which tuples are weight-aggregated on age, work, education and income. The size and dimensions of the datasets are:

Dataset	Dimension	# Tuples
US Census	$8 \times 16 \times 16$	15M
Adult	$8 \times 8 \times 16 \times 2$	33K

Table 5.1. The size and dimensions of the datasets

All experiments are executed on a quad-core 3.16GHz Intel CPU with 8 GB memory. Our Python implementation extends publicly-available code for the matrix mechanism [2] and also uses the `dsdp` solver [5] in the `cvxopt` [1] package. In addition, in order to present a more straightforward comparison between the total error and the relative error. Throughout the experiments, we present the square root of the total error, called the *workload error* instead of the total error.

5.3.2 Competing approaches

We compare the Eigen-Design strategy with the following four alternatives. Although originally proposed in the context of ϵ-differential privacy, each is easily adapted to (ϵ, δ)-differential privacy and the shift generally improves the relationship to the optimal error rate (with the exception of the Fourier strategy, noted below).

- **Fourier** is designed for workloads consisting of all k-way marginals, for given k [9]. The strategy transforms the cell counts with the Fourier transformation and computes the marginals from the Fourier parameters. When the workload is not full rank, the unnecessary queries of the Fourier basis are removed from the strategy to reduce sensitivity. The effectiveness of the Fourier strategy is somewhat reduced under (ϵ, δ)-differential privacy because dropping unnecessary queries results in a smaller sensitivity reduction using L_2.
- **DataCube** is an adaptive method that supports marginal workloads [19]. We implemented the BMAX algorithm, which chooses a subset of input marginals so as to minimize the maximum error when answering the input workload. To adapt the algorithm to (ϵ, δ)-differential privacy, sensitivity is measured under L_2 instead of L_1.
- **Wavelet** supports multi-dimensional range workloads by applying the Haar wavelet transformation to each dimension [70]. When using ϵ-differential privacy, Xiao et al. also introduced a hybrid algorithm that uses the identity strategy on dimensions with small size. This optimization is unnecessary under (ϵ, δ)-differential privacy: the hybrid algorithm does not lead to smaller error when sensitivity is measured under L_2.
- **Hierarchical** aims to answer workloads of range queries using a binary tree structure of queries: the first query is the sum of all cells and the rest of the queries recursively divide the first query into parts [40]. We test binary hierarchical

strategies (although higher orders are possible). The strategy in [40] supports one dimensional range workloads, but is adapted to multiple dimensions in a manner analogous to Wavelet [70].

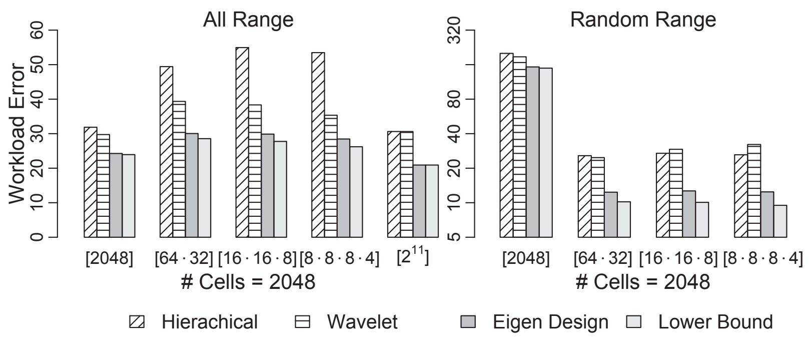
We do not compare with the error of the standard Gaussian mechanism, which, for the workloads considered, is far worse than all alternatives. Prior works [40, 70, 19] compared the error rates of their approaches with the identity strategy. We omit this explicit comparison, since the identity is always within the space of possible strategies the Eigen-Design could choose, but is not competitive.

5.3.3 Error of the Eigen-Design Algorithm

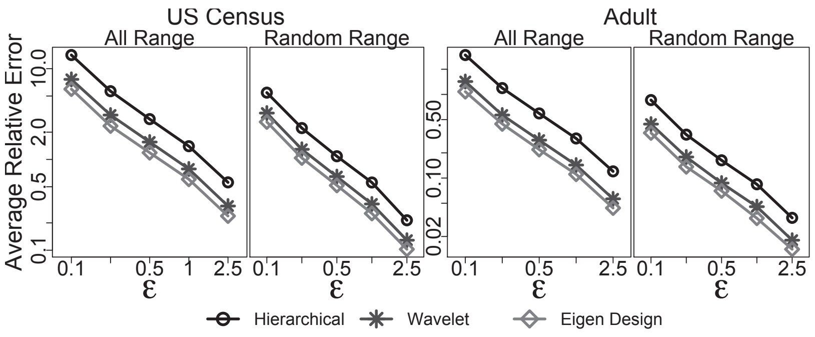
We now measure the improvement in absolute and relative error offered by the Eigen-Design algorithm along with its approximation to optimal absolute error. Below we refer to the strategy produced by the Eigen-Design algorithm, for a given workload, as the *eigen-strategy*. We consider three classes of workloads, beginning with workloads of range queries, then workloads of marginals, and then some alternative workloads designed to test the adaptivity of the mechanism.

5.3.3.1 Workloads of range queries

Figure 5.1 contain experiments on workloads of all range queries and random range queries. The random ranges are sampled with the two-step sampling method in [70]. Here the eigen-strategies are compared with Hierarchical and Wavelet strategy. The figures are in log scale, except Figure. 5.1(a) on all range queries. The results show that the eigen-design strategies reduce error by a factor of 1.2 to 2.1 in workload error and 1.3 to 1.5 in relative error compared to the best competing strategies. In addition, for workload error, the eigen-design strategy is within a factor of 1.3 to the lower bound.



(a) Workload errors on range queries

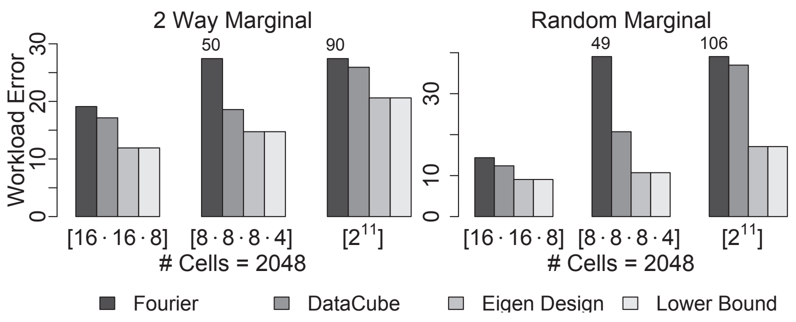


(b) Relative errors on range queries

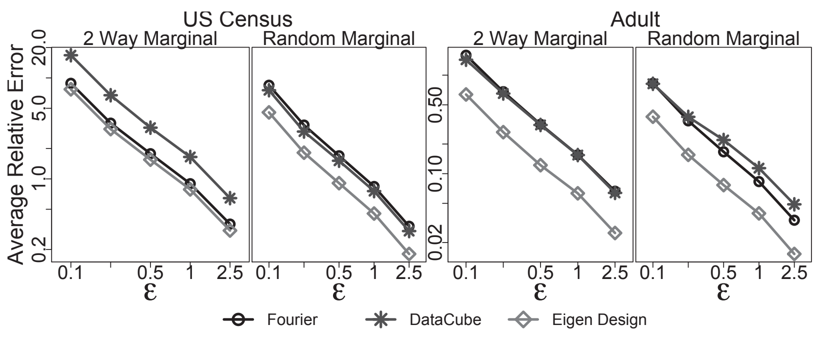
Figure 5.1. Absolute and relative error for the Eigen-Design algorithm and competitors, for range workloads, on 2048 cells. "Lower Bound" is a bound on the best possible error achievable by any strategy.

5.3.3.2 Workloads of marginals

Figure 5.2 contain experiments on workloads of 2-way marginal queries and random marginal queries, in which the random marginals are sampled with the sampling method in [19]. Here the eigen-strategies are compared with Fourier and DataCube. The figures are in linear scale for workload error and log scale for relative error. The results show that the eigen-design strategies reduce error by a factor of 1.3 to 2.2 compared to the best competing strategies in workload error, and by a factor of 1.1 to 2.7 in relative error. In addition, the error of eigen-design strategies match the lower bound of workload error, indicating that our algorithm found an optimal strategy with respect to workload error.



(a) Workload errors on marginal queries



(b) Relative errors on marginal queries

Figure 5.2. Absolute and relative error for the Eigen-Design algorithm and competitors, for marginal workloads, on 2048 cells. "Lower Bound" is a bound on the best possible error achievable by any strategy.

5.3.3.3 Alternative workloads

To demonstrate that our mechanism is adaptive over a variety of workloads, we also include other workloads that have not been studied in prior work. First we show that our mechanism adapts to semantically equivalent workloads, in which we repeat the experiment on range workload but randomly permute the order of cell conditions. The justification for this experiment comes from the fact that the user may wish to answer queries in which the order of the cell conditions is not obvious, such as predicate queries over categorial attributes.

In addition, we run experiments on three other workloads: the range marginals workload, the cumulative distribution (CDF) workload, and uniformly sampled predicate queries. The range marginals workload is important because most data analyses

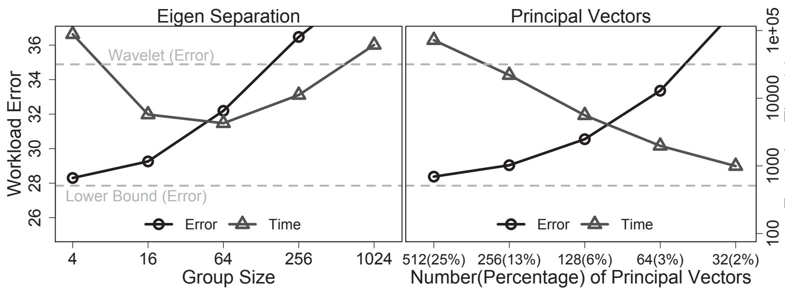
Workload	Error Ratio			Best/Worst Competitor
	Err Type	Best/Worst	Bound	
1D Range (Permuted)	workload	9.62/13.16	0.99	Wav./Hier.
	relative	1.51/2.43	-	Wav./Hier.
1Way Range Marginal	workload	1.30/7.69	0.98	D.Cube/Four.
	relative	1.36/4.93	-	D.Cube/Four.
2Way Range Marginal	workload	1.63/3.23	0.95	Hier./Four.
	relative	1.81/2.38	-	Wav./D.Cube
1D CDF	workload	1.01/1.01	0.80	Wav./Hier.
	relative	0.46/0.54	-	Wav./Hier.
Predicate	workload	1.39/1.94	1.00	Wav./Four.
	relative	1.42/3.55	-	Four./Hier.

Table 5.2. The factor of error reduced for the Eigen-Design algorithm w.r.t. the best/worst competitors strategies and the theoretical bound, for alternative workloads, on 2048 cells.

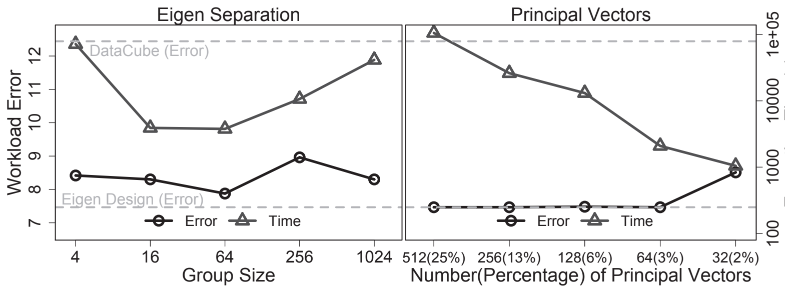
using marginals do not simply use individual counts, but also aggregate counts. If this is the case, simply computing the marginals workload privately is the wrong approach because error accumulates for aggregations. Last, the CDF workload is a highly-skewed set of one-dimensional range queries where the sensitivity in the first cell is n, decreasing linearly to 1 for the last cell.

We summarize the experimental results on alternative workloads in Table 5.2. For relative errors, due to space constraints, we only present results on US census data with $\epsilon = 0.5$ and $\delta = 0.0001$. We present, for each workload, the error reduction factor achieved by our algorithm compared to the best and worst competing approach, whose name is shown in the last column of the table. (Datacube is only considered for range marginals and Fourier is not considered on permuted range and CDF.) In addition, for workload error, we also include the ratio to the error lower bound.

The results show that the eigen-strategy can reduce workload error by as much as 13 times (on permuted range queries) and relative error by as much as 5 times (on one-way range marginals). The workload error of competing strategies is heavily impacted by the permutation but the relative errors are not as bad since queries



(a) Approximation approches over all 1D ranges



(b) Approximation approches over all 2D marginals

Figure 5.3. Quality and efficiency of approximation methods on 8192 cell conditions

of individual cells and small ranges dominate the workload, which do not change too much under permutation. On all workloads but one, the eigen-strategy beats every competitor by at least a factor of 1.3, and is very close to—or achieves—the theoretical error lower bound. The only exception is the CDF workload, in which the eigen-strategy is only a bit better than the competitor for workload error and worse (than Hierarchical and Wavelet) for relative error. Overall, the results for workload and relative error are largely similar for range marginals and the predicate workload.

5.3.4 Performance optimizations

Figure 5.3 illustrates the trade-off between computational speed-up and solution quality for the *eigen-separation* and *principal vector* performance optimizations described in Section 5.2. We only present results with workload errors here (the results with relative error are similar or even better). Error and computation time are plotted together using two y-axes: the left axis measures average per query error and the

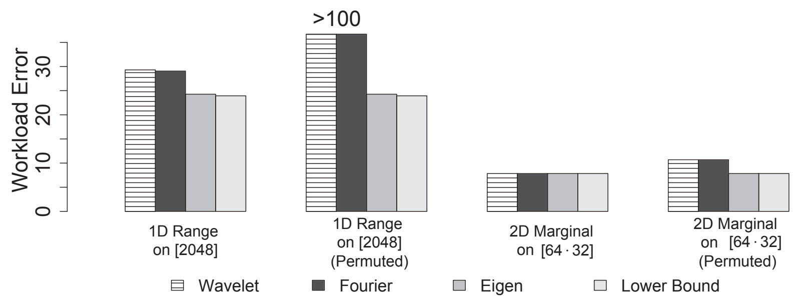


Figure 5.4. Comparison of design queries

right axis measures execution time in seconds. The baselines for error are the lower bound and the best competing technique.

The running time of using the standard Eigen-Design algorithm can be estimated from the running time of the principal vector method, which is more than an order of magnitude larger than the principal vector method with 25% of the eigenvectors. Both methods can reduce the running time by two orders of magnitude while the error they introduced is less than 12% over the lower bound. For the eigen-separation method, the computation in each group takes more time with larger group sizes while the computation of merging groups takes more time with smaller group sizes. Theoretically, the best choice for group size of the eigen-separation method is $n^{1/3}$, which is closest to 16 in this case. Using eigen-query separation with a group size of 16, the error is 5% higher on all range queries and 11% higher on all marginal queries. Using the principal vectors optimization with 6% of the eigenvectors, the error is 10% higher on all range queries and the same as the optimal on all marginal queries.

According to the results, the eigen-separation performs better on range queries while the principal vectors method is better on marginals. In either case, the performance improvements still produce results that are significantly better than competing techniques.

5.3.5 The Choice of design queries

To evaluate our claim in Section 5.1.2 that eigen-queries are an effective choice for the design queries we compare strategies computed by Program 5.1.1 using the eigen-queries, the Wavelet matrix and Fourier matrix as the design queries. Since using the eigen-queries introduces the same error to semantically equivalent workloads, we also empirically verify this property on other sets of designed queries. Figure 5.4 shows the results of those comparisons over two structured workloads considered above, as well as the same workloads with the order of the cell conditions permuted.

The results show that using the Fourier or the Wavelet strategy as the set of design queries introduces 20% more error over all one dimensional range queries and achieves the same error on two-way marginals. However these design queries cannot maintain their performance for workloads represented under a permutation of the cell conditions: they are worse than the eigen-queries by more than 4 times over the permuted one-dimensional range queries.

5.3.6 Experimental conclusions

The experimental results show that, for the workloads specifically targeted by competing techniques, those techniques achieve error that is not too far from optimal (usually a factor of about 1.2 to 3.4 times the lower bound on error). But for broader classes or workloads, or ad hoc subsets of structured workloads, existing techniques are limited and the adaptivity of the Eigen-Design can improve relative or absolute error by a larger factor. We have confirmed the versatility of our algorithm, as it improves on all competing techniques for virtually every workload considered. The one exception is the highly skewed CDF workload. The lowest error strategy we are aware of for this workload is produced by our design algorithm, but with an alternative basis.

CHAPTER 6

COMBINING THE MATRIX MECHANISM WITH DATA-AWARE MECHANISMS

Existing approaches for batch query answering broadly fall into two categories: *data-independent* mechanisms and *data-dependent* mechanisms. Data-independent mechanisms achieve the privacy condition by adding random noise that is independent of the input database. In previous chapters, we focus on the matrix mechanism with data-independent differentially private mechanisms. In this case, the matrix mechanism exploits properties of the workload to achieve greater accuracy, but the noise distribution (and therefore the error) is always fixed for *all* input databases.

Data-dependent mechanisms add noise that is customized to properties of the input database, producing different error rates on different input databases. In some cases, this can result in significantly lower error than data-independent approaches. These mechanisms typically need to use a portion of the privacy budget to learn about the data or the quality of a current estimate of the data. They then use the remaining privacy budget to privately answer the desired queries. In most cases, these approaches do not exploit workload.

A comparison of state-of-the-art mechanisms in each category reveals that each has advantages, depending on the "complexity" or "hardness" of the input database. If the database is viewed as a histogram, databases with large uniformly-distributed regions can be exploited by these algorithms allowing the data-dependent mechanisms to outperform data-independent competitors. But on more complex datasets, e.g. those with many regions of density, data-dependent mechanisms break down.

Consider a workload of random range queries and a dataset derived from an IP-level network trace. The state-of-the-art data-dependent mechanism *Multiplicative Weights and Exponential Mechanism* (MWEM) [38] can achieve 60.12 average per-query error when $\epsilon = 0.1$. The matrix mechanism using the wavelet basis as its query strategy offers per-query error of 196.6, for the same ϵ, a factor of 3.27 worse. But other datasets have properties that are difficult to exploit. On a dataset based on the HEP-PH citation network, the same workload evaluated by the MWEM algorithm has average per-query error of 722.3 with $\epsilon = 0.1$, while the error of the matrix mechanism with the same query strategy is still 196.6 for this workload, a factor of 3.67 better.

Such a large variation in the relative performance of mechanisms across data sets is a major limitation of current approaches. This is especially true because it is typically necessary to select a mechanism without seeing the data.

As described in Chapter 3, the matrix mechanism can be applied to any differentially private mechanism. However, applying the matrix mechanism to a data-dependent algorithm leads to much more complicated error analysis and our results in previous chapters are no longer be valid. In this chapter, we seek an alternative way to combine the matrix mechanism with a novel data-dependent algorithm, to form a novel 2-stage mechanism:

- The error of our mechanism is better by a factor up to 6.86 compared with the best state-of-art mechanisms on databases with large uniformly-distributed regions, and is comparable with state-of-art data-independent mechanisms on datasets that have properties that are difficult to exploit.

- We present an efficient algorithm in the first stage that partitions the domain into uniform regions. Compared with other differentially private partitioning algorithms, our algorithm generates much better partitions and runs in time that is only quasilinear in the size of the domain.

- We design a new, efficient algorithm in the second stage that adaptively distributes a privacy budget to ask a hierarchy of range queries of varying granularity. Unlike existing hierarchical strategies, our method allows a non-uniform budget distribution across queries of the same granularity, which we show leads to a strategy that is more finely tuned to the workload, and thus more accurate, than existing techniques.

To our knowledge, our mechanism is the first data-aware mechanism that provides significant improvement on databases with easy-to-exploit properties yet does not break-down on databases with complex distributions. Such property indicates that our mechanism can be successfully deployed without guessing about properties of the input database.

The rest of this chapter is organized as follows. We review notations in Sec. 6.1. An overview of the algorithm is presented in Sec. 6.2. The partitioning algorithm is presented in Sec. 6.3, and the bucket count estimating algorithm is included in Sec 6.4. We extensively compare our algorithm with state-of-the-art competing mechanisms in Sec. 6.5.

6.1 Histogram

In this section we review the concept of histogram. A histogram on $\mathbf{x}$ is a partition of $[1, n]$ into non-overlapping intervals, called buckets, along with a summary statistic for each bucket. We denote a histogram by $(\mathbf{B}, s)$ with $\mathbf{B}$ a set of buckets $\mathbf{B} = \{b_1 \dots b_k\}$ and s a set of corresponding statistics $s = s_1 \dots s_k$. Each b_i is described by an interval $[j_1, j_2]$ and the set of intervals covers $[1, n]$ and all intervals are disjoint. We define the length $|b_i|$ of bucket b_i to be $j_2 - j_1 + 1$.

We associate a summary statistic with each of the k buckets in a histogram. One way to do this is to treat the bucket intervals as range queries and evaluate them on $\mathbf{x}$. We denote this true statistic for bucket b_i by $b_i(\mathbf{x})$ and we use $\mathbf{B}(\mathbf{x})$ to denote the vector for true bucket counts. In other cases, the summary statistics are noisy estimates of $\mathbf{B}(\mathbf{x})$, denoted $s = s_1 \dots s_k$.

Throughout the paper we use the *uniform expansion* of a histogram $\mathbf{B}$ with k buckets. It is a data vector of length n derived from $\mathbf{B}$ by assuming uniformity for counts that fall within bucket ranges.

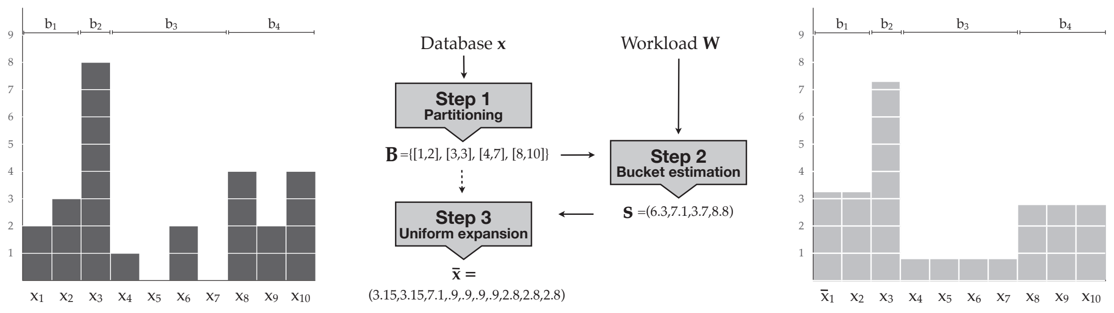
Definition 6.1. *Let expand be a function that takes a histogram $H = (\mathbf{B}, s)$ with buckets $\mathbf{B} = \{b_1 \dots b_k\}$ and statistics $s = s_1 \dots s_k$, and uniformly expands it. Thus, expand$(\mathbf{B}, s)$ outputs an n-length vector $\mathbf{y}$ defined as:*

$$y_j = \frac{s_{t(j)}}{|b_{t(j)}|}$$

where $t(j)$ is the function that maps position j to the index of the unique bucket in $\mathbf{B}$ that contains position j for $j \in [1, n]$.

In our algorithms, both the choice of a histogram and the value of the histogram statistics have the potential to leak sensitive information about $\mathbf{x}$. Both must be computed by a differentially private algorithm. Suppose that a differentially private algorithm returns histogram $H = (\mathbf{B}, s)$ where the statistics have noise added for privacy. We use $\hat{\mathbf{x}}$ to denote the uniform expansion of H, i.e., $\hat{\mathbf{x}} = \mathit{expand}(\mathbf{B}, s)$. Since the vector $\hat{\mathbf{x}}$ is a differentially private estimate for $\mathbf{x}$, we can use it to estimate answer any query w as $w(\hat{\mathbf{x}})$.

We are interested in how accurately $\hat{\mathbf{x}}$ approximates $\mathbf{x}$. The *absolute error* of $\hat{\mathbf{x}}$ is defined as $\|\mathbf{x} - \hat{\mathbf{x}}\|_1$. We are primarily interested in measuring the accuracy in terms of the answers to the workload queries. We define *average error* as the average L_1 error in the workload answers: $\frac{1}{m}\|\mathbf{W}(\mathbf{x}) - \mathbf{W}(\hat{\mathbf{x}})\|_1$. Our theoretical analysis considers expected error, where the expectation is taken over the randomness of $\hat{\mathbf{x}}$. For instance, $\mathbb{E}\|\mathbf{x} - \hat{\mathbf{x}}\|_1$ denotes *expected absolute error*.



(a) True database $\mathbf{x}$ (b) Algorithm flow chart (c) Private output $\hat{\mathbf{x}}$

Figure 6.1. Overview and example execution for the DAWA mechanism.

6.2 Algorithm overview

We give an overview to our new mechanism and an example below.

The *Data-Aware/Workload-Aware* (DAWA) mechanism is an ϵ-differentially-private algorithm that takes as input a workload of range queries, $\mathbf{W}$, and a database, $\mathbf{x}$, represented as a vector of counts. The output is an estimate $\hat{\mathbf{x}}$ of $\mathbf{x}$, where the noise added to achieve privacy is adapted to the input data and to the workload. The DAWA algorithm consists of the following three steps, the first two of which require private interactions with the database. To ensure that the overall algorithm satisfies ϵ-differential privacy, we split the total ϵ budget into ϵ_1 and ϵ_2 such that $\epsilon_1 + \epsilon_2 = \epsilon$ and use these two portions of the budget on the respective stages of the algorithm.

Step 1: private partitioning

The first step selects a partition of the domain that fits the input database. We describe (in Sec. 6.3) a novel differentially private algorithm that uses ϵ_1 budget to select a partition such that within each partition bucket, the dataset is approximately *uniform*. This notion of uniformity is later formalized as a cost function but the basic intuition is that if a region is uniform, then there is no benefit in using a limited privacy budget to ask queries at a finer granularity than these regions—the signal is too small to overcome the noise. The output of this step is $\mathbf{B}$, a partition of $\mathbf{x}$ into k buckets, *without* counts for the buckets.

Step 2: private bucket count estimation

Given the partition $\mathbf{B}$, the second step derives noisy estimates of the bucket counts. Rather than simply adding Laplace noise to the bucket counts, we use a workload-aware method. Conceptually, we re-express the workload over the new domain defined by the partition $\mathbf{B}$, with the buckets in the partition taking the place of $\mathbf{x}$. Then we have a well-studied problem of selecting unbiased measurements (i.e. linear functions of the bucket counts) in a manner that is optimized for the workload. This problem has received considerable attention in past work [45, 19, 17, 46, 76, 75]. We use the basic framework of the matrix mechanism [45], but we propose a new algorithm (described in Sec. 6.4) for efficiently approximating the optimal measurements for the workload.

Given the selected measurements, we then use the ϵ_2 privacy budget and Laplace noise to privately answer the measurement queries, followed by least-squares inference to derive the output of this step, a noisy estimate s for the buckets in $\mathbf{B}$.

Step 3: uniform expansion

In the last step we derive an estimate for the n components of $\mathbf{x}$ from the k components of the histogram $(\mathbf{B}, s)$. This is done by assuming uniformity: the count s_i for each bucket b_i is spread uniformly amongst each position of $\mathbf{x}$ that is contained in b_i. The result is the estimate $\hat{\mathbf{x}}$ for $\mathbf{x}$. Strictly speaking, any range query can be computed from $\hat{\mathbf{x}}$, but the noise is tuned to provide accuracy for precisely the queries in the workload.

The following example illustrates a sample execution of the DAWA algorithm.

Example 6.1. *For $n = 10$, Fig. 6.1 shows graphically a sample data vector $\mathbf{x} = (2, 3, 8, 1, 0, 2, 0, 4, 2, 4)$.*

- *A possible output of Step 1 is* $\mathbf{B} = \{b_1, b_2, b_3, b_4\}$ *where* $b_1 = [1,2]$, $b_2 = [3,3]$, $b_3 = [4,7]$, *and* $b_4 = [8,10]$. *This need not be the optimal partition, as defined in Sec. 6.3, because the partition selection is randomized. For the sample database* $\mathbf{x}$ *in the figure, the true bucket counts for the partition would be* $(5,8,3,10)$.
- *The result from Step 2 is a set of noisy bucket counts,* $s = (6.3, 7.1, 3.6, 8.4)$.
- *Step 3 then constructs* $\hat{\mathbf{x}}$ *by assuming a uniform distribution for values within each bucket. The final output,*

  $$\hat{\mathbf{x}} = (3.15, 3.15, 7.1, .9, .9, .9, .9, 2.8, 2.8, 2.8)$$

  is shown graphically in Fig. 6.1(c).

The novelty of our approach consists of splitting the overall private estimation problem into two phases: Step 1, which is data-dependent, and Step 2, which is workload-aware. Our main technical contributions are an effective and efficient private solution to the optimization problem underlying Step 1, and an effective and efficient solution to the optimization problem underlying Step 2. We also extend our methods to two-dimensional workloads using spatial decomposition techniques.

A number of recently-proposed methods [6, 71, 17, 74] share commonalities with one or more parts of our mechanism. But each omits or simplifies an important step and/or they use sub-optimal methods for solving related subproblems. In Sec. 6.5, an extensive experimental evaluation shows that the DAWA algorithm achieves lower error than all competitors on nearly every database and setting of ϵ tested, often by a significant margin.

6.3 Private partitioning

This section describes the partitioning phase of our algorithm. The output of this stage of the algorithm is a partition $\mathbf{B}$. In Section 6.3.1, we motivate the problem and argue that the quality of a partition depends on the data. We then describe our differentially private algorithm for finding a good partition in Section 6.3.2.

This stage of the algorithm is not tuned to the workload of queries and instead tries to select buckets such that after statistics have been computed for the buckets and the histogram is uniformly expanded, the resulting $\hat{\mathbf{x}}$ is as close to $\mathbf{x}$ as possible.

6.3.1 Cost of a partition

Recall that after the partition $\mathbf{B} = \{b_1 \dots b_k\}$ has been selected, statistics are computed for each bucket. The statistic is the total count for that bucket plus random noise to ensure privacy; thus, $s_i = b_i(\mathbf{x}) + Z_i$ where Z_i is a random variable representing the noise. Once the statistics have been computed, they are uniformly expanded into $\hat{\mathbf{x}} = \mathit{expand}(\mathbf{B}, s)$, which is an estimate for $\mathbf{x}$. If bucket b_i spans the interval $[j_1, j_2]$ we use $j \in b_i$ to denote $j \in [j_1, j_2]$. After applying uniform expansion, the resulting estimate for x_j for any $j \in b_i$ is:

$$\hat{x}_j = \frac{b_i(\mathbf{x})}{|b_i|} + \frac{Z_i}{|b_i|} \tag{6.1}$$

Equation (6.1) reveals that the accuracy of this estimate depends on two factors. First, the bucket size determines the scale of the noise. A fixed amount of noise is added to the bucket, so larger buckets have less noise per individual $\hat{x}_j$. The second factor is the degree of uniformity within the bucket—i.e., how close each x_j is to the mean value of the bucket $\frac{b_i(\mathbf{x})}{|b_i|}$. The more uniform the data values in the bucket, the more accurate the estimate $\hat{x}_j$.

We can translate these observations about $\hat{x}_j$ into a bound on the expected error of $\hat{\mathbf{x}}$. The bound depends on the amount the bucket *deviates* from being perfectly uniform. For a given bucket b_i, let *dev* be a function that measures the total absolute deviation:

$$\mathit{dev}(\mathbf{x}, b_i) = \sum_{j \in b_i} \left| x_j - \frac{b_i(\mathbf{x})}{|b_i|} \right| \tag{6.2}$$

The bound on the expected error of $\hat{\mathbf{x}}$ is in terms of deviation and the error due to added noise.

Proposition 6.1. *Given histogram $H = (\mathbf{B}, s)$ where $|\mathbf{B}| = k$ and for $i = 1 \dots k$, $s_i = b_i(\mathbf{x}) + Z_i$ where Z_i is a random variable. The uniform expansion, $\hat{\mathbf{x}} = expand(\mathbf{B}, s)$, has expected error of at most,*

$$\mathbb{E} \|\hat{\mathbf{x}} - \mathbf{x}\|_1 \le \sum_{i=1}^{k} dev(\mathbf{x}, b_i) + \sum_{i=1}^{k} \mathbb{E}|Z_i| \tag{6.3}$$

The proof of this bound follows from (6.1) and the fact that $|a+b| \le |a| + |b|$. Proof of a similar result is given in Acs et al. [6].

Proposition 6.1 reveals that the expected error of a histogram can be decomposed into two components: (a) *approximation error* due to approximating each x_j in the interval by the mean value $\frac{b_i(\mathbf{x})}{|b_i|}$ and (b) *perturbation error* due to the addition of random noise. The perturbation component is phrased in terms of random variables Z_i, which are not fully determined until the second stage of our algorithm. For the moment, let's make the simplifying assumption that $Z_i \sim \text{Laplace}(1/\epsilon_2)$. Then, $\sum_{i=1}^{k} \mathbb{E}|Z_i|$ would simplify to k/ϵ_2. Thus, to decrease perturbation error, we want as few buckets as possible; to decrease approximation error, we want buckets that are as close to uniform as possible. The optimal choice is going to depend on the uniformity of the particular dataset $\mathbf{x}$. (It also depends on ϵ_2 as smaller ϵ_2 increases perturbation error and makes deviation within the bucket relatively more tolerable.)

This analysis motivates our strategy for selecting a partition we will use Proposition 6.1 as the basis for a cost function and then find the partition with least cost.

Definition 6.2 (Cost of partition)**.** *Given a partition of the domain into buckets $\mathbf{B} = \{b_1, \dots, b_k\}$, the cost of this partition is*

$$pcost(\mathbf{x}, \mathbf{B}) = \sum_{i=1}^{k} dev(\mathbf{x}, b_i) \ + \ k/\epsilon_2 \tag{6.4}$$

This cost function is based on the simplifying assumption that $Z_i \sim \text{Laplace}(1/\epsilon_2)$. In fact, in our algorithm, the random variable Z_i is a weighted combination of Laplace random variables, where the weights are tuned to the workload in the second stage of the algorithm. Although we do not know the weights until the second stage completes, we do know that the weights are selected in such a way that $\mathbb{E}|Z_i| \geq 1/\epsilon_2$. The implication is that our choice of cost function is conservative in the sense that it will lead to a partition that is more fine grained than the partition that would have been selected with full knowledge of the noise distribution selected in the second stage.

Example 6.2. *Recall the partition* $\mathbf{B} = \{b_1, b_2, b_3, b_4\}$ *in Fig. 6.1 and assume* $\epsilon_2 = 1.0$.

- $b_1 = [1, 2]$, $\frac{b_1(\mathbf{x})}{|b_1|} = \frac{5}{2}$, $dev(\mathbf{x}, b_1) = \frac{1}{2} + \frac{1}{2} = 1$
- $b_2 = [3, 3]$, $\frac{b_2(\mathbf{x})}{|b_2|} = \frac{8}{1}$, $dev(\mathbf{x}, b_2) = 0$
- $b_3 = [4, 7]$, $\frac{b_3(\mathbf{x})}{|b_3|} = \frac{3}{4}$, $dev(\mathbf{x}, b_3) = \frac{1}{4} + \frac{3}{4} + \frac{5}{4} + \frac{3}{4} = 3$
- $b_4 = [8, 10]$, $\frac{b_4(\mathbf{x})}{|b_4|} = \frac{10}{3}$, $dev(\mathbf{x}, b_4) = \frac{2}{3} + \frac{4}{3} + \frac{2}{3} = 2\frac{2}{3}$

Therefore, $pcost(\mathbf{x}, B) = 6\frac{2}{3} + 4/\epsilon_2 = 10\frac{2}{3}$. *In comparison, the cost of partitioning* $\mathbf{x}$ *as a single bucket* $[1, 10]$ *leads to a deviation of 17.2 and total pcost of 18.2. Thus* B *is a lower cost partition and intuitively it captures the structure of* $\mathbf{x}$ *which has four regions of roughly uniform density. But note that with a more stringent privacy budget of* $\epsilon_2 = 0.1$, *the single-bucket partition would have lower cost. When* $\epsilon_2 = 0.1$, *the amount of noise added to the statistics will be on the order of* ± 10. *With this much noise, differences between the counts in* $\mathbf{x}$ *is too small to make it worth partitioning.*

Given this cost function, we can now formally state the problem that the first stage of the algorithm aims to solve.

Problem 6.1 (Least Cost Partition Problem)**.** *The least cost partition problem is to find the partition that minimizes the following objective:*

$$\begin{aligned}
&\underset{\mathbf{B}\subseteq\mathcal{B}}{\textit{minimize}} && \textit{pcost}(I,\mathbf{B})\\
&\textit{subject to} && \forall\, b, b' \in \mathbf{B}, b \cap b' = \varnothing\\
& && \bigcup_{b\in\mathbf{B}} b = [1,n]
\end{aligned}$$

where $\mathcal{B}$ is the set of all possible intervals $\mathcal{B} = \{[i,j] \mid 1 \le i \le j \le n\}$ and the constraints ensure that the collection of buckets $\mathbf{B}$ partitions [1,n].

The next section describes our algorithm for solving this optimization problem in a differentially private manner.

6.3.2 Finding a least cost partition

Since partition cost is data-dependent, we cannot solve Problem 6.1 exactly without violating privacy. Instead, we must introduce sufficient randomness to ensure differential privacy. Although it is theoretically possible to solve this problem using the exponential mechanism, in which a partition is sampled with probability proportional to $-\frac{\epsilon}{2\Delta_{pcost}} pcost(I,\mathbf{B})$, this approach is impractical as it requires enumerating all 2^{n-1} possible partitions.

Our approach is much more efficient and almost as simple. Our main contribution is in showing that this simple approach is in fact differentially private. In terms of computational efficiency, the main bottleneck is computing the partition cost. By storing intermediate results in a balanced tree, we show that we can achieve a runtime of $O(n^2 \log n)$. We can even further reduce the runtime to $O(n \log^2 n)$ by considering only partitions whose interval lengths are a power a two. Experiments in Section 6.5 suggest that this approximation has very little cost in terms of solution quality.

Our approach is based on the observation that the cost of a partition decomposes into a cost per bucket.

Definition 6.3. *Let bcost* $: \mathbb{Z}_{\geq 0}^n \times \wp([1,n]) \to \mathbb{R}_{\geq 0}$ *be a function that measures the cost of an individual bucket. For bucket* b*, the bucket cost is*

$$bcost(\mathbf{x}, b) = dev(\mathbf{x}, b) + 1/\epsilon_2.$$

For any $\mathbf{B}$, the partition cost is simply the sum of the bucket costs: $pcost(\mathbf{x}, \mathbf{B}) = \sum_{b \in \mathbf{B}} bcost(\mathbf{x}, b)$. Since the partition cost is a sum of individual bucket costs, to solve Problem 6.1, the only interaction with the private database necessary is in computing the cost of each individual bucket.

Our algorithm, which is shown in Algorithm 6.3.1, has three simple steps. First, it computes the cost for all possible buckets. There are at most $O(n^2)$ buckets and at most $O(n \log n)$ buckets if we restrict to buckets whose length is a power of two. The most obvious way to do this would require $O(n)$ time per bucket but, as described later, AllCosts is more efficient requiring only $O(\log n)$ time per bucket. Second, the algorithm adds noise to these costs. In the third and final step, it finds the partition with the least *noisy* cost. The LeastCostPartition takes the noisy costs and computes the least cost partition using dynamic programming, much like classical algorithms for v-optimal histograms [41].

We analyze the algorithm along three key dimensions: privacy, accuracy, and computational efficiency.

6.3.2.1 Privacy

The proof of privacy is a main challenge. Analyzing the privacy requires some subtlety in the sense that the noise by itself is not enough to guarantee privacy. To be precise, publishing the noisy costs of all buckets would violate differential privacy (unless the scale of the noise was inflated to be $\Omega(n)$). However, when the actual noisy costs are kept secret and the only published output is the partition with the least (noisy) cost, then a small amount of noise is sufficient to ensure privacy. The noise

Program 6.3.1 Private partition for intervals and L_1 cost function

```
procedure PRIVATE PARTITION($I$, $\epsilon_1$, $\epsilon_2$)
    // Let $\mathcal{B}$ be the set of all intervals on $[1, n]$
    // Compute cost $bcost(\mathbf{x}, b)$ for all $b \in \mathcal{B}$
    $cost \leftarrow$ ALLCOSTS($I, \epsilon_2$)
    // Add noise to each bucket cost
    for $b \in \mathcal{B}$ do
        $cost[b] \leftarrow cost[b] + \text{Laplace}(2\Delta_{bcost}/\epsilon_1)$
    end for
    // Find $\mathbf{B}$ with lowest total cost based on noisy bucket costs
    // stored in $cost$
    $\mathbf{B} \leftarrow$ LEASTCOSTPARTITION($\mathcal{B}, cost$)
    return $\mathbf{B}$
end procedure
```

is proportional to the sensitivity of computing bucket cost, but it is straightforward to show that $\bar{\Delta}_{bcost} \leq 2$.

Theorem 6.1. *Algorithm 6.3.1 is ϵ_1-differentially private.*

The intuition behind the privacy guarantee is similar to the intuition behind the exponential mechanism's privacy. The exponential mechanism secretly scores each item in a set of items, and then publishes the "best" item, where the best is selected in a noisy way. Our algorithm publishes the "best" partition, but keeps secret the noisy scores that were used to determine the best. In both cases, what matters is the the probability that a particular item is selected as the best, and this probability changes only slightly when the database changes by a tuple. Although the intuition is the same, the analysis in the proof is quite different.

Before giving the proof of Theorem 6.1, we first state two lemmas.

Lemma 4. *Let Z be a Laplace random variable with scale λ. For any z and any constant $c > 0$, we have*

$$P(Z < z - c) \geq e^{-c/\lambda} P(Z < z)$$

Lemma 5. *For any neighboring databases I_0, I_1 and any H, the cost of H can differ by at most $\bar{\Delta}_c$:*

$$\bar{\Delta}_c + \sum_{j \in H} c(I_1, B_j) \geq + \sum_{j \in H} c(I_0, B_j)$$

Proof. I_1 and I_0 differ by one record. There is exactly one bucket B_i in H that covers that record and the score of that bucket can increase by at most $\bar{\Delta}_c$. □

Proof of Theorem 6.1. For convenience of the proof, we change the notation slightly. First, we index the set $\mathcal{B}$. Let $\mathcal{B} = \{B_1, \ldots, B_M\}$ where $M = |\mathcal{B}|$. We can think of a histogram H as simply a subset of $[1, M]$, corresponding to a selection of buckets from $\mathcal{B}$. Formally, $H = \{i_1, \ldots, i_k\}$ is a histogram if for all $i, i' \in H$, we have $B_i \cap B_{i'} = \emptyset$ and $\bigcup_{i \in H} B_i = \mathbb{N}^n$. Let $\mathcal{H}$ be the set of all possible histograms that can be formed from $\mathcal{B}$.

Second, let $\mathbf{Z} = (Z_1, \ldots, Z_M)$ where for each $i \in [1, M]$, the random variable $Z_i \sim \text{Laplace}(\lambda)$ represents the noise added to the bucket cost for bucket B_i. Let $\mathbf{z} = (z_1, \ldots, z_M) \in \mathbb{R}^M$ denote a possible outcome (an assignment of $\mathbf{Z}$). We use $\mathbf{z}^{-i}$ as shorthand for $(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n)$.

Using this new notation, we observe that Algorithm 6.3.1, when run on input I, will output histogram H if and only if it is the histogram with the lowest noisy cost:

$$\sum_{j \in H} c(I, B_j) + Z_j < \min_{H' \in \mathcal{H} - \{H\}} \left\{ \sum_{k \in H'} c(I, B_k) + Z_k \right\}$$

Let I_0, I_1 be any pair of neighboring databases and let $H \in \mathcal{H}$ be any output of the algorithm. It suffices to prove

$$\frac{P(\mathcal{A}(I_0) = H)}{P(\mathcal{A}(I_1) = H)} \leq e^{\epsilon}$$

where $\mathcal{A}(I)$ denotes the algorithm running on input I and the probability distribution is over random variables $\mathbf{Z}$.

Since H defines a collection of disjoint buckets, there must be at most one $i \in H$ where $c(I_0, B_i) \neq c(I_1, B_i)$. (If the cost of H is the same on both databases, let i be any $i \in H$.) We will now derive an expression for the probability that H is selected that focuses on the effect of Z_i, the random noise added to the cost of bucket B_i. To focus on B_i, it will be convenient to partition the space of possible histograms into those that include bucket B_i and those that do not. Let $\mathcal{H}^+ = \{H \mid H \in \mathcal{H} \text{ and } i \in H\}$ and let $\mathcal{H}^- = \mathcal{H} - \mathcal{H}^+$.

We wish to define the set of outcomes $\mathbf{z} \in \mathbb{R}^M$ that cause H to be selected. H will be selected if and only if (a) H is the least cost histogram in $\mathcal{H}^+$ and (b) H has lower cost than any histogram in $\mathcal{H}^-$. We examine these two conditions in turn.

For condition (a), observe that all histograms in $\mathcal{H}^+$ use bucket B_i, thus whether (a) holds is independent of the outcome of Z_i since it has the same effect on the scores of all $H' \in \mathcal{H}^+$. Let $\phi : \mathbb{Z}^n_{\geq 0} \times \mathbb{R}^{M-1} \to \{\text{true}, \text{false}\}$ be a predicate that is true if and only if the assignment of $\mathbf{z}^{-i}$ makes H the least cost histogram among $\mathcal{H}^+$.

$$\begin{aligned} &\phi(I, \mathbf{z}^{-i}) \\ &= \sum_{j \in H - \{i\}} c(I, B_j) + z_j < \min_{H' \in \mathcal{H}^+ - \{H\}} \left\{ \sum_{k \in H' - \{i\}} c(I, B_k) + z_k \right\} \end{aligned}$$

Since I_0 and I_1 only differ in the score assigned to bucket B_i, $\phi(I_0, \mathbf{z}^{-i}) = \phi(I_1, \mathbf{z}^{-i})$ for all $\mathbf{z}^{-i} \in \mathbb{R}^{M-1}$.

For condition (b), let $\psi : \mathbb{Z}^n_{\geq 0} \times \mathbb{R}^M \to \{\text{true}, \text{false}\}$ be a predicate that is true if and only if the assignment of $\mathbf{z}$ makes H a lower cost histogram than any histogram in $\mathcal{H}^-$. A key insight is that if we fix $\mathbf{z}^{-i}$, then H will have lower cost provided that z_i is small enough.

$$\begin{aligned}
&\psi(I,\mathbf{z}) \\
&= \sum_{j\in H} c(I,B_j) + z_j < \min_{H'\in\mathcal{H}^-}\left\{\sum_{k\in H'} c(I,B_k) + z_k\right\} \\
&= z_i < \min_{H'\in\mathcal{H}^-}\left\{\sum_{k\in H'} c(I,B_k) + z_k\right\} - \sum_{j\in H} c(I,B_j) - \sum_{\ell\in H-\{i\}} z_\ell \\
&= z_i < C(I,\mathbf{z}^{-i})
\end{aligned}$$

The upper bound $C(I,\mathbf{z}^{-i})$ depends on the database. Given Lemma 5, we can say that for neighboring databases I_0 and I_1,

$$C(I_1,\mathbf{z}^{-i}) \geq C(I_0,\mathbf{z}^{-i}) - \Delta_c$$

because the score of the minimum cost histogram in $\mathcal{H}^-$ could decrease by at most Δ_c and the cost of H could increase by at most Δ_c.

We can now express the probability that the algorithm on input I outputs H in terms of ϕ and ψ. Let $f_{\mathbf{Z}}$ (respectively f_Z) denote the density function for a multivariate (respectively univariate) Laplace random variable.

$$\begin{aligned}
&P(\mathcal{A}(I) = H) = P(\phi(I,\mathbf{Z}^{-i}) \wedge \psi(I,\mathbf{Z})) \\
&= \int \mathbf{I}\left[\phi(I,\mathbf{z}^{-i}) \wedge \psi(I,\mathbf{z})\right] f_{\mathbf{Z}}(\mathbf{z})\mathrm{d}\mathbf{z} \\
&= \int \mathbf{I}\left[\phi(I,\mathbf{z}^{-i})\right] f_{\mathbf{Z}^{-i}}(\mathbf{z}^{-i})\left(\int \mathbf{I}\left[\psi(I,\mathbf{z})\right] f_{Z_i}(z_i)\mathrm{d}z_i\right)\mathrm{d}\mathbf{z}^{-i} \\
&= \int \mathbf{I}\left[\phi(I,\mathbf{z}^{-i})\right] f_{\mathbf{Z}^{-i}}(\mathbf{z}^{-i}) P(Z_i < C(I,\mathbf{z}^{-i}))\mathrm{d}\mathbf{z}^{-i}
\end{aligned}$$

Since $P(Z_i < C)$ decreases with decreasing C, we have for neighboring databases I_0 and I_1 and any $\mathbf{z}^{-i} \in \mathbb{R}^{M-1}$ that

$$
\begin{aligned}
&P(Z_i < C(I_1, \mathbf{z}^{-i})) \\
&\geq P(Z_i < C(I_0, \mathbf{z}^{-i}) - 2\bar{\Delta}_c) \\
&\geq e^{-2\bar{\Delta}_c/\lambda} P(Z_i < C(I_0, \mathbf{z}^{-i})) && \text{(by Lemma 4)}
\end{aligned}
$$

In addition, we observed earlier that $\phi(I_0, \mathbf{z}^{-i}) = \phi(I_1, \mathbf{z}^{-i})$ for all $\mathbf{z}^{-i} \in \mathbb{R}^{M-1}$. Therefore, we can express a lower bound for $P(\mathcal{A}(I_1) = H)$ strictly in terms of I_0:

$$
\begin{aligned}
&P(\mathcal{A}(I_1) = H) \\
&\geq \int \mathbf{I}\left[\phi(I_0, \mathbf{z}^{-i})\right] f_{\mathbf{Z}^{-i}}(\mathbf{z}^{-i}) e^{-2\bar{\Delta}_c/\lambda} P(Z_i < C(I_0, \mathbf{z}^{-i})) \mathrm{d}\mathbf{z}^{-i}
\end{aligned}
$$

which implies that

$$
\frac{P(\mathcal{A}(I_0) = H)}{P(\mathcal{A}(I_1) = H)} \leq e^{2\bar{\Delta}_c/\lambda} = e^{\epsilon}
$$

since, according the algorithm description, $\lambda = 2\bar{\Delta}_c/\epsilon$. □

Remark In Algorithm 6.3.1 we can reduce the noise from $2\Delta_{bcost}$ to Δ_{bcost} plus the sensitivity of the particular bucket. The benefit is a reduction in noise (by at most a factor of 2) for some buckets. This optimization is used in the experiments.

6.3.2.2 Accuracy

Accuracy is measured in terms of the difference in cost between the selected partition and the optimal choice (ignoring privacy).

Definition 6.4 (Useful approximation)**.** *A randomized algorithm is a (t, δ)-approximation if with probability at least $1 - \delta$, the algorithm returns a solution with cost at most $OPT + t$ where OPT is the cost of the least cost solution.*

We give the following bound on the accuracy of this algorithm.

Theorem 6.2. *Algorithm 6.3.1 is a (t,δ)-approximation with*

$$t = \frac{4\Delta_c\, n \log(|\mathcal{B}|/\delta)}{\epsilon}$$

In addition to a theoretical analysis, we do an extensive empirical evaluation in Section 6.5.

6.3.2.3 Efficiency

The runtime of Algorithm 6.3.1 is $O(n^2 \log n)$. The runtime can be reduced to $O(n \log^2 n)$ by considering only buckets whose lengths are a power of 2.

The dynamic program that computes the least cost partition is efficient, requiring time linear in n and the number of buckets. The computationally challenging part is computing bucket costs (Definition 6.2). When compared to computing costs for a v-optimal histogram (which is based on an L_2 metric), computing the costs for the L_1 metric used in this paper is more complicated because the cost can not be easily decomposed into sum and sum of squares terms.

To simplify the computation, the key idea is to remove the absolute values so that the computation can be decomposed to partial sums of x_j. For bucket b_i, let $I^+ = \left\{ j \mid j \in b_i \text{ and } x_j \geq \frac{b_i(\mathbf{x})}{|b_i|} \right\}$ and $I^- = b_i - I^+$. We can simplify $dev(\mathbf{x}, b_i)$ as follows:

$$\begin{aligned}
dev(\mathbf{x}, b_i) &= \sum_{j \in b_i} \left| x_j - \frac{b_i(\mathbf{x})}{|b_i|} \right| \\
&= \sum_{j \in I^+} \left(x_j - \frac{b_i(\mathbf{x})}{|b_i|} \right) + \sum_{j \in I^-} \left(\frac{b_i(\mathbf{x})}{|b_i|} - x_j \right) \\
&= 2 \sum_{j \in I^+} \left(x_j - \frac{b_i(\mathbf{x})}{|b_i|} \right) \\
&= 2 \sum_{j \in I^+} x_j - |I^+| \cdot \frac{b_i(\mathbf{x})}{|b_i|}
\end{aligned}$$

For bucket b_i, the total deviation can be computed knowing only $|I^+|$, the number of x_j who are larger than $\frac{b_i(\mathbf{x})}{|b_i|}$, and the sum of x_j for $j \in I^+$. Those quantities can be efficiently computed using a binary search tree of $x_{j_1}, \ldots, x_{j_2}$. Each node t in the tree records a value (x_t). In addition, each node t stores the sum of all values in its subtree (Σ_t), and the number of nodes in its subtree (c_t). For any constant a, we can then compute $\sum_{t \in T, x_t \geq a}(x_t - a)$ via binary search.

To compute the bucket costs for all intervals with length ℓ, we can dynamically update the search tree. After the cost for interval $[j, j+\ell]$ has been computed, we can update the tree to compute interval $[j+1, j+\ell+1]$ by removing x_j from the tree and adding $x_{j+\ell+1}$. By using a balanced binary search tree, computing all intervals of size ℓ requires only $O(n \log n)$ time. If we want to compute *all* intervals, then the total runtime is $O(n^2 \log n)$, but if we restrict to intervals whose length is a power of two, then the total runtime is $O(n \log^2 n)$.

This restriction on the length of the intervals is an approximation that has the potential to exclude the optimal solution. Empirically, we find that the big gain in efficiency does not lead to significant losses in accuracy: the algorithm remains almost as accurate as when it uses all intervals, and is always more accurate than competing techniques (Section 6.5.3). Furthermore, reducing the runtime to $O(n \log^2 n)$ makes it feasible to run on larger datasets.

6.4 Private bucket count estimation

Given the partition $B = \{b_1, \ldots, b_k\}$ determined by Step 1, it remains to privately estimate counts for each bucket, using budget ϵ_2. Thus the goal of this stage is to produce $s = s_1 \ldots s_k$. Naive solutions like adding Laplace noise to each bucket count result in high error for many workloads. In this section, we show how to adapt the existing framework of the matrix mechanism [45] to create a workload-adaptive algorithm for computing the bucket counts. Within this framework, we describe a novel greedy algorithm for minimizing error of the workload queries.

6.4.1 Transformation to the bucket domain

The first challenge in adapting to the workload is that W is expressed as queries on $\mathbf{x}$ whereas we ultimately must answer these queries using only the k statistics for the buckets in B. Recall that the statistics will be uniformly expanded into an estimate $\hat{\mathbf{x}}$, thus, any query can be answered after applying uniform expansion. However, we can equivalently transform the workload W into a new workload $\hat{W}$ that consists of queries on s. We describe this next.

Given a vector of statistics $s = s_1 \dots s_k$ for the corresponding buckets B, an estimate for the data vector $\mathbf{x}$ can be constructed by uniform expansion $\hat{\mathbf{x}} = expand(\mathbf{B}, s)$:

$$\hat{x}_i = \frac{s_j}{|b_j|}, \quad i \in b_j.$$

Given a query $q = (q_1, \dots, q_n)$ on $\mathbf{x}$, an estimated answer to q, $q(\hat{\mathbf{x}})$, is computed as

$$q(\hat{\mathbf{x}}) = \sum_{j=1}^{k} \sum_{i \in b_j} q_i \frac{s_j}{|b_j|} = \sum_{j=1}^{k} \left(\sum_{i \in b_j} \frac{q_i}{|b_j|} \right) s_j. \tag{6.5}$$

According to equation (6.5), query q on $\hat{\mathbf{x}}$ can be converted to a query on s, as described in the following proposition.

Proposition 6.2. *Let* $\hat{q} = (\hat{q}_1, \dots, \hat{q}_k)$ *where*

$$\hat{q}_j = \sum_{i \in b_j} \frac{q_i}{|b_j|}.$$

Evaluating the q on the uniform expansion of s is the same as evaluating the $\hat{q}$ over s directly—i.e., $q(\hat{\mathbf{x}}) = \hat{q}(s)$.

According to Proposition 6.2, given a workload W on $\mathbf{x}$, each of its queries can be converted to a query of s. We can then have a workload $\hat{W}$ on s such that the answers to W on the uniform expansion of s is the same as answer to $\hat{W}$ over s. The perturbation error of answering W is equivalent to the L_1 error of answering $\hat{W}$ on s. Therefore, instead of working on W and $\mathbf{x}$, this stage of the algorithm focuses on estimating $\hat{W}$ over s with the least error. In addition, since s is constructed by a data-aware histogram, this stage of the algorithm is data-independent.

Our approach relies on the matrix mechanism [45], which provides a framework to answer a batch of linear queries (i.e. a workload). Instead of answering the workload directly, the matrix mechanism poses another set of queries, called the query strategy, and uses the Laplace mechanism to obtain noisy answers. These noisy answers can then be used to derive an estimated data vector using ordinary least squares. The answers to the workload can then be computed form the estimated data vector. Key to the matrix mechanism is to find a query strategy, which is not necessarily the same as the workload, so as to minimize the mean square error of answering the workload.

In general, finding the query strategy that minimizes the error under the matrix mechanisms yields high complexity optimization problems [45, 76]. Hence we adapt an idea from prior work [46, 75]: fix the strategy to one that is well-suited for anticipated workloads, but then adjust the privacy budget so as to maximize accuracy on the specific workload. Essentially, queries in the strategy that play a significant role in answering the workload are given more weight. Since our anticipated workload is range queries, we adopt a hierarchical query strategy, similar to prior work [17, 40, 70].

The given query strategy is a hierarchy of queries, denoted as Y, which is a tree with branching factor t on s. Each node in the tree represents an interval query on s. Each leaf is a query of the respective entry in s. For each level of the tree, interval queries of every t nodes in the level are aggregated, and the aggregated query becomes their parent node in the upper level. Noticing that the number of nodes in each level may not be a multiple of t, the last nodes of each level is allowed to have less than t children. This aggregating process is repeated to create more levels until the topmost level only has one node, whose interval is the entire domain.

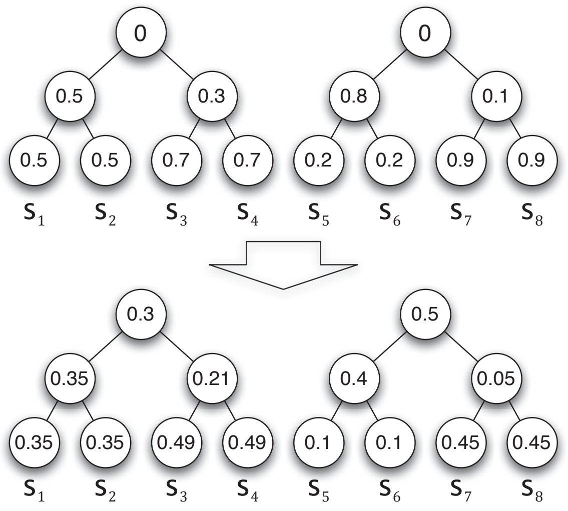


Figure 6.2. Budget allocation for the third level with $\epsilon = 1$. $\lambda = 0.3$ at the top left node, and $\lambda = 0.5$ at the top right node.

Our goal is to minimize the mean squared error of answering the workload by answering queries in Y with different privacy budgets. Although this goal is similar the goal of prior work [46, 75], their methods impose additional constraints that do not apply in our setting: either requiring the number of queries in the strategy to be no more than the size of the domain [46], or requiring the privacy budget to be equal for all queries on the same level [75].

6.4.2 Privacy budget allocation on the hierarchy

Let the privacy budget assigned to each interval query q in Y is represented by ϵ_q. We say a budget allocation satisfies privacy constraints if

$$\sum_{q(i) \neq 0, q \in Y} \epsilon_q \leq \epsilon_2, \quad i = 1, \ldots, k, \tag{6.6}$$

which means the accumulated privacy budget on any path from on Y from a leaf to the root is bounded by ϵ_2.

Since the answer of each query $\hat{q}(s)$ is a linear combination of entries in s, $\hat{q}$ can also be written as a vector $\hat{\mathbf{q}}$ such that $\hat{q}(s) = \hat{\mathbf{q}} s^T$. Similarly, we can write a workload

$\hat{W}$ as a matrix $\hat{\mathbf{W}}$, and the query hierarchy Y as a matrix $\mathbf{Y}$, where each row of $\hat{\mathbf{W}}$ and $\mathbf{Y}$ is the vector form of a query in $\hat{W}$ and Y, respectively. Given a budget distribution, $\mathbf{D}_{Y,\epsilon}$ is the diagonal matrix whose diagonal entries are the privacy budget of each query in Y.

According to the matrix mechanism, the mean squared error of answering $\hat{W}$ using Y is

$$||\hat{\mathbf{W}}(\mathbf{D}_{Y,\epsilon}^2\mathbf{Y}^T)^+||_F^2 = \text{trace}(\hat{\mathbf{W}}^T\hat{\mathbf{W}}(\mathbf{Y}^T\mathbf{D}_{Y,\epsilon}^2\mathbf{Y})^{-1}). \tag{6.7}$$

The goal of this stage of our algorithm, the optimal budget allocation problem, is defined as below.

Problem 6.2 (Optimal Budget Allocation Problem). *The optimal budget allocation problem is to find an ϵ_q for each query $q \in Y$ so as to minimize the following objective function:*

$$\begin{aligned} &\textit{minimize} && \textit{trace}(\hat{\mathbf{W}}^T\hat{\mathbf{W}}(\mathbf{Y}^T\mathbf{D}_{Y,\epsilon}^2\mathbf{Y})^{-1}) \\ &\textit{subject to} && \sum_{q(i)\neq 0, q\in Y} \epsilon_q \leq \epsilon_2, \quad i = 1,\ldots,k. \end{aligned}$$

Computing the optimal solution to Problem 6.2 appears difficult since equation (6.7) is non-convex. Instead of pursuing an optimal solution, we solve the problem approximately using the following greedy algorithm. The algorithm initially assigns all privacy budget to the leaves of Y, and determines the privacy budget of other queries in Y in a bottom-up manner. For each level of the tree, the algorithm chooses a $\lambda_q \in [0,1]$ for each query q at this level. The privacy budget is then reallocated as follows: for each query q at this level, the privacy budget on each of its descendent q' is reduced from $\epsilon_{q'}$ to $(1-\lambda_q)\epsilon_{q'}$ and the privacy budget on q is $\epsilon_q = \lambda_q\epsilon_2$. The value of λ_q is chosen to minimize Equation (6.7) after the budget reallocation. Notice that the new budget allocation still satisfies the privacy constraint in equation (6.6).

Example 6.3. *An example of the budget reallocation is shown in Figure 6.2, in which two different λ are chosen for two nodes (queries) at the third level.*

When the algorithm terminates, every query has been assigned a privacy budget. Then each query in the tree is asked using the Laplace mechanism with that query's allocated budget. After that, any inconsistencies among those noisy query answers are resolved using the ordinary least squares.

6.4.3 Efficient algorithm for budget allocation

In this section, we present how to efficiently perform the budget allocation described in the previous section. To simplify the presentation, we always assume the branching factor $t = 2$, though the discussion is valid for any branching factor.

For each interval query $q \in Y$, let $[i, j]$ be the corresponding interval of q. Use $\hat{\mathbf{W}}_q$ to denote the matrix consisting of the i^{th} to j^{th} column of $\hat{\mathbf{W}}$, and $\mathbf{Y}_q$ to denote the matrix consisting of the i^{th} to j^{th} column of the matrix of queries in the subtree of rooted at q. Let $\mathbf{D}_{q,\epsilon}$ be the diagonal matrix whose diagonal entries are ϵ'_q for all q' in the subtree rooted at q. For each query $q \in Y$ that is not on a leaf of Y, let q_1, q_2 be queries of its child nodes.

For each query $q \in Y$ that is not on a leaf of Y, according to the construction of Y, $q = q_1 + q_2$. Hence $\hat{\mathbf{W}}_q = [\hat{\mathbf{W}}_{q_1}\ \hat{\mathbf{W}}_{q_2}]$. Furthermore, since the queries in the subtree of q are the union of queries in the subtree of q_1, q_2, as well as query q itself, for a given λ_q,

$$\mathbf{D}_{q,\epsilon} = \begin{bmatrix} \lambda_q\epsilon & & \\ & (1-\lambda_q)\mathbf{D}_{q_1,\epsilon} & \\ & & (1-\lambda_q)\mathbf{D}_{q_2,\epsilon} \end{bmatrix}.$$

When choosing a λ_q, due to the fact that the privacy budgets on all ancestors of q in Y are 0 at this moment, the matrix $\mathbf{Y}^T\mathbf{D}^2_{Y,\epsilon}\mathbf{Y}$ becomes a block diagonal matrix, and $\mathbf{Y}_q^T\mathbf{D}^2_{q,\epsilon}\mathbf{Y}_q$ is one of its blocks. Therefore, the choice of λ_q only depends on $\hat{\mathbf{W}}_q$ and Y_q, which means λ_q can be determined locally, by minimizing

$$\text{trace}(\hat{\mathbf{W}}_q^T\hat{\mathbf{W}}_q(\mathbf{Y}_q^T\mathbf{D}^2_{q,\epsilon}\mathbf{Y}_q)^{-1}). \tag{6.8}$$

One of the problems of choosing λ_q using equation (6.8) is that it is biased towards q and λ_q is larger than required. When deciding the privacy budget on a query $q \in Y$, the privacy budgets on all the ancestors of q are 0. Hence the budget distribution is based on the assumption that all queries that contain q are answered by q, which is not true after some of ancestors of q are assigned non-zero budget allocations.

In order to reduce this bias, a heuristic decay factor μ is introduced to control the impact of q on queries that need to be answered with q. The following matrix is used in equation (6.8) to take the place of $\hat{\mathbf{W}}_q^T \hat{\mathbf{W}}_q$:

$$\mu \hat{\mathbf{W}}_q^T \hat{\mathbf{W}}_q + (1-\mu) \begin{bmatrix} \hat{\mathbf{W}}_{q_1}^T \hat{\mathbf{W}}_{q_1} & \\ & \hat{\mathbf{W}}_{q_2}^T \hat{\mathbf{W}}_{q_2} \end{bmatrix}. \tag{6.9}$$

As it is mentioned above, the bias of equation (6.8) comes from the assumption that the privacy budgets on all the ancestors of q are 0. Hence there will be less bias when q is more close to the root of Y. In our implementation, μ is set to be $t^{-\frac{l}{2}}$ where t is the branching factor of Y and l is the depth of q in Y. Our algorithm is then to minimize the following quantity instead of equation (6.8).

$$\operatorname{trace}\left(\left(t^{-\frac{l}{2}} \hat{\mathbf{W}}_q^T \hat{\mathbf{W}}_q + (1-t^{-\frac{l}{2}}) \begin{bmatrix} \hat{\mathbf{W}}_{q_1}^T \hat{\mathbf{W}}_{q_1} & \\ & \hat{\mathbf{W}}_{q_2}^T \hat{\mathbf{W}}_{q_2} \end{bmatrix}\right) (\mathbf{Y}_q^T \mathbf{D}_{q,\epsilon}^2 \mathbf{Y}_q)^{-1}\right). \tag{6.10}$$

In equation (6.10), $(\mathbf{Y}_q^T \mathbf{D}_{q,\epsilon}^2 \mathbf{Y}_q)^{-1}$ needs to be recomputed for each λ_q. Hence it is important to compute $(\mathbf{Y}_q^T \mathbf{D}_{q,\epsilon}^2 \mathbf{Y}_q)^{-1}$ efficiently. Let $\mathbf{M}_{q,\epsilon} = \mathbf{Y}_q^T \mathbf{D}_{q,\epsilon}^2 \mathbf{Y}_q$, and its inverse can be computed incrementally from $\mathbf{M}_{q_1,\epsilon}^{-1}$ and $\mathbf{M}_{q_2,\epsilon}^{-1}$:

$$\begin{aligned} \mathbf{M}_{q,\epsilon}^{-1} = \frac{1}{(1-\lambda_q)^2} \Bigg(& \begin{bmatrix} \mathbf{M}_{q_1,\epsilon}^{-1} & \\ & \mathbf{M}_{q_2,\epsilon}^{-1} \end{bmatrix} \\ & - \frac{\lambda_q^2}{(1-\lambda_q)^2 + \lambda_q^2 \mathbf{1}^T (\mathbf{M}_{q_1,\epsilon}^{-1} + \mathbf{M}_{q_2,\epsilon}^{-1}) \mathbf{1}} \begin{bmatrix} \mathbf{M}_{q_1,\epsilon}^{-1} \\ \mathbf{M}_{q_2,\epsilon}^{-1} \end{bmatrix} \mathbf{1}\mathbf{1}^T \begin{bmatrix} \mathbf{M}_{q_1,\epsilon}^{-1} \\ \mathbf{M}_{q_2,\epsilon}^{-1} \end{bmatrix}^T \Bigg). \end{aligned}$$

Here $\mathbf{1}$ is the column vector with all entries equal to 1. Therefore, with the following quantities given,

Program 6.4.1 Estimating bucket counts s.

```
procedure BucketCountEstimator($B$, $W$, $\mathbf{x}$, $\epsilon_2$)
    Given workload $W$ and buckets $B$, transform workload to $\hat{W}$
    Let $Y$ be a tree of queries over buckets
    Allocate all privacy budget $\epsilon_2$ to leaves of $Y$.
    for all $q \in Y$, from bottom to top do
        Compute $e_q$, $a_q$, $a'_q$, $d_q$.
        Numerically find $\lambda_q$ that minimizing Equation (6.11).
        Compute $\mathbf{D}_{q,\epsilon}$ according to $\lambda_q$.
        Compute $\mathbf{M}^{-1}_{q,\epsilon}$ and trace($\hat{\mathbf{W}}_q^T\hat{\mathbf{W}}_q\mathbf{M}^{-1}_{q,\epsilon}$) according to $\lambda_q$.
    end for
    For each $q \in Y$, let $\epsilon_q$ be the corresponding entry of $\mathbf{D}_{q,\epsilon}$.
    Let $\mathbf{y}$ be the vector of $q(B(\mathbf{x})) + \text{Laplace}(1/\epsilon_q)$ for all $q \in Y$.
    return $s = \mathbf{M}^{-1}_{q,\epsilon}(\mathbf{D}_{Y,\epsilon}\mathbf{Y})^T\mathbf{y}$
end procedure
```

$$\begin{aligned}
e_q &= \text{trace}\left(\hat{\mathbf{W}}_q^T\hat{\mathbf{W}}_q\begin{bmatrix}\mathbf{M}^{-1}_{q_1,\epsilon} & \\ & \mathbf{M}^{-1}_{q_2,\epsilon}\end{bmatrix}\right)\\
&= \text{trace}(\hat{\mathbf{W}}_{q_1}^T\hat{\mathbf{W}}_{q_1}\mathbf{M}^{-1}_{q_1,\epsilon}) + \text{trace}(\hat{\mathbf{W}}_{q_2}^T\hat{\mathbf{W}}_{q_2}\mathbf{M}^{-1}_{q_2,\epsilon}),\\
a_q &= \left\|\hat{\mathbf{W}}_q\begin{bmatrix}\mathbf{M}^{-1}_{q_1,\epsilon}\\ \mathbf{M}^{-1}_{q_2,\epsilon}\end{bmatrix}\mathbf{1}\right\|_2^2,\\
a'_q &= \|\hat{\mathbf{W}}_{q_1}\mathbf{M}^{-1}_{q_1,\epsilon}\mathbf{1}\|_2^2 + \|\hat{\mathbf{W}}_{q_2}\mathbf{M}^{-1}_{q_2,\epsilon}\mathbf{1}\|_2^2,\\
d_q &= \mathbf{1}^T(\mathbf{M}^{-1}_{q_1,\epsilon} + \mathbf{M}^{-1}_{q_2,\epsilon})\mathbf{1},
\end{aligned}$$

Equation (6.10) can be computed as:

$$\begin{aligned}
&\text{trace}\left(\left(t^{-\frac{l}{2}}\hat{\mathbf{W}}_q^T\hat{\mathbf{W}}_q + (1-t^{-\frac{l}{2}})\begin{bmatrix}\hat{\mathbf{W}}_{q_1}^T\hat{\mathbf{W}}_{q_1} & \\ & \hat{\mathbf{W}}_{q_2}^T\hat{\mathbf{W}}_{q_2}\end{bmatrix}\right)\mathbf{M}^{-1}_{q,\epsilon}\right)\\
&= \frac{1}{(1-\lambda_q)^2}\left(e_q - \frac{t^{-\frac{l}{2}}a_q + (1-t^{-\frac{l}{2}})a'_q}{(1-\lambda_q)^2 + \lambda_q^2 d_q}\lambda_q^2\right), \qquad (6.11)
\end{aligned}$$

for any λ_q in $O(1)$ time.

The entire process of computing bucket statistics is summarized in Algorithm 6.4.1.

Theorem 6.3. *Algorithm 6.4.1 is ϵ_2-differentially private.*

Proof. Since the only step in Algorithm 6.4.1 is computing $\mathbf{y}$, it is sufficient to prove that the computation of $\mathbf{y}$ satisfies ϵ_2-differential privacy. Recall the budget distribution in each loop of Algorithm 6.4.1 satisfies the constraint in equation (6.6). Hence, after the budget allocation, the maximum sum of the privacy budget in a path from a leaf of Y to the root of Y is at most ϵ_2. Moreover, for any queries in $q \in Y$, $\Delta_(q) = 1$ since each q queries the sum over some entries of $\mathbf{B}(\mathbf{x})$. Given the parallel and the sequential composition properties of differential privacy, a simple inductive argument shows that the tree of Laplace mechanism invocations is ϵ_2-differentially private. □

Theorem 6.4. *Algorithm 6.4.1 takes $O(mk \log k + k^2)$ time. In the worst case, $k = O(n)$, and Algorithm 6.4.1 takes $O(mn \log n + n^2)$ time.*

Proof. Recall $[i, j]$ is used to denote the interval that corresponding to q. Incrementally computing e_q, a_q, a'_q, d_q, $\mathbf{M}_{q,\epsilon}^{-1}$ and $tr(\mathbf{W}_q^{s\,T}\mathbf{W}_q^s\mathbf{M}_{q,\epsilon}^{-1})$ takes $O(1)$, $O(m(j-i+1)+(j-i+1)^2)$, $O((j-i+1)^2)$, $O((j-i+1)^2)$, $O((j-i+1)^2)$ and $O(1)$, respectively.

The intermediate results in Algorithm 6.4.1 can also accelerate the least square process. Since we have already computed $\mathbf{M}_{q,\epsilon}^{-1}$ in the loop of Algorithm 6.4.1, applying the ordinary least square method (the last step of Algorithm 6.4.1) only takes $O(k^2)$ time instead of $O(k^3)$ time in general cases. Summing the costs together proves the theorem. □

6.5 Experimental evaluation

The performance of DAWA is evaluated in this section. We start with a comparison to recently-proposed algorithms and evaluate each on multiple datasets and workloads (Section 6.5.2). We then examine the effectiveness of each of the two main steps of our algorithm (Sections 6.5.3 & 6.5.4). Finally, we also consider an extension of our technique to two-dimensional spatial data and compare it against the state-of-the-art for that setting (Section 6.5.5).

6.5.1 Experimental setup

In the experiments that follow, the primary metric for evaluation is the average L_1 error per query for answering the given workload queries[1]. Most workloads we use are generated randomly (as described below). Each experimental configuration is repeated on 5 random workloads with 3 trials for each workload. The results reported are the average across workloads and trials. The random workloads are generated once and used for all experiments.

The privacy budget in DAWA is set as $\epsilon_1 = 0.25\epsilon$ and $\epsilon_2 = 0.75\epsilon$. Unless otherwise specified, the first step of DAWA constructs a partition using intervals whose lengths must be a power of 2, an approximation that is described in Section 6.3. For the second step of the algorithm, the branching factor of the query tree is set to 2.

6.5.1.1 Datasets

There are seven different 1-dimensional datasets considered in our experiments. Although these datasets are publicly available, many of them describe a kind of data that could be potentially sensitive, including financial, medical, social, and search data. Adult is derived from U.S. Census data [8]: the histogram is built on the "capital loss" attribute, which is the same attribute used in [38]. Income is based on the IPUMS American community survey data from 2001-2011; the histogram attribute is personal income [62]. Medical Cost is a histogram of personal medical expenses based on a national home and hospice care survey from 2007 [65]. Nettrace is derived from an IP-level network trace collected at the gateway router of a major university. The histogram attribute is the IP address of internal hosts and so the histogram reports the number of external connections made by each internal host during the trace [40]. Search Logs is a dataset extracted from search query logs that

[1]This error measurement is different with previous chapters, since the algorithms are no longer instances of the matrix mechanism and are tuned to have better performance with L_1 error.

reports the frequency of the search term "Obama" over time (from 2004 to 2010) [40]. Furthermore, we consider two temporal datasets derived from two different kinds of network data. HepPh is a citation network among high energy physics pre-prints on arXiv and Patent is a citation network among a subset of US patents [3]. These last datasets describe public data but serve as a proxy for social network data, which can be highly sensitive. For both datasets, the histogram reports the number of new incoming links at each time stamp. To eliminate the impact of domain size in comparing the "hardness" of different datasets, all datasets above are aggregated so that the domain size n is 4096.

6.5.1.2 Query workloads

We run experiments on four different kinds of workloads. The *identity* workload consists of all unit-length intervals $[1, 1], [2, 2], \ldots, [n, n]$. The *uniform interval* workload samples 2000 interval queries uniformly at random. In addition, workloads that are not uniformly distributed over the domain are also included. The *clustered interval* workload first samples five numbers uniformly from $[1, n]$ to represent five cluster centers. For each cluster, 400 interval queries are sampled as follows: the left and right boundaries of each interval are sampled from the left and right halves, respectively, of a normal distribution with a standard deviation 256. The *large clustered interval* workload is generated in the same way but the standard deviation is 1024.

6.5.1.3 Competing algorithms

We compare DAWA with six competing algorithms. The following two are data-independent:

- **Identity [26]** uses the Laplace mechanism to release noisy counts for the individual unit-length intervals.

- **Privelet** [70] applies the Haar wavelet transform to the original data, adds Laplace noise to the wavelet coefficients and transforms the noisy coefficients back. When compared to identity, Privelet has been shown to achieve lower error on large interval queries.

We also compare with the following four data-dependent algorithms.

- **MWEM** [38] uses the exponential mechanism and multiplicative weights to iteratively refine an estimate of the data. A key parameter in the algorithm is the number of iterations T, which controls the number of measurements taken and has a significant impact on performance. Since an optimal strategy for selecting T is unknown, we empirically search for the best T on each dataset. We set T to the value in $\{10, 20, \ldots, 190, 200\}$ that achieves the lowest error when the workload is *uniform intervals* and $\epsilon = 0.1$. We use that T for all experiments on that dataset. Tuning T to the dataset benefits MWEM and results in higher accuracy than if T were fixed or set by a differentially private procedure.
- **EFPA** [6] is a method that applies a Fourier transform and then selects a subset of the Fourier coefficients using the exponential mechanism. Noisy coefficients are obtained and an estimate for the original dataset is computed via an inverse Fourier transformation.
- **P-HP** [6] uses the exponential mechanism to recursively bisect the domain into subintervals. The algorithm is designed to find a partition with minimal cost, where cost is defined exactly as in this paper (Definition 6.2).

StructureFirst [74] is motivated by classical work in V-optimal histograms. It is designed to construct a partition of the domain into k bins (where k is fixed) such that the variance within each bin is minimized. Rather than compute a

Nettrace	Adult	Medical Cost	Search Logs	Income	Patents	HepPh
22	29	20	500	1537	1870	2168

Table 6.1. Number of buckets in the optimal partition for each dataset when $\epsilon = 0.1$.

(a) Smallest ratio across datasets

ϵ	Identity	Privelet	MWEM	EFPA	P-HP	S. First
0.01	2.04	1.00	2.65	0.88	2.20	0.86
0.05	2.27	1.11	3.00	3.20	1.98	1.01
0.1	2.00	0.98	2.54	3.84	1.81	0.82
0.5	2.06	1.01	3.39	3.60	1.25	0.89

(b) Largest ratio across datasets

ϵ	Identity	Privelet	MWEM	EFPA	P-HP	S.First
0.01	26.42	12.93	17.00	18.94	7.09	10.85
0.05	22.97	11.24	19.14	43.58	17.57	9.77
0.1	20.85	10.20	22.54	41.09	31.41	8.32
0.5	25.47	12.46	68.75	43.69	138.14	10.89

Table 6.2. Ratio of algorithm error to DAWA error, for each competing algorithm and ϵ setting on *uniform intervals*: (a) smallest ratio observed across datasets; (b) largest ratio across datasets.

single statistic for each bin, it instead takes multiple measurements at varying granularity, using a hierarchical strategy.[2]

Among the algorithms introduced above, MWEM is the only algorithm that is also workload-aware. We use code from the original authors for EFPA, P-HP, and StructureFirst; we implemented the other algorithms based on the pseudocode described in the paper.

6.5.2 Accuracy on interval workloads

Figure 6.3 presents the main error comparison of all algorithms on workloads of *uniform intervals* across a range of datasets and settings of ϵ. The y-axis is the

[2]The other algorithms from Xu et al. [74] take more than 20 hours to complete a single trial. Therefore, they are not included.

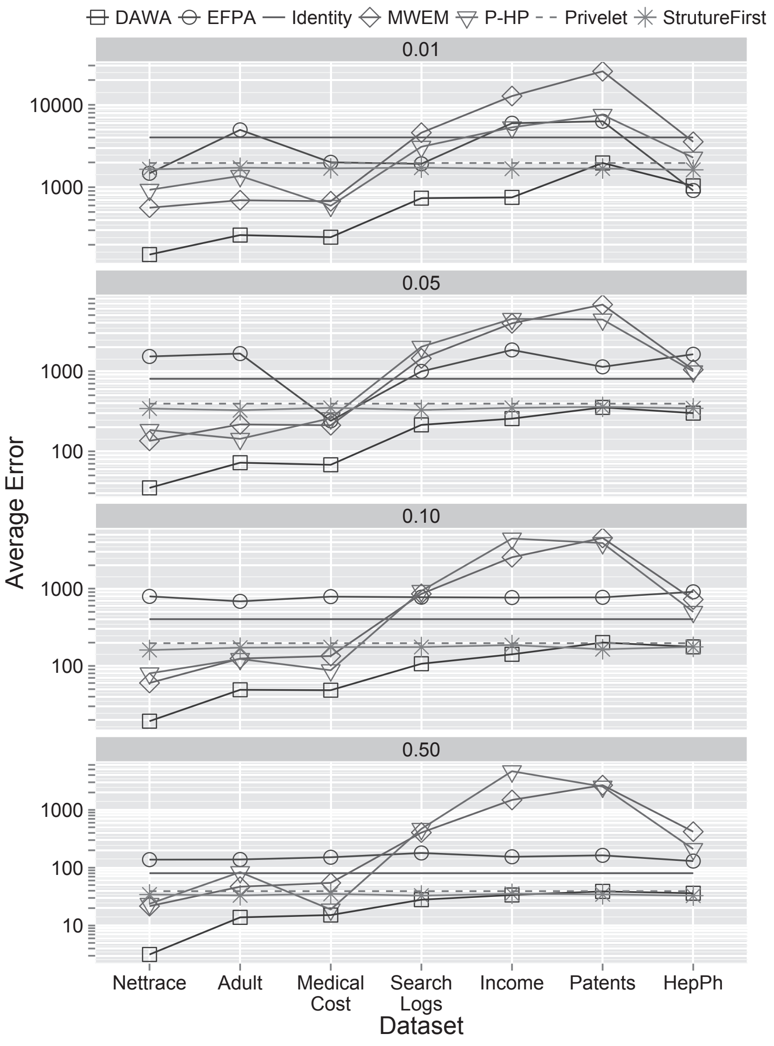


Figure 6.3. Average absolute error on the *uniform intervals* workload across multiple datasets. The privacy budget ranges from $\epsilon = 0.01$ (top) to $\epsilon = 0.5$ (bottom).

average error per query presented on a logarithmic scale. The datasets appear along the x-axis (in an order discussed below).

While data-independent algorithms like Privelet and Identity offer constant error across datasets, the error of data-dependent algorithms can vary significantly.[3] For

[3]StructureFirst is an exception to this trend, as its observed performance is almost totally independent of the dataset. The reason is that the algorithm chooses a partition based on a scoring function that has high sensitivity. Thus, the partition selection phase of its algorithm is very noisy and its choice of partition is close to random.

some datasets, data-dependent algorithms can be much more accurate. For example, on Nettrace with $\epsilon = 0.01$, *all* of the data-dependent algorithms have lower error than the best data-independent algorithm (Privelet). For this dataset, the error of DAWA is at least an order of magnitude lower than Privelet. These results suggest the potential power of data-dependence.

There are other datasets, however, where the competing data-dependent algorithms appear to break down. In the figure, the datasets are ordered by the cost of an optimal partition (i.e., an optimal solution to Step 1 of our algorithm) when $\epsilon_2 = 0.1$. This order appears to correlate with "hardness." Datasets on the left have low partition cost and appear to be relatively "easy," presumably because data-dependent algorithms are able to exploit uniformities in the data. However, as one moves to the right, the optimal partition cost increases and the datasets appear to get more difficult. It is on many of the "harder" datasets where competing data-dependent algorithms suffer: their error is larger than even a simple baseline approach like Identity.

In contrast, DAWA does not break down when the dataset is no longer "easy." On the moderately difficult dataset Search Logs, DAWA is the only data-dependent algorithm that outperforms data-independent algorithms. On the "hardest" datasets, its performance is comparable to data independent techniques like Privelet. DAWA comes close to achieving the best of both worlds: it offers very significant improvement on easier datasets, but even on hard datasets achieves roughly the same performance as data-independent techniques.

For the same workload, datasets, and algorithms, Table 6.2 reports the performance of DAWA relative to other algorithms. Each cell in the table reports the ratio of algorithm error to DAWA error. Table 6.2(a) reports the smallest ratio achieved over all datasets—i.e., how close the competing algorithm comes to matching, or in some cases beating, the performance of DAWA . Table 6.2(b) reports the largest ra-

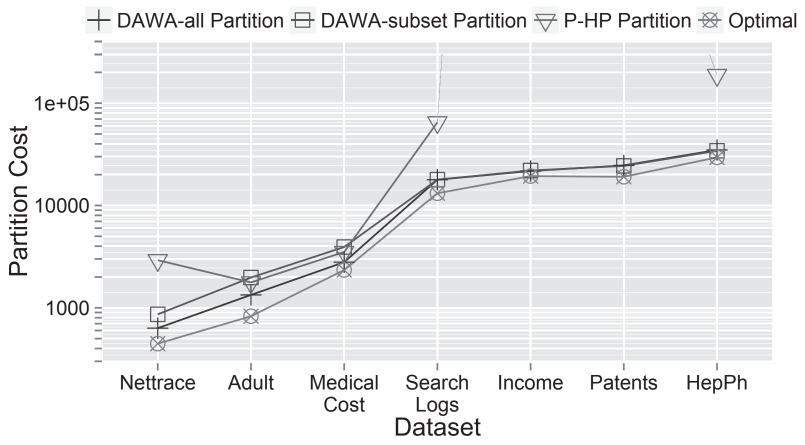


Figure 6.4. Cost of partitions selected by various algorithms with $\epsilon = 0.1$.

tio achieved—i.e., how much worse the competing algorithm can be on some dataset. Table 6.2(b) reveals that the error of every competing algorithm is at least 7.09 times larger than that DAWA on one dataset.

Table 6.2(a) reveals that DAWA is sometimes less accurate than another algorithm, but only moderately so. This occurs on the "hardest" datasets, Patents and HepPh, where DAWA has error that is at most $\frac{1}{0.82} \approx 22\%$ higher than other approaches. On these hard datasets, the optimal partition has thousands of buckets (see Table 6.1), indicating that it is highly non-uniform. On non-uniform data, the first stage of the DAWA algorithm spends ϵ_1 of the privacy budget only to select a partition that is similar to the base buckets. Despite the fact that the first stage of the algorithm does not help much on "hard" datasets, DAWA is still able to perform almost as well as the best data-independent technique, in contrast to the other data dependent strategies which perform poorly on such "hard" datasets.

In addition to *uniform interval* workload, we also ran experiments on the other three types of workloads. The performance of DAWA relative to its competitors is qualitatively similar to the performance on *uniform interval* workload shown above.

6.5.3 Effectiveness of partition selection

In addition to the strong performance of DAWA shown above, we would like to verify the effectiveness of our solutions in each of the main steps of the algorithm.

Recall that the first step of DAWA is to generate a differentially private partition of the domain using Algorithm 6.3.1. Here we evaluate the effectiveness of this first step by comparing the cost of the partition found by the algorithm to the cost of the optimal solution. The optimal solution is computed by solving Problem 6.1 directly, ignoring privacy considerations. In addition, we compare our algorithm to P-HP, which is also designed to solve Problem 6.1. To facilitate a fair comparison, for this experiment each algorithm spends all of its privacy budget on selecting the partition.

The results are shown in Figure 6.4 for $\epsilon = 0.1$. DAWA-all is the DAWA algorithm in which the partition is selected by considering all possible intervals. DAWA-subset computes the partition using the subset of intervals whose lengths are a power of two. As the figure shows, the partition of DAWA-all is close to optimal. The cost of the partition of DAWA-subset is sometimes higher than that of DAWA-all especially on "easier" datasets. Generally, however, DAWA-subset performs similarly to DAWA-all . This suggests that the gain in efficiency that comes from only considering a subset of the intervals does not come at the expense of lost utility.

The cost of the partition selected by P-HP, is almost as low as the cost of the DAWA-subset partition on the Adult and Medical Cost datasets, but is orders of magnitude larger on other datasets (on Income and Patents it is at least $1.6{\times}10^6$). This provides empirical evidence that Algorithm 6.3.1 is a much more accurate algorithm than the recursive bisection approach of P-HP. The results with $\epsilon \in \{0.01, 0.05, 0.5\}$ are similar and hence omitted.

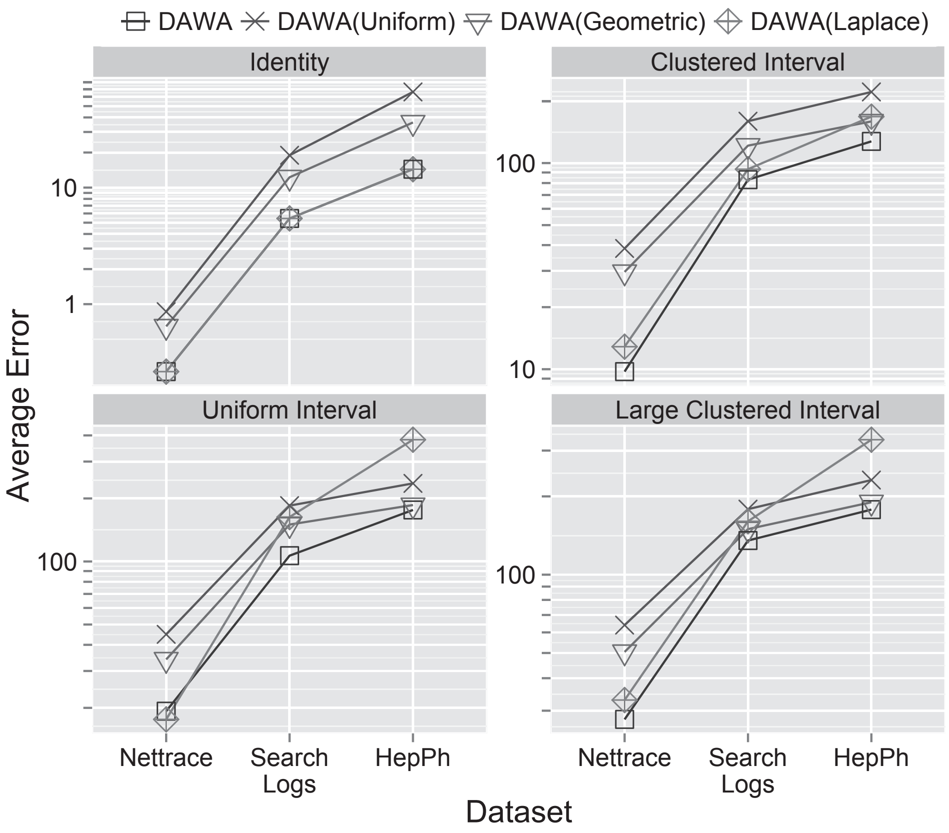


Figure 6.5. A comparison of alternative algorithms for the second step of DAWA across different workloads, with $\epsilon = 0.1$.

6.5.4 Effectiveness of adapting to workload

The second stage of DAWA designs a query strategy that is tuned to the workload, as described by Algorithm 6.4.1. Here we evaluate the effectiveness of this algorithm by comparing its performance with some alternative strategies. Two alternative ways to allocate the privacy budget on Y are considered: all queries have the same privacy budget (Uniform); the privacy budget decreases geometrically from leaves to root (Geometric) based on the approach described in [17]. The Laplace mechanism is also included as an alternative choice for the second stage. Among the alternative algorithms, Geometric Y is designed to answer *uniform interval* workloads, and the Laplace mechanism is known to be the optimal data-independent mechanism for the *identity* workload.

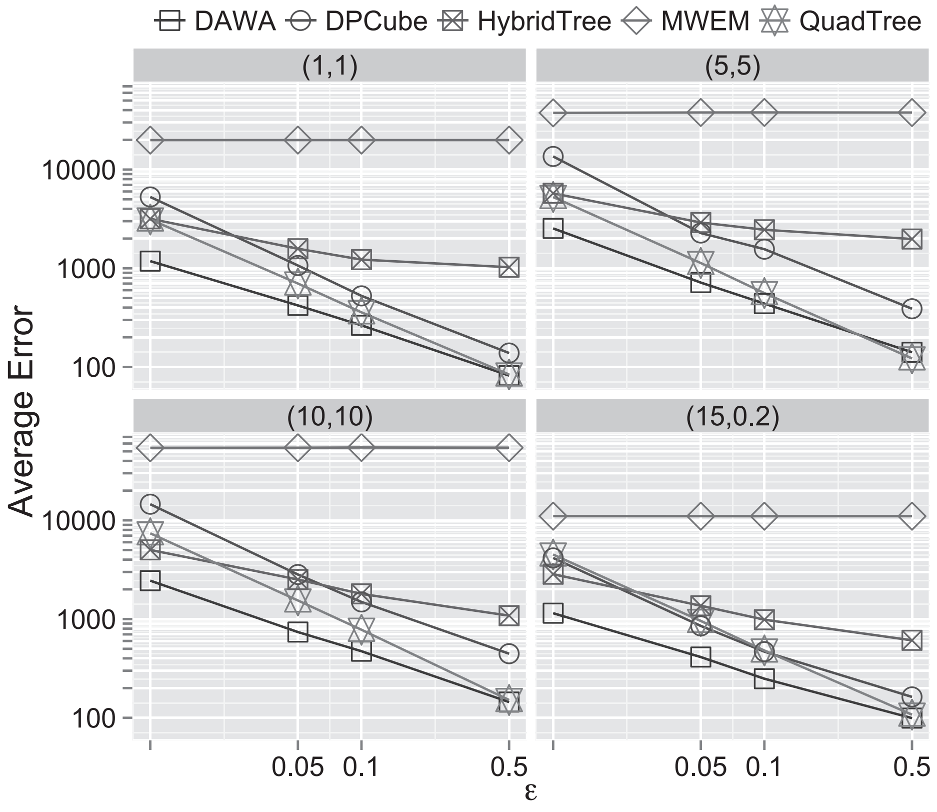


Figure 6.6. Average error answering query workloads on spatial data. Each workload is a batch of random rectangle queries of a given (x, y) shape.

The results are shown in Figure 6.5. Average error is measured across three datasets that span the range of difficulty: Nettrace ("easy" with the lowest partition cost), Search Logs ("moderate"), and HepPh ("hard"). In these experiments, $\epsilon = 0.1$. The original DAWA performs very well on all cases. In particular, it always outperforms Geometric on *uniform interval* workload and performs exactly same as the Laplace mechanism on *identity* workload. In the later case, we witness that the greedy algorithm in the second step outputs the initial budget allocation, which is exactly same as the Laplace mechanism.

6.5.5 Case study: spatial data workloads

Lastly, we evaluate an extension to our main algorithm to compute histograms over two dimensional spatial data. We use an experimental setup that is almost identical to previous work [17]; differences are highlighted below. The dataset describes the geographic coordinates (latitude and longitude) of road intersections across a wide region in the western U.S [4]. The frequency of road intersections serves as a proxy for human population. Over this region, we generate a workload of random rectangle queries of four different shapes: $(1, 1)$, $(5, 5)$, $(10, 10)$, and $(15, 0.2)$ where shape (x, y) is a rectangle that covers x degrees of longitude and y degrees of latitude.

We compare with algorithms from [17] and other recently published techniques.

- **QuadTree [17]** is a quadtree created over the domain where the privacy budget is allocated with weights that decrease geometrically from leaf to root.
- **HybridTree [17]** is a hybrid strategy that combines a differentially private kd-tree and at higher levels with a quadtree on lower levels with parameters set as described in [17].
- **DPCube [72]** obtains bin counts using the Laplace mechanism and then constructs a kd-tree based on those noisy counts.

For the above algorithms, we use code obtained from the authors. Also included in the study is MWEM (previously described). Among these algorithms, only QuadTree is data-independent and only MWEM is workload-aware. Since some algorithms expect discrete domains as input, we discretize the domain by partitioning the space into the finest granularity used by the QuadTree, whose height is 10 [17]. Thus, both longitude and latitude are split evenly into 2^{10} bins.

To extend the DAWA algorithm to two dimensional data, we use a Hilbert curve of order 20 to convert the $2^{10} \times 2^{10}$ grid into a 1-dimensional domain with size 2^{20}. In case the query region only partially covers some bins in the discretized domain, the query answer is estimated by assuming uniformity within each bin.

Figure 6.6 shows the results. Although DAWA is designed for interval workloads on 1-dimensional data, it performs as well or better than algorithms that were specifically designed to support rectangular range queries on 2-dimensional data. The performance gap between DAWA and its competitors increases as ϵ decreases. These results suggest that DAWA can work well even when the workload does not consist of interval queries. It may be possible to achieve even greater accuracy by extending the DAWA algorithm to operate directly on two-dimensional data. We leave this as future work.

CHAPTER 7
RELATED WORK

7.1 Private data analysis

The most straightforward idea in private data analysis is to remove the attributes that can directly identify each individual from each table, which was used in several releases of sensitive data. Barbaro et al. proposed an attack to the published AOL search log [10], where the partial information in search logs is used to re-identify an individual. Further, Narayanan et al. successfully identified an individual from the Netflix contest dataset [55] by joining the released information of Netflix users to the public accessible IMDB database.

To prevent linkage attacks, Sweeney et al. [63] presented k-anonymity, which divides the attributes into two groups: quasi-identifiers and sensitive information. Quasi-identifiers are defined as attributes that might be publicly available and can be used to partially identify one individual. The idea of k-anonymity is to guarantee that each quasi-identifier is shared by at least k tuples. However, there are cases that k-anonymized datasets leak private information. Many followup works on k-anonymity pointed out potential privacy leaks in previous works and proposed more sophisticated solutions. Machanavajjhala et al. presented l-diversity [50], which enforces the sensitive attributes in each of the k-anonymized group has at least l different values. M-invariance [68] and t-closeness [48], presented by Xiao et al. and Li. et al, respectively improve l-diversity by imposing more constraints to the distribution of values of sensitive attributes within each k-anonymized group. In addition, Xiao et al also introduced a new algorithm, Anatomy [69], that can be used to create a published database that satisfies any variation of k-anonymity.

Though there is no theoretical privacy guarantee with k-anonymity, the idea is widely accepted in many applications. Ganta et al. proposed an attack with auxiliary information that can be applied to all variation of k-anonymous above [31]. However, there are still more attempts to further improve k-anonymity and provide stronger privacy guarantee, such as β-likeness [16].

Other privacy definitions rely on the randomness of the output of algorithms. Dinur and Nissim [20] studied the least amount of perturbation to be added to query answers to avoid blatant non-privacy. Evfimievski et al. defined privacy by bounding the difference between the prior and posterior belief of adversaries. Blum et al. presented the SuLQ framework, which provides a privacy definition that can work against adversaries with arbitrary amount of auxiliary information. Differential privacy is presented in [26, 25] and draws lots of attentions since it has been introduced. Several systems are designed [54, 61, 52] to convert non-private programs into differentially private ones. A summary of recent theoretical works under differential privacy can be found in [24]. Recently, criticisms on differential privacy are also emerging [36, 43]. Kifer and Machanavajjhala also proposed a more general privacy framework in [44].

7.2 Linear query answering under differential privacy

There are two different types of approaches that answers linear queries under differential privacy. One is data-independent approaches, where the entire query answering process does not change with different input datasets. The existing data-independent approaches are either instances of the matrix mechanism or the matrix mechanism with some variation. Furthermore, we are not aware any data-independent approaches that work interactively with the database. The other one is data-aware approaches, where the query answering process is determined by the underlying database. Such approaches including both batch processing approaches, where all queries are answered at once and interactive approaches, where further queries can depends on answers of previous queries.

7.2.1 Data-independent approaches

The original work on the matrix mechanism is first introduced in [45]. It presents the error formula as well as the optimization problems. Originally, the matrix mechanism focuses mainly on ϵ-differential privacy, although (ϵ, δ)-differential privacy was also considered briefly. In addition, the original work also analyzes two prior techniques specifically tailored to range queries. The first uses a wavelet transformation [70]; the second uses a hierarchical set of queries followed by inference [40]. The analyses via the matrix mechanism indicate that those two techniques, though seemingly different, are actually very similar to each other. Further analyses also demonstrate that even though those techniques have much lower noise than the Laplace mechanism, none of them minimizes the amount of noise needed.

Meanwhile, other works that designed algorithms that answer marginal queries can also be considered as instances of the matrix mechanism. Each of those works designs special strategies for a specific class of workloads. Barak et al.[9] studied answering low order marginal queries using subsets of Fourier basis; Ding et al.[19] considered a special collection of marginal queries, called data cubes, which are answered by a subset of all data cube queries selected via a greedy algorithm. The algorithm adapts a known approximation algorithm for the subset sum problem and cannot be applied to general linear queries.

As follow up works to the matrix mechanism, we presented an adaptive algorithm that generates strategies for any workload under (ϵ, δ)-differential privacy in [46]. The algorithm generates different strategies by weighting queries that consist of the singular vectors of the query workload. An empirical study demonstrates that the strategies produced by this algorithm outperform previously designed algorithms on various workloads [70, 40, 9, 19]. The case of supporting low-rank workloads under ϵ-differential privacy are discussed by Yuan et al. in [76], which introduces lower error than [70, 40] when the number of queries is much smaller than the size of the domain. In order to avoid the hardness of solving the optimization problem under the matrix mechanism, [75] demonstrated an approach that has the similar form of the matrix mechanism, but relies on a fixed "recovery" matrix instead of the least square inference as the last step of the matrix mechanism. Such simplification leads to a much easier optimization formulation, though inference with the recovery matrix introduces more noise than inference with the least squares.

7.2.2 Data-aware approaches

Other works discuss algorithms whose query answering process is related to the underlying databases. Many of practical data-aware approaches are compared with DAWA in Chapter 6, which are all non-interactive approaches. Close to our work in Chapter 6, the P-HP [6] algorithm uses the same score function to choose a partition of the 1-dimensional domain. The algorithm differentially privately and recursively bisects the domain into small regions, which leads to an accumulated privacy budget cost from all levels of the recursion. In addition, the greedy bisection may lead to a suboptimal partition, and the algorithm is hence outperformed by our approach in experiments. Xu et al. [74] proposes an alternative approach (StructureFirst) where the number of partitions, k, is fixed and the algorithm aims to select a partition with minimum expected squared error (L_2), as opposed to the approach used here which is based on absolute error (L_1). We find that fixing k is disadvantageous as the optimal k is data-dependent, varying by orders of magnitude across different datasets. In addition, squared error has very high sensitivity, requiring a large amount noise to guarantee privacy and therefore resulting partitions do not capture the structure of the data very well. In addition, neither P-HP nor StrucureFirst tries to combine the dynamic strategy selection as our algorithm. There is another algorithm in [6] that dynamically chooses the granularity in a Fourier transformation based on the underlying database. Beyond 1-dimensional databases, linear query answering on 2-dimensional databases is studied in [17, 72], which create differentially private KD-trees. An alternative approach is presented in [13, 49], which, instead of perturbing the query answers, output a noisy compression of the database so as to support any kind of queries.

Further, there are more data-aware approaches from the theory community. Those approaches are both data-aware and interactive, and lead to smaller error than the matrix mechanism over sparse databases by analyzing the properties of the underlying database. Many of those theoretical approaches have been compared to the bound on the matrix mechanism in Chapter 4.4. The median mechanism [59] maintained a set of database instances that consist with historical query answers. The new query is either answered by the maintained set of the databases or by the original database, determined in a differentially private method. Dwork et al. [27] samples linear queries in each step and modifies the sample distribution with the new query answers. In [37, 35], the authors recursively update the estimated data vector to reduce the error on linear queries. In each of those algorithms, asymptotical error bounds are provided. More generally, Dwork et al. provide an error bound using an arbitrary differentially private mechanism [21] but not specifically for linear counting queries. Theoretically, those approaches have better dependency on privacy parameters and many not be applicable in practical.

7.2.3 Error bounds

The general error lower bound under the matrix mechanism was introduced in [47]. In recent work, Nikolov et al. [56] propose an algorithm whose error is within a ratio of $O(\log^2 \text{rank}(\mathbf{W}) \log(1/\delta))$ to the optimal error under *any* data-independent (ϵ, δ)-differentially private mechanism (not limited to instances of the matrix mechanism). Their algorithm is in fact a special case of the matrix mechanism, so this approximation ratio also bounds the ratio between the error lower bound under the matrix mechanism and the minimum achievable error of *all* possible data-independent (ϵ, δ)-differential private mechanisms. On ϵ-differential privacy, Hardt et al. [39] present a lower bound on error for low rank workloads. Similar to the SVD bound, this geometric bound can also be represented as a function of the singular values of the workload. The authors also provide a mechanism that is close to the lower bound on random matrices with high probability. Since the mechanism in the paper is a special case of the matrix mechanism, the result also indicates that the matrix mechanism can provide small enough error on such random matrices. In addition, this bound is not directly comparable with the SVD bound since it bounds the mean absolute error rather than mean squared error in the SVD bound.

Blum et al. [14] describe a very general mechanism for synthetic data release, in which error rates are related to the VC dimension of the workload. However, for many workloads of linear queries, VC dimension is too coarse-grained to provide a useful measure of workload error complexity. For example, the VC dimension for any workload of d-dimensional range queries that can not be embedded into $(d-1)$-dimensional spaces is always $d+1$, despite the fact that such workloads could have very different achievable error rates. Lower and upper bounds on answering all k-way marginals with a data dependent mechanism are discussed in [42]. Though it is clear that the SVD bound is tight in the case of all k-way marginals (since it is a special case of data cube) comparison with [42] requires a careful analysis of the singular values of workloads of k-way marginals and is a direction for future investigation.

CHAPTER 8
DISCUSSION AND CONCLUSION

Differential privacy draws a great amount of interest since it provides a promising privacy guarantee in theory. However, the original literature of differential privacy only provides high level ideas of the mechanism design and a mechanism (Laplace mechanism) that only works well for single query answering. With the query workload and the query strategy separated, the matrix mechanism takes the advantages of the correlation between queries to generate high quality query answers with limited privacy budget. In addition, the theoretical error analysis in the matrix mechanism makes it is possible to compare quality of different query strategies accurately without running experiments on concrete databases. As many proposed techniques [29, 47, 46, 76, 9, 70, 40, 19] can be formulated as instances of matrix mechanism with specific query strategies, the matrix mechanism allows a uniform analysis and comparison between all of those techniques.

We also discuss the optimal strategy that supports a certain workload under the matrix mechanism and show it can be computed by iteratively solving a pair of semidefinite programs. In order to cope with the high complexity of the original optimization formulation, we further proposed an approximated algorithm that generates strategies by weighting a set of designing queries, which produces strategies that outperform state-of-art mechanisms under (ϵ, δ)-differential privacy. Moreover, we present an error lower bound under the matrix mechanism, which characteristics the hardness of a workload using its spectral properties. Our error bound is proven to be tight in many cases under (ϵ, δ)-differential privacy and is shown empirically close to tight on many other commonly studied workloads. In addition, we design a data- and workload-aware algorithm for answering range queries under differential privacy. The algorithm first partitions the domain into approximately uniform regions and then estimates counts in each region using measurements of varying granularity that are tuned to the workload queries. Experimental results indicate our algorithm outperforms state-of-arts data-dependent algorithms, and is no worse than data-independent algorithms when other data-dependent algorithms yield to high error.

However, many questions are still open under the matrix mechanism and its data-dependent generalization. The first important direction is better algorithms and further analyses to the matrix mechanism under ϵ-differential privacy. Under (ϵ, δ)-differential privacy, past works have already designed an efficient workload adaptive algorithm [46], presented an almost tight error lower bound [47], and connected the error of the matrix mechanism to the error of any data-independent differentially private algorithms [56]. But there are no such general results under ϵ-differential privacy: the existing algorithms only work for specific workloads, the lower bound on the matrix mechanism is not as tight, and the connection to the general data-independent algorithms only existing on random matrices [39].

In addition, more researches on data-dependent algorithms under differential privacy is also an very important future direction. The power of data-dependent algorithms under differential privacy has already been shown in our experiments, but our current algorithm only works for 1-dimensional range queries. Existing algorithms for more general cases (e.g. MWEM) is shown to be not as good on many cases. Moreover, our experimental results indicate that some databases are significantly "easier" than other databases according to the performance of data-dependent algorithms. Formulate a measurement on the hardness of datasets or the hardness of datasets under a given workload is crucial for us understand the effectiveness of future differentially private algorithms.

BIBLIOGRAPHY

[1] http://abel.ee.ucla.edu/cvxopt/.

[2] http://dbgroup.cs.umass.edu/code/.

[3] http://snap.stanford.edu.

[4] http://www.census.gov/geo/maps-data/data/tiger.html.

[5] http://www.mcs.anl.gov/hs/software/dsdp/.

[6] Ács, Gergely, Castelluccia, Claude, and Chen, Rui. Differentially private histogram publishing through lossy compression. In *ICDM* (2012), pp. 1–10.

[7] Agrawal, S., Haritsa, J.R., and Prakash, B.A. Frapp: a framework for high-accuracy privacy-preserving mining. *Data Mining and Knowledge Discovery 18*, 1 (2009), 101–139.

[8] Bache, K., and Lichman, M. UCI machine learning repository, 2013.

[9] Barak, Boaz, Chaudhuri, Kamalika, Dwork, Cynthia, Kale, Satyen, McSherry, Frank, and Talwar, Kunal. Privacy, accuracy, and consistency too: A holistic solution to contingency table release. In *PODS* (2007).

[10] Barbaro, M., and Zeller, T. A face is exposed for aol searcher no. 4417749. *New York Times 9* (2006), 2008.

[11] Ben-Israel, A., and Greville, T.N.E. *Generalized inverses: Theory and applications*, vol. 15. Springer, 2003.

[12] Bhaskara, Aditya, Dadush, Daniel, Krishnaswamy, Ravishankar, and Talwar, Kunal. Unconditional differentially private mechanisms for linear queries. In *STOC* (New York, NY, USA, 2012), pp. 1269–1284.

[13] Blocki, Jeremiah, Blum, Avrim, Datta, Anupam, and Sheffet, Or. The johnson-lindenstrauss transform itself preserves differential privacy. In *FOCS* (2012), pp. 410–419.

[14] Blum, Avrim, Ligett, K, and Roth, Aaron. A learning theory approach to non-interactive database privacy. In *STOC* (2008), pp. 609–618.

[15] Boyd, S.P., and Vandenberghe, L. *Convex optimization.* Cambridge University Press, 2004.

[16] Cao, Jianneng, and Karras, Panagiotis. Publishing microdata with a robust privacy guarantee. *PVLDB 5*, 11 (2012), 1388–1399.

[17] Cormode, Graham, Procopiuc, Magda, Shen, Entong, Srivastava, Divesh, and Yu, Ting. Differentially private spatial decompositions. In *ICDE* (2012).

[18] Dattorro, J. *Convex optimization & Euclidean distance geometry.* Meboo Publishing USA, 2005.

[19] Ding, Bolin, Winslett, Marianne, Han, Jiawei, and Li, Zhenhui. Differentially private data cubes: optimizing noise sources and consistency. In *SIGMOD* (2011), pp. 217–228.

[20] Dinur, Irit, and Nissim, Kobbi. Revealing information while preserving privacy. In *PODS* (2003), pp. 202–210.

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

[22] Dwork, Cynthia. Differential privacy: A survey of results. In *TAMC* (2008).

[23] Dwork, Cynthia. The differential privacy frontier. In *TCC* (2009).

[24] Dwork, Cynthia. A firm foundation for private data analysis. *Communications of the ACM 54*, 1 (2011), 86–95.

[25] Dwork, Cynthia, Kenthapadi, Krishnaram, McSherry, Frank, Mironov, Ilya, and Naor, Moni. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT* (2006), pp. 486–503.

[26] Dwork, Cynthia, Nissim, Frank McSherry Kobbi, and Smith, Adam. Calibrating noise to sensitivity in private data analysis. In *TCC* (2006).

[27] Dwork, Cynthia, Rothblum, Guy N., and Vadhan, Salil P. Boosting and differential privacy. In *FOCS* (2010), pp. 51–60.

[28] Evfimievski, A., Gehrke, J., and Srikant, R. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (2003), ACM, pp. 211–222.

[29] Fawaz, Nadia, Muthukrishnan, S., and Nikolov, Aleksandar. Nearly optimal private convolution. *CoRR abs/1301.6447* (2013).

[30] Fulton, W. Eigenvalues, invariant factors, highest weights, and schubert calculus. *Bulletin of the AMS 37*, 3 (2000), 209–250.

[31] Ganta, S, Kasiviswanathan, S, and Smith, Adam. Composition attacks and auxiliary information in data privacy. In *KDD* (2008).

[32] Ghosh, Arpita, Roughgarden, T, and Sundararajan, M. Universally utility-maximizing privacy mechanisms. In *STOC* (2009).

[33] Gray, Jim, Chaudhuri, Surajit, Bosworth, Adam, Layman, Andrew, Reichart, Don, Venkatrao, Murali, Pellow, Frank, and Pirahesh, Hamid. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Min. Knowl. Discov. 1*, 1 (1997), 29–53.

[34] Gray, Robert M. *Toeplitz and circulant matrices: A review.* Now Pub, 2006.

[35] Gupta, Anupam, Roth, Aaron, and Ullman, Jonathan. Iterative constructions and private data release. In *TCC* (2012), pp. 339–356.

[36] Haeberlen, Andreas, Pierce, Benjamin C., and Narayan, Arjun. Differential privacy under fire. In *Proceedings of the 20th USENIX conference on Security* (Berkeley, CA, USA, 2011), SEC'11, USENIX Association, pp. 33–33.

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

[38] Hardt, Moritz, Ligett, Katrina, and McSherry, Frank. A simple and practical algorithm for differentially private data release. In *NIPS* (2012), pp. 2348–2356.

[39] Hardt, Moritz, and Talwar, Kunal. On the geometry of differential privacy. In *STOC* (2010), pp. 705–714.

[40] Hay, Michael, Rastogi, Vibhor, Miklau, Gerome, and Suciu, Dan. Boosting the accuracy of differentially-private histograms through consistency. *PVLDB 3*, 1-2 (2010), 1021–1032.

[41] Jagadish, H. V., Koudas, Nick, Muthukrishnan, S., Poosala, Viswanath, Sevcik, Kenneth C., and Suel, Torsten. Optimal histograms with quality guarantees. In *VLDB* (1998), pp. 275–286.

[42] Kasiviswanathan, S.P., Rudelson, M., Smith, A., and Ullman, J. The price of privately releasing contingency tables and the spectra of random matrices with correlated rows. In *STOC* (2010), pp. 775–784.

[43] Kifer, Daniel, and Machanavajjhala, Ashwin. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data* (New York, NY, USA, 2011), SIGMOD '11, ACM, pp. 193–204.

[44] Kifer, Daniel, and Machanavajjhala, Ashwin. A rigorous and customizable framework for privacy. In *Proceedings of the 31st symposium on Principles of Database Systems* (New York, NY, USA, 2012), PODS '12, ACM, pp. 77–88.

[45] Li, Chao, Hay, Michael, Rastogi, Vibhor, Miklau, Gerome, and McGregor, Andrew. Optimizing linear counting queries under differential privacy. In *PODS* (2010), pp. 123–134.

[46] Li, Chao, and Miklau, Gerome. An adaptive mechanism for accurate query answering under differential privacy. *PVLDB 5*, 6 (2012), 514–525.

[47] Li, Chao, and Miklau, Gerome. Optimal error of query sets under the differentially-private matrix mechanism. In *ICDT* (2013), pp. 272–283.

[48] Li, N., Li, T., and Venkatasubramanian, S. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on* (2007), IEEE, pp. 106–115.

[49] Li, Yang D., Zhang, Zhenjie, Winslett, Marianne, and Yang, Yin. Compressive mechanism: utilizing sparse representation in differential privacy. In *WPES* (2011), pp. 177–182.

[50] Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD) 1*, 1 (2007), 3–es.

[51] McSherry, F., and Mironov, I. Differentially Private Recommender Systems : Building Privacy into the Netflix Prize Contenders. In *SIGKDD* (2009).

[52] McSherry, Frank. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. *Commun. ACM 53*, 9 (2010), 89–97.

[53] McSherry, Frank D. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD* (2009), pp. 19–30.

[54] Mohan, Prashanth, Thakurta, Abhradeep, Shi, Elaine, and Culler, David E. Gupt: Privacy preserving data analysis made easy. In *SIGMOD* (2012).

[55] Narayanan, A., and Shmatikov, V. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on* (2008), IEEE, pp. 111–125.

[56] Nikolov, Aleksandar, Talwar, Kunal, and Zhang, Li. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing* (2013), STOC '13, pp. 351–360.

[57] Pukelsheim, Friedrich. *Optimal Design of Experiments.* Wiley & Sons, 1993.

[58] Rastogi, V., Suciu, D., and Hong, S. The boundary between privacy and utility in data publishing. In *Proceedings of the 33rd international conference on Very large data bases* (2007), VLDB Endowment, pp. 531–542.

[59] Roth, Aaron, and Roughgarden, Tim. Interactive privacy via the median mechanism. In *STOC* (2010), pp. 765–774.

[60] Roth, Aaron, and Roughgarden, Tim. The median mechanism: Interactive and efficient privacy with multiple queries. In *STOC* (2010).

[61] Roy, Indrajit, Setty, Srinath T. V., Kilzer, Ann, Shmatikov, Vitaly, and Witchel, Emmett. Airavat: Security and privacy for mapreduce. In *NSDI* (2010), pp. 297–312.

[62] Ruggles, Steven, Alexander, J., Genadek, K., Goeken, R, Schroeder, M., and Sobek, M. Integrated public use microdata series: Version 5.0, 2010.

[63] Sweeney, L., et al. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty Fuzziness and Knowledge Based Systems 10*, 5 (2002), 557–570.

[64] Tenorio, L, Fleck, A, and Moses, K. Confidence intervals for linear discrete inverse problems with a non-negativity constraint. *Inverse Problems 23*, 2 (2007), 669.

[65] United States Department of Health, Human Services. Centers for Disease Control, and Prevention. National Center for Health Statistics. National home and hospice care survey, 2007.

[66] Xiao, X., and Tao, Y. Anatomy: Simple and effective privacy preservation. In *Proceedings of the 32nd international conference on Very large data bases* (2006), VLDB Endowment, pp. 139–150.

[67] Xiao, Xiaokui, Bender, Gabriel, Hay, Michael, and Gehrke, Johannes. ireduct: Differential privacy with reduced relative errors. In *SIGMOD* (2011).

[68] Xiao, Xiaokui, and Tao, Yufei. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *SIGMOD* (2007), pp. 689–700.

[69] Xiao, Xiaokui, and Tao, Yufei. Output perturbation with query relaxation. In *VLDB* (2008).

[70] Xiao, Xiaokui, Wang, Guozhang, and Gehrke, Johannes. Differential privacy via wavelet transforms. In *ICDE* (2010), pp. 225–236.

[71] Xiao, Y., Xiong, L., and Yuan, C. Differentially private data release through multidimensional partitioning. In *SDM* (2010).

[72] Xiao, Yonghui, Gardner, James J., and Xiong, Li. Dpcube: Releasing differentially private data cubes for health information. In *ICDE* (2012), pp. 1305–1308.

[73] Xu, Jia, Zhang, Zhenjie, Xiao, Xiaokui, Yang, Yin, and Yu, Ge. Differentially private histogram publication. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on* (2012), pp. 32–43.

[74] Xu, Jia, Zhang, Zhenjie, Xiao, Xiaokui, Yang, Yin, Yu, Ge, and Winslett, Marianne. Differentially private histogram publication. *The VLDB Journal* (2013), 1–26.

[75] Yaroslavtsev, Grigory, Cormode, Graham, Procopiuc, Cecilia M., and Srivastava, Divesh. Accurate and efficient private release of datacubes and contingency tables. In *ICDE* (2013).

[76] Yuan, Ganzhao, Zhang, Zhenjie, Winslett, Marianne, Xiao, Xiaokui, Yang, Yin, and Hao, Zhifeng. Low-rank mechanism: Optimizing batch queries under differential privacy.