Journal of Machine Learning Research 17 (2016) 1-42 Submitted 9/14; Revised 9/15; Published 4/16

Differentially Private Data Releasing for Smooth Queries

Ziteng Wang WANGZT2012@GMAIL.COM
Key Laboratory of Machine Perception (MOE), School of EECS
Peking University
Beijing, 100871, China

Chi Jin CHIJIN@CS.BERKELEY.EDU
Department of Computer Science
University of California
Berkeley, CA 94720-1776, USA

Kai Fan KAI.FAN@DUKE.EDU
Computational Biology & Bioinformactics
Duke University
Durham, NC 27708, USA

Jiaqi Zhang ZHANGJQ@CIS.PKU.EDU.CN
Key Laboratory of Machine Perception (MOE), School of EECS
Peking University
Beijing, 100871, China

Junliang Huang HUANGJUNLIANG@PKU.EDU.CN
School of Mathematical Sciences
Peking University
Beijing, 100871, China

Yiqiao Zhong YIQIAOZHONG@PKU.EDU.CN
School of Mathematical Sciences
Peking University
Beijing, 100871, China

Liwei Wang WANGLW@CIS.PKU.EDU.CN
Key Laboratory of Machine Perception (MOE), School of EECS
Peking University
Beijing, 100871, China

Editor: Sanjoy Dasgupta

Abstract

In the past few years, differential privacy has become a standard concept in the area of privacy. One of the most important problems in this field is to answer queries while preserving differential privacy. In spite of extensive studies, most existing work on differentially private query answering assumes the data are *discrete* (i.e., in $\{0,1\}^d$) and focuses on queries induced by *Boolean* functions. In real applications however, *continuous* data are at least as common as binary data. Thus, in this work we explore a less studied topic, namely, differential privately query answering for continuous data with continuous function. As a first step

1. Part of this work has been appeared in (Wang et al., 2013).

towards the continuous case, we study a natural class of linear queries on continuous data which we refer to as *smooth* queries. A linear query is said to be K-smooth if it is specified by a function defined on $[-1,1]^d$ whose partial derivatives up to order K are all bounded. We develop two ϵ-differentially private mechanisms which are able to answer *all* smooth queries. The first mechanism outputs a summary of the database and can then give answers to the queries. The second mechanism is an improvement of the first one and it outputs a synthetic database. The two mechanisms both achieve an accuracy of $O(n^{-\frac{K}{2d+K}}/\epsilon)$. Here we assume that the dimension d is a constant. It turns out that even in this parameter setting (which is almost trivial in the discrete case), using existing discrete mechanisms to answer the smooth queries is difficult and requires more noise. Our mechanisms are based on L_∞-approximation of (transformed) smooth functions by low-degree even trigonometric polynomials with uniformly bounded coefficients. We also develop practically efficient variants of the mechanisms with promising experimental results.[1]

Keywords: differential privacy, smooth queries, synthetic dataset

1. Introduction

Statistical analysis and machine learning are often conducted on data sets containing sensitive information, such as medical records, commercial data, etc. The benefit of mining from such data is tremendous. But when releasing sensitive data, one must take privacy issue into consideration, and has to accept a tradeoff between the accuracy and the amount of privacy loss of the individuals in the database.

In this paper, we consider *differential privacy* (Dwork et al., 2006), which has become a standard concept in the area of privacy. Roughly speaking, a mechanism which releases information about some database is said to preserve differential privacy, if the change of a single database element does not affect the probability distribution of the output significantly. Differential privacy provides strong guarantees against attacks. It ensures that the risk of information leakage for any individual who submits her information to the database is very small, in the sense that an adversary can discover almost nothing new from the database that contains one individual's information compared with that from the database without that individual's information. Recently there have been extensive studies of machine learning, statistical estimation, and data mining under the differential privacy framework (Wasserman and Zhou, 2010; Chaudhuri et al., 2011; Lei, 2011; Kifer and Lin, 2010; Chaudhuri et al., 2012; Williams and McSherry, 2010; Jain et al., 2012; Chaudhuri and Hsu, 2011).

Accurately answering statistical queries is a well studied problem in differential privacy. A simple and efficient method is the Laplace mechanism (Dwork et al., 2006), which adds Laplace noise to the true answers. Laplace mechanism is especially useful for queries with low sensitivity, which is the maximal difference of the query values of two databases that are different in only one item. A typical class of queries that has low sensitivity is *linear* queries, whose sensitivity is $O(1/n)$, where n is the size of the database.

The Laplace mechanism has a limitation. It can answer at most $O(n^2)$ queries. If the number of queries is substantially larger than n^2, Laplace mechanism is not able to provide differentially private answers with nontrivial accuracy. Considering that potentially there are many users and each user may submit a set of queries, limiting the number of total

1. Source codes available at `http://www.cis.pku.edu.cn/faculty/vision/wangliwei/software.html`

queries to be smaller than n^2 is too restricted in some situations. A remarkable result due to Blum, Ligett and Roth (Blum et al., 2008) (will be referred to as BLR in this paper) shows that: information theoretically it is possible for a mechanism to answer far more than n^2 linear queries while preserving differential privacy and nontrivial accuracy simultaneously.

There is a series of works (Dwork et al., 2009, 2010; Roth and Roughgarden, 2010; Hardt and Rothblum, 2010) improving the result of (Blum et al., 2008). All these mechanisms are very powerful in the sense that they can answer general and adversely chosen queries. On the other hand, even the fastest algorithms for query answering (Hardt and Rothblum, 2010; Hardt et al., 2012a) run in time linear in the size of the data universe. Often the size of the data universe is much larger than that of the database, so these mechanisms are inefficient. Recently, Ullman (2013) shows that there is no polynomial time algorithm that can answer $n^{2+o(1)}$ *general* linear queries while preserving privacy and accuracy (assuming the existence of one-way function).

Recently, there are growing interests in studying differentially private mechanisms for *restricted* classes of queries, in particular queries useful in applications. One class of queries that attracts a lot of attentions are the k-way conjunctions. The data universe for this problem is $\{0, 1\}^d$. Thus each individual record has d binary attributes. A k-way conjunction query is specified by k features. The query asks what fraction of the individual records in the database has all these k features being 1. A series of works attack this problem using several different techniques (Barak et al., 2007; Gupta et al., 2011; Cheraghchi et al., 2012; Hardt et al., 2012b; Thaler et al., 2012) . They proposed elegant mechanisms which run in time poly(n) when k is a constant. Another class of queries that yields efficient mechanisms is the class of sparse query. A query is m-sparse if it takes non-zero values on at most m elements in the data universe. Blum and Roth (2013) developed mechanisms which are efficient when $m = \text{poly}(n)$.

The above differentially private mechanisms can also be categorized into two classes according to the output of the algorithm. The first class of algorithms output answers to the queries. The second class of algorithms output synthetic databases instead; the answers of the queries can then be simply computed from the synthetic database. The Laplace mechanism belongs to the first class. BLR and the offline version of the Private Multiplicative Weight updating (PMW) mechanism (Hardt and Rothblum, 2010; Hardt et al., 2012a) output synthetic database. From a practical point of view, the synthetic dataset output is appealing. In fact, before the notion of differential privacy was proposed, almost all practical techniques developed to preserve privacy against certain types of attacks output a synthetic dataset by modifying the raw dataset (please see the survey Aggarwal and Yu 2008 and the references therein).

Among the studies of differentially private query answering, most existing works focus on binary data and queries induced by Boolean functions[2]. In real applications however, continuous data are at least as common as binary data; and one has to use continuous functions in this scenario instead of Boolean functions. In this paper, we explore this relatively less studied topic, i.e., differentially private query answering with continuous

2. On the other hand, there are many results (e.g., Chaudhuri et al. 2011, Williams and McSherry 2010, Jain et al. 2012, Chaudhuri and Hsu 2011) on differentially private learning that study continuous data and continuous function.

functions on continuous data. We assume that 1) the data universe is $\mathcal{X} = [-1,1]^d$; 2) a linear query q_f is induced by a continuous function $f : \mathcal{X} \to \mathbb{R}$.

As will be clear soon, answering general linear queries in the continuous setting is considerably more difficult than that in the binary case. Therefore we will focus our analysis on a restricted class of linear queries. The first question is: which subclass of linear queries is commonly used in practice? Let $D = \{x_1, \ldots, x_n\}$ ($x_i \in [-1,1]^d$) be a database consisting of continuous data. A linear query q_f on D is defined as $q_f(D) := \frac{1}{n}\sum_{x \in D} f(x)$, where $f : [-1,1]^d \to \mathbb{R}$ is a continuous function. Thus, to find out a natural class of linear queries, one only needs to find out a natural class of continuous functions.

The set of continuous functions considered in this paper is the set of smooth queries. We say a function $f : [-1,1]^d \to \mathbb{R}$ is K-smooth, if all the partial derivatives of f up to the Kth order are bounded. We say a linear query q_f is K-smooth if f is a K-smooth function. We believe smooth function is one of the most natural and useful classes of continuous functions; and therefore the set of smooth queries is a natural class of linear queries worth studying. Our aim is to answer *all* smooth queries while preserving differential privacy and accuracy. See also section 4.2 for an explanation of why answering all smooth queries is useful for differentially private machine learning. As a first step towards differential privacy for continuous query, we study in this work the case where the dimension d of the data universe $[-1,1]^d$ is a constant. It turns out that the smooth query problem is very challenging even for $d = O(1)$ which is almost trivial in the discrete case (see below).

We develop two mechanisms for the smooth query. The two mechanisms both achieve accuracy of $O\left(\left(\frac{1}{n}\right)^{\frac{K}{2d+K}}/\epsilon\right)$ for *all* K-smooth queries. If the order of smoothness K is large compared to the dimension d, then the errors of the mechanisms are close to n^{-1}.

To be more concrete, the former mechanism outputs a summary of the database. To obtain an answer of a smooth query, the user runs a public evaluation procedure which contains no information of the database. Outputting the summary has running time $O\left(n^{1+\frac{d}{2d+K}}\right)$, and the evaluation procedure for answering a query runs in time $\tilde{O}(n^{\frac{d+2+\frac{2d}{K}}{2d+K}})$. It can be seen that if K is large compared to d, then the mechanism runs in almost linear time. For our second mechanism which outputs synthetic database, the running time is $O(n^{\frac{3dK+5d}{4d+2K}})$, polynomial in the size of the database. Theoretically, the second mechanism is far less efficient as the first one. This is reasonable since outputting synthetic database is much harder than outputting answers. We then develop practically efficient variants of this mechanism.

As a comparison, we will consider how to apply existing mechanisms (e.g., PMW, BLR) to solve the smooth query problem. As our goal is to answer all smooth queries while preserving privacy and accuracy, and because there are infinitely many K-smooth functions, one has to discretize the set of smooth functions before using any existing mechanism. It is not clear how to efficiently discretize this set of queries. More importantly, we show that even if one can discretize this query set in an optimal way, there is a lower bound for the number of discretized queries of this set. The lower bound is exponential in $1/\alpha$, where α is the desired error of the query answering mechanism; and as a result, the best existing mechanism has significantly larger error than that of the algorithms proposed in this paper. (See Section 3.2.1 and Proposition 18 in Section 4.1 for details.)

The mechanisms proposed in this paper are based on L_∞-approximation and is motivated by Thaler et al. (2012), which considers approximation of k-way conjunctions by low degree polynomials. Our basic idea is to approximate the whole query class by linear combination of a small set of basis functions. The main technical difficulty lies in that in order that the approximation induces an efficient and differentially private mechanism, all the linear coefficients of the basis functions must be small. To guarantee this, we first transform the query function. Then by using even trigonometric polynomials as basis functions we prove a constant upper bound for the linear coefficients. It is worth pointing out that to guarantee accuracy, we must consider L_∞-approximation. It is completely different from L_2-approximation, which is simply Fourier analysis when using trigonometric polynomial as basis. Another difficulty for the first mechanism is how to efficiently compute the linear coefficients. It turns out that the smoothness of the functions allows us to use an efficient numerical method to compute the coefficients to a precision so that the accuracy of the mechanism is not affected significantly.

We also point out that the basis functions described above have some relation to conjunctions. In fact, conjunctions can also be viewed as smooth functions as they are multilinear polynomials. Conjunctions form a small subset of smooth functions. Moreover, the set of conjunctions is also a strict subset of the basis function used in our mechanism. If we restrict to conjunctions, our algorithm essentially reduces to Laplace mechanism (see Section 3.2.2 for details). But different to all existing works dealing with conjunctions, our aim here is to answer all smooth queries while preserving privacy and accuracy.

Finally we conduct experiments on the efficient variant of the mechanism which outputs synthetic database as it may be more useful in practice. Experimental results demonstrate that the algorithm achieves good accuracy and are practically efficient on datasets of various sizes and of a number of attributes.

This work is a small step towards an understanding of smooth queries. Our mechanisms have obvious limitations: The performances decrease exponentially with respect to the data dimension d. Thus the algorithms cannot handle the case in which d is a super constant. The experimental results also demonstrate that our mechanisms work well mostly when d is no more than a few hundred. Studying the general case $d = \omega(1)$ is our future work.

The rest of the paper is organized as follows. Section 2 gives the background and all the definitions. In Section 3, we propose the query-answering mechanism. In Section 4, we give the mechanism which is able to output synthetic database. In Section 5, we develop a practical variant of the second mechanism and conduct experiments to evaluate its performance. Finally, we conclude in Section 6.

2. Preliminaries

Let D be a database containing n data points in the data universe $\mathcal{X}$. In this paper, we consider the case that $\mathcal{X} \subset \mathbb{R}^d$ where d is a constant. Typically, we assume that the data universe $\mathcal{X} = [-1, 1]^d$. Two databases D and D' are called neighbors if $|D| = |D'| = n$ and they differ in exactly one data point. The following is the formal definition of differential privacy.

Definition 1 ((ϵ, δ)-differential privacy) *A randomized mechanism $\mathcal{M}$ whose output lies in some range $\mathcal{S}$ is said to preserve (ϵ, δ)-differential privacy if for all measurable $S \in \mathcal{S}$,* *for all pairs of neighbor databases D, D', the following holds:*

$$\mathbb{P}(\mathcal{M}(D) \in S) \leq \mathbb{P}(\mathcal{M}(D') \in S) \cdot e^{\epsilon} + \delta,$$

where the probability is taken over the randomness of the $\mathcal{M}$. If $\mathcal{M}$ preserves $(\epsilon, 0)$-differential privacy, we say $\mathcal{M}$ is ϵ-differentially private.

We consider *linear queries.* Each linear query q_f is specified by a function f which maps the data universe $[-1,1]^d$ to $\mathbb{R}$. q_f is defined as

$$q_f(D) := \frac{1}{|D|} \sum_{\mathbf{x} \in D} f(\mathbf{x}).$$

Let Q be a set of queries. The accuracy of a mechanism with respect to Q is defined as follows.

Definition 2 ((α, β)-accuracy) *Let Q be a set of queries. A mechanism $\mathcal{M}$ is said to have (α, β)-accuracy for size n databases with respect to Q, if for every database D with $|D| = n$ the following holds*

$$\mathbb{P}(\exists q \in Q, \quad |\mathcal{M}(D, q) - q(D)| \geq \alpha) \leq \beta,$$

where $\mathcal{M}(D, q)$ is the answer to q given by $\mathcal{M}$, and the probability is over the internal randomness of the mechanism $\mathcal{M}$.

We will make use of the Laplace mechanism (Dwork et al., 2006) in our algorithm. Laplace mechanism adds Laplace noise to the output. We denote by $\text{Lap}(\sigma)$ the random variable distributed according to the Laplace distribution with parameter σ, whose density function is $\frac{1}{2\sigma}\exp(-|x|/\sigma)$.

Next, we formally define smooth queries, which is a special class of linear queries. Since each linear query q_f is specified by a function f, a set of queries Q_F can be specified by a set of functions F. Remember that each $f \in F$ maps $[-1,1]^d$ to $\mathbb{R}$. For any point $\mathbf{x} = (x_1, \ldots, x_d) \in [-1,1]^d$, if $\mathbf{k} = (k_1, \ldots, k_d)$ is a d-tuple of nonnegative integers, then we define

$$D^{\mathbf{k}} := D_1^{k_1} \cdots D_d^{k_d} := \frac{\partial^{k_1}}{\partial x_1^{k_1}} \cdots \frac{\partial^{k_d}}{\partial x_d^{k_d}}.$$

Let $|\mathbf{k}| := k_1 + \ldots + k_d$. Define the K-norm as

$$\|f\|_K := \sup_{|\mathbf{k}| \leq K} \sup_{\mathbf{x} \in [-1,1]^d} |D^{\mathbf{k}} f(\mathbf{x})|.$$

We will study the set C_B^K which contains all *smooth* functions whose derivatives up to order K have ∞-norm upper bounded by a constant $B > 0$. Formally,

$$C_B^K := \{f : \quad \|f\|_K \leq B\}.$$

The set of queries specified by C_B^K, denoted as $Q_{C_B^K}$, is our focus. Smooth functions have been studied in depth in machine learning (van der Vart and Wellner, 1996; Wahba et al., 1999; Smola et al., 1998; Wang, 2011). See Section 4.2 for examples of smooth functions and its relation to differentially private machine learning.

In this work we will frequently use *trigonometric polynomials*. For the univariate case, a function $p(\theta)$ is called a trigonometric polynomial of degree m if

$$p(\theta) = a_0 + \sum_{r=1}^{m} \left(a_r \cos r\theta + b_r \sin r\theta\right),$$

where a_r, b_r are constants. If $p(\theta)$ is an even function, we say that it is an even trigonometric polynomial, and

$$p(\theta) = a_0 + \sum_{r=1}^{m} a_r \cos r\theta.$$

For the multivariate case, if

$$p(\theta_1, \ldots, \theta_d) = \sum_{\mathbf{r}=(r_1,\ldots,r_d)} a_{\mathbf{r}} \cos(r_1\theta_1)\ldots\cos(r_d\theta_d),$$

then p is said to be an even trigonometric polynomial (with respect to each variable), and the degree of θ_i is $\max_{i\le d}\{r_i\}$.

3. The First Mechanism: Query Answering

In this section we propose our first mechanism which outputs answers to the queries. Our second mechanism which is able to output synthetic database will be given in the next section.

3.1 Mechanism and Main Results

The following theorem is our first main result. It says that if the query class is specified by smooth functions, then there is an efficient mechanism for query answering which preserves ϵ-differential privacy and good accuracy. The mechanism consists of two parts: One for outputting a summary of the database, the other for answering a query. The two parts are described in Algorithm 1 and Algorithm 2 respectively. The second part of the mechanism contains no private information of the database.

Theorem 3 *Let the query set be*

$$Q_{C_B^K} := \{q_f = \frac{1}{n}\sum_{\mathbf{x}\in D} f(\mathbf{x}) : \quad f \in C_B^K\},$$

where $K \in \mathbb{N}$ and $B > 0$ are constants. Let the data universe be $[-1,1]^d$, where $d \in \mathbb{N}$ is a constant. Then the mechanism $\mathcal{M}$ given in Algorithm 1 and Algorithm 2 satisfies that for any $\epsilon > 0$, the followings hold:

1) The mechanism is ϵ-differentially private.

Algorithm 1 Outputting the summary

Notations: $\mathcal{T}_t^d := \{0, 1, \dots, t-1\}^d$, $\mathbf{x} := (x_1, \cdots, x_d)$, $\theta_i(\mathbf{x}) := \arccos(x_i)$.
Parameters: Privacy parameters $\epsilon, \delta > 0$, Failure probability $\beta > 0$,
Smoothness order $K \in \mathbb{N}$.
Input: Database $D \in \left([-1,1]^d\right)^n$
Output: A t^d-dimensional vector as the summary $\hat{\mathbf{b}}$.

```
1: Set $t = \lceil n^{\frac{1}{2d+K}} \rceil$.
2: for all $\mathbf{r} = (r_1, \dots, r_d) \in \mathcal{T}_t^d$ do
3:     $b_{\mathbf{r}} \leftarrow \frac{1}{n} \sum_{\mathbf{x} \in D} \cos(r_1\theta_1(\mathbf{x})) \dots \cos(r_d\theta_d(\mathbf{x}))$
4:     $\hat{b}_{\mathbf{r}} \leftarrow b_{\mathbf{r}} + \mathrm{Lap}\left(\frac{t^d}{n\epsilon}\right)$
5: end for
6: $\hat{\mathbf{b}} \leftarrow (\hat{b}_{\mathbf{r}})_{\|\mathbf{r}\|_\infty \le t-1}$ ($\hat{\mathbf{b}}$ is a $t^d$ dimensional vector)
7: return: $\hat{\mathbf{b}}$
```

Algorithm 2 Answering a query

Input: A query q_f, where $f : [-1,1]^d \to \mathbb{R}$ and $f \in C_B^K$,
Summary $\hat{\mathbf{b}}$ returned by Algorithm 1
Output: Approximate answer to $q_f(D)$.

```
1: Set $t = \lceil n^{\frac{1}{2d+K}} \rceil$.
2: Let $g_f(\boldsymbol{\theta}) = f(\cos(\theta_1), \dots, \cos(\theta_d))$, $\boldsymbol{\theta} = (\theta_1, \dots, \theta_d) \in [-\pi, \pi]^d$.
3: Compute the a trigonometric polynomial approximation $p_t(\boldsymbol{\theta})$ of $g_f(\boldsymbol{\theta})$, where
   $p_t(\boldsymbol{\theta}) = \sum_{\mathbf{r}=(r_1,\dots,r_d), \|\mathbf{r}\|_\infty \le t-1} c_{\mathbf{r}} \cos(r_1\theta_1) \dots \cos(r_d\theta_d)$. (see Section 3.3 for details)
4: $\mathbf{c} \leftarrow (c_{\mathbf{r}})_{\|\mathbf{r}\|_\infty \le t-1}$ ($\mathbf{c}$ is a $t^d$ dimensional vector)
5: return: $\mathbf{c} \cdot \hat{\mathbf{b}}$
```

2) For any $\beta \ge 10 \cdot e^{-\frac{1}{5}(n^{\frac{d}{2d+K}})}$ the mechanism is (α, β)-accurate, where $\alpha = O\left(n^{-\frac{K}{2d+K}}/\epsilon\right)$, and the hidden constant depends only on d, K and B.

3) The running time for $\mathcal{M}$ to output the summary is $O(n^{\frac{3d+K}{2d+K}})$.

4) The running time for $\mathcal{M}$ to answer a query is $O(n^{\frac{d+2+\frac{2d}{K}}{2d+K}} \cdot \mathrm{polylog}(n))$.

The proof of Theorem 3 is essentially based on Theorem 4 given below. The detailed proof of Theorem 3 is given in the Section 3.4.

To have a better idea of how the performances depend on the order of smoothness, let us consider three cases. The first case is $K = 1$, i.e., the query functions only have the first order derivatives. Another extreme case is $K \gg d$, and we assume $d/K = \epsilon_0 \ll 1$. We also consider a case in the middle by assuming $K = 2d$. Table 1 gives simplified upper bounds for the error and running time in these cases. We have the following observations:

1) The accuracy α improves dramatically from roughly $O(n^{-\frac{1}{2d}})$ to nearly $O(n^{-1})$ as K increases. For $K > 2d$, the error is smaller than the sampling error $O(n^{-\frac{1}{2}})$.

Order of smoothness	Accuracy α	Running Time: Outputting summary	Running Time: Answering a query
$K = 1$	$O(n^{-\frac{1}{2d+1}})$	$O(n^{\frac{3}{2}})$	$\tilde{O}(n^{\frac{3}{2}+\frac{1}{4d+2}})$
$K = 2d$	$O(n^{-\frac{1}{2}})$	$O(n^{\frac{5}{4}})$	$\tilde{O}(n^{\frac{1}{4}+\frac{3/4}{d}})$
$\frac{d}{K} = \epsilon_0 \ll 1$	$O(n^{-(1-2\epsilon_0)})$	$O(n^{1+\epsilon_0})$	$\tilde{O}(n^{\epsilon_0(1+\frac{3}{d})})$

Table 1: Performance vs. Order of Smoothness for Query Answering

2) The running time for outputting the summary does not change too much, because reading through the database requires $\Omega(n)$ time.

3) The running time for answering a query reduces significantly from roughly $O(n^{3/2})$ to nearly $O(n^{\epsilon_0})$ as K getting large. When $K = 2d$, it is approximately $n^{1/4}$ if d is not too small.

Conceptually our mechanism is simple. First, by change of variables we have

$$g_f(\theta_1, \ldots, \theta_d) = f(\cos\theta_1, \ldots, \cos\theta_d).$$

It also transforms the data universe from $[-1, 1]^d$ to $[-\pi, \pi]^d$. Note that for each variable θ_i, g_f is an even function. To compute the summary, the mechanism just gives noisy answers to queries specified by *even trigonometric monomials* $\cos(r_1\theta_1)\ldots\cos(r_d\theta_d)$. For each trigonometric monomial, the highest degree of any variable is $\max_{1\le i\le d} r_i \le t = O(n^{\frac{1}{2d+K}})$. The summary is an $O(n^{\frac{d}{2d+K}})$-dimensional vector. To answer a query specified by a smooth function f, the mechanism computes a trigonometric polynomial approximation of g_f. The answer to the query q_f is a linear combination of the summary by the coefficients of the approximation trigonometric polynomial.

Our algorithm is an L_∞-approximation based mechanism, which is motivated by Thaler et al. (2012). An approximation based mechanism relies on three conditions:

1) There exists a small set of basis functions such that every query function can be well approximated by a linear combination of them.

2) All the linear coefficients are small.

3) The whole set of the linear coefficients can be computed efficiently.

If these conditions hold, then the mechanism just outputs noisy answers to the set of queries specified by the basis functions as the summary. When answering a query, the mechanism computes the coefficients with which the linear combination of the basis functions approximate the query function. The answer to the query is sidiscretizationmply the inner product of the coefficients and the summary vector.

The following theorem contains the main technical results based on which Theorem 3 holds. It guarantees that by change of variables and using even trigonometric polynomials as the basis functions, the class of smooth functions has all the three properties described above.

Theorem 4 *Let $\gamma > 0$. For every $f \in C_B^K$ defined on $[-1,1]^d$, let*

$$g_f(\theta_1, \dots, \theta_d) = f(\cos\theta_1, \dots, \cos\theta_d), \qquad \theta_i \in [-\pi, \pi].$$

Then, there is an even trigonometric polynomial p whose degree of each variable is $t(\gamma) = \left(\frac{1}{\gamma}\right)^{1/K}$:

$$p(\theta_1, \dots, \theta_d) = \sum_{0 \le r_1, \dots, r_d < t(\gamma)} c_{r_1,\dots,r_d} \cos(r_1\theta_1) \dots \cos(r_d\theta_d),$$

such that

1) $\|g_f - p\|_\infty \le \gamma$.
2) *All the linear coefficients $c_{r_1,\dots,r_d}$ can be uniformly upper bounded by a constant M independent of $t(\gamma)$ (i.e., M depends only on K, d, and B).*
3) *The whole set of the linear coefficients can be computed in time $O\left((\frac{1}{\gamma})^{\frac{d+2}{K}+\frac{2d}{K^2}} \cdot \mathrm{polylog}(\frac{1}{\gamma})\right)$.*

Theorem 4 is proved in Section 3.3. Based on Theorem 4, the proof of Theorem 3 is mainly the argument for Laplace mechanism together with an optimization of the approximation error γ trading-off with the Laplace noise. (Please see Section 3.4 for more details.)

3.2 Discussions

In this section, we show that our algorithm is more effective than methods based on discretization.

3.2.1 Performance of Existing Mechanisms for Smooth Queries

As mentioned in Introduction, in order to apply existing mechanisms to smooth query problem, one has to conduct discretization, both to the data universe and the set of queries. Here, we show that even if one can discretize this function set and even if the it is implemented in an optimal way, the discretized set is exponentially large, and all existing mechanisms have accuracies significantly worse than that of our algorithm.

Proposition 5 *Let $0 < \alpha < 1$. Let $C_B^K(\alpha)$ be a subset of C_B^K so that for every $f, g \in C_B^K(\alpha)$, $\|f - g\|_{[-1,1]^d} \ge \alpha$. In other words, $|C_B^K(\alpha)|$ is the packing number of C_B^K. We have*

$$\log\left|C_B^K(\alpha)\right| \ge \Omega\left(\left(\frac{1}{\alpha}\right)^{d/K}\right).$$

If we further define $Q_{C_B^K}^\alpha$ as the corresponding discretization of $Q_{C_B^K}$ with precision α, then

$$\log\left|Q_{C_B^K}^\alpha\right| \ge \Omega\left(\left(\frac{1}{\alpha}\right)^{d/K}\right).$$

Corollary 6 *Suppose the query set $Q_{C_B^K}$ can be discretized in an optimal way. Then the accuracy guarantee of the PMW mechanism is at best $O(n^{-\frac{K}{2(d+K)}}/\epsilon)$, and the running time is $O(n^{\frac{Kd}{2(K+d)}})$ per query.*

Compare to the accuracy of our mechanism $O(n^{-\frac{K}{2d+K}}/\epsilon)$, our algorithm has significantly better accuracy especially when K is relatively large.

3.2.2 Connection with Conjunctions

In a sense, the well studied query class conjunctions defined on $\{0,1\}^d$ can be seen as smooth functions, by simply extending the domain from $\{0,1\}^d$ to $[-1,1]^d$ and extrapolating $x_{i_1} \wedge \ldots \wedge x_{i_J}$ to $x_{i_1} \cdots x_{i_J}$. Clearly $f(\mathbf{x}) := \prod_{j=1}^{J} x_{i_j}$ is multilinear and smooth.

Note that for multilinear function f induced query, our mechanism will transform it to $g_f(\theta_1, \ldots, \theta_d) := f(\cos\theta_1, \ldots, \cos\theta_d) = \prod_{j=1}^{J} \cos\theta_{i_j}$. Thus g_f is simply a multilinear trigonometric polynomial and is one of the basis functions used in our algorithm. Recall that the mechanism adds Laplace noise to the basis functions. Therefore, if we focus only on the conjunction queries, our algorithm simply reduces to Laplace mechanism.

3.3 L_∞-approximation of smooth functions: small and efficiently computable coefficients

In this section we prove Theorem 4. That is, for every $f \in C_B^K$ the corresponding g_f can be approximated by a low degree trigonometric polynomial in L_∞-norm. We also require that the linear coefficients of the trigonometric polynomial are all small and can be computed efficiently. These properties are crucial for the differentially private mechanism to be accurate and efficient.

In fact, L_∞-approximation of smooth functions in C_B^K by polynomial (and other basis functions) is an important topic in approximation theory. It is well-known that for every $f \in C_B^K$ there is a low degree polynomial with small approximation error. However, it is not clear whether there is an upper bound for the linear coefficients that is sufficiently good for our purpose. Instead we transform f to g_f and use trigonometric polynomials as the basis functions in the mechanism. Then we are able to give a constant upper bound for the linear coefficients. We also need to compute the coefficients efficiently. But results from approximation theory give the coefficients as complicated integrals. We adopt an algorithm which fully exploits the smoothness of the function and thus can efficiently compute approximations of the coefficients to certain precision so that the errors involved do not affect the accuracy of the differentially private mechanism too much.

Below, Section 3.3.1 describes the classical theory on trigonometric polynomial approximation of smooth functions. Section 3.3.2 shows that the coefficients have a small upper bound and can be efficiently computed. Theorem 4 then follows from these results.

3.3.1 Trigonometric Polynomial Approximation with Generalized Jackson Kernel

This section mainly contains known results of trigonometric polynomial approximation, stated in a way tailored to our problem. For a comprehensive description of univariate approximation theory, please refer to the excellent book of DeVore and Lorentz (1993) and to Temlyakov (1994) for multivariate approximation theory.

Let g_f be the function obtained from $f \in C_B^K([-1,1]^d)$:

$$g_f(\theta_1, \ldots, \theta_d) = f(\cos\theta_1, \ldots, \cos\theta_d).$$

Note that $g_f \in C_{B'}^K([-\pi,\pi]^d)$ for some constant B' depending only on B, K, d, and g_f is even with respect to each variable. The key tool in trigonometric polynomial approximation of smooth functions is the generalized Jackson kernel.

Definition 7 *Define the generalized Jackson kernel as*

$$J_{t,r}(s) = \frac{1}{\lambda_{t,r}} \left(\frac{\sin(ts/2)}{\sin(s/2)} \right)^{2r},$$

where $\lambda_{t,r}$ is determined by $\int_{-\pi}^{\pi} J_{t,r}(s)\mathrm{d}s = 1$.

$J_{t,r}(s)$ is an even trigonometric polynomial of degree $r(t-1)$. Let $H_{t,r}(s) = J_{t',r}(s)$, where $t' = \lfloor t/r \rfloor + 1$. Then $H_{t,r}$ is an even trigonometric polynomial of degree at most t. We write

$$H_{t,r}(s) = a_0 + \sum_{l=1}^{t} a_l \cos ls. \tag{1}$$

Suppose that g is a univariate function defined on $[-\pi,\pi]$ which satisfies that $g(-\pi) = g(\pi)$. Define the approximation operator $I_{t,K}$ as

$$I_{t,K}(g)(x) = -\int_{-\pi}^{\pi} H_{t,r}(s) \sum_{l=1}^{K+1} (-1)^l \binom{K+1}{l} g(x+ls)\mathrm{d}s, \tag{2}$$

where $r = \lceil \frac{K+3}{2} \rceil$. It is not difficult to see that $I_{t,K}$ maps g to a trigonometric polynomial of degree at most t.

Next suppose that g is a d-variate function defined on $[-\pi,\pi]^d$, and is even with respect to each variable. Define an operator $I_{t,K}^d$ as sequential composition of $I_{t,K,1}, \ldots, I_{t,K,d}$, where $I_{t,K,j}$ is the approximation operator given in (2) with respect to the jth variable of g. Thus $I_{t,K}^d(g)$ is a trigonometric polynomial of d-variables and each variable has degree at most t.

Theorem 8 *Suppose that g is a d-variate function defined on $[-\pi,\pi]^d$, and is even with respect to each variable. Let $D_j^{(K)}g$ be the Kth order partial derivative of g respect to the j-th variable. If $\|D_j^{(K)}g\|_\infty \le M$ for some constant M for all $1 \le j \le d$, then there is a constant C such that*

$$\|g - I_{t,K}^d(g)\|_\infty \le \frac{C}{t^{K+1}},$$

where C depends only on M, d and K.

3.3.2 The Linear Coefficients

In this subsection we study the linear coefficients in the trigonometric polynomial $I^d_{t,K}(g_f)$. The previous subsection established that g_f can be approximated by $I^d_{t,K}(g_f)$ for a small t. Here we consider the upper bound and approximate computation of the coefficients. Since $I^d_{t,K}(g_f)(\theta_1,\ldots,\theta_d)$ is even with respect to each variable, we write

$$I^d_{t,K}(g_f)(\theta_1,\ldots,\theta_d) = \sum_{0\le n_1,\ldots,n_d\le t} c_{n_1,\ldots,n_d}\cos(n_1\theta_1)\ldots\cos(n_d\theta_d). \tag{3}$$

Fact 9 *The coefficients $c_{n_1,\ldots,n_d}$ of $I^d_{t,K}(g_f)$ can be written as*

$$c_{n_1,\ldots,n_d} = (-1)^d \sum_{\substack{1\le k_1,\ldots,k_d\le K+1\\ 0\le l_1,\ldots,l_d\le t\\ l_i=k_i\cdot n_i \forall i\in[d]}} m_{l_1,k_1,\ldots,l_d,k_d}, \tag{4}$$

where

$$m_{l_1,k_1,\ldots,l_d,k_d} = \prod_{i=1}^{d}(-1)^{k_i} a_{l_i}\binom{K+1}{k_i}\left(\int_{[-\pi,\pi]^d}\prod_{i=1}^{d}\cos\left(\frac{l_i}{k_i}\theta_i\right)g_f(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}\right), \tag{5}$$

and a_{l_i} is the linear coefficient of $\cos(l_i s)$ in $H_{t,r}(s)$ as given in (1).

The following lemma shows that the coefficients $c_{n_1,\ldots,n_d}$ of $I^d_{t,K}(g_f)$ can be uniformly upper bounded by a constant independent of t.

Lemma 10 *There exists a constant M which depends only on K, B, d but independent of t, such that for every $f \in C^K_B$, all the linear coefficients $c_{n_1,\ldots,n_d}$ of $I^d_{t,K}(g_f)$ satisfy*

$$|c_{n_1,\ldots,n_d}| \le M.$$

For clarity, we postpone the proof in Section 3.3.3.

Now we consider the computation of the coefficients $c_{n_1,\ldots,n_d}$ of $I^d_{t,K}(g_f)$. Note that each coefficient involves d-dimensional integrations of smooth functions, so we have to numerically compute approximations of them. For function class C^K_B defined on $[-1,1]^d$, traditional numerical integration methods run in time $O((\frac{1}{\tau})^{d/K})$ in order that the error is less than τ. Here we adopt the sparse grids algorithm due to Gerstner and Griebel (1998) which fully exploits the smoothness of the integrand. By choosing a particular quadrature rule as the algorithm's subroutine, we are able to prove that the running time of the sparse grids is bounded by $O((\frac{1}{\tau})^{2/K})$. The sparse grids algorithm, the theorem giving the bound for the running time and its proof are all given in the follow section 3.3.4 and 3.3.5. Based on these results, we establish the running time for computing the approximate coefficients of the trigonometric polynomial, which is stated in the following Lemma.

Lemma 11 *Let $\hat{c}_{n_1,\ldots,n_d}$ be an approximation of the coefficient $c_{n_1,\ldots,n_d}$ of $I^d_{t,K}(g_f)$ obtained by approximately computing the integral in* (5) *with a version of the sparse grids algorithm (Gerstner and Griebel, 1998) (given in the section 3.3.4). Let*

$$\hat{I}^d_{t,K}(g_f)(\theta_1,\ldots,\theta_d) = \sum_{0\le n_1,\ldots,n_d\le t} \hat{c}_{n_1,\ldots,n_d}\cos(n_1\theta_1)\ldots\cos(n_d\theta_d).$$

Then for every $f \in C_B^K$, in order that

$$\|\hat{I}_{t,K}^d(g_f) - I_{t,K}^d(g_f)\|_\infty \le O\left(t^{-K}\right),$$

it suffices that the computation of all the coefficients $\hat{c}_{n_1,\ldots,n_d}$ runs in time $O\left(t^{(1+\frac{2}{K})d+2} \cdot \mathrm{polylog}(t)\right)$. In addition, $\max_{n_1,\ldots,n_d} |\hat{c}_{n_1,\ldots,n_d} - c_{n_1,\ldots,n_d}| = o(1)$ as $t \to \infty$.

The proof of Lemma 11 is given in the Section 3.3.5. Theorem 4 then follows easily from Lemma 10 and Lemma 11.

Proof of Theorem 4

Setting $t = t(\gamma) = \left(\frac{1}{\gamma}\right)^{1/K}$. Let $p = \hat{I}_{m,K}^d(g_f)$. Combining Lemma 10 and Lemma 11, and note that the coefficients $\hat{c}_{n_1,\ldots,n_d}$ are upper bounded by a constant, the theorem follows.

■

3.3.3 Proof of Lemma 10

We first give a simple lemma.

Lemma 12 *Let*

$$H_{t,r}(s) = \sum_{l=0}^{t} a_l \cos ls. \tag{6}$$

Then for all $l = 0, 1, \ldots, t$

$$|a_l| \le 1/\pi.$$

Proof For any $l \in \{0, 1, \ldots, t\}$, multiplying $\cos ls$ on both sides of (6) and integrating from $-\pi$ to π, we obtain that for some $\xi \in [-\pi, \pi]$,

$$a_l = \frac{1}{\pi}\int_{-\pi}^{\pi} H_{t,r}(s)\cos ls \mathrm{d}s = \frac{\cos l\xi}{\pi}\int_{-\pi}^{\pi} H_{t,r}(s)\mathrm{d}s = \frac{\cos l\xi}{\pi}.$$

where in the last equation we use the identity

$$\int_{-\pi}^{\pi} H_{t,r}(s)\mathrm{d}s = 1.$$

This completes the proof. ■

Proof of Lemma 10

We first bound $m_{l_1,k_1,\ldots,l_d,k_d}$. Recall that (see also (5) in Fact 9)

$$m_{l_1,k_1,\ldots,l_d,k_d} = \prod_{i=1}^{d}(-1)^{k_i} a_{l_i}\binom{K+1}{k_i}\left(\int_{[-\pi,\pi]^d}\prod_{i=1}^{d}\cos\left(\frac{l_i}{k_i}\theta_i\right) g_f(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}\right).$$

It is not difficult to see that $|m_{l_1,k_1,\ldots,l_d,k_d}|$ can be upper bounded by a constant depending only on d, K and B, but independent of t. This is because that the previous lemma shows $|a_{l_i}| \leq \frac{1}{\pi}$ and g_f is upper bounded by a constant.

Now consider $c_{n_1,\ldots,n_d}$. Recall that

$$c_{n_1,\ldots,n_d} = (-1)^d \sum_{\substack{1\leq k_1,\ldots,k_d\leq K+1 \\ 0\leq l_1,\ldots,l_d\leq t \\ l_i=k_i\cdot n_i \forall i\in[d]}} m_{l_1,k_1,\ldots,l_d,k_d}.$$

We need to show that all $|c_{n_1,\ldots,n_d}|$ are upper bounded by a constant independent of t. Note that although each l_i takes $t+1$ values, l_i and k_i must satisfy the constraint $l_i/k_i = n_i$. Since k_i can take at most $K+1$ values, the number of $m_{l_1,k_1,\ldots,l_d,k_d}$ appeared in the summation is at most $(K+1)^d$. Therefore all $c_{n_1,\ldots,n_d}$ are bounded by a constant depending only on d, K and B, and is independent of t. ■

3.3.4 The Sparse Grids Algorithm

In this section we briefly describe the sparse grids numerical integration algorithm due to Gerstner and Griebel. (Please refer to Gerstner and Griebel 1998 for a complete introduction.) We also specify a subroutine used by this algorithm, which is important for proving the running time.

Numerical integration algorithms dicretize the space and use weighted sum to approximate the integration. Traditional methods for the multidimensional case usually discretize each dimension to the same precision level. In contrast, the sparse grids methods, first proposed by Smolyak (1963), discretize each dimension to carefully chosen and possibly different precision levels, and finally combine many such discretization results. When the integrand has bounded mixed derivatives, as in our case that the integrand is in C_B^K, one can use very few grids in most dimension and still achieve high accuracy.

The sparse grids method is based on one dimensional quadrature (i.e., numerical integration). There are many candidates for one dimensional quadrature. In order to prove an upper bound for the running time, we choose the Clenshaw-Curtis rule (Clenshaw and Curtis, 1960) as the subroutine. This also makes the analysis simpler.

Let $h : [-1,1]^d \to \mathbb{R}$ be the integrand. Let $SG(h)$ be the output of the sparse grids algorithm. Let l be the *level* parameter of the algorithm.

Let $\mathbf{k} = (k_1, \ldots, k_d)$ and $\mathbf{j} = (j_1, \ldots, j_d)$ be d-tuples of positive integers. Then $SG(h)$ is given as a combination of weighted sum:

$$SG(h) := \sum_{|\mathbf{k}|\leq l+d-1} \sum_{j_1=1}^{m(k_1)} \cdots \sum_{j_d=1}^{m(k_d)} u_{\mathbf{k},\mathbf{j}} f(\mathbf{x}_{\mathbf{k},\mathbf{j}}). \tag{7}$$

Below we describe $m(k_i)$, $\mathbf{x}_{\mathbf{k},\mathbf{j}}$ and $u_{\mathbf{k},\mathbf{j}}$ respectively.

1) For any $k \in \mathbb{N}$, $m(k) := 2^k$.

2) For each $\mathbf{k} = (k_1, \ldots, k_d)$ and $\mathbf{j} = (j_1, \ldots, j_d)$, define $\mathbf{x}_{\mathbf{k},\mathbf{j}} := (x_{k_1,j_1}, \ldots, x_{k_d,j_d})$, and x_{k_i,j_i} is the j_ith zero of the Chebyshev polynomial with degree $m(k_i)$. Denote by $T_{m(k_i)}$ the

Chebyshev polynomial. Its zeros are given by the following formula.

$$x_{k_i,j_i} = \cos\left(\frac{(2j_i-1)\pi}{2m(k_i)}\right), \qquad j_i = 1,2,\ldots,m(k_i). \tag{8}$$

3) Now we define the weights $u_{\mathbf{k},\mathbf{j}}$. First let $w_{k,j}$ be the weight of $x_{k,j}$ in the one-dimensional Clenshaw-Curtis quadrature rule given by

$$\begin{aligned} w_{k,1} &= \frac{1}{(m(k)+1)(m(k)-1)}, \\ w_{k,j} &= \frac{2}{m(k)}\left(1 + 2\sum_{r=1}^{m(k)/2}{}' \frac{1}{1-4r^2}\cos\left(\frac{2\pi(j-1)r}{m(k)}\right)\right), \qquad \text{for } 2 \le j \le m(k), \end{aligned} \tag{9}$$

where $\sum'$ means that the last term of the summation is halved.

Next, for any fixed k and j, define

$$v_{(k+q),j} = \begin{cases} w_{k,j} & \text{if } q = 1, \\ w_{(k+q-1),r} - w_{(k+q-2),s} & \text{if } q > 1, \text{ and } r,s \text{ with } x_{k,j} = x_{(k+q-1),r} = x_{(k+q-2),s}\ , \end{cases}$$

where $x_{k,j}$ is the zero of Chebyshev polynomial defined above.

Finally, the weight $u_{\mathbf{k},\mathbf{j}}$ is given by

$$u_{\mathbf{k},\mathbf{j}} = \sum_{|\mathbf{k}+\mathbf{q}| \le l+2d-1} v_{(k_1+q_1),j_1} \cdots v_{(k_d+q_d),j_d},$$

where $\mathbf{k} = (k_1,\ldots,k_d)$ and $\mathbf{q} = (q_1,\cdots,q_d)$. This completes the description of the sparse grids algorithm.

3.3.5 Proof of Lemma 11

We first give the result that characterizes the running time of the Gerstner-Griebel sparse grids algorithm in order to achieve a given accuracy.

Lemma 13 *Let $h \in C_B^K([-\pi,\pi]^d)$ for some constants K and B. Let $SG(h)$ be the numerical integration of h using the sparse grids algorithm described in the previous section. Given any desired accuracy parameter $\tau > 0$, the algorithm achieves*

$$\left|\int_{[\pi,\pi]^d} h(\theta)\mathrm{d}\theta - \mathrm{SG}(h)\right| \le \tau,$$

with running time at most $O\left(\left(\frac{1}{\tau}\right)^{\frac{2}{K}}(\log\frac{1}{\tau})^{3d+\frac{2d}{K}+1}\right)$.

Proof

Let $L = 2^{l+1}-2$, where l is the level parameter of the sparse grids algorithm. l and L will be determined by the desired accuracy τ later. In fact, L is the maximum number of grid points of one dimension. By (7) it is easy to see that the total number of grid points, denoted by N_l^d, is given by

$$\begin{aligned} N_l^d &= \sum_{|\mathbf{k}|\le l+d-1} m(k_1)\cdots m(k_d) \\ &= O(l^{d-1}L) \\ &= O(L(\log_2 L)^{d-1}). \end{aligned} \tag{10}$$

In Gerstner and Griebel (1998), it is shown that the approximation error τ can be bounded by the maximal number of grid points per dimension as follows.

$$\tau = O(L^{-K}(\log L)^{(K+1)(d-1)}). \tag{11}$$

Next, let us consider the computational cost per grid point. Since we assume that $h(\mathbf{x})$ can be computed in unit time, and the zeros of Chebyshev polynomials can be computed according to (8), then computing the weights $u_{\mathbf{k},\mathbf{j}}$ dominates the running time. Fix $k \in \mathbb{N}$, consider $w_{k,j}$, $1 \le j \le m(k)$. From (9), it is not difficult to see that the set of $w_{k,j}$ can be computed by Fast Fourier Transform (FFT). Therefore the computation cost is $O(m(k)\log m(k))$. Some calculations yield that for a fixed $\mathbf{k},\mathbf{j}$, the computational cost for $u_{\mathbf{k},\mathbf{j}}$ is $O(dL\log L)$. Combining this with (10) and (11) the lemma follows.

■

Next we turn to prove Lemma 11. First, we need the following famous result.

Lemma 14 *Let m be a positive integer, let $\sigma(m)$ denotes the number of divisors of m, then for large t*

$$\sum_{m=1}^{t} \sigma(m) = t\ln t + (2c-1)t + O(t^{1/2}),$$

where c is Euler's constant.

To analyze the running time, we also need a result about the normalizing constant of the generalized Jackson kernel (Vyazovskaya and Pupashenko, 2006).

Lemma 15 (Vyazovskaya and Pupashenko 2006) *Let*

$$J_{t,r} = \frac{1}{\lambda_{t,r}}\left(\frac{\sin(ts/2)}{\sin(s/2)}\right)^{2r},$$

be the generalized Jackson kernel as given in Definition 4.1, and the normalizing constant $\lambda_{t,r}$ is determined by

$$\int_{-\pi}^{\pi} J_{t,r}(s)\mathrm{d}s = 1.$$

Then the following identity of the normalizing constant $\lambda_{t,r}$ holds

$$\lambda_{t,r} = 2\pi \sum_{k=0}^{[r-r/t]} (-1)^k \binom{2r}{k}\binom{r(t+1)-tk-1}{r(t-1)-tk}. \tag{12}$$

Now we are ready to prove Lemma 11.

Proof of Lemma 11

Assume that the error induced by the sparse grids algorithm is at most τ per integration. That is, for every $\mathbf{k} = (k_1, \ldots, k_d)$, $\mathbf{l} = (l_1, \ldots, l_d)$

$$\left| \int_{[-\pi,\pi]^d} \prod_{i=1}^{d} \cos\left(\frac{l_i}{k_i}\theta_i\right) g(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} - SG\left(\prod_{i=1}^{d} \cos\left(\frac{l_i}{k_i}\theta_i\right) g(\boldsymbol{\theta})\right) \right| \le \tau.$$

Then

$$\sup_{n_1,\ldots,n_d} |c_{n_1,\ldots,n_d} - \hat{c}_{n_1,\ldots,n_d}| \le \sup_{n_1,\ldots,n_d} \left| \sum_{l_i/k_i = n_i} \prod_{i=1}^{d} (-1)^{k_i} \binom{K+1}{k_i} a_{l_i} \right| \cdot \tau.$$

By Lemma 12, $|a_{l_i}| \le \frac{1}{\pi}$. We obtain that

$$\sup_{n_1,\ldots,n_d} |c_{n_1,\ldots,n_d} - \hat{c}_{n_1,\ldots,n_d}| \le M \cdot \tau, \tag{13}$$

for some constant M independent of t.

Similarly, we have

$$\left\| I_{t,K}^d(g) - \hat{I}_{t,K}^d(g) \right\|_\infty \le O(t^d \tau). \tag{14}$$

Since in the statement of the lemma the desired approximation error is $O(t^{-K})$, we have

$$\tau = t^{-(K+d)}. \tag{15}$$

It is also clear that

$$\max_{n_1,\ldots,n_d} |\hat{c}_{n_1,\ldots,n_d} - c_{n_1,\ldots,n_d}| = o(1), \qquad \text{as } t \to \infty.$$

Now let us consider the computation cost. Recall that the kernel $H_{t,r}$ is an even trigonometric of degree at most t:

$$H_{t,r}(s) = a_0 + \sum_{l=1}^{t} a_l \cos ls, \tag{16}$$

where $H_{t,r}(s) = J_{t',r}(s)$ and $J_{t',r}$ is the generalized Jackson kernel given in Definition 4.1. First we need to compute the value of the linear coefficient a_l of $H_{t,r}$. By Lemma 15, one can compute the linear coefficients a_l by solving a system of $t+1$ linear equations. That is, we choose arbitrary $t+1$ points in $[-\pi, \pi]$ and solve (16), since we can compute the value of $H_{t,r}(s)$ directly based on the value of $\lambda_{t,r}$. Clearly, the running time is $O(t^3)$.

Having a_{l_i}, let us consider the computational cost for calculating $\hat{c}_{n_1,\ldots,n_d}$. According to Lemma 13, the running time for the sparse grids algorithm to compute one integration is

$$O\left((\frac{1}{\tau})^{\frac{2}{K}} (\log(1/\tau))^{3d + \frac{2d}{K} + 1} \right) = O\left(t^{\frac{2(K+d)}{K}} \mathrm{polylog}(t) \right).$$

Since we only need to compute the integration when $l_i | k_i$ for all $i \in [d]$, by Lemma 14 the number of integrations to compute is at most

$$(K + 1 + \sigma(1) + \ldots + \sigma(t))^d = O\left((t \log t)^d\right).$$

Thus the total time cost for all numerical integration is $O\left(t^{(1+\frac{2}{K})d+2}\text{polylog}(t)\right)$. Since

$$(1 + \frac{2}{K})d + 2 \geq 3,$$

the computation time for obtaining the coefficients a_l in $H_{t,r}$ is dominated by the running time of the sparse grids algorithm. It is also easy to see that all other computation costs are dominated by that of the numerical integration. The lemma follows. ■

3.4 Proof of Theorem 3

Here we provide the full proof of our first main theorem (Theorem 3).

Proof of Theorem 3

We prove the four results separately.

3.4.1 Differential Privacy

The summary is a t^d-dimensional vector with sensitivity $\frac{t^d}{n}$. By the standard argument for Laplace mechanism, adding t^d i.i.d. Laplace noise $\text{Lap}(\frac{t^d}{n\epsilon})$ preserves ϵ-differential privacy.

3.4.2 Accuracy

The error of the answer to each query consists of two parts: the approximation error and the noise error. Setting the approximation error γ in Theorem 4 as $\gamma = n^{-\frac{K}{2d+K}}$. Then the degree of each variable in $g(\boldsymbol{\theta})$ is

$$t(\gamma) = \left(\frac{1}{\gamma}\right)^{1/K} = n^{\frac{1}{2d+K}},$$

which is the same as t given in Algorithm 1. Now consider the error induced by the Laplace noise. The noise error is simply the inner product of the t^d linear coefficients $c_{l_1,\ldots,l_d}$ and t^d i.i.d. $\text{Lap}(\frac{t^d}{n\epsilon})$. Since the coefficients are uniformly bounded by a constant, the noise error is bounded by the sum of t^d independent and exponentially distributed random variables (i.e., $|\text{Lap}(\frac{t^d}{n\epsilon})|$). The following lemma gives it an upper bound.

Lemma 16 *Let $X_1, \ldots, X_N$ be i.i.d. random variables with p.d.f. $\mathbb{P}(X_i = x) = \frac{1}{\sigma}e^{-x/\sigma}$ for $x \geq 0$. Then*

$$\mathbb{P}(\sum_{i=1}^{N} X_i \geq 2N\sigma) \leq 10 \cdot e^{-\frac{N}{5}}.$$

Proof Let $Y = \sum_{i=1}^{N} X_i$. It is well-known that Y satisfies the gamma distribution, and for $\forall u > 0$

$$\mathbb{P}(Y \geq u) \leq e^{-\frac{u}{\sigma}} \sum_{n=0}^{N-1} \frac{1}{n!} \left(\frac{u}{\sigma}\right)^n .$$

Thus

$$\mathbb{P}(Y \geq 2N\sigma) \leq e^{-2N} \sum_{n=0}^{N-1} \frac{1}{n!} (2N)^n .$$

Note that for $n < N$

$$\frac{\frac{1}{n!}(2N)^n}{e^{2N}} \leq \frac{\frac{1}{N!}(2N)^N}{\frac{1}{(2N)!}(2N)^{2N}} \leq \prod_{n=1}^{N-1} (1 - \frac{n}{2N}) \leq e^{-\frac{N-1}{4}} .$$

Thus

$$\mathbb{P}(Y \geq 2N\sigma) \leq e^{-2N} \left(e^{2N} N e^{-\frac{N-1}{4}}\right) \leq 10 \cdot e^{-\frac{N}{5}} .$$

■

Part 2) of Theorem 3 then follows from Lemma 16.

3.4.3 Running Time for Outputing Summary

This is straightforward since the summary is a t^d-dimensional vector and for each item the running time is $O(n)$.

3.4.4 Running Time for Answering a Query

According to our setting of t, it is easy to check that the error induced by Laplace noise and that of approximation have the same order. Then by the third part of Theorem 4 we have the running time for computing the coefficients of the trigonometric polynomial is $O\left(n^{\frac{d+2+\frac{2d}{K}}{2d+K}} \cdot \mathrm{polylog}(n)\right)$. The result follows since computing the inner product has running time $O(n^{\frac{d}{2d+K}})$, which is much less than computing the coefficients.

■

4. An Improved Mechanism: Output Synthetic Database

Although the mechanism given in the previous section achieves good accuracy for smooth queries and is efficient, it has a disadvantage: The mechanism can only output answers to given queries, and therefore is not convenient for most machine learning tasks which involve optimizations.

In this section we propose an improved mechanism. The mechanism has the advantage that it is able to output a synthetic database. From a practical point of view this is very appealing, because users can simply use this differentially private database to learn whatever they want. Of course, utility will be guaranteed only for restricted types of tasks as described below. Also, the price for outputting synthetic database is that theoretically this mechanism is less efficient than the previous one, although it runs in polynomial time. Please see Section 5.2 for practically efficient variations.

4.1 The Mechanism and Theoretical Results

The following theorem is our second main result. It says that if the query class is specified by smooth functions, then there is a polynomial time mechanism which preserves ϵ-differential privacy and achieves good accuracy. The output of the mechanism is a synthetic dataset. A formal description of the mechanism is given in Algorithm 3.

Theorem 17 *Let the query set be*

$$Q_{C_B^K} := \{q_f(D) = \frac{1}{n}\sum_{\mathbf{x}\in D} f(\mathbf{x}) : \quad f \in C_B^K\},$$

where $K \in \mathbb{N}$ and $B > 0$ are constants. Let the data universe be $[-1,1]^d$, where d is a constant. Then the mechanism described in Algorithm 3 satisfies that for any $\epsilon > 0$, the followings hold:

1) The mechanism preserves ϵ-differential privacy.

2) There is an absolute constant c such that for every $\beta \geq c \cdot e^{-n^{\frac{1}{2d+K}}}$ the mechanism is (α, β)-accurate, where $\alpha = O(n^{-\frac{K}{2d+K}}/\epsilon)$, and the hidden constant depends only on d, K and B.

3) The running time of the mechanism is $O(n^{\frac{3dK+5d}{4d+2K}})$. (This is dominated by solving the linear programming problem in step 20 of the algorithm.)

4) The size of the output synthetic database is $O(n^{1+\frac{K+1}{2d+K}})$.

The proof of Theorem 17 is given in the Section 4.3. Note that the accuracy of this new mechanism is of the same order as that of our first mechanism.

In Table 2, we illustrate the results with typical parameters as we did in the previous section. From Table 2 we can see, as before, the same accuracy improvement as K/d increases. On the other hand, the running time of the mechanism increases if one wants better accuracy for highly smooth queries. Finally, the size of the output synthetic database also increases in order to have better accuracy: roughly, $O(n^{-1})$ accuracy requires an $O(n^2)$-size synthetic database.

Now we explain the mechanism in detail. Part of Algorithm 3 is the same as Algorithm 1. In particular, Algorithm 3 still employs L_∞ approximation with trigonometric polynomials as basis for transformed smooth functions

$$g_f(\theta_1, \ldots, \theta_d) = f(\cos\theta_1, \ldots, \cos\theta_d).$$

Here we view the trigonometric polynomial functions as a set of basis queries. As in Algorithm 1, we compute the answers to the basis queries and add Laplace noise. So far, if we

Algorithm 3 Private Synthetic DB for Smooth Queries

```
Notations: $\mathcal{T}_t^d := \{0,1,\dots,t-1\}^d$,  $\mathbf{x} := (x_1,\cdots,x_d)$,  $\theta_i(\mathbf{x}) := \arccos(x_i)$,
    $a_k := \frac{2k+1-N}{N}$,  $\mathcal{A} := \{a_k | k = 0,1,\dots,N-1\}$,  $\mathcal{L} := \{\frac{i}{L} | i = -L,-L+1,\dots,L-1,L\}$.
Parameters: Privacy parameters $\epsilon,\delta > 0$,  Failure probability $\beta > 0$,
    Smoothness order $K \in \mathbb{N}$.
Input: Database $D \in \left([-1,1]^d\right)^n$
Output: Synthetic database $\tilde{D} \in \left([-1,1]^d\right)^m$
1:  Set $t = \lceil n^{\frac{1}{2d+K}} \rceil$, $N = \lceil n^{\frac{K}{2d+K}} \rceil$, $m = \lceil n^{1+\frac{K+1}{2d+K}} \rceil$, $L = \lceil n^{\frac{d+K}{2d+K}} \rceil$.
2:  Initialize: $\underline{D} \leftarrow \emptyset$, $\tilde{D} \leftarrow \emptyset$, $\mathbf{u} \leftarrow \mathbf{0}_{N^d}$
3:  for all $\mathbf{x} = (x_1,\dots,x_d) \in D$ do
4:      $\underline{x_i} \leftarrow \arg\min_{a\in\mathcal{A}} |x_i - a|$, $i = 1,\dots,d$
5:      Add $\underline{\mathbf{x}} = (\underline{x_1},\dots,\underline{x_d})$ to $\underline{D}$
6:  end for
7:  for all $\mathbf{r} = (r_1,\dots,r_d) \in \mathcal{T}_t^d$ do
8:      $b_{\mathbf{r}} \leftarrow \frac{1}{n}\sum_{\underline{\mathbf{x}}\in\underline{D}} \cos(r_1\theta_1(\underline{\mathbf{x}}))\dots\cos(r_d\theta_d(\underline{\mathbf{x}}))$
9:      $\hat{b}_{\mathbf{r}} \leftarrow b_{\mathbf{r}} + \mathrm{Lap}\left(\frac{t^d}{n\epsilon}\right)$
10:     $\underline{\hat{b}_{\mathbf{r}}} \leftarrow \arg\min_{l\in\mathcal{L}} |\hat{b}_{\mathbf{r}} - l|$
11: end for
12: for all $\mathbf{k} = (k_1,\dots,k_d) \in \mathcal{T}_N^d$ do
13:     for all $\mathbf{r} = (r_1,\dots,r_d) \in \mathcal{T}_t^d$ do
14:         $W_{\mathbf{rk}} \leftarrow \cos(r_1\arccos(a_{k_1}))\dots\cos(r_d\arccos(a_{k_d}))$
15:         $\underline{W_{\mathbf{rk}}} \leftarrow \arg\min_{l\in\mathcal{L}} |W_{\mathbf{rk}} - l|$
16:     end for
17: end for
18: $\underline{\hat{\mathbf{b}}} \leftarrow (\underline{\hat{b}_{\mathbf{r}}})_{\|\mathbf{r}\|_\infty\le t-1}$ ($\underline{\hat{\mathbf{b}}}$ is a $t^d$ dimensional vector)
19: $\underline{\mathbf{W}} \leftarrow (\underline{W_{\mathbf{rk}}})_{\|\mathbf{r}\|_\infty\le t-1,\|\mathbf{k}\|_\infty\le N-1}$ (a $t^d \times N^d$ matrix)
20: Solve the following LP problem: $\min_{\mathbf{u}} \|\underline{\mathbf{W}}\mathbf{u} - \underline{\hat{\mathbf{b}}}\|_1$, subject to $\mathbf{u} \succeq 0$, $\|\mathbf{u}\|_1 = 1$.
    Obtain the optimal solution $\mathbf{u}^*$.
21: repeat
22:     Sample $\mathbf{y}$ according to distribution $\mathbf{u}^*$
23:     Add $\mathbf{y}$ to $\tilde{D}$
24: until $|\tilde{D}| = m$
25: return: $\tilde{D}$
```

ignore the discretization steps (step 3-6), Algorithm 3 is essentially the same as Algorithm 1.

Recall that the next step in our first mechanism is that, for any given smooth query function, we compute the linear coefficients with which the combination of the trigonometric polynomial basis functions is a good approximation of it. The key idea (and the main advantage) of Algorithm 3 is that we do not even need to know the linear coefficients. We merely need to know that there *exist* such coefficients which leads to a good approximation of the smooth function. Now, the goal of the algorithm is to generate a synthetic dataset so that if we evaluate all the basis queries on this synthetic database, all the answers will

Order of smoothness	Accuracy α	Running time	Size of synthetic DB
$K = 1$	$O(n^{-\frac{1}{2d+1}})$	$O(n^2)$	$O(n^{1+\frac{2}{2d+1}})$
$K = 2d$	$O(n^{-\frac{1}{2}})$	$O(n^{\frac{3}{4}d+\frac{5}{8}})$	$O(n^{\frac{3}{2}+\frac{1}{4d}})$
$\frac{d}{K} = \epsilon_0 \ll 1$	$O(n^{-(1-2\epsilon_0)})$	$O(n^{d(\frac{3}{2}-\frac{\epsilon_0}{2})})$	$O(n^{2-\frac{\epsilon_0}{2}})$

Table 2: Performance vs. Order of Smoothness for Outputing Synthetic Database

be close to the noisy answers obtained from the original dataset. The key observation is that if we have such a synthetic dataset, then the evaluation of any smooth query on this synthetic dataset is an answer both differentially private and accurate. To generate such a dataset, we first learn a probability distribution over $[-1, 1]^d$ so that the answers of the basis queries with respect to this distribution are close to the noisy answers. Observe that such a distribution must exist, because the uniform distribution over the original dataset satisfies this requirement. However, learning a continuous distribution is computationally intractable. So we discretize the domain (as well as the original data (step 4)) and consider distributions over the discretized data universe. Because the queries are smooth, the error involved by discretization can be controlled. Learning the distribution can be formulated as a linear programming problem (step 20). Note that in the LP problem we minimize l_1 error instead of l_∞ error because it results in slightly better accuracy. Finally, we randomly draw sufficiently large number of data from this probability distribution, and these data form the output synthetic database.

The running time of the mechanism is dominated by the linear programming step. It is known that the worst-case time complexity of the interior point method is upper bounded in terms of the number of variables, number of constraints, and the number of bits to encode the problem. It is easy to see that there are only poly(n) variables and constraints. To control the number of bits, we round each number in the linear programming problem at a certain precision level (step 10 and 15). Because all the numbers after rounding are bounded uniformly by a constant, the total number of bits to encode the problem is not too large.

We can also analyze the performance of BLR for the smooth query problem and compare to our mechanism proposed in this section as both of them output synthetic database. The analysis is almost identical to that for PMW in Section 3.2.1. Even if one can discretize the query set $Q_{C_B^K}$ in an optimal way, the error of BLR is still considerably larger than that of our algorithm, as described in the following Proposition.

Proposition 18 *Suppose the query set $Q_{C_B^K}$ in an optimal way. The accuracy guarantee of the BLR is at best $O\left(n^{-\frac{K}{d+3K}}\right)$; and the running time is super-exponential in n.*

4.2 Examples of Smooth Queries and Application to Learning

Almost all widely used continuous functions are smooth up to a certain order. Here we list some simple examples. The smoothness of these functions are either obvious or are well known. (See Section 4.4 for proofs.)

1) Linear functions: $f(\mathbf{x}) = \mathbf{w}^T x$ is infinitely smooth if $\|\mathbf{w}\|$ is bounded.

2) Gaussian kernel functions: $f(\mathbf{x}) = \exp(\frac{\|\mathbf{x}-\mathbf{x}_0\|^2}{2\sigma^2})$ (for some $\mathbf{x}_0 \in \mathbb{R}^d$) is in $C_1^{\sigma^2}$, i.e., its derivatives up to order σ^2 is bounded by 1.

3) Logistic function: $f(\mathbf{x}) = \frac{1}{1+e^{-x/\sigma}}$ (for $\sigma > 2$) is K-smooth for any K such that $\max_{k\le K} B_k \le 1$, where B_k is the kth Bernoulli number.

We also point out that many loss functions used in machine learning, composed with smooth functions, are still smooth. Here we just give one example. Suppose the data record in the database are of the form $(\mathbf{x}, y)$. Then the square loss for a smooth regression function f defined as

$$l(f; \mathbf{x}, y) = (y - f(x))^2 ,$$

is smooth.

Now let us discuss applications of our mechanisms to differentially private machine learning. Suppose people want to learn a regression function using the database, where for each data record the first $d-1$ features are independent variables and the dth feature is the dependent variable. In fact, there are excellent algorithms that can do this and guarantee differential privacy. However, consider the following setting: There are many users each wants to learn a regressor and each uses a different type of smooth functions (e.g., linear, polynomial, Ridge, kernel, etc.). Suppose there are totally M users. If we want to guarantee ϵ-differential privacy, we have to require each learner achieve ϵ/M-differential privacy; or equivalently an M-times accuracy decrease in accuracy. As the major goal of differential privacy is to safely release the data and allows everyone to use it, the number of learners M can be very large. Thus a differentially private regression algorithm is not sufficient for this aim. Recently, Ullman (2015) applied PMW to answering multiple convex optimization problems. Using their method, the accuracy decreases only $\log M$-times for M learning problems. However, their algorithm only works for convex loss of linear learning models, while ours work for all smooth functions.

On the other hand, our mechanism can achieve this goal easily: just run the mechanism and output the synthetic database. It is clear that the mechanism guarantees ϵ-differential privacy no matter how many learners use it. The mechanism also guarantees accuracy to *all* learners who use smooth regression functions and the least square criterion, because the accuracy provided by our mechanism hold for *all* smooth functions simultaneously. Therefore, there is no accuracy decrease or privacy loss increase as the number of users grow.

4.3 Proof of Theorem 17

In this section we prove Theorem 17.

Proof of Theorem 17

We first define some notations repeatedly used in this proof.Let the input database be $D = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(n)}\}$. Let the discretized dataset be (please see step 5 in Algorithm 3) $\underline{D} = \{\underline{\mathbf{x}}^{(1)}, \underline{\mathbf{x}}^{(2)}, \cdots, \underline{\mathbf{x}}^{(n)}\}$. Also let the output synthetic dataset be $\tilde{D} = \{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \cdots, \mathbf{y}^{(m)}\}$.

Let

$$\mathbf{b} = (b_{\mathbf{r}})_{\|\mathbf{r}\|_\infty \le t-1}.$$

be a t^d dimensional vector, where $b_{\mathbf{r}}$ is defined in step 8 of the Algorithm 3. Similarly, Let

$$\hat{\mathbf{b}} = (\hat{b}_{\mathbf{r}})_{\|\mathbf{r}\|_\infty \le t-1}$$

and

$$\mathbf{W} = (W_{\mathbf{rk}})_{\|\mathbf{r}\|_\infty \le t-1, \|\mathbf{k}\|_\infty \le N-1},$$

where $\hat{b}_{\mathbf{r}}$ and $W_{\mathbf{rk}}$ are defined as in step 9 and 14 of the algorithm respectively. Let $\mathbf{\Delta} = \hat{\mathbf{b}} - \mathbf{b}$ be the t^d dimensional Laplace noise. Finally let

$$\tilde{\mathbf{b}} = (\tilde{b}_{\mathbf{r}})_{\|\mathbf{r}\|_\infty \le t-1},$$

where

$$\tilde{b}_{\mathbf{r}} = \frac{1}{m} \sum_{\mathbf{y} \in \tilde{D}} \cos(r_1 \theta_1(\mathbf{y})) \ldots \cos(r_d \theta_d(\mathbf{y})).$$

(Recall that $\theta_i(\mathbf{y}) = \arccos(y_i)$. Please see also the Notations in Algorithm 3.)

Now we prove the four results in the theorem one by one.

4.3.1 Differential Privacy

That the mechanism preserves ϵ-differential privacy is straightforward. Note that the output synthetic database $\tilde{D}$ contains no private information other than that obtained from $\hat{\mathbf{b}}$. So we only need to show that $\hat{\mathbf{b}}$ is differentially private. But this is immediate from the privacy of Laplace mechanism.

4.3.2 Accuracy

Let $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)$. For any $f(\mathbf{x}) \in C_B^K$, where $\mathbf{x} \in [-1,1]^d$, let

$$g_f(\boldsymbol{\theta}) := f(\cos\theta_1, \ldots, \cos\theta_d).$$

By Theorem 4, we know, there's an even trignometric polynomial:

$$h_f^{M,t}(\boldsymbol{\theta}) := \sum_{\mathbf{r}=(r_1,\ldots,r_d), \|\mathbf{r}\|_\infty \le t-1} c_{\mathbf{r}}^* \cos(r_1\theta_1) \ldots \cos(r_d\theta_d) \tag{17}$$

satisfy: (Denote $\mathbf{c}^* = (c_{\mathbf{r}}^*)_{\|\mathbf{r}\|_\infty \le t-1}$)

$$\|g_f - h_f^{M,t}\|_\infty \le O(\frac{1}{t^{K+1}}) \tag{18}$$

$$\|\mathbf{c}^*\|_\infty \le M \tag{19}$$

Where constant M only dependents on K, d, B.

Recall that $\boldsymbol{\theta}(\mathbf{x}) := (\arccos x_1, \ldots, \arccos x_d)$. Now we are ready to decompose the error of the mechanism into several terms:

$$\begin{aligned}
\left| q_f(\tilde{D}) - q_f(D) \right| &= \left| \frac{1}{m} \sum_{\mathbf{y} \in \tilde{D}} f(\mathbf{y}) - \frac{1}{n} \sum_{\mathbf{x} \in D} f(\mathbf{x}) \right| \\
\le \left| \frac{1}{m} \sum_{\mathbf{y} \in \tilde{D}} f(\mathbf{y}) - \frac{1}{m} \sum_{\mathbf{y} \in \tilde{D}} h_f^{M,t}(\boldsymbol{\theta}(\mathbf{y})) \right| &+ \left| \frac{1}{m} \sum_{\mathbf{y} \in \tilde{D}} h_f^{M,t}(\boldsymbol{\theta}(\mathbf{y})) - \frac{1}{n} \sum_{\underline{\mathbf{x}} \in \underline{D}} h_f^{M,t}(\boldsymbol{\theta}(\underline{\mathbf{x}})) \right| \\
+ \left| \frac{1}{n} \sum_{\underline{\mathbf{x}} \in \underline{D}} h_f^{M,t}(\boldsymbol{\theta}(\underline{\mathbf{x}})) - \frac{1}{n} \sum_{\underline{\mathbf{x}} \in \underline{D}} f(\underline{\mathbf{x}}) \right| &+ \left| \frac{1}{n} \sum_{\underline{\mathbf{x}} \in \underline{D}} f(\underline{\mathbf{x}}) - \frac{1}{n} \sum_{\mathbf{x} \in D} f(\mathbf{x}) \right|.
\end{aligned} \tag{20}$$

We further decompose the second term in the last row of the above inequality. We have

$$\begin{aligned}
&\left|\frac{1}{m}\sum_{\mathbf{y}\in\tilde{D}} h_f^{M,t}\left(\boldsymbol{\theta}(\mathbf{y})\right) - \frac{1}{n}\sum_{\underline{\mathbf{x}}\in\underline{D}} h_f^{M,t}\left(\boldsymbol{\theta}(\underline{\mathbf{x}})\right)\right| \\
&= \left|\mathbf{c}^*\cdot(\tilde{\mathbf{b}}-\mathbf{b})\right| \le \left(\|\tilde{\mathbf{b}}-\hat{\mathbf{b}}\|_1 + \|\boldsymbol{\Delta}\|_1\right)\|\mathbf{c}^*\|_\infty \\
&\le \left(\|\tilde{\mathbf{b}}-\mathbf{W}\mathbf{u}^*\|_1 + \|\mathbf{W}\mathbf{u}^*-\underline{\mathbf{W}}\mathbf{u}^*\|_1 + \|\underline{\mathbf{W}}\mathbf{u}^*-\hat{\underline{\mathbf{b}}}\|_1 + \|\hat{\underline{\mathbf{b}}}-\hat{\mathbf{b}}\|_1 + \|\boldsymbol{\Delta}\|_1\right)\|\mathbf{c}^*\|_\infty \\
&\le \Big(\|\tilde{\mathbf{b}}-\mathbf{W}\mathbf{u}^*\|_1 + \|(\mathbf{W}-\underline{\mathbf{W}})\mathbf{u}^*\|_1 + \|\mathbf{W}\underline{\mathbf{u}}-\hat{\mathbf{b}}\|_1 + \|(\mathbf{W}-\underline{\mathbf{W}})\underline{\mathbf{u}}\|_1 \\
&\qquad + 2\|\hat{\underline{\mathbf{b}}}-\hat{\mathbf{b}}\|_1 + \|\boldsymbol{\Delta}\|_1\Big)\|\mathbf{c}^*\|_\infty \\
&\le \left(\|\tilde{\mathbf{b}}-\mathbf{W}\mathbf{u}^*\|_1 + \frac{4t^d}{L} + 4\|\boldsymbol{\Delta}\|_1\right)\|\mathbf{c}^*\|_\infty
\end{aligned} \tag{21}$$

where L is the number of grid points in each dimension for rounding, $\hat{\underline{\mathbf{b}}}$ is rounded version of $\hat{\mathbf{b}}, \underline{\mathbf{W}}$ is rounded version of $\mathbf{W}, \mathbf{u}^*$ is optimal distribution by solving LP in step 20 in Algorithm 3, and $\underline{\mathbf{u}}$ is the uniform distribution on $\underline{D}$. Note that the second last inequality holds because

$$\|\underline{\mathbf{W}}\mathbf{u}^* - \hat{\underline{\mathbf{b}}}\|_1 \le \|\underline{\mathbf{W}}\cdot\underline{\mathbf{u}} - \hat{\underline{\mathbf{b}}}\|_1.$$

Also, the last inequality in (21) follows from

$$\|\hat{\underline{\mathbf{b}}}-\hat{\mathbf{b}}\|_1 \le \frac{t^d}{L} + \|\boldsymbol{\Delta}\|_1,$$

and

$$\|\mathbf{W}\underline{\mathbf{u}}-\hat{\mathbf{b}}\|_1 \le \|\mathbf{W}\underline{\mathbf{u}}-\mathbf{b}\|_1 + \|\boldsymbol{\Delta}\|_1 = \|\boldsymbol{\Delta}\|_1,$$

where the last equality holds since $\mathbf{W}\underline{\mathbf{u}} = \mathbf{b}$.

Define

$$\begin{aligned}
\eta_d &= \left|\frac{1}{n}\sum_{\underline{\mathbf{x}}\in\underline{D}} f(\underline{\mathbf{x}}) - \frac{1}{n}\sum_{\mathbf{x}\in D} f(\mathbf{x})\right|, \\
\eta_n &= 4\|\boldsymbol{\Delta}\|_1\|\mathbf{c}^*\|_\infty, \\
\eta_a &= \left|\frac{1}{m}\sum_{\mathbf{y}\in\tilde{D}} f(\mathbf{y}) - \frac{1}{m}\sum_{\mathbf{y}\in\tilde{D}} h_f^{M,t}\left(\boldsymbol{\theta}(\mathbf{y})\right)\right| + \left|\frac{1}{n}\sum_{\underline{\mathbf{x}}\in\underline{D}} h_f^{M,t}\left(\boldsymbol{\theta}(\underline{\mathbf{x}})\right) - \frac{1}{n}\sum_{\underline{\mathbf{x}}\in\underline{D}} f(\underline{\mathbf{x}})\right|, \\
\eta_s &= \|\tilde{\mathbf{b}}-\mathbf{W}\mathbf{u}^*\|_1\|\mathbf{c}^*\|_\infty, \\
\eta_r &= \frac{4t^d}{L}\|\mathbf{c}^*\|_\infty,
\end{aligned}$$

where η_d, η_n, η_a, η_s, η_r correspond to the discretization error, noise error, approximation error, sampling error and rounding error, respectively. Combining (20), (21) and the equations above, we have the error of the mechanism bounded by the sum of these five types of errors:

$$\left|q_f(\tilde{D}) - q_f(D)\right| \le \eta_d + \eta_n + \eta_a + \eta_s + \eta_r.$$

We now bound the five errors separately.

Discretization error η_d: Since $f \in C_B^K$ $(K \geq 1)$, the first order derivatives of f are all bounded by B. Also the discretization precision of $[-1,1]^d$ is $\frac{1}{N}$, so the distance between the data in D and the corresponding data in $\underline{D}$ is $O(\frac{1}{N})$. Recall that N is number of grid points in each dimension for discretization. Thus we have

$$\eta_d = \left| \frac{1}{n} \sum_{\underline{\mathbf{x}} \in \underline{D}} f(\underline{\mathbf{x}}) - \frac{1}{n} \sum_{\mathbf{x} \in D} f(\mathbf{x}) \right| \leq \frac{dB}{N} = O\left(n^{-\frac{K}{2d+K}}\right).$$

Noise error η_n: Since M in Equation (19) is a constant only depending on d, K, B, We know $\|\mathbf{c}^*\|_\infty = O(1)$. Thus to bound $\eta_n = \|\mathbf{\Delta}\|_1 \cdot \|\mathbf{c}^*\|_\infty$, we only need to bound the l_1 norm of the t^d-dimensional vector $\mathbf{\Delta}$ which contains i.i.d. random variables Lap $\left(\frac{t^d}{n\epsilon}\right)$; or equivalently bound the sum of t^d i.i.d. random variables with exponential distribution. It is well known that such a sum satisfies gamma distribution. Simple calculations yields

$$\mathbb{P}\left(\|\mathbf{\Delta}\|_1 \leq 2\frac{t^{2d}}{n\epsilon}\right) \geq 1 - 10e^{-\frac{t^d}{5}}.$$

Thus, with probability $1 - 10e^{-\frac{t^d}{5}}$, we have

$$\eta_n = \|\mathbf{\Delta}\|_1 \|\mathbf{c}^*\|_\infty \leq O\left(\frac{t^{2d}}{n\epsilon}\right).$$

Approximation error η_a: Recall that for any $\mathbf{x}$,

$$g_f(\boldsymbol{\theta}(\mathbf{x})) = f(\mathbf{x}).$$

Denote

$$\|g_f - h_f^{M,t}\|_{[-\pi,\pi]^d} := \sup_{\boldsymbol{\theta} \in [-\pi,\pi]^d} \left| g_f(\boldsymbol{\theta}) - h_f^{M,t}(\boldsymbol{\theta}) \right|.$$

We have

$$\begin{aligned} \eta_a &= \left| \frac{1}{m} \sum_{\mathbf{y} \in \tilde{D}} f(\mathbf{y}) - \frac{1}{m} \sum_{\mathbf{y} \in \tilde{D}} h_f^{M,t}(\boldsymbol{\theta}(\mathbf{y})) \right| + \left| \frac{1}{n} \sum_{\underline{\mathbf{x}} \in \underline{D}} h_f^{M,t}(\boldsymbol{\theta}(\underline{\mathbf{x}})) - \frac{1}{n} \sum_{\underline{\mathbf{x}} \in \underline{D}} f(\underline{\mathbf{x}}) \right| \\ &\leq 2\|g_f - h_f^{M,t}\|_\infty \leq O\left(\frac{1}{t^{K+1}}\right). \end{aligned}$$

Sampling error η_s: It is easy to bound sampling error. Let $\mathbf{W_{r\cdot}}$ be the row vector of matrix $\mathbf{W}$ indexed by $\mathbf{r}$. Recall that $-1 \leq W_{\mathbf{rk}} \leq 1$. Thus for each $\mathbf{r}$, by Chernoff bound we have that for any $\tau > 0$:

$$\mathbb{P}\left(|\tilde{b}_{\mathbf{r}} - \mathbf{W_{r\cdot}}\mathbf{u}^*| \geq \tau\right) \leq 2e^{-\frac{m\tau^2}{2}},$$

since $\tilde{b}_{\mathbf{r}}$ is just the average of m i.i.d. samples and $\mathbf{W}\mathbf{u}^*$ is its expectation. Next by union bound

$$\mathbb{P}\left(\|\tilde{\mathbf{b}} - \mathbf{W}\mathbf{u}^*\|_\infty \geq \tau\right) \leq 2t^d e^{-\frac{m\tau^2}{2}},$$

and therefore

$$\mathbb{P}\left(\|\tilde{\mathbf{b}} - \mathbf{W}\mathbf{u}^*\|_1 \geq t^d \tau\right) \leq 2t^d e^{-\frac{m\tau^2}{2}}.$$

Setting τ such that $2t^d e^{-\frac{m\tau^2}{2}} = e^{-t}$, we have that with probability $1 - e^{-t}$,

$$\|\tilde{\mathbf{b}} - \mathbf{W}\mathbf{u}^*\|_1 \leq O\left(\frac{t^{d+1/2}}{\sqrt{m}}\right).$$

Rounding error η_r: Since $\|\mathbf{c}^*\|_\infty$ is upper bounded by a constant, we have

$$\eta_r \leq O\left(\frac{t^d}{L}\right).$$

Putting it together: Combining the five types of errors, we have that with probability $1 - e^{-t} - 10e^{-\frac{t^d}{5}}$, the error of the mechanism satisfies

$$\left|\frac{1}{m}\sum_{\mathbf{y}\in\tilde{D}} f(\mathbf{y}) - \frac{1}{n}\sum_{\mathbf{x}\in D} f(\mathbf{x})\right| \leq O\left(\frac{1}{N} + \frac{1}{t^{K+1}} + \frac{t^{2d}}{n\epsilon} + \frac{t^{d+\frac{1}{2}}}{\sqrt{m}} + \frac{t^d}{L}\right). \tag{22}$$

Recall that the mechanism sets

$$t = \lceil n^{\frac{1}{2d+K}} \rceil,\ N = \lceil n^{\frac{K}{2d+K}} \rceil,$$
$$m = \lceil n^{1+\frac{K+1}{2d+K}} \rceil,\ L = \lceil n^{\frac{d+K}{2d+K}} \rceil.$$

The theorem follows after some simple calculation.

4.3.3 Running Time

It is not difficult to see that in this case the running time of the mechanism is dominated by solving the Linear Programming problem in step 20. (Because the time complexity of linear programming is with respect to arithmetic operations, all running time discussed here should be understand in this way.) To analyze the running time of the LP problem, observe that it could be rewritten in following standard form:

$$\begin{aligned} \max_{\bar{\mathbf{x}}} \quad & \bar{\mathbf{c}}^T \bar{\mathbf{x}} \\ \text{s.t.} \quad & \bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}} \\ & \bar{\mathbf{x}} \succeq \mathbf{0} \end{aligned} \tag{23}$$

where

$$\bar{\mathbf{A}} = \begin{pmatrix} L \cdot \underline{\mathbf{W}} & L \cdot \mathbf{I}_{t^d} & -L \cdot \mathbf{I}_{t^d} \\ \mathbf{1}_{N^d}^T & 0 & 0 \end{pmatrix},$$

$$\bar{\mathbf{b}} = \begin{pmatrix} L \cdot \hat{\underline{\mathbf{b}}} \\ 1 \end{pmatrix}, \quad \bar{\mathbf{c}} = \begin{pmatrix} \mathbf{0} \\ \mathbf{1}_{t^d} \\ \mathbf{1}_{t^d} \end{pmatrix}, \quad \bar{\mathbf{x}} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \end{pmatrix}.$$

$\bar{\mathbf{A}}$ is a $\bar{m} \times \bar{n}$ matrix where $\bar{m} = t^d + 1$ and $\bar{n} = N^d + 2t^d$. Note that 1) each element of $\underline{\mathbf{W}}$ is in $[-1, 1]$; 2) each element of $\hat{\underline{\mathbf{b}}}$ is in $[-1, 1]$; and 3) each element of $\underline{\mathbf{W}}$ and $\hat{\underline{b}}$ is rounded to precision $1/L$. So actually we have reduce to a LP problem (23), with elements of $\bar{\mathbf{A}}$, $\bar{b}$, $\bar{c}$ are all integers and bounded by L.

The most well-known worst-case complexity of the interior point algorithm for linear programming with integer parameters is $O(\bar{n}^3\tilde{L})$, where $\bar{n}$ is the number of variables and $\tilde{L}$ is the number of bits to encode the linear programming problem. Here we use a more refined bound given in Anstreicher (1999). By using this bound, we are able to prove a much better time complexity for our algorithm; because in the linear programming problem (23), the number of constraints is often much smaller than the number of variables. The bound we make use of for the complexity of linear programming is $O(\frac{\bar{n}^{1.5}\bar{m}^{1.5}}{\ln \bar{m}}\bar{L})$ (Anstreicher, 1999). Here, $\bar{L}$ is the size of LP problem in standard form defined as follows (Monteiro and Adler, 1989):

$$\begin{aligned}\bar{L} =& \lceil \log(1 + |\det(\bar{\mathbf{A}}_{max})|) \rceil + \lceil \log(1 + \|\bar{\mathbf{c}}\|_\infty) \rceil \\ &+ \lceil \log(1 + \|\bar{\mathbf{b}}\|_\infty) \rceil + \lceil \log(\bar{m} + \bar{n}) \rceil,\end{aligned}$$

where

$$\bar{\mathbf{A}}_{max} = \underset{\mathbf{X}\text{is a square submatrix of}\bar{\mathbf{A}}}{\arg\max} |\det(\mathbf{X})|.$$

Note that $\bar{m} < \bar{n}$, so the size of $\bar{\mathbf{A}}_{max}$ is at most $\bar{m} \times \bar{m}$. Therefore, we have

$$|\det(\bar{\mathbf{A}}_{max})| \le \bar{m}! L^{\bar{m}},$$

and

$$\bar{L} = O(\bar{m}(\log \bar{m} + \log L) + \log \bar{n}).$$

Given $\bar{m} = O(t^d)$ and $\bar{n} = O(N^d)$, simple calculation shows that the total time complexity is

$$O\left(\frac{\bar{n}^{1.5}\bar{m}^{1.5}}{\ln \bar{m}}\bar{L}\right) = O\left(N^{1.5d}t^{2.5d}\right) = O\left(n^{\frac{3dK+5d}{4d+2K}}\right).$$

4.3.4 Size of the Output Synthetic Database

The size of synthetic dataset m is set in step 1 of the algorithm. ■

4.4 Proof of Smoothness

Here we prove the smoothness of the functions listed in Section 4.2. We only give the proof for Gaussian kernel function. The smoothness for logistic function follows directly from the derivative of hyperbolic tangent.

Proposition 19 *Let*

$$f(\mathbf{x}) = \sum_{j=1}^{J} \alpha_j \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_j\|^2}{2\sigma^2}\right),$$

where $\mathbf{x} \in \mathbb{R}^d$. *Let* $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_J)$. *Suppose* $\|\boldsymbol{\alpha}\|_1 \le 1$. *Then for every* $K \le \sigma^2$, $\|f\|_K \le 1$.

Proof We first give a well-known inequality for Hermite polynomial. Proposition 19 follows immediately from this lemma.

Lemma 20 (Indritz 1961) *For Hermite polynomial of degree k defined as*

$$H_k(x) = (-1)^k e^{x^2} \frac{\mathrm{d}^k}{\mathrm{d}x^k} e^{-x^2},$$

where $k \in \mathbb{N}$ and $x \in (-\infty, \infty)$, it satisfies following inequality:

$$|H_k(x)| \le (2^k k!)^{\frac{1}{2}} e^{\frac{1}{2}x^2}.$$

To prove Proposition 19, we only need to show that the K-norm of the Gaussian kernel function is bounded by 1 since $\|\boldsymbol{\alpha}\|_1 \le 1$.

Let $g(x) = e^{-x^2}$. From Lemma 20 we directly have:

$$|\frac{\mathrm{d}^k}{\mathrm{d}x^k} g(x)| = |H_k(x) e^{-x^2}| \le (2^k k!)^{\frac{1}{2}}.$$

Let $\mathbf{k} = (k_1, \ldots, k_d)$, and $|\mathbf{k}| = K$. Therefore, for $f(\mathbf{x})$ defined in Proposition 19, we have:

$$|D^{\mathbf{k}} f(\mathbf{x})| = \prod_{j=1}^{d} \frac{\mathrm{d}^{k_j}}{\mathrm{d}x_j^{k_j}} g\left(\frac{x_j - y_j}{\sqrt{2}\sigma}\right) \le \left(\frac{1}{\sqrt{2}\sigma}\right)^K \left(\prod_{j=1}^{d} (2^{k_j} k_j!)\right)^{\frac{1}{2}} \le \frac{(K!)^{\frac{1}{2}}}{\sigma^K}.$$

Obviously, when $K \le \sigma^2$,

$$|D^{\mathbf{k}} f(\mathbf{x})| \le \frac{K^{\frac{K}{2}}}{\sigma^K} \le 1.$$

The proposition follows. ■

5. Additional Results

In this section we provide some additional results. In Section 5.1 we show that the two mechanisms described in Section 3 and Section 4 respectively can be modified to achieve slightly better accuracy and preserve (ϵ, δ)-differential privacy. In Section 5.2 we give a variant of our second mechanism which outputs synthetic database. The goal is to make the algorithm practically efficient, since as stated in Theorem 17 the running time of that mechanism is approximately $O(n^{3d/2})$, which is not acceptable in real applications.

5.1 (ϵ, δ)-Differentially Private Mechanisms

The two mechanisms given in Section 3 and Section 4 respectively preserve ϵ-differential privacy. It is easy to generalize them to preserve (ϵ, δ)-differential privacy and have slightly better accuracy.

For the first mechanism, simply setting

$$t = \lceil n^{\frac{2}{3d+2K}} (\log \frac{1}{\delta})^{-\frac{1}{3d+2K}} \rceil,$$

in step 1 of Algorithm 1 and Algorithm 2 respectively we have the following result.

Theorem 21 *Let the query set be*

$$Q_{C_B^K} = \{q_f = \frac{1}{n}\sum_{\mathbf{x}\in D} f(\mathbf{x}) : \quad f \in C_B^K\},$$

where $K \in \mathbb{N}$ and $B > 0$ are constants. Let the data universe be $[-1,1]^d$, where $d \in \mathbb{N}$ is a constant. Then the new mechanism related to Algorithm 1 and Algorithm 2 satisfies that for any $\epsilon > 0$, $\delta > 0$, the following hold:

1) The mechanism is (ϵ, δ)-differentially private.

2) There is an absolute constant c such that for any $\beta \geq c \cdot e^{-n^{\frac{2}{3d+2k}}(\log\frac{1}{\delta})^{-\frac{1}{3d+2K}}}$, the mechanism is (α, β)-accurate, where $\alpha = O\left(n^{-\frac{2K}{3d+2K}}(\log\frac{1}{\delta})^{\frac{K}{3d+2K}}/\epsilon\right)$, and the hidden constant depends only on d, K and B.

3) The running time of the mechanism is $O\left(n^{\frac{5d+2K}{3d+2K}}(\log\frac{1}{\delta})^{-\frac{d}{3d+2K}}\right)$.

4) The running time for $\mathcal{S}$ to answer a query is $O\left(n^{\frac{2d+4+\frac{2d}{K}}{3d+2K}}(\log\frac{1}{\delta})^{-\frac{d+2+\frac{d}{K}}{3d+2K}}\log(n)\right)$.

The proof of Theorem 21 is along the same line as the proof of Theorem 3 plus standard use of the composition theorem (Dwork et al., 2010). We omit the details.

For the second mechanism, replacing step 1 and step 9 of Algorithm 3 by the following we obtain an (ϵ, δ)-differentially private mechanism.

1) Step 1. Set $t = \lceil n^{\frac{2}{3d+2K}}(\log\frac{1}{\delta})^{-\frac{1}{3d+2K}}\rceil$, $N = \lceil n^{\frac{2K}{3d+2K}}(\log\frac{1}{\delta})^{-\frac{K}{3d+2K}}\rceil$,
$m = \lceil n^{\frac{4d+4K+2}{3d+2K}}(\log\frac{1}{\delta})^{-\frac{2d+2K+1}{3d+2K}}\rceil$, and $L = \lceil n^{\frac{2d+2K}{3d+2K}}(\log\frac{1}{\delta})^{-\frac{d+K}{3d+2K}}\rceil$.

2) Step 9. $\hat{b}_{\mathbf{r}} = b_{\mathbf{r}} + \mathrm{Lap}\left(\frac{(t^d \log\frac{1}{\delta})^{\frac{1}{2}}}{n\epsilon}\right)$.

Theorem 22 *Let the query set $Q_{C_B^K}$ be defined as in Theorem 17. Let the data universe be $[-1,1]^d$, where $d \in \mathbb{N}$ is a constant. Then the new mechanism related to Algorithm 3 satisfies that for any $\epsilon > 0$, $\delta > 0$, the followings hold:*

1) The mechanism is (ϵ, δ)-differentially private.

2) There is an absolute constant c such that for any $\beta \geq c \cdot e^{-n^{\frac{2}{3d+2k}}(\log\frac{1}{\delta})^{-\frac{1}{3d+2K}}}$, the mechanism is (α, β)-accurate, where $\alpha = O\left(n^{-\frac{2K}{3d+2K}}(\log\frac{1}{\delta})^{\frac{K}{3d+2K}}/\epsilon\right)$, and the hidden constant depends only on d, K and B.

3) The running time of the mechanism is $O\left(n^{\frac{3dK+5d}{3d+2K}}(\log\frac{1}{\delta})^{-\frac{3dK+5d}{6d+4K}}\right)$.

4) The size of synthetic database is $O\left(n^{\frac{4d+4K+2}{3d+2K}}(\log\frac{1}{\delta})^{-\frac{2d+2K+1}{3d+2K}}\right)$.

The proof of Theorem 22 is omitted for the same reason as above.

5.2 A Practical Variant of the Synthetic-Database-Output Mechanism

The time complexity of our second mechanism, which outputs synthetic database, can be nearly $n^{\frac{3d}{2}}$ to achieve n^{-1} accuracy for highly smooth queries, where d is the dimension of the data. In real application such a running time is unacceptable. We thus consider a variant of Algorithm 3 which turns out to be very efficient and suffers only from minor loss in accuracy in our experiments. (On the other hand we do not have theoretical guarantee for the utility of this algorithm any more.) Note that the running time of Algorithm 3 is dominated by the linear programming step. This LP problem has $O(N^d)$ variables and $O(t^d)$ constraints, where N^d is the number of discretized grids in $[-1,1]^d$ and t^d is the number of trigonometric polynomial basis functions. To make our algorithm practical, we consider a subset $\mathcal{S}$ of the N^d grids with size $C := |\mathcal{S}| \ll N^d$ and restrict the probability distribution $\mathbf{u}$ on this subset of grids (see step 20 in Algorithm 3). Similarly, we may use a subset of size R of the t^d trigonometric polynomial basis functions preferred to lower degrees. By doing this, the LP problem has C variables and R constraints.

The simplest approach to obtain a small subset $\mathcal{S}$ is sampling from the N^d grids in $[-1,1]^d$ uniformly. However, this approach suffers from substantial loss in accuracy (see the supplementary material for experimental results of this method), because $|\mathcal{S}|$ is extremely small compared to N^d, the probability that $\mathcal{S}$ contains data points in D (or close to D) is very small. In order to reduce the size of the LP problem and preserve the accuracy, we need a better approach to obtain $\mathcal{S}$. Formally, the problem of choosing a subset $\mathcal{S}$ for our purpose can be formulated as follows: We want a subset $\mathcal{S}$ so that

1) $\mathcal{S}$ is differentially private;

2) $|\mathcal{S}|$ is small;

3) For almost every data point x in D, there is a point in $\mathcal{S}$ close to x.

Note that without the privacy concern, one can simply let $\mathcal{S} = D$. But under the requirement of privacy, this problem is non-trivial. Here we adopt private PCA to obtain a low dimensional ellipsoid. The ellipsoid is spanned by the (private) top eigenvectors of the data covariance matrix with the square root of the (private) eigenvalues as the radius. In particular, we use a slightly modified version of the Private Subspace Iteration (PSI) mechanism (Hardt, 2013) to compute the private eigenpairs. The mechanism is described in Algorithm 4. Finally, we uniformly sample C points from the ellipsoid to form $\mathcal{S}$.

In the following three results, we show that the PSI mechanism is differentially private and accurate for the top eigenvectors and eigenvalues respectively. Note that Hardt (2013) shows that the principal angle between the space spanned by the top-k leading eigenvectors of the true data covariance matrix and the the space spanned by the output column vectors of $\mathbf{X}^{(T)}$ is small. However, it does not directly imply that the output private ellipsoid converges to the true PCA ellipsoid. Our results slightly strengthen the results in Hardt (2013). We show each private top-k eigenvalue is close to the true eigenvalue and each private top-k eigenvector is close to the true eigenvector, provided the true eigenvalues are well separated.

Theorem 23 (Accuracy of the eigenvectors) *Given a database D with $|D| = n$, let*

$$\mathbf{A} = \frac{1}{n}\sum_{\mathbf{y}\in D}(\mathbf{y}-\bar{\mathbf{y}})(\mathbf{y}-\bar{\mathbf{y}})^T$$

Algorithm 4 Private Subspace Iteration (Hardt, 2013)

```
Input: Database $D \in ([-1,1]^d)^n$
Output: $\boldsymbol{\lambda}$ (as private top-$k$ eigenvalues), $\mathbf{X}^{(T)}$ (columns as private top-$k$ eigenvectors).
Parameters: Number of iterations $T \in \mathbb{N}$, dimension $k$, privacy parameters $\epsilon, \delta > 0$,
    Denote GS as the Gram-Schmidt orthonormalization algorithm.
1: Set $\sigma = \frac{5d\sqrt{4k(T+1)\log(1/\delta)}}{n\epsilon}$,
   $\mathbf{A} = \frac{1}{n}\sum_{\mathbf{y}\in D}(\mathbf{y}-\bar{\mathbf{y}})(\mathbf{y}-\bar{\mathbf{y}})^T$, where $\bar{\mathbf{y}} = \frac{1}{n}\sum_{\mathbf{y}\in D}\mathbf{y}$. Thus $\mathbf{A}$ is the data covariance
   matrix.
2: Initialize: $\mathbf{G}^{(0)} \sim N(0,1)^{d\times k}$ (i.i.d. Gaussian distribution), $\mathbf{X}^{(0)} \leftarrow \mathrm{GS}(\mathbf{G}^0)$
3: for all $l = 1, 2, \ldots, T$ do
4:     Sample $\mathbf{G}^{(l)} \sim N(0,\sigma^2)^{n\times k}$.
5:     $\mathbf{W}^{(l)} = \mathbf{A}\mathbf{X}^{(l-1)} + \mathbf{G}^{(l)}$
6:     $\mathbf{X}^{(l)} \leftarrow \mathrm{GS}(\mathbf{W}^{(l)})$
7: end for
8: Set $\hat{\lambda}_s = \|\mathbf{w}_s^{(T)}\|_2$ for $s = [k]$, where $\mathbf{w}_s^{(T)}$ is the $s$-th column of $\mathbf{W}^{(T)}$. Set $\boldsymbol{\lambda} = (\hat{\lambda}_s)_{s\le k}$.
```

be the data covariance matrix. Also let $\lambda_1 \ge \cdots \ge \lambda_d$ be the eigenvalues of A and $\gamma_k = \lambda_k/\lambda_{k+1} - 1$. Let

$$\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_k) \in \mathbb{R}^{d\times k},$$

where $\mathbf{u}_1, \ldots, \mathbf{u}_k$ are the top k eigenvectors of $\mathbf{A}$. Denote $\theta(\mathbf{u},\mathbf{v})$ as the angle between two vectors $\mathbf{u}$ and $\mathbf{v}$. Then, with parameters $k \le d/2$ and $T \ge C(\min_{s\le k}\gamma_s)^{-1}\log d$ for some sufficiently large constant C, the matrix $\mathbf{X}^{(T)} = (\mathbf{x}_1^{(T)}, \ldots, \mathbf{x}_k^{(T)}) \in \mathbb{R}^{d\times k}$ returned by Algorithm 4 satisfies that: with probability $1-o(1)$, for all $s \le k$

$$\sin\theta(\mathbf{u}_s, \mathbf{x}_s^{(T)}) \le O\left(\frac{\omega_s d^{\frac{3}{2}}\sqrt{kT\log T\log\left(\frac{1}{\delta}\right)}}{n\epsilon}\right),$$

where

$$\omega_s = \begin{cases} \frac{1}{\gamma_1\lambda_1} & s = 1, \\ \max\{\frac{1}{\gamma_s\lambda_s}, \frac{1}{\gamma_{s-1}\lambda_{s-1}}\} & 2 \le s \le k. \end{cases}$$

Theorem 24 (Accuracy of the eigenvalues) *Using the same notions as in Theorem 23, let $\hat{\lambda}_s = \|\mathbf{w}_s^{(T)}\|_2$ for $s \le k$, where $\mathbf{w}_s^{(T)}$ is the sth column of $\mathbf{W}^{(T)}$. Then with probability $1-o(1)$, we have for all $s \le k$*

$$|\hat{\lambda}_s - \lambda_s| \le O\left(\frac{d^3kT\log T\log\left(\frac{1}{\delta}\right)}{n^2\epsilon^2\gamma_s^2\lambda_s^2} + \frac{kd\sqrt{T\log T\log\left(\frac{1}{\delta}\right)}}{n\epsilon}\right).$$

Theorem 25 (Privacy) *If Algorithm 4 is executed with each $\mathbf{G}^{(l)}$ independently sampled as*

$$\mathbf{G}^{(l)} \sim N(0,\sigma^2)^{d\times k},$$

with

$$\sigma = \frac{5d\sqrt{4kT\log(1/\delta)}}{n\epsilon},$$

then Algorithm 4 satisfies (ϵ, δ)-differential privacy.

If Algorithm 4 is executed with each $\mathbf{G}^{(l)}$ independently sampled as

$$\mathbf{G}^{(l)} \sim Lap(\sigma)^{d \times k},$$

with

$$\sigma = \frac{50 d^{3/2} k T}{n\epsilon},$$

then Algorithm 4 satisfies ϵ-differential privacy.

The proof of Theorem 23, Theorem 24 and Theorem 25 are given below.

Before we state it formally, let us take a closer look at the Theorem 3.3 in Hardt (2013): With high probability, the tangent of the angle between the space spanned by the top-k leading eigenvectors, namely eigenspace, and the space spanned by the output columns, namely output-space, is *small*, given regularity conditions. Our goal is the column-wise convergence between eigenvectors and output columns, which can be concluded from the simultaneous convergence between the increasing sequence of eigenspaces and the increasing sequence of output-spaces, given that they shared the same dimension. This constraint leads us to utilize a weaker version of Theorem 3.3 by specifying $r = k$, but the favored column-wise convergence at least compensated for the loss of tuning parameter r. Note that simply applying Theorem 3.3 consecutively for the sequence will not assure the high convergence probability $1 - o(1)$ and our analysis can be extended to the case $k = O(d)$, where the dimension d can grow as the size of database given the aptitude of added noise is adequate.

Our results generalize Theorem 3.3 in Hardt (2013) which proves that the principal angle between the subspace spanned by the private top-k eigenvectors and the subspace spanned by the true eigenvectors is small. But what we need in this paper is that each private eigenvector (and eigenvalue) converges to the true eigenvector (and eigenvalue). It is worth pointing out that simply applying Theorem 3.3 in Hardt (2013) consecutively for each k does not obtain our result, although our proofs rely on some results in Hardt (2013).

We first give three lemmas. The proof of the first two lemmas are straightforward and are omitted. For simplicity, below we denote $\|\mathbf{X}\|$ the spectral norm of a matrix $\mathbf{X}$.

Lemma 26 *Assuming the data universe $\mathcal{X} = [-1, 1]^d$, for all pairs of neighbor databases D, D' with $|D| = |D'| = n$, let*

$$\mathbf{A}(D) = \frac{1}{n} \sum_{\mathbf{y} \in D} (\mathbf{y} - \bar{\mathbf{y}})(\mathbf{y} - \bar{\mathbf{y}})^T,$$

be the data covariance matrix. It holds that

$$\|\mathbf{A}(D) - \mathbf{A}(D')\| \leq \frac{5d}{n}.$$

Lemma 27 *Let $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{n \times d}$, denote $\mathbf{A}_{kl} = (a_{ij})_{i \leq k, j \leq l}$ the (k, l)-sub matrix of $\mathbf{A}$ for any $k \leq n$ and $l \leq d$, then $\|\mathbf{A}_{kl}\| \leq \|\mathbf{A}\|$.*

Lemma 28 *Let $\mathbf{U} \in \mathbb{R}^{d\times k}$ be a matrix with orthonormal columns. Let*

$$\mathbf{G}^{(1)}, ..., \mathbf{G}^{(T)} \sim N(0, \sigma^2)^{d\times k},$$

with $k \le d$ and assume that $T \le d$. Let $\mathbf{G}_s^{(l)}$ and $\mathbf{U}_s$ be the (d, s)-sub matrix of $\mathbf{G}^{(l)}$ and $\mathbf{U}$ respectively for $s \in [k]$. Then, with probability $1 - o(1)$,

$$\max_{l\in[T]} \|\mathbf{U}_s^T \mathbf{G}_s^{(l)}\| \le O(\sigma\sqrt{k \log T}), \quad \forall\, s \in [k].$$

Proof

By Lemma A.3 in Hardt (2013),

$$\|\mathbf{U}_k^T \mathbf{G}_k^{(l)}\| \le O(\sigma\sqrt{k \log T}).$$

The desired result follows from Lemma 27 since $\mathbf{U}_s^T \mathbf{G}_s^{(l)}$ is the (s, s)-sub matrix of $\mathbf{U}_k^T \mathbf{G}_k^{(l)}$. ■

Proof of Theorem 23 Consider the spectral decomposition $\mathbf{A} = \mathbf{Z}\mathbf{\Lambda}\mathbf{Z}^{-1}$, and denote $\mathbf{\Lambda} = \begin{pmatrix}\mathbf{\Lambda}_1 & \\ & \mathbf{\Lambda}_2\end{pmatrix}$, and $\mathbf{Z} = \begin{pmatrix}\mathbf{Z}_1 & \mathbf{Z}_2\end{pmatrix}$, where $\mathbf{\Lambda}_1 \in \mathbb{R}^{s\times s}$ and $\mathbf{Z}_1 \in \mathbb{R}^{d\times s}$. Next we denote $\mathbf{U}_s = \mathbf{Z}_1\mathbf{\Lambda}_1\mathbf{Z}_1^T$ and $\mathbf{V}_s = \mathbf{Z}_2\mathbf{\Lambda}_2\mathbf{Z}_2^T$. Obviously we have

$$\mathbf{A} = \mathbf{U}_s\mathbf{\Lambda}_1\mathbf{U}_s^T + \mathbf{V}_s\mathbf{\Lambda}_2\mathbf{V}_s^T.$$

Let

$$\Delta(\mathbf{U}_s) = \max_l \|\mathbf{U}_s^T \mathbf{G}_s^{(l)}\|,$$

and

$$\Delta(\mathbf{V}_s) = \max_l \|\mathbf{V}_s^T \mathbf{G}_s^{(l)}\|,$$

where $\mathbf{G}_s^{(l)}$ is the (d, s)-sub matrix of $\mathbf{G}^{(l)}$. By Lemma 28, we concludes that with probability $1 - o(1)$, the following events occur simultaneously:

1. $\forall\, s \in [k], \Delta(\mathbf{U}_s) \le O(\sigma\sqrt{k \log T})$,
2. $\forall\, s \in [k], \Delta(\mathbf{V}_s) \le O(\sigma\sqrt{d \log T})$.

Notice that for all $s \le k$, we have with probability $1 - o(1)$ that $\Delta(\mathbf{U}_s) \le \Delta(\mathbf{V}_s)$ as we set $s \le k \le d/2$. Since $\arccos\theta(\mathbf{U}_s, \mathbf{X}_s^{(0)})$ is bounded, where $\mathbf{X}_s^{(0)}$ is the (d, s)-sub matrix of $\mathbf{X}^{(0)}$, we have for all $s \le k$

$$\Delta(\mathbf{U}_s) \arccos\theta(\mathbf{U}_s, \mathbf{X}_s^{(0)}) \le O(\sigma\sqrt{k \log T}).$$

Applying Theorem 2.9 in Hardt (2013), we have with probability of $1 - o(1)$, for all $s \in [k]$

$$\tan\theta(\mathbf{U}_s, \mathbf{X}_s^{(T)}) \le O\left(\frac{\sigma}{\gamma_s\lambda_s}\sqrt{d \log T}\right). \tag{24}$$

For the case $s = 1$, the theorem is proved. Now for any fixed $s \in [k]$, notice that $\mathbf{u}_s$ is in the space spanned by $(\mathbf{u}_1, \ldots, \mathbf{u}_s)$ as well as the orthogonal complement of the space spanned by $(\mathbf{u}_1, \ldots, \mathbf{u}_{s-1})$, we have

$$\begin{aligned}
&\sin^2 \theta(\mathbf{u}_s, \mathbf{x}_s^{(T)}) \\
&= \|\mathbf{U}_{s-1}\mathbf{U}_{s-1}^T \mathbf{x}_s^{(T)} + (\mathbf{I} - \mathbf{U}_s\mathbf{U}_s^T)\mathbf{x}_s^{(T)}\|^2 \\
&= \|\mathbf{U}_{s-1}\mathbf{U}_{s-1}^T \mathbf{x}_s^{(T)}\|^2 + \|(\mathbf{I} - \mathbf{U}_s\mathbf{U}_s^T)\mathbf{x}_s^{(T)}\|^2 \\
&\leq \sin^2 \theta(\mathbf{U}_{s-1}, \mathbf{X}_{s-1}^{(T)}) + \sin^2 \theta(\mathbf{U}_s, \mathbf{X}_s^{(T)}) \\
&\leq 2(\max\{\sin^2 \theta(\mathbf{U}_{s-1}, \mathbf{X}_{s-1}^{(T)}), \sin^2 \theta(\mathbf{U}_s, \mathbf{X}_s^{(T)})\}) \\
&\leq 2(\max\{\tan^2 \theta(\mathbf{U}_{s-1}, \mathbf{X}_{s-1}^{(T)}), \tan^2 \theta(\mathbf{U}_s, \mathbf{X}_s^{(T)})\}).
\end{aligned} \tag{25}$$

The theorem, for the case $s \geq 2$, is proved by substituting (24) into (25). ■

Proof of Theorem 24 Denote $\mathbf{x}_s = \mathbf{x}_s^{(T-1)}$, $\mathbf{w}_s = \mathbf{w}_s^{(T)}$, $\mathbf{g}_s$ as the s-column of $\mathbf{G}^{(T)}$, and $\theta^{(T)} = \theta(\mathbf{U}_s, \mathbf{X}_s^{(T)})$ for short. Let $\mathbf{x}_s = \mathbf{u} + \mathbf{u}^{\perp}$, where $\mathbf{u}$ is the eigenvector corresponding to λ_s. Then, since $\mathbf{u} = \mathbf{x}_s \cos\phi$ and $\mathbf{u}^{\perp} = \mathbf{x}_s \sin\phi$ for a $\phi \leq \theta^{(T)}$, we have

$$\begin{aligned}
\hat{\lambda}_s^2 &= \mathbf{w}_s^T \mathbf{w}_s \\
&= (\mathbf{A}\mathbf{x}_s + \mathbf{g}_s)^T (\mathbf{A}\mathbf{x}_s + \mathbf{g}_s) \quad , \\
&= \mathbf{x}_s^T \mathbf{A}^2 \mathbf{x}_s + 2\mathbf{x}_s^T \mathbf{A}\mathbf{g}_s + \mathbf{g}_s^T \mathbf{g}_s
\end{aligned}$$

Let $R = 2\mathbf{x}_s^T \mathbf{A}\mathbf{g}_s + \mathbf{g}_s^T \mathbf{g}_s$, then by Lemma 28, with probability $1 - o(1)$, $R \leq O(\sigma\sqrt{k \log T}) + O(d\sigma^2)$.

$$\begin{aligned}
\mathbf{x}_s^T \mathbf{A}^2 \mathbf{x}_s &= \mathbf{u}^T \mathbf{A}^2 \mathbf{u} + \mathbf{u}^{\perp T} \mathbf{A}^2 \mathbf{u}^{\perp} \\
&= \lambda_s^2 \mathbf{u}^T \mathbf{u} + \mathbf{u}^{\perp T} \mathbf{A}^2 \mathbf{u}^{\perp} \\
&\leq \lambda_s^2 \|\mathbf{u}\|^2 + \lambda_1^2 \|\mathbf{u}^{\perp}\|^2 \\
&\leq \lambda_s^2 \cos^2 \theta^{(T)} + \lambda_1^2 \sin^2 \theta^{(T)} \\
&= \lambda_s^2 (1 - \sin^2 \theta^{(T)}) + \lambda_1^2 \sin^2 \theta^{(T)} \\
&= \lambda_s^2 + (\lambda_1^2 - \lambda_s^2) \sin^2 \theta^{(T)}.
\end{aligned}$$

Thus,

$$\begin{aligned}
|\hat{\lambda}_s - \lambda_s| &\leq \frac{(\lambda_1^2 - \lambda_s^2)}{\hat{\lambda}_s + \lambda_s} \sin^2 \theta^{(T)} + O(\sigma\sqrt{k \log T}) + O(d\sigma^2) \\
&= O(\frac{\sigma^2 d \log T}{\gamma_s^2 \lambda_s^2}) + O(\sigma\sqrt{k \log T}).
\end{aligned}$$

Plug in the setting of σ the corollary follows. ■

Dataset	Size (n)	# Attributes (d)
NOM	20643	116
NUR	12958	28
CTG	2126	42

Table 3: Summary of the dataset

Remark 29 *The accuracy of both the private eigenvectors and the private eigenvalues can be improved by considering the matrix coherence of A, as analyzed in Hardt (2013). Typically, the accuracy can be improved by a factor of $O\left(\frac{1}{d} \cdot \operatorname{polylog}(d)\right)$.*

Proof of Theorem 25 The theorem follows immediately from Lemma 26 and Lemma 3.6 in Hardt (2013). ∎

5.2.1 Experimental Results

Here we provide experimental results for the practical variant mechanism described above. It turns out that this algorithm is also far more efficient than our first mechanism given in Section 3 as answering a query is still time consuming in practice.

We adopt three datasets all from the UCI repository. A summary of the size and the number of attributes of these datasets is given in Table 3. Since the data universe considered in this paper is $[-1,1]^d$, we normalize each attribute to $[-1,1]$.

We conduct two sets of experiments in order to have a relatively comprehensive understanding of the *utility* of the differentially private synthetic database generated by our mechanism. In both sets of experiments we first output the synthetic database. In one set of experiments, we test the accuracy of query answering, as described in the previous sections. In the other set of experiments, we consider a very different task. We learn a SVM classifier from the synthetic database, and evaluate its accuracy (on the original data). We think that this task may be more useful from a practical point of view. It is worth pointing out that in the scenario of classification, the classification error ($0-1$ loss) is not a smooth function. Therefore the second set of experiments might be viewed, in a sense, as a test of how well our mechanism generalize to non-smooth functions.

The queries employed in the first set of experiments are linear combinations of Gaussian kernel functions. We use this type of functions because 1) These functions possess good smoothness property as stated in Section 2, and 2) linear combinations of Gaussian are universal approximators.

Detailed parameter setting of the query functions is as follows. We consider

$$f(\mathbf{x}) = \sum_{j=1}^{J} \alpha_j \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_j\|^2}{2\sigma^2}\right).$$

In all experiments, we set $J = 10$; α_j is randomly chosen from $[0,1]$, and $\mathbf{x}_j$ is randomly chosen from $[-1,1]^d$. We test various values of σ and various ϵ to see how the smoothness

Dataset	ϵ	Error	σ					Time (s)
			2	4	6	8	10	
NOM	0.1	Abs	0.0276	0.0859	0.0750	0.0520	0.0374	8.5
		Rel	0.6800	0.2192	0.1138	0.0657	0.0433	
	1	Abs	0.0155	0.0506	0.0330	0.0286	0.0202	8.2
		Rel	0.4600	0.1286	0.0521	0.0373	0.0237	
	10	Abs	0.0047	0.0199	0.0070	0.0096	0.0065	8.5
		Rel	0.1660	0.0594	0.0109	0.0128	0.0077	
NUR	0.1	Abs	0.0050	0.0018	0.0015	0.0004	0.0001	2.3
		Rel	0.0057	0.0018	0.0015	0.0004	0.0001	
	1	Abs	0.0015	0.0007	0.0003	0.0003	0.0002	3.1
		Rel	0.0018	0.0008	0.0003	0.0004	0.0001	
	10	Abs	0.0037	0.0001	0.0001	0.0005	0.0001	2.7
		Rel	0.0042	0.0001	0.0001	0.0006	0.0001	
CTG	0.1	Abs	0.1022	0.2113	0.1479	0.0984	0.0693	0.7
		Rel	0.8590	0.3871	0.1981	0.1139	0.0770	
	1	Abs	0.0970	0.1605	0.1193	0.0770	0.0505	0.7
		Rel	0.8007	0.2923	0.1597	0.0891	0.0560	
	10	Abs	0.0462	0.0632	0.0430	0.0353	0.0147	0.8
		Rel	0.3881	0.1184	0.0602	0.0415	0.0165	

Table 4: Worst-case error for the variant of the database-outputting mechanism

of the query function and privacy parameter affect the performance of the algorithm (see below for detailed results).

We use different performance measures to evaluate the algorithm. The goal is to have a comprehensive understanding of the performance of the mechanism. We consider the worst-case error of the mechanism over the set of queries. Because our query set, i.e., linear combination of Gaussian Kernels, contains infinitely many functions, we randomly choose 10^4 queries in each experiment. The worst-case error is over these 10^4 queries.

We give both absolute error and relative error for all experiments. The absolute error of a query q_f is defined as $|q_f(D) - q_f(\tilde{D})|$; and relative error is defined as $|\frac{q_f(D)-q_f(\tilde{D})}{q_f(D)}|$. We present relative error because in certain cases (e.g. when σ is small) $f(\mathbf{x})$ is very small for most $\mathbf{x} \in D$. Therefore in this case a small absolute error does not necessarily imply good performance, and relative error is more informative[3].

We present the running time of the mechanism for outputting the synthetic database in each experiment. The computer used in all the experiments is a workstation with 2 Intel Xeon X5650 processors of 2.67GHz and 32GB RAM.

We present the performance of the ϵ-differentially private algorithm in Table 4. For each dataset, both absolute error and relative error, as average of 20 rounds, are reported

3. One also needs to be careful when using relative error. In our experiments, we deliberately set $\alpha_j \in [0, 1]$. So $f(\mathbf{x}) \geq 0$ for all $\mathbf{x}$. If instead we set $\alpha_j \in [-1, 1]$, then $f(\mathbf{x})$ can be either positive or negative, and it is possible that $q_f(D)$ is close to zero while $f(\mathbf{x})$ is not small for most $\mathbf{x} \in D$. In such a case, a large relative error does not necessarily imply a bad performance.

Dataset	Original	ϵ		
		0.1	1	10
NOM	0.9353	0.6328	0.7959	0.8491
NUR	0.9081	0.5361	0.7748	0.8164
CTG	0.9755	0.5309	0.5853	0.6116

Table 5: AUC for the variant of the database-outputting mechanism

sequentially. We make use of linear combination of Gaussian with different values of σ as the query functions. The last column of the table lists the running time with respect to the worst σ of the algorithm for outputting the synthetic database.

Now we analyze the experimental results in Table 4 in greater detail. First, the algorithm is quite efficient. On all datasets, the mechanism outputs the synthetic databases in a few seconds. Next consider the accuracy. As explained earlier, the relative error is more meaningful in our experiments. It can be seen that except for the case $\sigma = 2$ (recall that $K = \sigma^2$), the accuracy is reasonably good. The relative errors decrease monotonically as the the order of smoothness of the queries increases.

We then turn to describe the second set of experiments. We randomly partition each dataset into two subsets of equal size. One subset is used as *training* dataset, the other as *test* dataset. By running our mechanism, we generate a differentially private synthetic database using the training dataset as input. Since our task is classification, and the attribute values in the synthetic dataset are continuous ($[-1, 1]$), we round the label attribute to $\{-1, 1\}$. We then learn a SVM classifier from the synthetic dataset. Finally we test the classifier's performance on the *test* dataset. As a comparison, we also list the performance of the SVM classifier learned from the non-private training data, as shown in the second column of Table 5. Here, the performance measures are Area Under ROC Curve (AUC).

We test three values of ϵ, and the performances are given in Table 5. For each dataset, the AUC is an average of 10 rounds. It can be seen that when ϵ is small, there is usually a relatively big utility gap. But as ϵ getting large, the performances improve quickly.

6. Conclusion

In this paper, we study differentially private mechanisms with respect to the K-smooth queries. We first propose a differentially private mechanism for efficiently answering smooth queries. The running time for outputting the summary is $O(n^{1\sim 1.5})$. For queries of high order smoothness, the running time for answering a query is sublinear, and nearly $O(n^{\epsilon_0})$. Furthermore, the mechanisms achieve an accuracy nearly $O(n^{-1})$ in highly smooth case, much better than the sampling error $O(n^{-1/2})$ which is inherent to differentially private mechanisms answering general queries.

From a practical viewpoint, outputting a synthetic database while preserving differential privacy is more appealing. In this paper, we also propose a differentially private mechanism which output synthetic database. The user can obtain accurate answers to all smooth queries from the synthetic database. Our mechanisms run in polynomial time, while existing algorithms run in super-exponential time. This synthetic dataset outputing mechanism achieves almost the same level of accuracy as the query-answering mechanism.

There is a future direction we think worth exploring. As mentioned in Introduction, there exists an efficient and differentially private algorithm which outputs a synthetic database and is accurate to the class of rectangle queries defined on $[-1, 1]^d$ (Blum et al., 2008). Rectangle queries are not smooth. They are specified by indicator functions which are not even continuous. The mechanism proposed in Blum et al. (2008) is completely different to the mechanism for smooth queries given in this paper. Thus an immediate question is: can we develop efficient mechanisms which output synthetic database and preserve differential privacy for a natural class of queries containing both smooth and important non-smooth functions?

Acknowledgments

This work was supported by National Basic Research Program of China (973 Program)(grant no. 2015CB352502), NSFC(61573026, 61222307), IBM Faculty Award and a grant from Microsoft Research Asia.

References

C. Aggarwal and P. Yu. A general survey of privacy preserving data mining models and algorithms. In *Privacy-Preserving Data Mining*, chapter 2, pages 11–52. Springer, 2008.

K. M. Anstreicher. Linear programming in $O(\frac{n^3}{\ln n}L)$ operations. *SIAM J. on Optimization*, 9(4):803–812, April 1999.

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

A. Blum and A. Roth. Fast private data release algorithms for sparse queries. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 395–410. Springer, 2013.

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

K. Chaudhuri and D. Hsu. Sample complexity bounds for differentially private learning. In *COLT*, 2011.

K. Chaudhuri, C. Monteleoni, and A.D. Sarwate. Differentially private empirical risk minimization. *JMLR*, 12:1069–1109, 2011.

K. Chaudhuri, A. Sarwate, and K. Sinha. Near-optimal differentially private principal components. In *NIPS*, pages 998–1006, 2012.

M. Cheraghchi, A. Klivans, P. Kothari, and H.K. Lee. Submodular functions are noise stable. In *SODA*, pages 1586–1592. SIAM, 2012.

C.W. Clenshaw and A.R. Curtis. A method for numerical integration on an automatic computer. *Numerische Mathematik*, 2(1):197–205, 1960.

R.A. DeVore and G. G. Lorentz. *Constructive approximation*, volume 303. Springer Verlag, 1993.

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

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

C. Dwork, G.N. Rothblum, and S. Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60. IEEE, 2010.

T. Gerstner and M. Griebel. Numerical integration using sparse grids. *Numerical algorithms*, 18(3-4):209–232, 1998.

A. Gupta, M. Hardt, A. Roth, and J. Ullman. Privately releasing conjunctions and the statistical query barrier. In *STOC*, pages 803–812. ACM, 2011.

M. Hardt. Robust subspace iteration and privacy-preserving spectral analysis. *arXiv preprint arXiv:1311.2495*, 2013.

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

M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. In *NIPS*, 2012a.

M. Hardt, G. N. Rothblum, and R. A. Servedio. Private data release via learning thresholds. In *SODA*, pages 168–187. SIAM, 2012b.

J. Indritz. An inequality for hermite polynomials. *Proceedings of the American Mathematical Society*, 12(6):981–983, 1961.

P. Jain, P. Kothari, and A. Thakurta. Differentially private online learning. In *COLT*, 2012.

D. Kifer and B.R. Lin. Towards an axiomatization of statistical privacy and utility. In *PODS*, pages 147–158. ACM, 2010.

J. Lei. Differentially private M-estimators. In *NIPS*, 2011.

R. D.C. Monteiro and I. Adler. Interior path following primal-dual algorithms. Part I: Linear programming. *Math. Program.*, 44(1):27–41, June 1989.

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

A. Smola, B. Schölkopf, and K. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11(4):637–649, 1998.

S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. In *Dokl. Akad. Nauk SSSR*, volume 4, page 123, 1963.

V.N. Temlyakov. *Approximation of periodic functions.* Nova Science Pub Inc, 1994.

J. Thaler, J. Ullman, and S. Vadhan. Faster algorithms for privately releasing marginals. In *ICALP*, pages 810–821. Springer, 2012.

J. Ullman. Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. In *STOC*. ACM, 2013.

J. Ullman. Private multiplicative weights beyond linear queries. In *PODS*, 2015.

A. van der Vart and J.A. Wellner. *Weak Convergence and Empirical Processes.* Springer, 1996.

M. Vyazovskaya and N. Pupashenko. On the normalizing multiplier of the generalized jackson kernel. *Mathematical Notes*, 80(1-2):19–26, 2006.

G. Wahba et al. Support vector machines, reproducing kernel Hilbert spaces and the randomized gacv. *Advances in Kernel Methods-Support Vector Learning*, 6:69–87, 1999.

L. Wang. Smoothness, disagreement coefficient, and the label complexity of agnostic active learning. *JMLR*, 12:2269–2292, 2011.

Z. Wang, K. Fan, J. Zhang, and L. Wang. Efficient algorithm for privately releasing smooth queries. In *NIPS*, 2013.

L. Wasserman and S. Zhou. A statistical framework for differential privacy. *Journal of the American Statistical Association*, 105(489):375–389, 2010.

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