Efficient Algorithm for Privately Releasing Smooth Queries

Ziteng Wang
Key Laboratory of Machine Perception, MOE
School of EECS
Peking University
wangzt@cis.pku.edu.cn

Kai Fan
Key Laboratory of Machine Perception, MOE
School of EECS
Peking University
interfk@hotmail.com

Jiaqi Zhang
Key Laboratory of Machine Perception, MOE
School of EECS
Peking University
grzhang.jq@gmail.com

Liwei Wang
Key Laboratory of Machine Perception, MOE
School of EECS
Peking University
wanglw@cis.pku.edu.cn

Abstract

We study differentially private mechanisms for answering *smooth* queries on databases consisting of data points in $\mathbb{R}^d$. A K-smooth query is specified by a function whose partial derivatives up to order K are all bounded. We develop an ϵ-differentially private mechanism which for the class of K-smooth queries has accuracy $O(\left(\frac{1}{n}\right)^{\frac{K}{2d+K}}/\epsilon)$. The mechanism first outputs a summary of the database. To obtain an answer of a query, the user runs a public evaluation algorithm which contains no information of the database. Outputting the summary runs in time $O(n^{1+\frac{d}{2d+K}})$, and the evaluation algorithm for answering a query runs in time $\tilde{O}(n^{\frac{d+2+\frac{2d}{K}}{2d+K}})$. Our mechanism is based on L_∞-approximation of (transformed) smooth functions by low degree even trigonometric polynomials with small and efficiently computable coefficients.

1 Introduction

Privacy is an important problem in data analysis. Often people want to learn useful information from data that are sensitive. But when releasing statistics of sensitive data, one must tradeoff between the accuracy and the amount of privacy loss of the individuals in the database.

In this paper we consider *differential privacy* [10], which has become a standard concept of privacy. Roughly speaking, a mechanism which releases information about the 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 any individual to submit her information to the database is very small. An adversary can discover almost nothing new from the database that contains the individual's information compared with that from the database without the individual's information. Recently there have been extensive studies of machine learning, statistical estimation, and data mining under the differential privacy framework [33, 5, 19, 18, 6, 34, 22, 4].

Accurately answering statistical queries is a well studied problem in differential privacy. A simple and efficient method is the Laplace mechanism [10], which adds Laplace noise to the true answers. Laplace mechanism is especially useful for query functions with low sensitivity, which is the max-

imal 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 queries to be smaller than n^2 is too restricted in some situations. A remarkable result due to Blum, Ligett and Roth [2] 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 are a series of works [11, 12, 23, 17] improving the result of [2]. 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 [17, 15] run in time linear in the size of the data universe to answer a query. Often the size of the data universe is much larger than that of the database, so these mechanisms are inefficient. Recently, [28] shows that there is no polynomial time algorithm that can answer $n^{2+o(1)}$ general queries while preserving privacy and accuracy (assuming the existence of one-way function).

Given the hardness result, recently there are growing interests in studying efficient and differentially private mechanisms for restricted class of queries. From a practical point of view, if there exists a class of queries which is rich enough to contain most queries used in applications and allows one to develop fast mechanisms, then the hardness result is not a serious barrier for differential privacy.

One class of queries that attracts a lot of attentions is 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 [1, 14, 7, 16, 27] . They propose elegant mechanisms which run in time $\mathrm{poly}(n)$ when k is a constant. Another class of queries that yields efficient mechanisms is sparse query. A query is m-sparse if it takes non-zero values on at most m elements in the data universe. [3] develops mechanisms which are efficient when $m = \mathrm{poly}(n)$.

When the data universe is $[-1,1]^d$, where d is a constant, [2] considers rectangle queries. A rectangle query is specified by an axis-aligned rectangle. The answer to the query is the fraction of the data points that lie in the rectangle. [2] shows that if $[-1,1]^d$ is discretized to $\mathrm{poly}(n)$ bits of precision, then there are efficient mechanisms for the class of rectangle queries. There are also works studying related range queries [20].

In this paper we study *smooth* queries defined also on data universe $[-1,1]^d$ for constant d. A smooth query is specified by a smooth function, which has bounded partial derivatives up to a certain order. The answer to the query is the average of the function values on data points in the database. Smooth functions are widely used in machine learning and data analysis [32]. There are extensive studies on the relation between smoothness, regularization, reproducing kernels and generalization ability [31, 24].

Our main result is an ϵ-differentially private mechanism for the class of K-smooth queries, which are specified by functions with bounded partial derivatives up to order K. The mechanism has (α,β)-accuracy, where $\alpha = O\left(\left(\frac{1}{n}\right)^{\frac{K}{2d+K}}/\epsilon\right)$ for $\beta \geq e^{-O(n^{\frac{d}{2d+K}})}$. The mechanism first 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 $\tilde{O}(n^{\frac{d+2+\frac{2d}{K}}{2d+K}})$. The mechanism has the advantage that both the accuracy and the running time for answering a query improve quickly as K/d increases (see also Table 1 in Section 3).

Our algorithm is a L_∞-approximation based mechanism and is motivated by [27], which considers approximation of k-way conjunctions by low degree polynomials. The basic idea is to approximate the whole query class by linear combination of a small set of basis functions. The technical difficulties lie 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 and efficiently computable.

To guarantee these properties, 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. The smoothness of the functions also 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.

2 Background

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 2.1 ((ϵ, δ)-differential privacy)**.** A sanitizer $\mathcal{S}$ which is an algorithm that maps input database into some range $\mathcal{R}$ is said to preserve (ϵ, δ)-differential privacy, if for all pairs of neighbor databases D, D' and for any subset $A \subset \mathcal{R}$, it holds that

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

If $\mathcal{S}$ preserves $(\epsilon, 0)$-differential privacy, we say $\mathcal{S}$ is ϵ-differentially private.

We consider *linear queries*. Each linear query q_f is specified by a function f which maps data universe $[-1, 1]^d$ to $\mathbb{R}$, and q_f is defined by $q_f(D) := \frac{1}{|D|} \sum_{x \in D} f(x)$.

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

Definition 2.2 ((α, β)-accuracy)**.** Let Q be a set of queries. A sanitizer $\mathcal{S}$ 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

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

where $\mathcal{S}(D, q)$ is the answer to q given by $\mathcal{S}$.

We will make use of Laplace mechanism [10] in our algorithm. Laplace mechanism adds Laplace noise to the output. We denote by $\mathrm{Lap}(\sigma)$ the random variable distributed according to the Laplace distribution with parameter σ: $\mathbb{P}(\mathrm{Lap}(\sigma) = x) = \frac{1}{2\sigma} \exp(-|x|/\sigma)$.

We will design a differentially private mechanism which is accurate with respect to a query set Q possibly consisting of infinite number of queries. Given a database D, the sanitizer outputs a *summary* which preserves differential privacy. For any $q_f \in Q$, the user makes use of an evaluation procedure to measure f on the summary and obtain an approximate answer of $q_f(D)$. Although we may think of the evaluation procedure as part of the mechanism, it does not contain any information of the database and therefore is public. We will study the running time for the sanitizer outputting the summary. Ideally it is $O(n^c)$ for some constant c not much larger than 1. For the evaluation procedure, the running time *per query* is the focus. Ideally it is sublinear in n. Here and in the rest of the paper, we assume that calculating the value of f on a data point x can be done in unit time.

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 $p(\theta) = a_0 + \sum_{l=1}^{m} (a_l \cos l\theta + b_l \sin l\theta)$, where a_l, b_l are constants. If $p(\theta)$ is an even function, we say that it is an even trigonometric polynomial, and $p(\theta) = a_0 + \sum_{l=1}^{m} a_l \cos l\theta$. For the multivariate case, if $p(\theta_1, \ldots, \theta_d) = \sum_{\mathbf{l}=(l_1,\ldots,l_d)} a_{\mathbf{l}} \cos(l_1\theta_1) \ldots \cos(l_d\theta_d)$, then p is said to be an even trigonometric polynomial (with respect to each variable), and the degree of θ_i is the upper limit of l_i.

3 Efficient differentially private mechanism

Let us first describe the set of queries considered in this work. Since each 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 with nonnegative integers, then we define

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

Parameters: Privacy parameters $\epsilon, \delta > 0$; Failure probability $\beta > 0$;
Smoothness order $K \in \mathbb{N}$; Set $t = n^{\frac{1}{2d+K}}$.
Input: Database $D \in \left([-1,1]^d\right)^n$.
Output: A t^d-dimensional vector as the summary.
Algorithm:
For each $\mathbf{x} = (x_1, \ldots, x_d) \in D$:
Set: $\theta_i(\mathbf{x}) = \arccos(x_i)$, $i = 1, \ldots, d$;
For every d-tuple of nonnegative integers $\mathbf{m} = (m_1, \ldots, m_d)$, where $\|\mathbf{m}\|_\infty \leq t-1$
Compute: $\mathrm{Su}_{\mathbf{m}}(D) = \frac{1}{n}\sum_{\mathbf{x}\in D} \cos(m_1\theta_1(\mathbf{x})) \ldots \cos(m_d\theta_d(\mathbf{x}))$;
$\widehat{\mathrm{Su}}_{\mathbf{m}}(D) \leftarrow \mathrm{Su}_{\mathbf{m}}(D) + \mathrm{Lap}\left(\frac{t^d}{n\epsilon}\right)$;
Let $\widehat{\mathrm{Su}}(D) = \left(\widehat{\mathrm{Su}}_{\mathbf{m}}(D)\right)_{\|\mathbf{m}\|_\infty \leq t-1}$ be a t^d dimensional vector;
Return: $\widehat{\mathrm{Su}}(D)$.

Algorithm 1: Outputting the summary

Parameters: $t = n^{\frac{1}{2d+K}}$.
Input: A query q_f, where $f : [-1,1]^d \to \mathbb{R}$ and $f \in C_B^K$,
Summary $\widehat{\mathrm{Su}}(D)$ (a t^d-dimensional vector).
Output: Approximate answer to $q_f(D)$.
Algorithm:
Let $g_f(\boldsymbol{\theta}) = f(\cos(\theta_1), \ldots, \cos(\theta_d))$, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d) \in [-\pi, \pi]^d$;
Compute a trigonometric polynomial approximation $p_t(\boldsymbol{\theta})$ of $g_f(\boldsymbol{\theta})$,
where the degree of each θ_i is t; // see Section 4 for details of computation.
Denote $p_t(\boldsymbol{\theta}) = \sum_{\mathbf{m}=(m_1,\ldots,m_d), \|\mathbf{m}\|_\infty < t} c_{\mathbf{m}} \cos(m_1\theta_1) \ldots \cos(m_d\theta_d)$;
Let $\mathbf{c} = (c_{\mathbf{m}})_{\|\mathbf{m}\|_\infty < t}$ be a t^d-dimensional vector;
Return: the inner product $< \mathbf{c}, \widehat{\mathrm{Su}}(D) >$.

Algorithm 2: Answering a query

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, $C_B^K := \{f : \|f\|_K \leq B\}$. 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 [29, 32, 31] and found wide applications [24].

The following theorem is our main result. It says that if the query class is specified by smooth functions, then there is a very efficient mechanism 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 3.1. *Let the query set be $Q_{C_B^K} = \{q_f = \frac{1}{n}\sum_{\mathbf{x}\in D} f(\mathbf{x}) : f \in C_B^K\}$, 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 $\mathcal{S}$ given in Algorithm 1 and Algorithm 2 satisfies that for any $\epsilon > 0$, the following hold:*

1) The mechanism is ϵ-differentially private.

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

Table 1: Performances vs. Order of smoothness

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

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

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

The proof of Theorem 3.1 is given in the appendix.

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 $O(\frac{1}{\sqrt{n}})$.

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 about $n^{1/4}$ if d is not too small. In practice, the speed for answering a query may be more important than that for outputting the summary since the sanitizer only output the summary once. Thus having an n^c-time ($c \ll 1$) algorithm for query answering will be appealing.

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* $\cos(m_1\theta_1)\ldots\cos(m_d\theta_d)$. For each trigonometric monomial, the highest degree of any variable is $t := \max_d m_d = O(n^{\frac{1}{2d+K}})$. The summary is a $O(n^{\frac{d}{2d+K}})$-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 [27]. An approximation based mechanism relies on three conditions:

- There exists a small set of basis functions such that every query function can be well approximated by a linear combination of them.
- All the linear coefficients are small.
- 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 simply the inner product of the coefficients and the summary vector.

The following theorem 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 3.2. *Let $\gamma > 0$. For every $f \in C_B^K$ defined on $[-1,1]^d$, let*

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

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

$$p(\theta_1, \ldots, \theta_d) = \sum_{0 \le l_1, \ldots, l_d < t(\gamma)} c_{l_1,\ldots,l_d} \cos(l_1\theta_1) \ldots \cos(l_d\theta_d),$$

such that

1) $\|g_f - p\|_\infty \le \gamma$.

2) All the linear coefficients $c_{l_1,\ldots,l_d}$ 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 3.2 is proved in Section 4. Based on Theorem 3.2, the proof of Theorem 3.1 is mainly the argument for Laplace mechanism together with an optimization of the approximation error γ trading-off with the Laplace noise. (Please see the appendix.)

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

In this section we prove Theorem 3.2. 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 4.1 describes the classical theory on trigonometric polynomial approximation of smooth functions. Section 4.2 shows that the coefficients have a small upper bound and can be efficiently computed. Theorem 3.2 then follows from these results.

4.1 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 [9]; and to [26] 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 4.1. 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 $I^d_{t,K}$ 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 $I^d_{t,K}(g)$ is a trigonometric polynomial of d-variables and each variable has degree at most t.

Theorem 4.1. *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*

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

where C depends only on M, d and K.

4.2 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 4.2. *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 4.3. *There exists a constant M which depends only on K, B, d but independent of t, such that for every $f \in C_B^K$, all the linear coefficients $c_{n_1,\ldots,n_d}$ of $I^d_{t,K}(g_f)$ satisfy*

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

The proof of Lemma 4.3 is given in the appendix.

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 [13] 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 appendix. 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 4.4. *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 [13] (given in the appendix). 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 $f \in C^K_B$, in order that $\|\hat{I}^d_{t,K}(g_f) - I^d_{t,K}(g_f)\|_\infty \le O\left(t^{-K}\right)$, 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 4.4 is given in the appendix. Theorem 3.2 then follows easily from Lemma 4.3 and Lemma 4.4.

Proof of Theorem 3.2. Setting $t = t(\gamma) = \left(\frac{1}{\gamma}\right)^{1/K}$. Let $p = \hat{I}^d_{m,K}(g_f)$. Combining Lemma 4.3 and Lemma 4.4, and note that the coefficients $\hat{c}_{n_1,\ldots,n_d}$ are upper bounded by a constant, the theorem follows.

□

5 Conclusion

In this paper we propose an ϵ-differentially private mechanism for efficiently releasing K-smooth queries. The accuracy of the mechanism is $O(\left(\frac{1}{n}\right)^{\frac{K}{2d+K}})$. The running time for outputting the summary is $O(n^{1+\frac{d}{2d+K}})$, and is $\tilde{O}(n^{\frac{d+2+2d/K}{2d+K}})$ for answering a query. The result can be generalized to (ϵ,δ)-differential privacy straightforwardly using the composition theorem [12]. The accuracy improves slightly to $O((\frac{1}{n})^{\frac{2K}{3d+2K}}\log(\frac{1}{\delta})^{\frac{K}{3d+2K}})$, while the running time for outputting the summary and answering the query increase slightly. Our mechanism is based on approximation of smooth functions by linear combination of a small set of basis functions with small and efficiently computable coefficients. Directly approximating functions in $C^K_B([-1,1]^d)$ by polynomials does not guarantee small coefficients and is less efficient. To achieve these goals we use trigonometric polynomials to approximate a transformation of the query functions.

It is worth pointing out that the approximation considered here for differential privacy is L_∞-approximation, because the accuracy is defined in the worst case sense with respect to databases and queries. L_∞-approximation is different to L_2-approximation, which is simply the Fourier transform if we use trigonometric polynomials as the basis functions. L_2-approximation does not guarantee (worst case) accuracy.

For the class of smooth functions defined on $[-1,1]^d$ where d is a constant, in fact it is not difficult to design a $\mathrm{poly}(n)$ time differentially private mechanism. One can discretize $[-1,1]^d$ to $O(\frac{1}{\sqrt{n}})$ precision, and use the differentially private mechanism for answering general queries (e.g., PMW [17]). However the mechanism runs in time $\tilde{O}(n^{d/2})$ to answer a query, and provides $\tilde{O}(n^{-1/2})$ accuracy. In contrast our mechanism exploits higher order smoothness of the queries. It is always more efficient, and for queries highly smooth it is more accurate.

Finally, the smooth query should not be confused with the smooth sensitivity [21], which is an upper bound of the local sensitivity. Different to the global sensitivity, local sensitivity measures the sensitivity of a query on a data element and can be smaller than global sensitivity. For our problem, both global and local sensitivities are $O(\frac{1}{n})$.

Acknowledgments

This work was supported by NSFC(61222307, 61075003) and a grant from MOE-Microsoft Key Laboratory of Statistics and Information Technology of Peking University. We also thank Di He for very helpful discussions.

Appendix

Here we give proofs of the theorems and lemmas, as well as a description of the sparse grids algorithm.

Proof of Theorem 3.1

Proof. We prove the four results separately.

1) 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 $\mathrm{Lap}(\frac{t^d}{n\epsilon})$ preserves ϵ-differential privacy.

2) 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 3.2 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. $\mathrm{Lap}(\frac{t^d}{n\epsilon})$. 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., $|\mathrm{Lap}(\frac{t^d}{n\epsilon})|$). The following lemma gives it an upper bound.

Lemma 5.1. *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

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

□

Part 2) of Theorem 3.1 then follows from Lemma 5.1.

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

4) 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 3.2 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.

□

Proof of Lemma 4.3 We first give a simple lemma.

Lemma 5.2. *Let*

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

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

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

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 4.3. We first bound $m_{l_1,k_1,\ldots,l_d,k_d}$. Recall that (see also (5) in Fact 4.2)

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

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.

□

The sparse grids algorithm In this section we briefly describe the sparse grids numerical integration algorithm due to Gerstner and Griebel. (Please refer to [13] 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 Smolyaks [25], 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 [8] 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:

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

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

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

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

Next, for any fixed k and j, define

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

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.

Proof of Lemma 4.4 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 5.3. *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 $\left|\int_{[\pi,\pi]^d} h(\theta)\mathrm{d}\theta - \mathrm{SG}(h)\right| \leq \tau$, 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 of Lemma 5.3. 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

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

In [13] 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 $w_{k,j}$, $1 \leq j \leq m(k)$. 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 4.4. First, we need the following famous result.

Lemma 5.4. *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 [30].

Lemma 5.5 ([30])**.** *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 4.4.

Proof of Lemma 4.4. Assume that the error induced by the sparse grids algorithm is at most τ per integration. That is, for every $\mathbf{k} = (k_1, \ldots, k_d), \mathbf{l} = (l_1, \ldots, l_d)$

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

Then

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

By Lemma 5.2, $|a_{l_i}| \leq \frac{1}{\pi}$. We obtain that

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

for some constant M independent of t.

Similarly, we have

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

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

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

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 5.5, one can compute the linear coefficients a_l by solving a system of $t+1$ linear equations. That is, we choose an 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 5.3, 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 5.4 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 $O\left(t^{(1+\frac{2}{K})d+2} \mathrm{polylog}(t)\right)$. 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.

□

References

[1] 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.

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

[3] A. Blum and A. Roth. Fast private data release algorithms for sparse queries. *arXiv preprint arXiv:1111.6842*, 2011.

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

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

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

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

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

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

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

[11] 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.

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

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

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

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

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

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

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

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

[20] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, pages 123–134. ACM, 2010.

[21] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84. ACM, 2007.

[22] Pravesh K. Prateek J. and Abhradeep T. Differentially private online learning. In *COLT*, 2012.

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

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

[25] S.A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. In *Dokl. Akad. Nauk SSSR*, volume 4, page 111, 1963.

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

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

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

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

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

[31] 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.

[32] L. Wang. Smoothness, disagreement coefficient, and the label complexity of agnostic active learning. *Journal of Machine Learning Research*, 12(2269-2292):5–2, 2011.

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

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