Fast Private Kernel Density Estimation via Locality Sensitive Quantization

Tal Wagner [1] **Yonatan Naamad** [1] **Nina Mishra** [1]

Abstract

We study efficient mechanisms for differentially private kernel density estimation (DP-KDE). Prior work for the Gaussian kernel described algorithms that run in time exponential in the number of dimensions d. This paper breaks the exponential barrier, and shows how the KDE can privately be approximated in time linear in d, making it feasible for high-dimensional data. We also present improved bounds for low-dimensional data.

Our results are obtained through a general framework, which we term Locality Sensitive Quantization (LSQ), for constructing private KDE mechanisms where existing KDE approximation techniques can be applied. It lets us leverage several efficient non-private KDE methods—like Random Fourier Features, the Fast Gauss Transform, and Locality Sensitive Hashing—and "privatize" them in a black-box manner. Our experiments demonstrate that our resulting DP-KDE mechanisms are fast and accurate on large datasets in both high and low dimensions.

1. Introduction

Private analysis of massive-scale data is a prominent current challenge in computing and machine learning. On the one hand, it is widely acknowledged that big datasets drive advances and progress in many important problem spaces. On the other hand, when the data contains sensitive information such as personal or medical details, it is often necessary to preserve the privacy of individual dataset records. Scalable methods for private computations are therefore crucial for progress in medical, financial and many other domains.

Differential privacy (DP) (Dwork et al., 2006) is a rigorous and powerful notion of privacy-preserving computation, widely accepted in machine learning. Unfortunately, it often comes at a high computational cost, and many DP computations are dramatically less efficient than their non-private counterparts. This makes them infeasible for use on data of the size and dimensionality that matches present-day scale.

DP-KDE. In this paper we focus on private density estimation, a fundamental problem with numerous applications in data analysis and machine learning. A popular way to convert a collection of data points into a smoothed probability distribution is the kernel density method, wherein a certain probability measure—say, a Gaussian—is centered at each data point, and the mixture of these measures is formed over the space. The kernel density estimate (KDE) at every point y is the mean of all such Gaussians evaluated at y. This method has a long history in statistics and machine learning (e.g., (Shawe-Taylor et al., 2004; Hofmann et al., 2008)). Under private computation, it has recently been used for private crowdsourcing and location sharing (Huai et al., 2019; Cunningham et al., 2021).

The associated algorithmic task is: given a dataset $X \subset \mathbb{R}^d$, return a map $\hat{e}_X : \mathbb{R}^d \to \mathbb{R}$ that approximates the KDE map $y \mapsto \frac{1}{|X|} \sum_{x \in X} e^{-\|x-y\|_2^2/\sigma^2}$. In DP-KDE, $\hat{e}_X$ must also be private w.r.t. X, no matter how many times it is queried.

Absent privacy limitations, the Gaussian KDE at a point y can be evaluated in time $O(nd)$, where n is the number of data points and d is their dimension. Many efficient approximation methods have been developed to speed this up even further for large-scale data (e.g., (Greengard & Strain, 1991; Rahimi & Recht, 2007; Charikar & Siminelakis, 2017; Phillips & Tai, 2020). In sharp contrast, in the DP setting, existing methods for privately estimating the Gaussian KDE have running time exponential in d (Hall et al., 2013; Hall, 2013; Wang et al., 2016; Alda & Rubinstein, 2017). This makes them infeasible in many important cases where KDE is utilized in high-dimensional feature spaces.

In this paper, we close this gap by systematically studying efficient mechanisms for DP-KDE, and obtain improved results in both the high and low dimensional regimes.

1.1. Our Results

We focus on the Gaussian kernel, although we will discuss other kernels as well. Our first result is an ϵ-DP function release mechanism for Gaussian KDE (see Sections 1.2

[1] Amazon. Correspondence to: Tal Wagner <tal.wagner@gmail.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Table 1. ϵ-DP KDE function release mechanisms for the Gaussian kernel, that satisfy (α, η)-approximation (Definition 1.3). The dataset contains n points in $\mathbb{R}^d$. Where n appears in the curator time, note that it must be at least as large as the sample complexity. (*) SmallDB, PMW and LSQ-FGT assume that all points lie in a ball of radius Φ. (**) EvenTrig and Bernstein assume that all points lie in $[-1, 1]^d$, and their performance depends on the bandwidth σ under this scaling. (†) For EvenTrig, $\eta = \exp(-\Omega(n^{d/(2d+O(\sigma^2))}))$.

	Mechanism	Curator time	Sample complexity	
Prior	SmallDB	$\exp\left(\frac{\min\{\sqrt{d\log(1/\alpha)},1/\alpha\}\cdot d^2\log^2(\Phi/\alpha)}{\epsilon\cdot\alpha^2}\right)$	$O\left(\frac{\min\{\sqrt{d\log(1/\alpha)},1/\alpha\}\cdot d\log(\Phi/\alpha)}{\epsilon\cdot\alpha^2}\right)$	(*)
	PMW	$\tilde{O}\left(d\cdot(\Phi/\alpha)^d\right)$	$\tilde{O}\left(\frac{d^2\log^2(\Phi/\alpha)}{\epsilon\cdot\alpha^3}\right)$	(*)
	EvenTrig	$O\left(2^d + dn^{1+d/(2d+\Theta(\sigma^2))}\right)$	$O\left(\frac{1}{(\epsilon\cdot\alpha)^{1+\Theta(d/\sigma^2)}}\right)$	(**), (†)
	Bernstein	$O\left(dn\cdot\left(2^d + (\frac{\epsilon\cdot n}{\log(1/\eta)})^{d/(d+\Theta(\sigma^2))}\right)\right)$	$O\left(\frac{\log(1/\eta)}{\epsilon\cdot\alpha^{1+\Theta(d/\sigma^2)}}\right)$	(**)
Ours	LSQ-RFF	$O(dn\cdot\frac{\log(1/\eta)}{\epsilon\cdot\alpha^2})$	$O(\frac{\log(1/\eta)}{\epsilon\cdot\alpha^2})$	
	LSQ-FGT	$(dn + (\frac{\Phi}{\sqrt{d}})^d)\cdot O(\log(1/\alpha))^d\cdot\log(1/\eta)$	$\frac{\log(1/\eta)}{\epsilon\cdot\alpha}\cdot(\log(1/\alpha))^{O(d)}$	(*)

and 1.3 for formal definitions), whose running time is only linear in d, making it suitable for high-dimensional data.

Theorem 1.1 (Gaussian DP-KDE in high dimensions)**.** *There is an ϵ-DP function release mechanism for (α, η)-approximation of Gaussian KDE on datasets in $\mathbb{R}^d$ of size $n \geq O(\log(1/\eta)/(\epsilon\alpha^2))$, and:*

- *The curator runs in time $O(nd\log(1/\eta)/\alpha^2)$.*
- *The output size is $O(d\log(1/\eta)/\alpha^2)$.*
- *The client runs in time $O(d\log(1/\eta)/\alpha^2)$.*

Our second result is a Gaussian DP-KDE mechanism for low-dimensional data, if the points reside in a bounded region. It improves the dependence on α to nearly linear.

Theorem 1.2 (Gaussian DP-KDE in low dimensions)**.** *There is an ϵ-DP function release mechanism for (α, η)-approximation of Gaussian KDE on datasets in $\mathbb{R}^d$ of size $n \geq \log(1/\eta)\cdot(\log(1/\alpha))^{O(d)}/(\epsilon\alpha)$ and that are contained in a ball of radius Φ, and:*

- *The curator runs in time $(nd + (\frac{\Phi}{\sqrt{d}})^d)\cdot O(\log(1/\alpha))^{O(d)}\cdot\log(1/\eta)$.*
- *The output size is $O((1 + \frac{\Phi}{\sqrt{d}})(\log(1/\alpha)))^d\log(1/\eta)$.*
- *The client runs in time $(\log(1/\alpha))^{O(d)}\log(1/\eta)$.*

Our approach. We obtain our results by introducing a framework we call *locality sensitive quantization* (LSQ). It captures a certain type of KDE approximation algorithms, which are based on point quantization. On the one hand, we show that the LSQ properties are by themselves sufficient to imply an efficient DP-KDE mechanism. On the other hand, we show that many popular approximation methods for KDE already possess these properties—including random Fourier features (RFF) (Rahimi & Recht, 2007), the Fast Gauss Transform (FGT) (Greengard & Strain, 1991), and locality sensitive hashing (LSH) (Charikar & Siminelakis, 2017). Thus, by plugging each of these methods into the LSQ framework, we immediately get efficient DP-KDE mechanisms for the kernels they approximate.

The key properties highlighted in the LSQ framework are *quantization* (i.e., the dataset is quantized into a small number of values), *range* (these values are bounded), and *sparsity* (each single point affects only a small number of values). As mentioned, several non-private KDE algorithms operate in this manner, as it can lead to efficient and accurate approximation. The reason it is also useful for efficient DP mechanisms is roughly that quantization lets us add noise to a compact representation of the data, saving time; bounded range means the noise can have small magnitude; and sparsity ensures the noise does not add up too much.

On a broader conceptual level, there is a fundamental connection between DP and non-private approximation algorithms based on sketching (or quantization). Indeed, many recent works have uncovered such connections (Blocki et al., 2012; Feldman & Talwar, 2021; Aumüller et al., 2021; Coleman & Shrivastava, 2021; Pagh & Thorup, 2022; Nikolov, 2023). Our work adds to this growing line of research.

1.2. Preliminaries: Kernel Density Estimation (KDE)

A *kernel* is a function $k : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$ that measures similarity between points in $\mathbb{R}^d$. Popular kernels include:

- Gaussian kernel: $k(x,y)=\exp(-\|x-y\|_2^2/\sigma^2)$
- Laplacian kernel: $k(x,y)=\exp(-\|x-y\|_1/\sigma)$
- Cauchy kernel: $k(x,y)=\prod_{j=1}^{d} 2/(1+(x_j-y_j)^2/\sigma^2)$

Here, $\sigma>0$ is the *bandwidth* parameter. For simplicity, we set $\sigma=1$ throughout; this does not limit generality, as we can scale the point coordinates accordingly.

Let $X\subset\mathbb{R}^d$ be a finite dataset. The *kernel density estimation (KDE)* map $KDE_X:\mathbb{R}^d\rightarrow[0,1]$ is defined as

$$KDE_X(y)=\frac{1}{|X|}\sum_{x\in X}k(x,y).$$

Our goal will be to approximate the KDE map in the following formal sense.

Definition 1.3. Let $\hat{e}:\mathbb{R}^d\rightarrow[0,1]$ be a randomized mapping, and let $\alpha,\eta\in(0,1)$. We say that $\hat{e}$ is an (α,η)-*approximation* for KDE_X if for every $y\in\mathbb{R}^d$,

$$\Pr[|\hat{e}(y)-KDE_X(y)|\leq\alpha]\geq 1-\eta.$$

1.3. Preliminaries: Differential Privacy (DP)

Differential privacy (Dwork et al., 2006) is a setting that involves communication between two parties: the *curator*, who holds a dataset X, and the *client*, who wishes to obtain the result of some computation on the dataset. We say that two datasets X,X' are *neighboring* if omitting a single data point from one of them yields the other.

Definition 1.4. Let M be a randomized algorithm (called a *mechanism*) that maps an input dataset to a range of outputs $\mathcal{O}$. For $\epsilon,\delta>0$, we say that M is (ϵ,δ)*-DP if* for every neighboring datasets X,X' and every $O\subset\mathcal{O}$,

$$\Pr[M(X)\in O]\leq\exp(\epsilon)\cdot\Pr[M(X')\in O]+\delta.$$

The case $\delta=0$ is called *pure* differential privacy, and in that case we say that M is ϵ*-DP*.

In this paper we focus on pure differential privacy—given a desired privacy level $\epsilon>0$, the curator is only allowed to release the results of ϵ-DP computations on X. See Appendix C.3 for a discussion of non-pure DP-KDE.

Function release. We focus on the differentially private function release communication model. In this model, the curator releases an ϵ-DP description of a function $\hat{e}(\cdot)$ that satisfies Definition 1.3 for the dataset X, without seeing any queries in advance. The client can then use this description to compute $\hat{e}(y)$ for any query y. Note that since $\hat{e}(\cdot)$ itself is ϵ-DP, the client can use it for infinitely many queries without compromising the privacy of the dataset. However, as more queries are computed, the overall number of inaccurate estimates is expected to grow (as only an expected $(1-\eta)$-fraction of them is guaranteed to have error within $\pm\alpha$).

Sample complexity. There is an inherent trade-off between privacy and approximation (or utility). It can be expressed as the minimal dataset size for which both are simultaneously possible—a quantity known as the *sample complexity*. Intuitively, the larger the dataset is, the easier it is to maintain the privacy of each point while releasing accurate global information. Formally, given $\epsilon,\alpha,\eta>0$, the sample complexity $\mathrm{sc}(M)$ of a mechanism M is the smallest s such that M is both ϵ-DP and satisfies (α,η)-approximation for all datasets of size at least s.

The sample complexity affects the running time: On the one hand, the dataset size n must be at least $\mathrm{sc}(M)$. On the other hand, since the KDE at any query point is the mean of values in $[0,1]$, the curator can initially subsample the dataset down to size $O(\log(1/\eta)/\alpha^2)$ while maintaining (α,η)-approximation, by Hoeffding's inequality. The upshot is that w.l.o.g., n can always be assumed to satisfy $\mathrm{sc}(M)\leq n\leq O(\max\{\mathrm{sc}(M),\log(1/\eta)/\alpha^2\})$.

Computational efficiency. In addition to privacy and utility, we also want the curator and client algorithms to be time-efficient, and the curator output size to be space-efficient.

1.4. Prior Work

Generic linear queries. KDE queries belong to a broader class of linear queries, which are extensively studied in the DP literature. Two classical mechanisms for them are SmallDB (Blum et al., 2013) and Private Multiplicative Weights (PMW) (Hardt & Rothblum, 2010; Gupta et al., 2012). These mechanisms are designed for the DP *query release* model, where the curator only releases responses to queries provided by the client. Nonetheless they can be adapted to the more general function release model, if the KDE problem is restricted to points contained in a ball of radius Φ. We provide more details on this transformation in Appendix C. In either the query or function release model, these mechanisms run in time at least exponential in d.

DP-KDE in low dimensions. Several authors explored mechanisms specifically for DP-KDE. (Hall et al., 2013) presented a non-pure DP mechanism, based on noise correlation, in the query release model. However, when used for function release, its running time is exponential in d (see Appendix C.3 for details). (Wang et al., 2016) introduced an ϵ-DP function release mechanism for (α,η)-approximation of smooth functions, assuming all points lie in $[-1,1]^d$, using a basis of even trigonometric polynomials. Its performance for DP-KDE depends the bandwidth σ (under scaling the data into $[-1,1]^d$), and has a fixed value for η. It also entails computations that do not admit a closed form and require numerical methods. (Alda & Rubinstein, 2017) introduced the Bernstein mechanism, based on Bernstein basis polynomials, and obtained similar guarantees with any $\eta\in(0,1)$ and in a closed-form computation. The running time of both of these mechanisms is exponential in d.

Locality sensitive hashing (LSH). Recently, (Coleman & Shrivastava, 2021) broke the $\exp(d)$ barrier for DP-KDE by using LSH (Indyk & Motwani, 1998). The usefulness of LSH for non-private KDE has been observed in (Andoni & Indyk, 2009), and recently regained much attention (Charikar & Siminelakis, 2017; Siminelakis et al., 2019; Coleman & Shrivastava, 2020; Backurs et al., 2019; 2021). Then, (Coleman & Shrivastava, 2021) showed it is also useful for DP-KDE. They obtained an ϵ-DP mechanism with (α, η)-approximation and running time only linear in d.

However, their result does not apply to the Gaussian kernel. It is restricted to kernels that satisfy a property known as *LSHability*, which roughly means they can be accurately described by LSH (see Section 3.3 for the formal definition). While some popular kernels possess this property—perhaps most notably, the Laplacian kernel (Rahimi & Recht, 2007; Andoni & Indyk, 2009; Backurs et al., 2019)—other important kernels, like Gaussian and Cauchy, are not known nor believed to be LSHable (see, e.g., (Backurs et al., 2018)). See Appendix C.4 for specific LSHable kernels.

Comparison to our results. The comparison is summarized in Table 1. Our LSQ-RFF mechanism runs in time linear in d and polynomial in $1/\alpha$. Its sample complexity and computational efficiency match those of (Coleman & Shrivastava, 2021), but it works for a wider class of kernels. For the Gaussian kernel, it is the first to avoid an exponential dependence on d in the running time. Furthermore, it does not require the data to be contained in a bounded region. In the low-dimensional setting $d = O(1)$, our LSQ-FGT mechanism is the first to attain a nearly linear dependence of $O(\alpha^{-1}\log^{O(1)}(\alpha^{-1}))$ on the error α.[1]

Adaptive queries. The transformation of SmallDB and PMW from query release to function release, mentioned above, in fact endows them with a stronger property than Definition 1.3: not only they succeed on every query with probability $1 - \eta$, but they succeed on all queries *simultaneously* with a fixed probability (say 0.9). This enables the client to adaptively choose queries based on the results of previous queries, which is useful for data exploration, among other benefits (see (Cherapanamjeri & Nelson, 2020)). The same transformation can be applied to our mechanisms as well; see Appendix C.2.

2. Locality Sensitive Quantization

The following is the main definition for this paper.

Definition 2.1. Let $Q, S \geq 0$ be integers and $\alpha, R > 0$. Let $\mathcal{Q}$ be a distribution over pairs (f, g) such that:

- f and g are maps $f, g : \mathbb{R}^d \to [-R, R]^Q$.
- For every $x, y \in \mathbb{R}^d$, the Q-dimensional vectors $f(x)$ and $g(y)$ have each at most S non-zero entries.

We say that $\mathcal{Q}$ is an *α-approximate (Q, R, S)-locality sensitive quantization (abbrev. (Q, R, S)-LSQ)* family for a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$, if for every $x, y \in \mathbb{R}^d$,

$$\left|k(x, y) - \mathbb{E}_{(f,g)\sim\mathcal{Q}}[f(x)^T g(y)]\right| \leq \alpha.$$

We call k an *α-approximate (Q, R, S)-LSQable* kernel. If $\alpha = 0$, we say that $\mathcal{Q}$ is an *exact* (Q, R, S)-LSQ family for k, and that k is *(Q, R, S)-LSQable*.

Algorithm 1 : LSQ Mechanism for DP-KDE

```
Curator
  Input: Dataset $X \subset \mathbb{R}^d$; $(Q, R, S)$-LSQ family $\mathcal{Q}$; privacy parameter $\epsilon > 0$; integers $I \geq J > 0$.
  for $i = 1, \ldots, I$ do
    Sample $(f_i, g_i) \sim \mathcal{Q}$
    $F_i \leftarrow \frac{1}{|X|}\sum_{x \in X} f_i(x)$        // note: $F_i \in [-R, R]^Q$
    $\widetilde{F}_i \leftarrow F_i$ with an i.i.d. sample from Laplace$(IRS/(\epsilon|X|))$ added to each coordinate
  release $f_i, g_i, \widetilde{F}_i$ for all $i = 1, \ldots, I$.

Client
  Input: Query point $y \in \mathbb{R}^d$; the released $\{f_i, g_i, \widetilde{F}_i\}_{i=1}^I$.
  $I' \leftarrow \lfloor I/J \rfloor$
  for $j = 1, \ldots, J$ do
    $m_j \leftarrow \frac{1}{I'}\sum_{i=I'(j-1)+1}^{I'j} \widetilde{F}_i^T g_i(y)$
  return $\hat{e}(y) := \mathrm{median}(m_1, \ldots, m_J)$.
```

Intuitively, an LSQ family expresses the kernel as the expected inner product between vectors with a small number of entries (Q), bounded range (R), and bounded sparsity (S). The definition is reminiscent of random features, Fast Multipole Methods (Greengard & Rokhlin, 1987), and LSHability (Definition 3.3)—indeed, as we will see, it captures all of these. Its goal is to form an abstraction of the key properties that on the one hand "automatically" suffice for an efficient DP-KDE mechanism, and on the other hand are already shared by many prominent KDE methods.

2.1. LSQ Mechanism for DP-KDE

Let k be a kernel with an α-approximate (Q, R, S)-LSQ family $\mathcal{Q}$. The LSQ mechanism for DP-KDE is specified in Algorithm 1. It is parameterized by the privacy level ϵ, and by integers $I \geq J > 0$ that govern the efficiency/utility trade-off. We discuss their role and how to set them in more detail in Appendix E.1. The formal properties of the mechanism are stated next, with proofs deferred to Appendix A.

[1] Note that this is the dependence on α in both the sample complexity and the curator running time, since n is lower bounded by the sample complexity.

Lemma 2.2 (privacy)**.** *The mechanism is ϵ-DP.*

Lemma 2.3 (efficiency)**.** *Denote by $T_{\mathcal{Q}}$ the time to sample $(f,g)\sim\mathcal{Q}$, by T_f, T_g the time to compute $f(x), g(y)$ given $x,y\in\mathbb{R}^d$ respectively, and by $L_{\mathcal{Q}}$ the description size in machine words of a pair (f,g) sampled from $\mathcal{Q}$. Then,*

- *The curator runs in time $O(I(T_{\mathcal{Q}}+|X|T_f+Q))$.*
- *The curator output size is $O(I(L_{\mathcal{Q}}+Q))$.*
- *The client runs in time $O(I(T_g+S))$.*

For utility, we start with bounding the simpler case where $\mathcal{Q}$ contains just a single pair of functions.

Lemma 2.4 (single pair utility)**.** *Suppose $\mathcal{Q}$ is supported on a single pair (f,g), and the mechanism is run with $I=J=\Theta(\log(1/\eta))$. For every $y\in\mathbb{R}^d$, with probability $1-\eta$, the client output $\hat{e}(y)$ that satisfies*

$$|\hat{e}(y)-KDE_X(y)|\le\alpha+O\left(\frac{S^{1.5}R^2\log(\frac{1}{\eta})}{\epsilon|X|}\right).$$

The next utility bound is for large or infinite $\mathcal{Q}$.

Lemma 2.5 (utility)**.** *Suppose the mechanism is run with $J=\Theta(\log(1/\eta))$ and $I=\Theta(J/\alpha^2)$. For every $y\in\mathbb{R}^d$, with probability $1-\eta$, the client output $\hat{e}(y)$ that satisfies*

$$|\hat{e}(y)-KDE_X(y)|<\alpha+O\left(\alpha SR^2+\frac{S^{1.5}R^2\log(\frac{1}{\eta})}{\alpha\epsilon|X|}\right).$$

3. DP-KDE for LSQable Kernels

3.1. DP-KDE via Random Fourier Features (RFF)

We recall the construction of RFF for the Gaussian kernel. To sample a random feature, one draws $\omega\sim N(0,I_d)$ and β uniformly at random over $[0,2\pi)$, and defines the Fourier feature $z_{\omega,\beta}:\mathbb{R}^d\to\mathbb{R}$ as $z_{\omega,\beta}(x)=\sqrt{2}\cos(\sqrt{2}\omega^Tx+\beta)$. For every $x,y,\in\mathbb{R}^d$ it holds that

$$e^{-\|x-y\|_2^2}=\mathbb{E}_{\omega,\beta}[z_{\omega,\beta}(x)\cdot z_{\omega,\beta}(y)].$$

This clearly implies an LSQ family $\mathcal{Q}$, given by sampling ω and β as above and returning the pair $(z_{\omega,\beta},z_{\omega,\beta})$. Since $z_{\omega,\beta}$ takes values in $[-\sqrt{2},\sqrt{2}]$, we obtain,

Proposition 3.1. *The Gaussian kernel admits an exact $(1,\sqrt{2},1)$-LSQ family.*

This leads to our first Gaussian DP-KDE mechanism.

Proof of Theorem 1.1. Privacy is guaranteed by Lemma 2.2. For accuracy we use Lemma 2.5, plugging $S=1$, $R=\sqrt{2}$ and $|X|\ge O(\log(1/\eta)/(\epsilon\cdot\alpha^2))$ which holds by the theorem's premise. We get that for every $y\in\mathbb{R}^d$, the client outputs $\hat{e}(y)$ that with probability $1-\eta$ is off by an additive error of $O(\alpha)$ from the subsampled KDE, and we can scale α by the appropriate constant. For computational efficiency, note that sampling $(f,g)\sim\mathcal{Q}$ means sampling $\omega\sim N(0,I_d)$ and $\beta\sim[0,2\pi)$, and takes time $O(d)$; the pair (f,g) can be specified by the $d+1$ machine words ω,β; and evaluating f or g on a point in $\mathbb{R}^d$ takes $O(d)$ time. Plugging these with $I=O(\log(1/\eta)/\alpha^2)$ (from Lemma 2.5) into Lemma 2.3, we obtain Theorem 1.1. □

Other kernels. (Rahimi & Recht, 2007) showed that random Fourier features exist all shift-invariant positive definite kernels. For those kernels, the LSQ framework yields DP-KDE mechanisms with the same error and sample complexity guarantees as the Gaussian kernel in Theorem 1.1. However, their computational efficiency may be different, depending on their specific RFF distribution. See Appendix D.

3.2. DP-KDE via the Fast Gauss Transform (FGT)

We review the Fast Gauss Transform. Let all data and query points be contained in a ball $\mathcal{B}_\Phi$ of radius $\Phi>0$. Let $\mathcal{G}$ be the grid with side-length 1 in $\mathbb{R}^d$ whose nodes are $\mathbb{Z}^d$. Let $\mathcal{G}_\Phi$ denote the set of $\mathcal{G}$-grid cells that intersect $\mathcal{B}_\Phi$. For every cell $H\in\mathcal{G}_\Phi$, let $z^H\in\mathbb{R}^d$ denote its center point.

The FGT is based on the Hermite expansion of the Gaussian kernel. Let $\xi:\mathbb{R}\to\mathbb{R}$ be defined as $\xi(x)=e^{-x^2}$, and let $\xi^{(r)}$ denote the rth derivative of ξ for every $r\ge0$. The Hermite function $h_r:\mathbb{R}\to\mathbb{R}$ is defined as $h_r(x)=(-1)^r\xi^{(r)}(x)$. By substituting Taylor series, it can be shown (see Appendix B.1) that for any given $z\in\mathbb{R}^d$, the Gaussian kernel over points in $\mathbb{R}^d$ admits the Hermite expansion,

$$e^{-\|x-y\|_2^2}=\sum_{r_1=1}^{\infty}\cdots\sum_{r_d=1}^{\infty}\prod_{j=1}^{d}(x_j-z_j)^{r_j}\cdot\frac{1}{r_j!}h_{r_j}(y_j-z_j).$$

Truncating each of the d sums after $\rho=O(\log(1/\alpha))$ terms leads to an additive error of at most α. We can then define the following pair of functions f,g on $\mathbb{R}^d$. Each of their coordinates is indexed by a pair $H\in\mathcal{G}_\Phi$ and $r\in\mathbb{R}^d$, where r has coordinates in $\{0,\dots,\rho\}$, and is set as follows:

$$f_{H,r}(x)=\begin{cases}\prod_{j=1}^d\left(x_j-z_j^H\right)^{r_j} & \text{if } x\in H\\ 0 & \text{otherwise,}\end{cases}$$

$$g_{H,r}(y)=\begin{cases}\prod_{j=1}^d\frac{1}{r_j!}h_{r_j}\left(y_j-z_j^H\right) & \text{if } \|y-z^H\|_2^2\le\rho\\ 0 & \text{otherwise.}\end{cases}$$

For usual FGT, one may compute $F(X)=\frac{1}{|X|}\sum_{x\in X}f(x)$ on the dataset X, and return $F(X)^Tg(y)$ given a query point y. To our end, we view (f,g) as an LSQ "family" with just one pair. In Appendix B.1 we show the following.

Proposition 3.2. *Let $\alpha>0$ be smaller than a sufficiently small constant, and suppose $d=O(\log(1/\alpha))$.*

The Gaussian kernel over points contained in a Euclidean ball of radius Φ in $\mathbb{R}^d$ admits an α-approximate $(O((1+\frac{\Phi}{\sqrt{d}})(\log(1/\alpha)))^d, O(1)^d, (\log(1/\alpha))^{O(d)})$-LSQ family, supported on a single pair of functions (f,g). Furthermore, the evaluation times of f on $x \in \mathbb{R}^d$ and of g on $y \in \mathbb{R}^d$ are both $(\log(1/\alpha))^{O(d)}$.

This yields our second Gaussian DP-KDE mechanism.

Proof of Theorem 1.2. We may assume w.l.o.g. that $d = O(\log(1/\alpha))$, since otherwise Theorem 1.1 subsumes Theorem 1.2. Privacy follows from Lemma 2.2. For utility we use Lemma 2.4. By plugging R, S from Proposition 3.2, the additive error is, with probability $1-\eta$, at most $\alpha + (\epsilon|X|)^{-1}\log(1/\eta)\cdot(\log(1/\alpha))^{O(d)}$. By the lower bound on $|X|$ in the theorem statement, this error is at most $O(\alpha)$, and we can scale α by a constant. For efficiency, note that having only one pair in $\mathcal{Q}$ means that $T_{\mathcal{Q}} = O(1)$ and $L_{\mathcal{Q}} = 0$. Plugging these and Q, R, S, T_f, T_g from Proposition 3.2 into Lemma 2.3 yields the theorem. □

3.3. DP-KDE via Locality Sensitive Hashing (LSH)

In this section we observe that if a kernel is LSHable then it is also LSQable, thereby recovering the results of (Coleman & Shrivastava, 2021) for LSHable kernels (which do not include the Gaussian kernel) within the LSQ framework. We recall the relevant definition of kernel LSHability:

Definition 3.3. A kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$ is α-approximate LSHable if there is a distribution $\mathcal{H}$ over hash functions $h : \mathbb{R}^d \to \{0,1\}^*$, such that for every $x, y \in \mathbb{R}^d$,

$$\left| k(x,y) - \Pr_{h\sim\mathcal{H}}[h(x)=h(y)] \right| \le \alpha.$$

Suppose the hash functions in $\mathcal{H}$ map points in $\mathbb{R}^d$ to one of B hash buckets. For every $h \in \mathcal{H}$, let $f_h : \mathbb{R}^d \to \{0,1\}^B$ map x to the indicator vector of $h(x)$. To get an LSQ family $\mathcal{Q}$ from $\mathcal{H}$, we may sample $h \sim \mathcal{H}$ and return the pair (f_h, f_h). For all $x, y \in \mathbb{R}^d$ we thus get $f_h(x)^T f_h(y) = 1$ if $h(x) = h(y)$ and $f_h(x)^T f_h(y) = 0$ if $h(x) \ne h(y)$, hence $\mathbb{E}_{(f,g)\sim\mathcal{Q}}[f(x)^T g(y)] = \Pr_{h\sim\mathcal{H}}[h(x)=h(y)]$. Therefore,

Proposition 3.4. *If k is α-approximate LSHable with B hash buckets, then k is α-approximate $(B,1,1)$-LSQable.*

This does not immediately lead to efficient DP-KDE, since B can be very large. For example, all known LSHability results for the Laplacian kernel use $B = \exp(d)$ (Rahimi & Recht, 2007; Andoni & Indyk, 2009; Backurs et al., 2019). This issue does not typically interfere with non-private applications of LSH, due to sparsity (only one bucket is non-empty per point), but in the DP case, this would disclose information about which buckets are empty. Our LSQ mechanism adds noise to each bucket, which would take time proportional to B. Nonetheless, this can be remedied by standard universal hashing; see Appendix B.2.

Proposition 3.5. *If k is α-approximate LSHable, then k is 2α-approximate $(\lceil 1/\alpha \rceil, 1, 1)$-LSQable.*

Together with Lemmas 2.2 to 2.5, this recovers the DP-KDE results for LSHable kernels within the LSQ framework. As a concrete example, we re-derive a result of (Coleman & Shrivastava, 2021) for the Laplacian kernel.

Theorem 3.6. *There is an ϵ-DP function release mechanism for (α,η)-approximation of Laplacian KDE on datasets in $\mathbb{R}^d$ of size $n \ge O(\log(1/\eta)/(\epsilon\alpha^2))$, and:*

- *The curator runs in time $O(nd\log(1/\eta)/\alpha^2)$.*
- *The output size is $O(d\log(1/\eta)/\alpha^2)$.*
- *The client runs in time $O(d\log(1/\eta)/\alpha^2)$.*

Proof. The Laplacian kernel is LSHable, hence by Proposition 3.5, it is 2α-approximate $(\lceil 1/\alpha \rceil, 1, 1)$-LSQable. By Lemmas 2.2 and 2.5, this implies an ϵ-DP mechanism with (α,η)-approximation for Laplacian KDE. Furthermore, the Laplacian kernel LSH families from (Rahimi & Recht, 2007; Andoni & Indyk, 2009; Backurs et al., 2019) have $O(d)$ evaluation time, hashing time and description size. Viewed as LSQ families, they satisfy $T_{\mathcal{Q}}, T_f, T_g, L_{\mathcal{Q}} = O(d)$ in the notation of Lemma 2.3, which yields the theorem. □

The Laplacian kernel DP-KDE bounds in Theorem 3.6 are the same those of the Gaussian kernel in Theorem 1.1. We also remark that the Laplacian kernel admits an efficient RFF distribution, different than its LSH families. Thus, we can also instantiate the LSQ-RFF mechanism for it. This would lead to an alternative proof of Theorem 3.6, yielding the same asymptotic bounds via a different DP-KDE mechanism; see Appendix D.1.

See Appendix C.4 for an overview of other LSHable kernels.

4. Experiments

We evaluate our mechanisms on public benchmark datasets in both the high- and low-dimensional regimes. For compatibility, we select datasets often used in prior work on density estimation and clustering:

- Covertype: forest cover types (n = 581,012, d = 55) (Blackard & Dean, 1999)
- GloVe: word embeddings (n = 1,000,000, d = 100) (Pennington et al., 2014)
- Diabetes: age and days in hospital (n = 101,766, d = 2) (Strack et al., 2014)
- NYC Taxi: longitude and latitude (n = 100,000, d = 2) (Chavez et al., 2018)

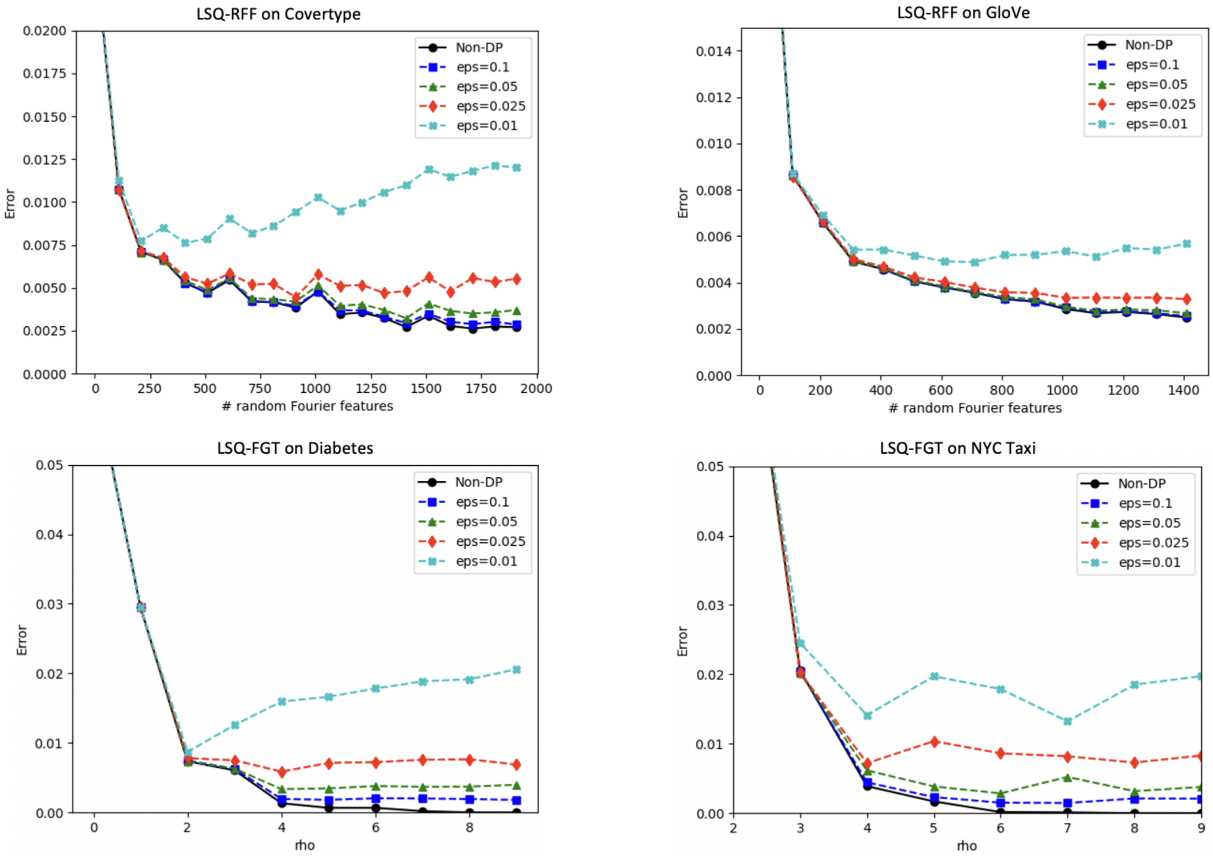


Figure 1. Error vs. computational budget

Query points are chosen at random from each dataset and are held out from it. The reported experimental results are averaged over 100 queries and 10 trials with independent random seeds. Our code is available online.[2] Appendix E includes additional details on the implementation of our mechanisms, additional experiments, and more details on our experimental framework and bandwidth selections.

4.1. Parameter Selection

In the first experiment, we measure the KDE approximation error of our mechanisms as we increase the parameter that governs their computational budget—the number of Fourier features in RFF, and ρ in FGT. Figure 1 displays the results for several values of ϵ, as well as for a non-private variant of each mechanism, that elides the Laplace noise addition step in Algorithm 1 (i.e., it sets $\widetilde{F}_i = F_i$).

The results highlight a key difference between the DP and non-DP variants: while the error of the non-DP variants converges to zero as we increase their computational budget, the error of the DP mechanisms begins to diverge at a certain point, which corresponds to a smaller parameter setting for smaller ϵ.[3] This behavior stems from the interplay between non-private approximation and privacy-preserving noise: as we increase the computational budget, the non-private approximation component of the mechanism becomes more accurate, thus disclosing more information about the dataset, that needs to be offset with a larger magnitude of privacy-preserving noise. The optimal parameter setting corresponds to the point of balance between the non-private approximation error and the privacy noise error.

For LSQ-RFF, as we increase the number of Fourier features m, the error of approximating the Gaussian kernel with (non-private) RFF decays like $1/\sqrt{m}$ by Hoeffding's inequality, while the Laplace noise magnitude grows like $\sqrt{m}/(\epsilon n)$. Hence, the optimal number of Fourier features is $m = \Theta(\epsilon n)$. Using more features would increase the overall error while having higher computational cost.

For LSQ-FGT, as we increase ρ, the non-private truncation error of the Hermite expansion decays like $\exp(-\rho)$, while the Laplace noise magnitude grows like $\rho^{O(d)}/(\epsilon n)$, hence the optimal setting is $\rho = \Theta(\log(\epsilon n)) - O(d \log\log(\epsilon n))$.

[2] https://github.com/talwagner/lsq

[3] Convergence to zero error is impossible for DP mechanisms due to the sample complexity limitation: for a given dataset size n and $\epsilon > 0$, the error α must be large enough to render $n \geq \mathrm{sc}(M)$.

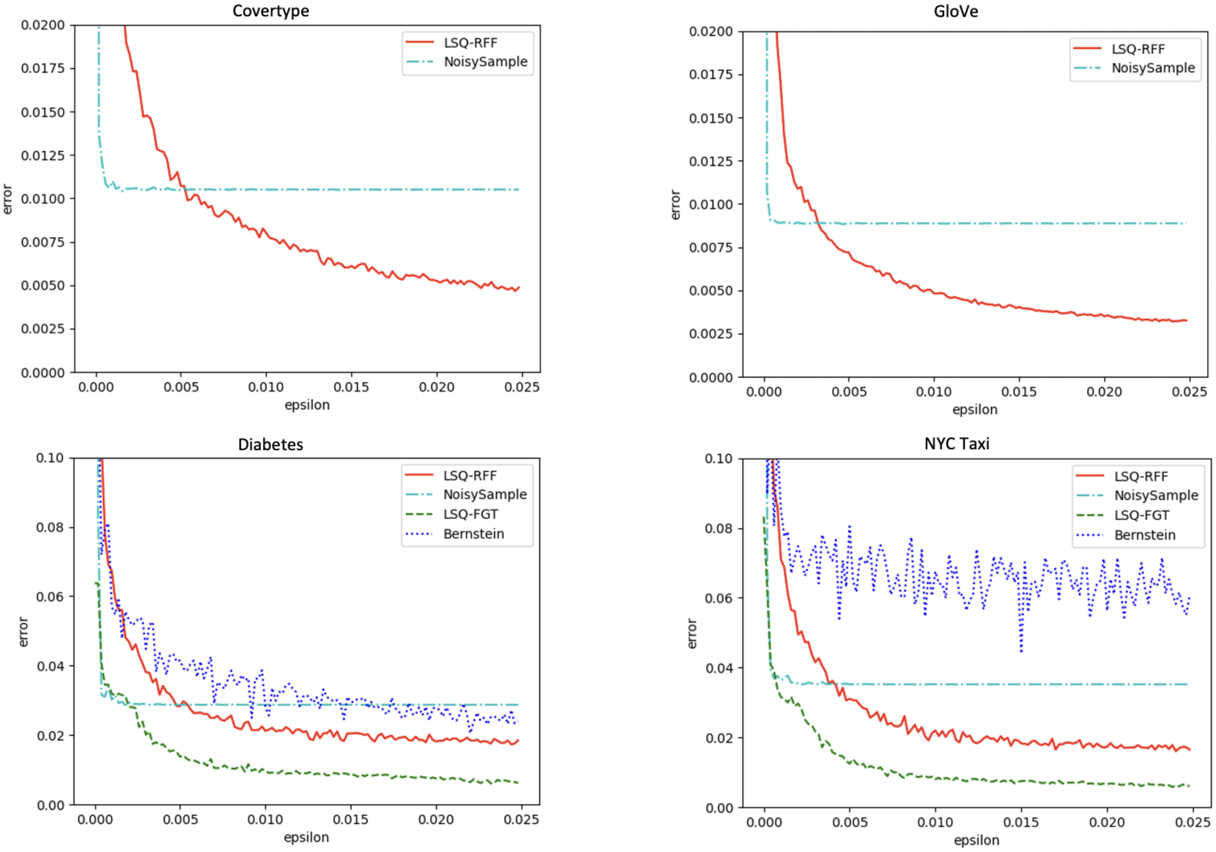


Figure 2. Error vs. privacy

The upshot is that the parameters should be chosen not only by the available computational budget, but also the desired privacy ϵ and available dataset size n.[4]

4.2. Performance

Error vs. privacy. We measure the privacy to error trade-off, with each algorithm evaluated at its optimal setting of parameters for the given value of ϵ. We compare our mechanisms to the following baselines:

- NoisySample: A vanilla mechanism that samples 100 points from the dataset, computes the average of their true KDEs plus a sample from $\text{Laplace}(1/(\epsilon|X|))$, and returns this value as the estimate for any query KDE. The mechanism is ϵ-DP w.r.t. the non-sampled points. It helps verify that the KDE function is not degenerate and essentially constant (and thus trivial to approximate).
- The Bernstein mechanism (Alda & Rubinstein, 2017), prior state of the art for Gaussian DP-KDE (with pure DP). It has the same error divergence behavior discussed in Section 4.1, and we evaluate it too at its optimal parameter setting (see Appendix E.2 for details on Bernstein).

The results are in Figure 2. LSQ-FGT and Bernstein are evaluated only on the on the low-dimensional datasets, as they are infeasible for the high-dimensional datasets.

The results show that our LSQ-based mechanisms outperform the baselines by large margins, and procude accurate KDE estimates in desirable privacy regimes. On the low-dimensional datasets, the results corroborate the privacy/error trade-offs predicted by the sample complexity of the mechanisms, as listed in Table 1. Note that LSQ-FGT is expected to outperform LSQ-RFF in this regime, due to the near-linear dependence of its sample complexity on α^{-1}, compared to the quadratic dependence of LSQ-RFF. On the high-dimensional datasets, only LSQ-RFF is feasible.

The performance of Bernstein depends on the smoothness of the KDE function, which is determined by the bandwidth σ under scaling the data into a unit hypercube (cf. Section 1.4).

[4] Note that setting the parameters of the mechanism according to the dataset size n—e.g., choosing $m \sim \epsilon n$ or $\rho \sim \log(\epsilon n)$—leaks information about n and could affect privacy. This can be easily avoided, for example by using $\tilde{n} = n + \text{Laplace}(1/\epsilon)$ instead of n. It can be easily checked that $\tilde{n}$ is ϵ-DP and that using it instead of n would only change m or ρ by an additive constant.

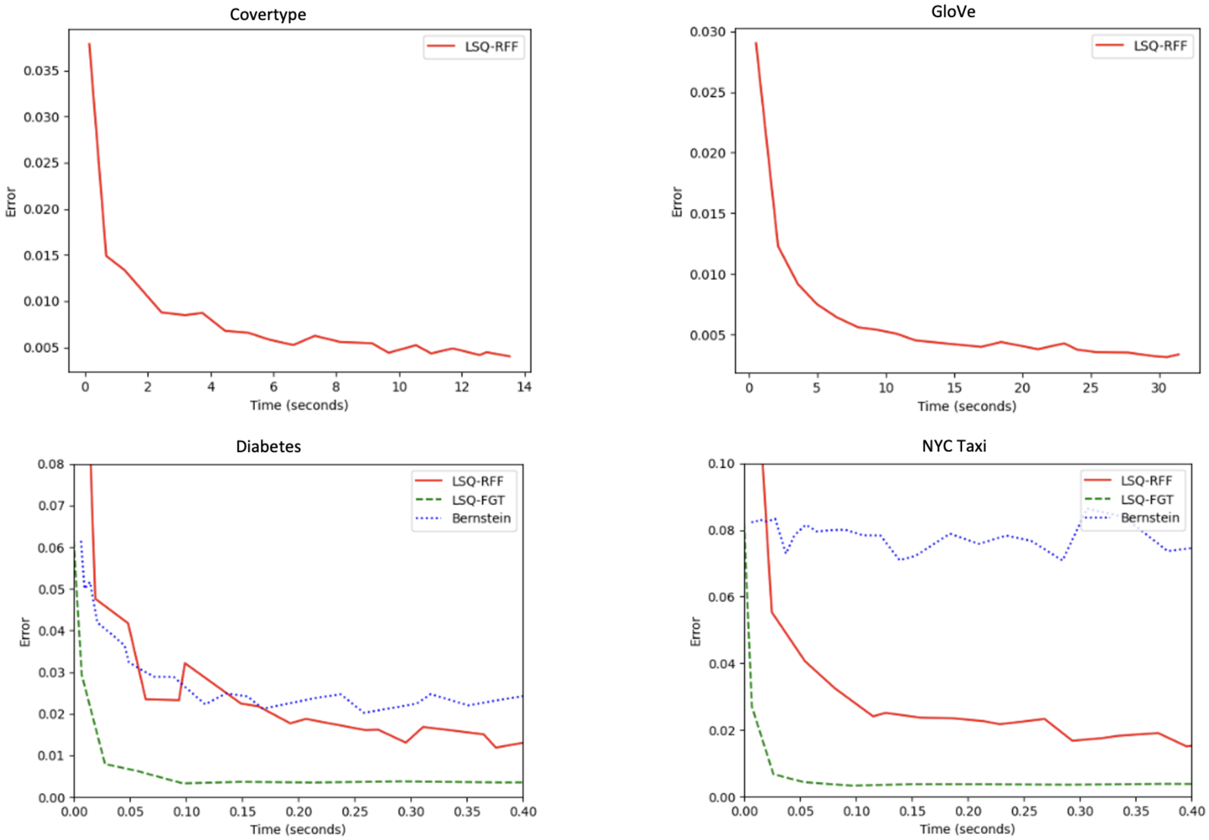


Figure 3. Error vs. curator running times with $\epsilon = 0.05$

In particular, its sample complexity depends on d/σ^2. For NYC Taxi, this quantity is much larger than for Diabetes (cf. Appendix E), accounting for the degraded performance of Bernstein on NYC Taxi compared to Diabetes.

Running times. We plot the error attained by the mechanisms versus their curator running time. Figure 3 reports the results with $\epsilon = 0.05$, and Figure 5 (in the appendix) repeats the experiment with $\epsilon = 0.02$. Here too, in the high-dimensional regime LSQ-RFF is the only feasible mechanism, while in the low-dimensional regime LSQ-FGT has the best performance.

Acknowledgements

We thank James Cook, Christos Faloutsos, Hakan Ferhatosmanoglu, Supriya Nagesh, Matthew Reimherr, Aaron Roth, Milind Shyani, Doug Terry, Elizabeth Yang, and the anonymous reviewers for helpful feedback, comments and suggestions on this work.

References

Alda, F. and Rubinstein, B. I. The bernstein mechanism: Function release under differential privacy. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Alman, J., Chu, T., Schild, A., and Song, Z. Algorithms and hardness for linear algebra on geometric graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 541–552. IEEE, 2020.

Andoni, A. and Indyk, P. Dimension reduction in kernel spaces from locality-sensitive hashing. *Maniscript, also available in Andoni A.,"Nearest neighbor search: the old, the new, and the impossible", PhD thesis, Massachusetts Institute of Technology*, 2009.

Aumüller, M., Lebeda, C. J., and Pagh, R. Differentially private sparse vectors with low error, optimal space, and fast access. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1223–1236, 2021.

Backurs, A., Charikar, M., Indyk, P., and Siminelakis, P. Efficient density evaluation for smooth kernels. In *2018*

IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS), pp. 615–626. IEEE, 2018.

Backurs, A., Indyk, P., and Wagner, T. Space and time efficient kernel density estimation in high dimensions. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Backurs, A., Indyk, P., Musco, C., and Wagner, T. Faster kernel matrix algebra via density estimation. In *International Conference on Machine Learning (ICML)*, 2021.

Blackard, J. A. and Dean, D. J. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3): 131–151, 1999. URL https://archive.ics.uci.edu/ml/datasets/covertype.

Blocki, J., Blum, A., Datta, A., and Sheffet, O. The johnson-lindenstrauss transform itself preserves differential privacy. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 410–419. IEEE, 2012.

Blum, A., Ligett, K., and Roth, A. A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM)*, 60(2):1–25, 2013.

Carter, J. L. and Wegman, M. N. Universal classes of hash functions. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pp. 106–112, 1977.

Charikar, M. and Siminelakis, P. Hashing-based-estimators for kernel density in high dimensions. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 1032–1043. IEEE, 2017.

Chavez, A., Sterling, D., Elliott, J., V, L., Sagar, and Cukierski, W. New york city taxi fare prediction, 2018. URL https://kaggle.com/competitions/new-york-city-taxi-fare-prediction.

Cherapanamjeri, Y. and Nelson, J. On adaptive distance estimation. *Advances in Neural Information Processing Systems*, 33:11178–11190, 2020.

Chierichetti, F. and Kumar, R. Lsh-preserving functions and their applications. *Journal of the ACM (JACM)*, 62(5): 1–25, 2015.

Coleman, B. and Shrivastava, A. Sub-linear race sketches for approximate kernel density estimation on streaming data. In *Proceedings of The Web Conference 2020*, pp. 1739–1749, 2020.

Coleman, B. and Shrivastava, A. A one-pass distributed and private sketch for kernel sums with applications to machine learning at scale. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 3252–3265, 2021.

Cunningham, T., Cormode, G., and Ferhatosmanoglu, H. Privacy-preserving synthetic location data in the real world. In *17th International Symposium on Spatial and Temporal Databases*, pp. 23–33, 2021.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference (TCC)*, pp. 265–284. Springer, 2006.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Feldman, V. and Talwar, K. Lossless compression of efficient private local randomizers. In *International Conference on Machine Learning (ICML)*, 2021.

Greengard, L. and Rokhlin, V. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2): 325–348, 1987.

Greengard, L. and Strain, J. The fast gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1): 79–94, 1991.

Gupta, A., Roth, A., and Ullman, J. Iterative constructions and private data release. In *Theory of cryptography conference*, pp. 339–356. Springer, 2012.

Hall, R. *New Statistical Applications for Differential Privacy*. PhD thesis, Carnegie Mellon University, 2013.

Hall, R., Rinaldo, A., and Wasserman, L. Differential privacy for functions and functional data. *Journal of Machine Learning Research*, 14(Feb):703–727, 2013.

Hardt, M. and Rothblum, G. N. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st annual symposium on foundations of computer science*, pp. 61–70. IEEE, 2010.

Hofmann, T., Schölkopf, B., and Smola, A. J. Kernel methods in machine learning. *The annals of statistics*, 36(3): 1171–1220, 2008.

Huai, M., 0015, D. W., Miao, C., Xu, J., and Zhang, A. Privacy-aware synthesizing for crowdsourced data. In *IJCAI*, pp. 2542–2548, 2019.

Indyk, P. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

Indyk, P. and Motwani, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC)*, 1998.

Jaakkola, T. S., Diekhans, M., Haussler, D., et al. Using the fisher kernel method to detect remote protein homologies. In *ISMB*, volume 99, pp. 149–158, 1999.

Johnson, W. B. and Schechtman, G. Embedding l _pˆm into l _1ˆn. 1982.

Karnin, Z. and Liberty, E. Discrepancy, coresets, and sketches in machine learning. In *Conference on Learning Theory*, pp. 1975–1993. PMLR, 2019.

Lacoste-Julien, S., Lindsten, F., and Bach, F. Sequential kernel herding: Frank-wolfe optimization for particle filtering. In *Artificial Intelligence and Statistics*, pp. 544–552. PMLR, 2015.

Lopez-Paz, D., Muandet, K., Schölkopf, B., and Tolstikhin, I. Towards a learning theory of cause-effect inference. In *International Conference on Machine Learning*, pp. 1452–1461. PMLR, 2015.

Nikolov, A. Private query release via the johnson-lindenstrauss transform. *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2023.

Pagh, R. and Thorup, M. Improved utility analysis of private countsketch. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014. URL https://nlp.stanford.edu/projects/glove/.

Phillips, J. M. and Tai, W. M. Near-optimal coresets of kernel density estimates. *Discrete & Computational Geometry*, 63(4):867–887, 2020.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Shawe-Taylor, J., Cristianini, N., et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

Siminelakis, P., Rong, K., Bailis, P., Charikar, M., and Levis, P. Rehashing kernel evaluation in high dimensions. In *International Conference on Machine Learning*, pp. 5789–5798. PMLR, 2019.

Strack, B., DeShazo, J. P., Gennings, C., Olmo, J. L., Ventura, S., Cios, K. J., and Clore, J. N. Impact of hba1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records. *BioMed research international*, 2014, 2014. URL https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008.

Wang, Z., Jin, C., Fan, K., Zhang, J., Huang, J., Zhong, Y., and Wang, L. Differentially private data releasing for smooth queries. *The Journal of Machine Learning Research*, 17(1):1779–1820, 2016.

A. Analysis of the LSQ Mechanism

Proof of Lemma 2.2. The (Q, R, S)-LSQ property is easily seen to imply that the sensitivity of each F_i in Algorithm 1 is $SR/|X|$, and we have I of them, thus the lemma follows from the classical Laplace DP mechanism.

We expand on the details for completeness. To this end we recall some DP fundamentals. Let F be a function that maps a dataset to $\mathbb{R}^m$. The ℓ_1-*sensitivity* of F is defined as $\Delta F = \max_{X,X'} \|F(X) - F(X')\|_1$, where the maximum is taken over all pairs of neighboring datasets X, X' (the definition of neigboring datasets is given in Section 1.3). Given a function F, a dataset X, and $\epsilon > 0$, the Laplace mechanism (Dwork et al., 2006) releases $F(X) + N$, where $N \in \mathbb{R}^m$ has entries sampled i.i.d. from $\mathrm{Laplace}(\Delta F/\epsilon)$. This mechanism is ϵ-DP (Dwork et al., 2006).

In Algorithm 1 we have $F_i = \frac{1}{|X|}\sum_{x\in X} f_i(x)$, where f_i is sampled from a (Q, R, S)-LSQ family, and thus every $f_i(x)$ has at most S non-zero entries, each of contained in $[-R, R]$. Therefore, F_i has ℓ_1-sensitivity $RS/|X|$, and the sequence $(F_1, \ldots, F_I)$ has ℓ_1-sensitivity $IRS/|X|$. The curator in Algorithm 1 releases $(\widetilde{F}_1, \ldots, \widetilde{F}_I)$, which we observe is but the output of the Laplace mechanism on this function, and is thus ϵ-DP. The curator also releases $(f_i, g_i)_{i=1}^{I}$, which are sampled obliviously to the dataset and have no effect on differential privacy. □

Proof of Lemma 2.3. The lemma follows by tracking the steps of the curator and client algorithms.

Curator running time: In each of $i = 1, \ldots, I$ iterations, it samples $(f_i, g_i) \sim \mathcal{Q}$ in time $T_{\mathcal{Q}}$, evaluates f_i on every $x \in X$ in total time $|X|T_f$, and adds Laplace noise to each of Q coordinates in total time $O(Q)$.

Curator output size: For every $i = 1, \ldots, I$, it outputs the pair (f_i, g_i) which is described using $L_{\mathcal{Q}}$ machine words, and the Q-dimensional vector $\widetilde{F}_i$ which occupies Q machine words.

Client running time: For every $i = 1, \ldots, I$, it evaluates $g_i(y)$ in time T_g, and computes the inner product $\widetilde{F}_i^T g_i(y)$, which can be done in time $O(S)$ since $g_i(y)$ has at most S non-zero entries. This takes total time $O(I(T_g + S))$. It then returns the median of J values, each one of whom is the mean of I' values, which takes additional time $O(I'J) = O(I)$. □

Proof of Lemma 2.4. Let $j \in [J]$. By plugging $I' = 1$ (since in this lemma we have $I = J$) and $(f_i, g_i) = (f, g)$ (since we have a single pair (f, g)) into the definition of m_j in the client algorithm, we have

$$m_j = \widetilde{F}_j^T g(y) = \frac{1}{|X|}\sum_{x\in X} f(x)^T g(y) + N_j^T g(y), \tag{1}$$

where $N_j = (N_j^{(1)}, \ldots, N_j^{(Q)})$ is a random vector whose entries are drawn i.i.d. from $\mathrm{Laplace}(IRS/(\epsilon|X|))$. By properties of the Laplace distribution, each entry $N_j^{(q)}$ has variance $\mathbf{Var}(N_j^{(q)}) = 2(IRS/(\epsilon|X|))^2$. Since $\mathbf{Var}(N_j^T g_j(y)) = \sum_{q=1}^{Q} g_j(y)^2 \mathbf{Var}(N_j^{(q)})$, and $g_j(y)$ has at most S non-zero entries contained in $[-R, R]$, we have $\mathbf{Var}(N_j^T g_j(y)) \leq SR^2 \cdot 2(IRS/(\epsilon|X|))^2$. Thus by Chebyshev's inequality (recalling that $I = \Theta(\log(1/\eta))$),

$$\Pr\left[\left|N_j^T g_j(y)\right| > \frac{O(1)\cdot S^{1.5}R^2\log(1/\eta)}{\epsilon|X|}\right] < \frac{1}{6}.$$

Now by the Chernoff inequality, the median of $J = \Theta(\log(1/\eta))$ independent copies $N_1^T g(y), \ldots, N_J^T g(y)$ satisfies

$$\Pr\left[\left|\mathrm{median}(N_1^T g(y), \ldots, N_J^T g(y))\right| \geq \frac{O(1)\cdot S^{1.5}R^2\log(1/\eta)}{\epsilon|X|}\right] < \eta. \tag{2}$$

The client output in Algorithm 1 equals $\hat{e}(y) = \mathrm{median}(m_1, \ldots, m_J)$. By noting in Equation (1) that the term $\frac{1}{|X|}\sum_{x\in X} f(x)^T g(y)$ does not depend on j, we have

$$\hat{e}(y) = \mathrm{median}(m_1, \ldots, m_J) = \frac{1}{|X|}\sum_{x\in X} f(x)^T g(y) + \mathrm{median}(N_1^T g(y), \ldots, N_J^T g(y)).$$

The α-approximate LSQ property, in the case of $\mathcal{Q}$ supported on a single pair, guarantees that $|k(x, y) - f(x)^T g(y)| \leq \alpha$.

Thus,

$$
\begin{aligned}
|\hat{e}(y) - KDE_X(y)| &= \left| \frac{1}{|X|} \sum_{x \in X} f(x)^T g(y) + \text{median}(N_1^T g(y), \ldots, N_J^T g(y)) - KDE_X(y) \right| \\
&\leq \left| \frac{1}{|X|} \sum_{x \in X} f(x)^T g(y) - KDE_X(y) \right| + |\text{median}(N_1^T g(y), \ldots, N_J^T g(y))| \\
&\leq \frac{1}{|X|} \sum_{x \in X} \left| f(x)^T g(y) - k(x,y) \right| + |\text{median}(N_1^T g(y), \ldots, N_J^T g(y))| \\
&\leq \alpha + |\text{median}(N_1^T g(y), \ldots, N_J^T g(y))| \\
&\leq \alpha + \frac{O(1) \cdot S^{1.5} R^2 \log(1/\eta)}{\epsilon |X|},
\end{aligned}
$$

where the final inequality holds with probability $1 - \eta$ by Equation (2), as was to be shown. □

Proof of Lemma 2.5. For every $i = 1, \ldots, I$ we have

$$\widetilde{F}_i^T g_i(y) = F_i^T g_i(y) + N_i^T g_i(y),$$

where N_i is a random vector whose entries are drawn i.i.d. from Laplace$(IRS/(\epsilon|X|))$. Therefore, for every $j = 1, \ldots, J$,

$$m_j = \frac{1}{I'} \sum_{i=I'(j-1)+1}^{I'j} \widetilde{F}_i^T g_i(y) = \frac{1}{I'} \sum_{i=I'(j-1)+1}^{I'j} F_i(x)^T g_i(y) + \frac{1}{I'} \sum_{i=I'(j-1)+1}^{I'j} N_i^T g_i(y), \tag{3}$$

where we recall that $I' = \lfloor I/J \rfloor = \Theta(1/\alpha^2)$. We handle the two sums in turn.

For the first sum, consider a random choice of $(f_i, g_i) \sim \mathcal{Q}$, and recall that $F_i(x) = \frac{1}{|X|} \sum_{x \in X} f_i(x)$. By the (Q, R, S)-LSQ property of $\mathcal{Q}$, for every x, y it holds that both $f_i(x)$ and $g_i(y)$ have at most S non-zero entries of magnitude at most R, hence $|f_i(x)^T g_j(y)| \leq SR^2$. Therefore,

$$\left| F_i(x)^T g_i(y) \right| \leq \frac{1}{|X|} \sum_{x \in X} \left| f_i(x)^T g_i(y) \right| \leq SR^2.$$

This holds for every supported pair (f_i, g_i). Consequently, Hoeffding's concentration inequality ensures that averaging $I' = \Theta(1/\alpha^2)$ independent copies of $F_i(x)^T g_i(y)$ yields

$$\Pr\left[\left| \frac{1}{I'} \sum_{i=I'(j-1)+1}^{I'j} F_i(x)^T g_i(y) - \mathbb{E}\left[F_i(x)^T g_i(y) \right] \right| > O(1) \cdot \alpha S R^2 \right] < \frac{1}{6}.$$

Moreover, the expectation $\mathbb{E}\left[F_i(x)^T g_i(y) \right]$ satisfies

$$
\begin{aligned}
\left| \mathbb{E}\left[F_i(x)^T g_i(y) \right] - KDE_X(y) \right| &= \left| \mathbb{E}\left[\frac{1}{|X|} \sum_{x \in X} f_i(x)^T g_i(y) \right] - \frac{1}{|X|} \sum_{x \in X} k(x,y) \right| \\
&\leq \frac{1}{|X|} \sum_{x \in X} \left| \mathbb{E}\left[f_i(x)^T g_i(y) \right] - k(x,y) \right| \\
&\leq \alpha,
\end{aligned}
$$

where the final inequality is an application of the α-approximate LSQ property of $\mathcal{Q}$, i.e., $\left| \mathbb{E}\left[f_i(x)^T g_i(y) \right] - k(x,y) \right| \leq \alpha$. Combining these, we get

$$\Pr\left[\left| \frac{1}{I'} \sum_{i=I'(j-1)+1}^{I'j} F_i(x)^T g_i(y) - KDE_X(y) \right| > \alpha + O(1) \cdot \alpha S R^2 \right] < \frac{1}{6}. \tag{4}$$

For the second sum in Equation (3), recall that in the above proof of Lemma 2.4 it was shown that $\mathbf{Var}(N_i^T g_i(y)) \leq SR^2 \cdot 2(IRS/(\epsilon|X|))^2$ for every i. Averaging over $I' = \Theta(1/\alpha^2)$ independent copies scales the variance down by $1/|I'|$, ensuring it is at most $O(1) \cdot \alpha^2 SR^2 \cdot (IRS/(\epsilon|X|))^2$. Plugging $I = \Theta(\log(1/\eta)/\alpha^2)$, we have by Chebyshev's inequality,

$$\Pr\left[\left|\frac{1}{I'}\sum_{i=I'(j-1)+1}^{I'j} N_i^T g_i(y)\right| > \frac{O(1)\cdot S^{1.5}R^2\log(1/\eta)}{\alpha\epsilon|X|}\right] < \frac{1}{6}. \tag{5}$$

Taking a union bound over Equations (4) and (5) and plugging both into Equation (3), we get

$$\Pr\left[|m_j - KDE_X(y)| > \alpha + O(1)\cdot\left(\alpha SR^2 + \frac{S^{1.5}R^2\log(1/\eta)}{\alpha\epsilon|X|}\right)\right] < \frac{1}{3}.$$

The client output is $\hat{e}(y) = \mathrm{median}(m_1, \ldots, m_J)$. Since $J = \Theta(\log(1/\eta))$, we get by Chernoff's inequality,

$$\Pr\left[|\hat{e}(y) - KDE_X(y)| < \alpha + O(1)\cdot\left(\alpha SR^2 + \frac{S^{1.5}R^2\log(1/\eta)}{\alpha\epsilon|X|}\right)\right] \geq 1 - \eta,$$

as desired. □

B. Additional Omitted Analysis

B.1. Fast Gauss Transform (Section 3.2)

For context, we start by deriving the Hermite expansion of the Gaussian kernel. Let $x, y, z \in \mathbb{R}^d$. We may write,

$$\begin{aligned}
e^{-\|y-x\|_2^2} &= \prod_{j=1}^{d} e^{-(y_j-x_j)^2} \\
&= \prod_{j=1}^{d} \xi(y_j - x_j) \\
&= \prod_{j=1}^{d}\left(\sum_{r_j=1}^{\infty}\frac{(z_j-x_j)^{r_j}}{r_j!}\cdot\xi^{(r_j)}(y_j-z_j)\right) \\
&= \prod_{j=1}^{d}\left(\sum_{r_j=1}^{\infty}\frac{(x_j-z_j)^{r_j}}{r_j!}\cdot h_{r_j}(y_j-z_j)\right) \\
&= \sum_{r_1=1}^{\infty}\cdots\sum_{r_d=1}^{\infty}\prod_{j=1}^{d}\frac{(x_j-z_j)^{r_j}}{r_j!}\cdot h_{r_j}(y_j-z_j),
\end{aligned}$$

where we recall from Section 3.2 that ξ denotes the univariate function $\xi(\gamma) = e^{-\gamma^2}$, that $\xi^{(r)}$ denotes its rth derivative, and that h_r denotes the Hermite function of order r. With this notation, the third equality above is by replacing each $\xi(y_j - x_j)$ with its Taylor expansion about $y_j - z_j$. The fourth equality is by recalling that $h_r = (-1)^r\xi^{(r)}$. The fifth equality is by rewriting the product of sums as the sum of products.

(Greengard & Strain, 1991) show that truncating each of the d sums after $\rho = O(\log(1/\alpha))$ terms leads to an additive error of at most α. Thus,

$$\forall z \in \mathbb{R}^d, \qquad \left|e^{-\frac{1}{2}\|x-y\|_2^2} - \sum_{r_1=1}^{\rho}\cdots\sum_{r_d=1}^{\rho}\prod_{j=1}^{d}(x_j-z_j)^{r_j}\cdot\frac{1}{r_j!}\cdot h_{r_j}(y_j-z_j)\right| \leq \alpha. \tag{6}$$

We now prove Proposition 3.2 with the pair of functions f, g as defined in Section 3.2.

Proof of Proposition 3.2. The LSQ family is supported on the single pair of functions (f, g). Note that by the premise $d = O(\log(1/\alpha))$, we may choose ρ that satisfies $\rho \geq d$.

We start by showing the α-approximate LSQ property, which here means that $|f(x)^T g(y) - e^{-\|x-y\|_2^2}| \leq \alpha$ for every $x, y \in \mathbb{R}^d$. Let $x, y \in \mathbb{R}^d$. Let H_x be the grid cell that contains x. Recall that $z^{H_x} \in \mathbb{R}^d$ denotes its center point. Note that $f(x)$ is non-zero only in those entries $f_{H,r}(x)$ for which $H = H^x$. We consider two cases:

- If $\|y - z^{H_x}\|_2 \leq \sqrt{\rho}$, then by the definition of f and g we have

  $$f(x)^T g(y) = \sum_{r_1=1}^{\rho} \cdots \sum_{r_d=1}^{\rho} \prod_{j=1}^{d} (x_j - z_j^{H_x})^{r_j} \cdot \frac{1}{r_j!} \cdot h_{r_j}(y_j - z_j^{H_x}),$$

  hence by Equation (6), $|f(x)^T g(y) - e^{-\|x-y\|_2^2}| \leq \alpha$.

- If $\|y - z^{H_x}\|_2 > \sqrt{\rho}$, then $f(x)^T g(y) = 0$, since there are no entries where both $f(x)$ and $g(y)$ are non-zero. Thus, in this case it suffices to show that $e^{-\|x-y\|_2^2} \leq \alpha$. Recall that H_x is a hypercube with side-length 1 centered at z^{H_x} and contains x, hence $\|x - z^{H_x}\| \leq \frac{1}{2}\sqrt{d} \leq \frac{1}{2}\sqrt{\rho}$. Therefore, by the triangle inequality,

  $$\|x - y\|_2^2 \geq (\|y - z^{H_x}\|_2 - \|x - z^{H_x}\|_2)^2 \geq (\sqrt{\rho} - \tfrac{1}{2}\sqrt{\rho})^2 = \tfrac{1}{4}\rho,$$

  and thus $e^{-\|x-y\|_2^2} \leq e^{-\rho/4} \leq \alpha$, which holds provided we choose $\rho = O(\log(1/\alpha))$ with an appropriate hidden constant.

In both cases we have $|f(x)^T g(y) - e^{-\|x-y\|_2^2}| \leq \alpha$, so α-approximate LSQability holds.

Next, we bound the parameters (Q, R, S) of this LSQ pair.

- Quantization Q: Each coordinate is indexed by a pair $H \in \mathcal{G}_\Phi$ and $r \in \{0, \ldots, \rho\}^d$. We recall that $\mathcal{G}_\Phi$ is the set of grid cells with side-length 1 that intersect a ball of radius Φ, hence $|\mathcal{G}_\Phi| = O(1 + \frac{\Phi}{\sqrt{d}})^d$ by a standard volume argument. The number of choices for r is $(\rho + 1)^d$, thus $Q = O((1 + \frac{\Phi}{\sqrt{d}}) \cdot \rho)^d$.

- Range R: Observe that $f(x)$ is zero in all coordinates $(H, r_1, \ldots, r_d)$ except those where H is the (unique) grid cell H_x that contains x. Since H_x has side-length 1 and its center point is z^{H_x}, we have $\forall_j |x_j - z_j^{H_x}| \leq 0.5$, and therefore the magnitude of $f(x)$ at each non-zero coordinate can be bounded as $\left|\prod_{j=1}^{d} \left(x_j - z_j^{H_x}\right)^{r_j}\right| \leq 1$.

  For $g(y)$, we use the following bound from (Greengard & Strain, 1991), which is a consequence of Cramer's inequality for Hermite functions: for every $r_1, \ldots, r_d$ and $y \in \mathbb{R}^d$,

  $$\left|\prod_{j=1}^{d} \frac{1}{r_j!} \cdot h_{r_j}(y_j)\right| \leq e^{-\|y\|_2^2} \prod_{j=1}^{d} \frac{1.09 \cdot (\sqrt{2})^{r_j}}{\sqrt{r_j!}}.$$

  It is not hard to verify that the term $1.09 \cdot (\sqrt{2})^r / \sqrt{r!}$ is maximized over non-negative integers r at $r = 1$ and is bounded by $1.09 \cdot \sqrt{2} < 1.6$, hence the right-hand size is upper bounded by 1.6^d.

- Sparsity S: Again, $f(x)$ is non-zero only at coordinates $f_{H,r}(x)$ such that $H = H_x$, of which there are only $(\rho + 1)^d$ (the number of choices for $r \in \{0, \ldots, \rho\}^d$).

  As for $g(y)$, it is non-zero only in coordinates $g_{H,r}(x)$ where H is one of the grid cells of $\mathcal{G}$ that satisfies $\|y - z^H\|_2 \leq \sqrt{\rho}$. Since the grid has side-length 1, the number of cells at distance at most $\sqrt{\rho}$ from any given point y is at most $O(1 + \sqrt{\rho/d})^d$, again by a standard volume argument. Accounting also for the $(\rho + 1)^d$ possible choices for r, the number of non-zero coordinates $g_{H,r}(x)$ is at most $O(1 + \sqrt{\rho/d})^d \cdot (\rho + 1)^d \leq \rho^{O(d)}$.

Finally, we bound the evaluation times T_f and T_g of f and g respectively.

- T_f: Every non-zero entry of $f(x)$ is the product of d terms, which takes $O(d)$ time to compute. As shown above, $f(x)$ has $(\rho+1)^d$ non-zero entries, thus its total evaluation time of is thus $O(d)\cdot(\rho+1)^d$.
- T_g: Let $i\geq 0$ be an integer. The hermite function $h_i(\gamma)$ is equal to $e^{-\gamma^2}P_i(\gamma)$ for every $\gamma\in\mathbb{R}$, where P_i is the ("physicist's") Hermite polynomial of degree i. Fix a grid cell $H\in\mathcal{G}_\Phi$. Since $P_i(\gamma)$ and thus $h_i(\gamma)$ can be evaluated in time $O(i)$ for every i and γ, all values $\{h_i(y_j-z_j^H):i=0,\dots,\rho\}$ can be computed in time $O(\rho^2)$. With these at hand, for every $r\in\{0,\dots,\rho\}^d$ and our fixed H we can compute $g_{H,r}(y)$ in time $O(d)$, by multiplying the appropriate pre-computed values. The total evaluation time for a fixed H is thus $O(\rho^2+d)$. As shown above, the number of cells H whose corresponding entries in $g(y)$ are non-zero is $O(1+\sqrt{\rho/d})^d$, leading to a total computation time of $O(1+\sqrt{\rho/d})^d\cdot(\rho^2+d)\leq\rho^{O(d)}$.

Recalling that $d\leq\rho=O(\log(1/\alpha))$, the proof is complete. □

Refined LSQ for sharper implementation. In Section 2.1, for the purpose of asymptotic analysis, we defined LSQ with a uniform bound R on the range of all coordinates in f and g. Nonetheless, the coordinates can have different ranges, as the above proof shows for FGT. While it does not change the asymptotic bounds, it can have practical importance in implementation.

Concretely, let $f,g:\mathbb{R}^d\to\mathbb{R}^Q$. Suppose we have $S,R^g,R_1^f,\dots,R_Q^f\geq 0$ such that for every $x,y\in\mathbb{R}^d$:

- $g(y)$ has at most S non-zero coordinates;
- Each coordinate of $g(y)$ is in $[-R^g,R^g]$;
- For $i=1,\dots,Q$, coordinate i of $f(x)$ is in $[-R_i^f,R_i^f]$.

The LSQ mechanism in Algorithm 1 adds a sample from $\text{Laplace}((\epsilon|X|)^{-1}IRS)$ to each coordinate, to ensure ϵ-DP via the Laplace mechanism. In the refined form of LSQ stated above, since f has sensitivity $\sum_{i=1}^Q R_i^f$, it suffices to add a sample from $\text{Laplace}((\epsilon|X|)^{-1}I\sum_{i=1}^Q R_i^f)$ to each coordinate to ensure ϵ-DP.

In the case of FGT, the above proof of Proposition 3.2 shows that if a coordinate of $f_{H,r}$ is indexed by a pair $H\in\mathcal{G}_\Phi$ and $r\in\{0,\dots,\rho\}^d$, then $f_{H,r}(x)=0$ if $x\notin H$, and otherwise,

$$|f_{H,r}(x)|\leq\left|\prod_{j=1}^d(x_j-z_j^H)^{r_j}\right|\leq\prod_{j=1}^d\frac{1}{2^{r_j}}=\frac{1}{2^{\sum_{j=1}^d r_j}}.$$

Therefore,

$$\sum_{i=1}^Q R_i^f=\sum_{r_1=0}^{\rho}\cdots\sum_{r_d=0}^{\rho}\frac{1}{2^{\sum_{j=1}^d r_j}}=\left(\sum_{r=0}^{\rho}\frac{1}{2^r}\right)^d=\left(2\left(1-2^{-(\rho-1)}\right)\right)^d.$$

Asymptotically, this makes no difference to the analysis: by retracing the proof of Lemma 2.4 with this refined LSQ, we get that the error term $O((\epsilon|X|)^{-1}\log(1/\eta)\cdot S^{1.5}R^2)$ from Lemma 2.4 becomes $O((\epsilon|X|)^{-1}\log(1/\eta)\cdot\sqrt{S}\cdot R^g\cdot\sum_{i=1}^Q R_i^f)$. Since $R^g=1.6^d$ and $S=\rho^{O(d)}$ in Proposition 3.2, the resulting error is the same in both cases up to hidden constants. However, in practice, adding noise of magnitude only $\left(2\left(1-2^{-(\rho-1)}\right)\right)^d$ instead of $\rho^{O(d)}$ to each coordinate noticeably improves the empirical performance of FGT, while retaining its theoretical guarantees.

B.2. Locality Sensitive Hashing (Section 3.3)

Proof of Proposition 3.5. The proof is by composing a usual pairwise independent hash function over the LSH function. Let $\mathcal{U}$ be a universal family of hash functions from $\{1,\dots,B\}$ to $\{1,\dots,B'\}$, where B is the number of buckets in the range of $\mathcal{H}$, and $B'>0$ is an integer of our choice. We recall that, as per the definition of universal hashing, $\mathcal{U}$ satisfies $\Pr_{u\sim\mathcal{U}}[u(b)=u(b')]\leq 1/B'$ for every b,b'.

We define an LSQ family $\mathcal{Q}$ as follows: to sample from it, we draw $h\sim\mathcal{H}$ and $u\sim\mathcal{U}$, and for every $x\in\mathbb{R}^d$ we let $f_{h,u}(x)\in\{0,1\}^{B'}$ be the indicator vector for $u(h(x))$. We return $(f_{h,u},f_{h,u})$ as the sampled pair from $\mathcal{Q}$.

A union bound over the collision probabilities of h and u yields that for all $x, y \in \mathbb{R}^d$,

$$\Pr_{u,h}[u(h(x)) = u(h(y))] \leq \Pr_{h}[h(x) = h(y)] + \tfrac{1}{B'}.$$

Consequently, if $\mathcal{H}$ is an α-approximate LSH family for k, then $\mathcal{Q}$ is an $(\alpha + \frac{1}{B'})$-approximate $(B', 1, 1)$-LSQ family for k. Proposition 3.5 follows by choosing $B' = \lceil 1/\alpha \rceil$. □

C. Expanded Discussion on Related Work

C.1. Generic Linear Queries

For completeness of the discussion of prior work from Section 1.4, we expand on some aspects of SmallDB and PMW for Gaussian DP-KDE in the function release model. These mechanisms are designed to answer generic linear queries. Let $\mathcal{X}$ denote the universe in which the elements of the dataset X are contained. The goal of a DP linear query is to estimate the quantity $\phi(X) := \frac{1}{|X|}\sum_{x\in X}\phi(x)$, where $\phi : \mathcal{X} \to [0,1]$ is a query function chosen by the client. In the case of KDE, we have $\mathcal{X} = \mathbb{R}^d$, and each query point $y \in \mathbb{R}^d$ corresponds to the query function $\phi_y(x) = k(x,y)$. Since SmallDB and PMW require $\mathcal{X}$ to be finite, we next discuss discretization.

Discretization for Gaussian KDE. If all points are assumed to be contained in ball of radius $\Phi \geq 1$ in $\mathbb{R}^d$, then for the purpose of approximation of Gaussian KDE (Definition 1.3), one can round every point coordinate to its nearest integer multiple of $\alpha/(4\Phi\sqrt{d})$. Thus, we can without loss of generality assume that $\mathcal{X}$ contains only those points in the ball that have such coordinates, of which there are $O(\Phi^2/\alpha)^d$ by a standard volume argument.

To see why this suffices for Gaussian KDE, let $x, y \in \mathbb{R}^d$, and let $\bar{x}$ be the result of rounding x. Then,

$$e^{-\|y-\bar{x}\|_2^2} = e^{-\|(y-x)-(x-\bar{x})\|_2^2} = e^{-\|y-x\|_2^2} \cdot e^{-\|x-\bar{x}\|_2^2} \cdot e^{2(y-x)^T(x-\bar{x})}.$$

Since $\|x-\bar{x}\|_2 \leq \alpha/(4\Phi)$,

$$1 \geq e^{-\|x-\bar{x}\|_2^2} \geq e^{-(\alpha/(4\Phi))^2} \geq e^{-\alpha},$$

and, by Cauchy-Schwartz and the fact that $\|y-x\|_2 \leq 2\Phi$,

$$|2(y-x)^T(x-\bar{x})| \leq 2\|y-x\|_2\|x-\bar{x}\|_2 \leq 2\cdot 2\Phi \cdot \frac{\alpha}{4\Phi} = \alpha,$$

which implies

$$e^{-\alpha} \leq e^{2(y-x)^T(x-\bar{x})} \leq e^{\alpha}.$$

Noting that $1 \leq e^{\alpha} \leq 1+2\alpha$ and $1-\alpha \leq e^{-\alpha} \leq 1$ for all $\alpha \in (0,1)$, we plug these back above and get,

$$(1-\alpha)^2 e^{-\|y-x\|_2^2} \leq e^{-\|y-\bar{x}\|_2^2} \leq (1+2\alpha)e^{-\|y-x\|_2^2},$$

thus $|e^{-\|y-\bar{x}\|_2^2} - e^{-\|y-x\|_2^2}| \leq 2\alpha \cdot e^{-\|y-x\|_2^2} \leq 2\alpha$. Therefore rounding up to this precision introduces an additive error of only $O(\alpha)$ to every kernel evaluation and hence to every KDE evaluation, and we can scale α down by an appropriate constant.

SmallDB. The mechanism works as follows: Let X be the curator dataset, and let Q be a set of client queries. Suppose we know of $s(\alpha, Q) \geq 0$ such that there exists a dataset Z of size $s(\alpha, Q)$ that satisfies $|\phi(Z) - \phi(X)| \leq \alpha$ for all $\phi \in Q$ simultaneously. SmallDB selects a dataset $\widetilde{Z}$ of size $s(\alpha, Q)$ using the DP exponential mechanism, and, in the query release model, releases the answers $\{\phi(\widetilde{Z}) : \phi \in Q\}$ to the client queries Q. When the goal is to release ϵ-DP accurate answers to all queries in Q simultaneously with constant probability (say 0.9), SmallDB has sample complexity $O(s(\alpha, Q)\cdot \log(|\mathcal{X}|)/(\epsilon\alpha))$.

(The exponential mechanism entails iterating over all possible datasets of size $s(\alpha, Q)$—that is, all $|\mathcal{X}|^{s(\alpha,Q)}$ subsets of $\mathcal{X}$ of that size—and computing their utility with respect to Q, which leads to the inefficient running time of SmallDB.)

By standard concentration (Hoeffding's inequality), it is well-known that $s(\alpha, Q) = O(\log(|Q|)/\alpha^2)$ for every Q and α, yielding a sample complexity of $O(\log(|Q|)\log(|\mathcal{X}|)/(\epsilon\alpha^3))$ for generic linear queries. In the transformation from query release to function release for Gaussian DP-KDE, we set $Q = \mathcal{X}$. By discretization we have $|\mathcal{X}| = O(\Phi^2/\alpha)^d$, hence the above sample complexity becomes $O(d^2\log^2(\Phi/\alpha)/(\epsilon\alpha^3))$. However, it can be improved somewhat further, due to the existence of coresets for Gaussian KDE. An α-coreset for X is a dataset Z such that $|KDE_X(y) - KDE_Z(y)| \leq \alpha$ for all $y \in \mathbb{R}^d$ simultaneously. It is known that every dataset has an α-coreset for Gaussian KDE of size

$$C_{d,\alpha} = O(\min\{\alpha^{-1}\sqrt{d\log(1/\alpha)}, \alpha^{-2}\}),$$

see (Lopez-Paz et al., 2015; Lacoste-Julien et al., 2015; Phillips & Tai, 2020; Karnin & Liberty, 2019). Therefore, $C_{d,\alpha}$ is an upper bound on $s(\alpha, Q)$ for every Q and α. This yields the SmallDB sample complexity bound listed in Table 1.

The curator running time, which as mentioned above depends on enumerating over all datasets of size $s(\alpha, Q)$, is similarly improved. The curator output is the synthetic dataset $\widetilde{Z}$ released by the exponential mechanism, and it contains $C_{d,\alpha}$ points in $\mathbb{R}^d$, hence its size is $O(d \cdot C_{d,\alpha})$ words. The client can estimate $KDE_X(y)$ on this output by computing $KDE_{\widetilde{Z}}(y)$, which takes time $O(d \cdot C_{d,\alpha})$.

PMW. The mechanism has sample complexity $\tilde{O}(\log(|Q|)\log(|\mathcal{X}|)/(\epsilon\alpha^3))$ for generic linear queries. It is similar to that of SmallDB up to log factors, but stems from a different analysis (that we do not revisit here) which is not immediately improved by the existence of coresets. In the DP-KDE function release case we have, as above, $|Q| = |\mathcal{X}| = O(\Phi^2/\alpha)^d$, leading to the sample complexity listed in Table 1.

(We remark that PMW, unlike SmallDB, allows for adaptive queries in the query release model. Since we transform both mechanisms to the function release model for DP-KDE, this distinction between them does not apply in our setting.)

Like SmallDB, the output of PMW (in the function release model) is a synthetic private dataset $\widetilde{Z}$ on which the KDE of every query point can be directly evaluated. Initially $\widetilde{Z}$ can be as large as $\mathcal{X}$, but it too can be replaced by a coreset of itself, increasing the additive error of every query by at most α. The coresets bounds listed above are constructive (in particular, a uniformly random sample of $O(1/\alpha^2)$ from $\tilde{Z}$ yields an α-coreset for it with constant probability (Lopez-Paz et al., 2015)), and since the released coreset would be computed from $\tilde{Z}$ which is already ϵ-DP, the coreset too would be ϵ-DP by immunity of differential privacy to post-processing. Consequently, like SmallDB, the curator output size and client running time of PMW are both $O(d \cdot C_{d,\alpha})$.

From query release to function release: uniform convergence and running time. As alluded to above, a naïve way to transform a query release mechanism into a function release mechanism is to invoke it with all possible queries, of which (by the above discretization argument) we have $O(\Phi^2/\alpha)^d$. This was used above to determine the sample complexity bounds for SmallDB and PMW. In fact, by uniform convergence results from learning theory, invoking these mechanisms with a small random sample of queries (instead of all possible queries) suffices to turn them into function release mechanisms. The reason is that the functions these mechanisms release admit a short description (a small synthetic dataset in the case of SmallDB, or a short transcript that describes the synthetic dataset in the case of PMW), and therefore the released functions can be "learned" on a small sample of queries and still generalize (in the learning theory sense) to all queries. We omit further details. This argument does not change the sample complexity of these mechanisms, but it somewhat improves the curator running time (albeit it remains at least exponential in d, as listed in Table 1).

C.2. Adaptive Queries

In Section 1.4 we mentioned that SmallDB and PMW, when used in the function release model, have the property that with a fixed probability of say 0.9, they release a function[5] which returns the correct answer up to an additive error of at most α for all queries simultaneously (assuming all points are contained in a ball of radius Φ). This is a stronger guranatee than (α, η)-approximation. In particular, it allows to use the released function for adaptive queries.

We can achieve the same stronger guarantee for our mechanisms (and similarly for the Bernstein mechanism), by setting η sufficiently small so as to allow for a union bound over all queries (namely, by the above discretization bound, $\eta = \Theta(\alpha/\Phi^2)^d$). We get the following corollaries of Theorems 1.1 and 1.2 respectively.

Corollary C.1 (high dimensions)**.** *There is an ϵ-DP function release mechanism for Gaussian KDE on datasets in $\mathbb{R}^d$ of size $n \geq O(d\log(\Phi/\alpha)/(\epsilon\alpha^2))$ and that are contained in a ball of radius Φ, such that with probability* 0.9*, the released function has additive error at most α on every query simultaneously. Furthermore:*

- *The curator runs in time $O(nd^2\log(\Phi/\alpha)/\alpha^2)$.*

[5] In the case of SmallDB and PMW, the released function in fact takes the form of a synthetic dataset.

- *The output size is* $O(d^2 \log(\Phi/\alpha)/\alpha^2)$.
- *The client runs in time* $O(d^2 \log(\Phi/\alpha)/\alpha^2)$.

Corollary C.2 (low dimensions)**.** *There is an* ϵ*-DP function release mechanism for Gaussian KDE on datasets in* $\mathbb{R}^d$ *of size* $n \geq \log(1/\eta) \cdot (\log(1/\alpha))^{O(d)}/(\epsilon\alpha)$ *and that are contained in a ball of radius* Φ*, such that with probability* 0.9*, the released function has additive error at most* α *on every query simultaneously. Furthermore:*

- *The curator runs in time* $(nd + (\frac{\Phi}{\sqrt{d}})^d) \cdot O(\log(1/\alpha))^{O(d)} \cdot d\log(\Phi/\alpha)$.
- *The output size is* $O((1 + \frac{\Phi}{\sqrt{d}})(\log(1/\alpha)))^d \cdot d\log(\Phi/\alpha)$.
- *The client runs in time* $(\log(1/\alpha))^{O(d)} \cdot d\log(\Phi/\alpha)$.

Note that the dependence on Φ remains polylogarithmic, and for the first mechanism, the dependence on the dimension remains polynomial.

C.3. (ϵ, δ)-DP and Query Release

When (ϵ, δ)-DP with $\delta > 0$ (a.k.a. approximate DP) is allowed, the most notable prior result on Gaussian DP-KDE is due to (Hall et al., 2013), which we call the HRW mechanism. Their mechanism is time-efficient in the query release model, albeit not in the function release model. To describe it, we define the query release model as follows. First, the client sends the curator q query points, $y_1, \ldots, y_q \in \mathbb{R}^d$. In response the curator, who holds a dataset X, releases a sequence of answers $A = (a_1, \ldots, a_q)$. We require that *(i)* A is differentially private w.r.t. X, and *(ii)* with probability (say) 0.99, it holds that $\max_{i=1,\ldots,q} |a_i - KDE_X(y_i)| \leq \alpha$.[6]

Note that in the query release model, no "curse of dimensionality" immediately arises at all: the curator can simply compute the true KDE values of all queries in time $O(dnq)$, and release them after adding appropriate privacy-preserving noise.[7] However, such naïve mechanisms lead to an undesirably large sample complexity (or equivalently, undesirably large error α), and improving the sample complexity while avoiding exponential dependence on d turns out to be challenging. This is manifested in the following discussion, whose quantitative results are summarized in Table 2.

Query release with pure DP. For context, let us start with DP-KDE in the query release model under pure DP, that is, where the released sequence of answers A must be ϵ-DP w.r.t. X. As alluded to above, the curator can invoke the vanilla Laplace mechanism: compute the true KDE values of the q queries, and add noise sampled independently from $\text{Laplace}(q/(\epsilon n))$ to each. It is not hard to verify that A has ℓ_1-sensitivity q/n, hence the mechanism is ϵ-DP. The running time is $O(dnq)$. The resulting sample complexity is $O(q\log(q)/(\epsilon\alpha))$. While the dependence on d, α, ϵ is desirable, the dependence on q in the sample complexity impedes the usability of this mechanism if the number of queries is large.

Instead of the Laplace mechanism, one could use SmallDB or PMW, whose sample complexity has better dependence on q in some regimes, albeit their running time is (at least) exponential in d. LSQ-RFF achieves a sample complexity of $O(\log(q)/(\epsilon\alpha^2))$ and running time linear in d, subsuming SmallDB and PMW on both counts.[8] Comparing its sample complexity to the Laplace mechanism, the dependence on q is exponentially better, while the dependence on α is quadratically worse. LSQ-FGT has sample complexity $O(\log(q) \cdot (\log(1/\alpha))^d/(\epsilon\alpha))$ and running time exponential in d, improving over the above mentioned results only when d is small.

Query release with approximate DP: the HRW mechanism. Now suppose approximate DP is allowed—that is, the curator is allowed to release an answer sequence A which is (ϵ, δ)-DP w.r.t. X. The natural analog of the vanilla Laplace mechanism from the pure DP case is the vanilla Gaussian mechanism (see (Dwork et al., 2014)): the curator computes the true KDE values of all queries, and adds independent Gaussian noise $N(0, 2q\log(1.25/\delta)/(\epsilon n)^2)$ to each. It is not hard to verify that A has ℓ_2-sensitivity $\sqrt{q}/n$, hence the mechanism is (ϵ, δ)-DP. The running time is $O(dnq)$. The resulting sample complexity is $O(\sqrt{q\log(q) \cdot \log(1/\delta)}/(\epsilon\alpha))$. While the dependence on q is quadratically better than the pure-DP Laplace

[6] This is the *batch* query release model. In the *online* query release model, the client may send the curator additional queries after seeing the answers to previous ones. The results we describe in this section extend to the online variant as well.

[7] Note that this would not have been possible in the function release model, where the curator has no access to the queries, and no party has to access to both the dataset and the queries simultaneously, thus the true KDE values cannot be computed at all—unless the curator enumerates over all possible queries in advance, before receiving any specific queries from the client.

[8] Of course, LSQ-RFF is specialized for KDE queries, while SmallDB and PMW apply to general linear queries.

Table 2. ϵ-DP and (ϵ, δ)-DP KDE query release mechanisms for the Gaussian kernel, that receive q queries and approximate each KDE up to additive error α. (*) SmallDB, PMW and LSQ-FGT assume that all points lie in a ball of radius Φ. (‡) Recall that $O(\Phi^2/\alpha)^d$ is an upper bound on the number of possible points (cf. discretization in Appendix C.1), hence on the number of queries q, hence the $d\log(\Phi/\alpha)$ term in the sample complexity of SmallDB is at least $\Omega(\log q)$.

MECHANISM	PURE DP?	SAMPLE COMPLEXITY	RUNTIME DEPENDENCE ON d	
Laplace	Yes	$O(\frac{q\log q}{\epsilon\cdot\alpha})$	Linear	
SmallDB	Yes	$O\left(\frac{\min\{\sqrt{d\log(1/\alpha)},1/\alpha\}\cdot d\log(\Phi/\alpha)}{\epsilon\cdot\alpha^2}\right)$	Exponential	(*), (‡)
PMW	Yes	$\tilde{O}\left(\frac{\log(q)\cdot d\log(\Phi/\alpha)}{\epsilon\cdot\alpha^3}\right)$	Exponential	(*)
LSQ-RFF	Yes	$O(\frac{\log q}{\epsilon\cdot\alpha^2})$	Linear	
LSQ-FGT	Yes	$\frac{\log q}{\epsilon\cdot\alpha}\cdot(\log(1/\alpha))^{O(d)}$	Exponential	(*)
Gaussian	No	$O(\frac{\sqrt{q\log q\log(1/\delta)}}{\epsilon\cdot\alpha})$	Linear	
PMW	No	$\tilde{O}\left(\frac{\log q\sqrt{d\log(\Phi/\alpha)\log(1/\delta)}}{\epsilon\cdot\alpha^2}\right)$	Exponential	(*)
HRW	No	$O(\frac{\sqrt{\log q\log(1/\delta)}}{\epsilon\cdot\alpha})$	Linear	

mechanism, it is still undesirably large. Again, one could use SmallDB or PMW, but they are subsumed by the pure-DP LSQ-RFF mechanism, even when approximate DP is allowed.[9]

(Hall et al., 2013) presented the HRW mechanism, which is (ϵ, δ)-DP, runs in time $O(dq(n+q))$, and achieves sample complexity $O(\sqrt{\log(q)\cdot\log(1/\delta)}/(\epsilon\alpha))$. It operates similarly to the Gaussian mechanism, except that the noise samples added to different answers are not independent, but correlated via an appropriate Gaussian process, allowing for much less noise per query. Namely, the mechanism returns $a_i = KDE_X(y_i) + Z_i$, where $Z_i \sim N(0, 2\log(2/\delta)/(\epsilon n)^2)$ and $\mathrm{Cov}(Z_i, Z_j) = k(y_i, y_j) = e^{-\|y_i - y_j\|_2^2}$. They prove that the mechanism is (ϵ, δ)-DP for arbitrarily many queries, even though the noise magnitude per query does not grow with q at all. (The extra $\sqrt{\log q}$ term in the sample complexity is from a standard bound on the maximum of this finite Gaussian process, ensuring that all q queries are answered accurately simultaneously.) The HRW sample complexity is better than all previously mentioned results if approximate DP with sufficiently large δ (say a small constant $\delta = \Omega(1)$) is allowed.

Query release vs. function release. The HRW mechanism runs in time linear in d in the query release model, but in order to use it for function release, the curator must release answers to all possible queries, which entails running time exponential in d. Thus, in the function release model, to our knowledge, the LSQ-RFF mechanism, despite being pure-DP, is currently the only DP-KDE mechanism for the Gaussian kernel that achieves (α, η)-approximation with running time linear in d, even if approximate DP is allowed.

C.4. Overview of LSHable Kernels

As mentioned in the introduction, the Laplacian kernel $k(x, y) = e^{-\|x-y\|_1}$ is likely the most popular LSHable kernel over $\mathbb{R}^d$. For completeness, in this section we give an overview of other kernels known to be LSHable.

(Rahimi & Recht, 2007) introduced a family of LSHable kernels (although they did not use this terminology) in their Random Binning Features construction. They start by showing that the *hat kernel* over $x, y \in \mathbb{R}$, $\hat{k}_\sigma(x, y) = \max\{0, 1 - |x-y|/\sigma\}$, is LSHable. They then show this implies the LSHability of shift-invariant kernels over $\mathbb{R}$ that can be written as convex

[9] PMW has an (ϵ, δ)-DP variant with better bounds than its pure-DP variant. SmallDB has no (ϵ, δ)-DP variant. See Table 2.

combinations of such hat kernels on a compact subsets of $\mathbb{R} \times \mathbb{R}$ (this includes the one-dimensional Laplacian kernel $k(x, y) = e^{-|x-y|}$), and of kernels over $\mathbb{R}^d$ that can be written as the product of one-dimensional LSHable kernels over the coordinates (this includes the d-dimensional Laplacian kernel $k(x, y) = e^{-\|x-y\|_1} = \prod_{i=1}^{d} e^{-|x_i - y_i|}$). They note that this family does not include the Gaussian kernel.

(Andoni & Indyk, 2009) discussed additional LSHable kernels over $\mathbb{R}^d$: the *exponential* kernel $k(x, y) = e^{-\|x-y\|_2}$, whose LSHability follows from that of the Laplacian kernel essentially by an (efficient and approximate) isometric embedding of ℓ_2 into ℓ_1;[10] the *geodesic* kernel over the unit sphere, $k(x, y) = 1 - \pi^{-1}\theta(x, y)$, where $\theta(x, y)$ denotes the angle between x and y; and the Erfc kernel $k(x, y) = \frac{\mathrm{erfc}(\|x-y\|_2)}{2-\mathrm{erfc}(\|x-y\|_2)}$, where $\mathrm{erfc}(z) = \frac{2}{\sqrt{\pi}} \int_z^\infty e^{-t^2} \mathrm{d}t$ is the complementary Gauss error function. Regarding the lack of LSHability results for the Gaussian kernel, they suggest using the Erfc kernel as a proxy (naming it a "near-Gaussian kernel"), showing it approximates the Gaussian kernel at every up point up to an additive error of 0.16. Unfortunately, this error is far too large for most KDE applications. Furthermore, the LSH family associated with the Erfc kernel has running time that depends exponentially on the additive error α (where α is the approximation error for the Erfc kernel, leading to an error of $0.16 + \alpha$ for the Gaussian kernel), making it infeasible when α is small.

The lack of available LSHability results for the Gaussian and Cauchy kernel is also discussed in (Backurs et al., 2018; Siminelakis et al., 2019), who develop alternative methods for (non-private) approximation of these kernels where normally LSHability would be used.

Finally, apart from $\mathbb{R}^d$, some LSHability results are available for kernels that measure similarity over finite spaces. (Andoni & Indyk, 2009) observe that the Jaccard kernel is LSHable, while (Chierichetti & Kumar, 2015) discuss transformations that preserve the LSHability of such kernels.

D. Extensions to Other Kernels

In this section we discuss the applicability of our results beyond the Gaussian kernel. The key distinction to draw here is between the *sample complexity* of the DP-KDE mechanism (i.e., the tradeoff between the privacy parameter ϵ and the additive error parameters α, η), for which we can make general statements for some families of kernels, to the *computational efficiency* of the mechanism (i.e. the running times of the curator and the client, and the curator output size), which would generally depend on the specific properties of each kernel.

D.1. LSQ with RFF

(Rahimi & Recht, 2007) showed that every positive definite shift-invariant kernel (abbreviated henceforth as a PDSI kernel) admits a family of random Fourier features. More precisely, for every such kernel k defined over $\mathbb{R}^d$, there exists a distribution $\mathcal{D}_k^{RFF}$ over $\mathbb{R}^d$ such that

$$\forall x, y \in \mathbb{R}^d \quad , \quad k(x, y) = \mathbb{E}_{\omega \sim \mathcal{D}_k^{RFF}, \beta \sim \mathrm{Uniform}[0, 2\pi)}[\sqrt{2}\cos(\omega^T x + \beta) \cdot \sqrt{2}\cos(\omega^T y + \beta)].$$

This implies that every PDSI kernel is $(1, \sqrt{2}, 1)$-LSQable. Therefore, from Lemmas 2.2 and 2.5 we get the following result.

Theorem D.1. *For every PDSI kernel over $\mathbb{R}^d$, there is an ϵ-DP function release mechanism for (α, η)-approximation of its KDE, on datasets of size at least $n \geq O(\log(1/\eta)/(\epsilon\alpha^2))$.*

These are the same privacy, utility and sample complexity guarantees as we get for the Gaussian kernel in Theorem 1.1. However, the computational efficiency (and more specifically in the case, the curator running time) depends on the computational properties of D_k^{RFF} for each specific kernel k. Namely, it hinges on whether one can sample ω from $\mathcal{D}_k^{RFF}$ efficiently. Formally, by Lemma 2.3, we get:

Proposition D.2. *Let k be a PDSI kernel. Let T_k^{RFF} be the time complexity of drawing a sample ω from $\mathcal{D}_k^{RFF}$. Then, the LSQ-RFF DP-KDE mechanism from Theorem D.1 satisfies the following:*

- *The curator runs in time $O((nd + T_k^{RFF})\log(1/\eta)/\alpha^2)$.*
- *The curator output size is $O(d\log(1/\eta)/\alpha^2)$.*

[10] Note that the exponential kernel is different from the Laplacian kernel in that the norm in the exponent is ℓ_2 and not ℓ_1, and is different from the Gaussian kernel in that the norm is not squared.

- *The client runs in time $O(d\log(1/\eta)/\alpha^2)$.*

Proof. In the notation of Lemma 2.3, we have $T_Q = T_k^{RFF}$. Furthermore, $L_Q = d+1$ since this is the number of machine words needed to describe a pair ω, β (regardless of the time it took to sample ω), and $T_f = T_g = O(d)$ since computing $\sqrt{2}\cos(\omega^T x + \beta)$ given x, ω, β takes time $O(d)$. We plug these into Lemma 2.3 together with $I = O(\log(1/\eta)/\alpha^2)$, the setting of I used in Lemma 2.5 to obtain Theorem D.1, and the proposition follows. □

Let us give some examples of $\mathcal{D}_k^{RFF}$ and T_k^{RFF} for specific kernels, and observe how they affect the efficiency of the LSQ-RFF mechanism.

- *Gaussian, Laplacian and Cauchy kernels:* For these three kernels, mentioned in Section 1.2, (Rahimi & Recht, 2007) derived the corresponding RFF distributions (we list them here with bandwidth $\sigma = 1$):
  - For the Gaussian kernel $k(x,y) = \exp(-\|x-y\|_2^2)$, $\mathcal{D}_k^{RFF}$ is the d-dimensional Gaussian distribution $\sqrt{2}\cdot N(0, I_d)$.
  - For the Laplacian kernel $k(x,y) = \exp(-\|x-y\|_1)$, $\mathcal{D}_k^{RFF}$ is the d-dimensional Cauchy distribution, whose density at $\omega \in \mathbb{R}^d$ is $\prod_{j=1}^d (\pi(1+\omega_j^2))^{-1}$.
  - For the Cauchy kernel $k(x,y) = \prod_{j=1}^d 2/(1+(x_j-y_j)^2)$, $\mathcal{D}_k^{RFF}$ is the d-dimensional Laplace distribution $\mathrm{Laplace}(0, I_d)$.

  Each of these distributions is a d-dimensional product distribution where each coordinate can be sampled in time $O(1)$, hence $T_k^{RFF} = O(d)$. Therefore, for these kernels, we get the same DP-KDE results as stated for the Gaussian kernel in Theorem 1.1.
- *Exponential ℓ_p^p kernels:* Let $p \in [1,2]$. Consider the kernel $k(x,y) = \exp(-\|x-y\|_p^p)$. This can be seen as a generalization of the Gaussian and Laplacian kernels (which correspond to $p=2$ and $p=1$ respectively). For this kernel, it can be checked that $\mathcal{D}_k^{RFF}$ is the d-dimensional product distribution whose coordinates are i.i.d. samples from the p-stable distribution, and furthermore, each coordinate can be sampled in time $O(1)$. See (Indyk, 2006) for the definition of the p-stable distribution and for how to efficiently sample from it. Therefore, for these kernels too we have $T_k^{RFF} = O(d)$, and we get the same DP-KDE result as in Theorem 1.1.
- *Exponential ℓ_p kernels:* Again let $p \in [1,2]$, and consider the kernel $k(x,y) = \exp(-\|x-y\|_p)$. Note that, in contrast to the previous case, the ℓ_p-norm in the exponent is not raised to the power p. The $p=1$ case again coincides with the Laplacian kernel, while the $p=2$ case coincides with the exponential kernel mentioned in Appendix C.4. These kernels are PDSI, hence Theorem D.1 and Proposition D.4 hold for them. However, we do not immediately see how to efficiently sample from their RFF distribution $\mathcal{D}_k^{RFF}$ (even though it may be possible), and are therefore unable to determine T_k^{RFF} and bound the curator running time of their LSQ-RFF DP-KDE mechanism.

D.2. LSQ with FGT

The Fast Gauss Transform is rather specialized to the Gaussian kernel. Nonetheless, it can be extended to certain kernels with sufficiently similar properties, like those discussed in (Alman et al., 2020), section 9.3. For those kernels, we get the same DP-KDE results as we get for the Gaussian kernel in Theorem 1.2.

D.3. LSQ with LSH

With LSH, the situation is similar to LSQ-RFF: for every LSHable kernel we can get a DP-KDE mechanism with the same privacy and utility guarantees as Theorem D.1, but the computational efficiency depends on the properties of the LSH family associated with that specific kernel. More precisely, we have the following result, which we recall follows already from the prior work of (Coleman & Shrivastava, 2021).

Theorem D.3. *For every α-approximate LSHable kernel over $\mathbb{R}^d$, there is an ϵ-DP function release mechanism for (α,η)-approximation of its KDE, on datasets of size at least $n \geq O(\log(1/\eta)/(\epsilon\alpha^2))$.*

Proof. By Proposition 3.5, k is 2α-approximate $(\lceil 1/\alpha \rceil, 1, 1)$-LSQable, hence the theorem follows from Lemmas 2.2 and 2.5. □

These are the same privacy, utility and sample complexity guarantees as we get for the Gaussian kernel in Theorem D.1 (however, note that PDSI kernels and LSHable kernels are distinct classes of kernels). The computational efficiency of the LSH-based mechanism depends on the computational properties of the LSH family as follows.

Proposition D.4. *Let k be an α-approximate LSHable kernel over $\mathbb{R}^d$. Let $\mathcal{H}$ be the associated LSH family. Let B be range size (i.e., number of hash buckets) of the hash functions in $\mathcal{H}$. Let $T_{\mathcal{H}}$ be the time to sample $h \sim \mathcal{H}$, let T_h be the time to evaluate $h(x)$ given $h \in \mathcal{H}$ and $x \in \mathbb{R}^D$, and let $L_{\mathcal{H}}$ be the description size of $h \in \mathcal{H}$. Then, the LSQ-LSH DP-KDE mechanism from Theorem D.3 satisfies the following:*

- *The curator runs in time $O((nT_h + T_{\mathcal{H}})\log(1/\eta)/\alpha^2)$.*
- *The curator output size is $O((L_{\mathcal{H}} + \min\{B, \log B + 1/\alpha\}) \cdot \log(1/\eta)/\alpha^2)$.*
- *The client runs in time $O(T_h \log(1/\eta)/\alpha^2)$.*

Proof. Let $I = O(\log(1/\eta)/\alpha^2)$, noting this is the setting of I used in Lemma 2.5 to obtain Theorem D.3.

Recall that we have two options to transform the LSH family into an LSQ family: either by Proposition 3.4 or by Proposition 3.5. We analyze both cases. If we use Proposition 3.4, then k is $(B, 1, 1)$-LSQable, and in the notation of Lemma 2.3 we have $T_{\mathcal{Q}} = T_{\mathcal{H}}$, $T_f = T_g = T_h$, and $L_{\mathcal{Q}} = L_{\mathcal{H}}$. Applying Lemma 2.3, the curator running time is $O(I(nT_h + T_{\mathcal{H}} + B))$, the curator output size is $O(I(L_{\mathcal{H}} + B))$, and the client running in time $O(I \cdot T_h)$.

Alternatively, if we use Proposition 3.5, then k is $(\lceil 1/\alpha \rceil, 1, 1)$-LSQable. The proof of Proposition 3.5 (cf. Appendix B.2) obtains the LSQ family by composing over $\mathcal{H}$ a universal hash family $\mathcal{U}$ that hashes a domain of size B into $\lceil 1/\alpha \rceil$ hash buckets. There are well-known choices for $\mathcal{U}$ (e.g., (Carter & Wegman, 1977)) with sampling and evaluation times $O(1)$ and description size $O(\log B)$. Hence, for the composition of $\mathcal{U}$ over $\mathcal{H}$, we have in the notation of Lemma 2.3 $T_{\mathcal{Q}} = T_{\mathcal{H}} + O(1)$, $T_f = T_g = T_h + O(1)$, and $L_{\mathcal{Q}} = L_{\mathcal{H}} + O(\log B)$. Applying Lemma 2.3, the curator running time is $O(I(nT_h + T_{\mathcal{H}} + 1/\alpha))$, the curator output size is $O(I(L_{\mathcal{H}} + \log B + 1/\alpha))$, and the client running in time $O(I \cdot T_h)$.

Putting these together, the curator running time is $O(I(nT_h + T_{\mathcal{H}} + \min\{B, 1/\alpha\}))$, the curator output size is $O(I(L_{\mathcal{H}} + \min\{B, \log B + 1/\alpha\}))$, and the client running in time $O(I \cdot T_h)$. Note that the approximation guarantee in Theorem D.3 requires $n \geq O(\log(1/\eta)/(\epsilon\alpha^2))$, hence $nT_h \geq n \geq O(1/\alpha)$, and hence the curator running time becomes $O(I(nT_h + T_{\mathcal{H}}))$. These are the bounds claimed in the proposition. □

Here too, let us give some examples of how different LSH families affect the computational efficiency of the DP-KDE mechanism.

- *Laplacian, exponential and geodesic kernels:* as already mentioned in Section 3.3, the Laplacian kernel admits an LSH family that satisfies $T_{\mathcal{Q}}, T_f, T_g, L_{\mathcal{Q}} = O(d)$ in the notation of Lemma 2.3. Therefore, we get an efficient DP-KDE mechanism for it, as stated in Theorem 3.6. The exponential kernel and the geodesic kernel, mentioned as LSHable in Appendix C.4, also have LSH families with similar (though perhaps slightly different) efficiency properties, given in (Andoni & Indyk, 2009).
- *Erfc kernel:* In Appendix C.4 we defined the Erfc kernel, and mentioned that (Andoni & Indyk, 2009) showed it is α-approximate LSHable, albeit with an LSH family that takes time exponential in α to sample from. Therefore, for this kernel we get a DP-KDE mechanism with the privacy, utility and sample complexity stated in Theorem D.3, but with running time exponential in α.
- *Exponential ℓ_p kernels:* Let us revisit the family of kernels $k(x, y) = \exp(-\|x - y\|_p)$ with $p \in [1, 2]$. We discussed these kernels in the context of LSQ-RFF, and showed that while we have DP-KDE mechanisms for them, we do not know them to be computationally efficient. This result also follows by LSHability. The reason is that ℓ_p is known to embed isometrically into ℓ_1 (Johnson & Schechtman, 1982). This implies that the kernel $k(x, y) = \exp(-\|x - y\|_p)$ with any $p \in [1, 2]$ is LSHable, by first applying an isometric embedding of the ℓ_p distances into ℓ_1, and then using the LSHability of the Laplacian kernel. However, except in the $p = 2$ case, it is not known how to compute an (approximately) isometric embedding of ℓ_p into ℓ_1 efficiently. Therefore, while for these kernels we can get DP-KDE mechanisms from Theorem D.3, we are unable to bound their computational efficiency.

E. Additional Experiments and Implementation Details

E.1. Mechanism Implementation

In this section we provide details on how we instantiate the LSQ mechanism from Algorithm 1 into the LSQ-RFF and LSQ-FGT mechanisms included in our code and used in our experiments, and on how these mechanisms are parameterized.

The efficiency/utility trade-off of the LSQ mechanism in Algorithm 1 is governed by the input parameters I, J, which are non-negative integers such that J is a divisor of I. (Observe that the computational efficiency bounds in Lemma 2.3 grow linearly with I.) Their role is simply to determine the number of repetitions in a standard median-of-means (MoM) averaging scheme, to induce the desired probabilistic concentration. The mechanism performs a total of I independent repetitions, and uses them to return the median of J terms, where each term is the average of $I' = I/J$ repetitions. As usual with MoM, I' governs the additive error α that we consider "successful", while J governs the probability η of failing to achieve that successful additive error.

From a typical theoretical perspective, one would like to select the desired utility parameters α and η, and ensure that the mechanism rigorously satisfies (α, η)-approximation. To this end, Lemmas 2.4 and 2.5 specify the setting of I and J that formally guarantees (α, η)-approximation and leads to our theoretical results, Theorems 1.1 and 1.2.

For our experiments, however, we would like to directly control the computational cost of our mechanisms, and measure their empirical utility as we vary the computational cost. To this end, we parameterize each of our two implemented mechanisms—LSQ-RFF and LSQ-FGT—by a single parameter that governs their computational efficiency, as follows. In both mechanisms, for simplicity, we use $J = 1$, which means we do not perform a median operation at all. One can always increase J and return the median over J independent repetitions in order to boost the success probability of each individual query, at the expense of degrading ϵ (by a factor of J) for releasing more information in those additional repetitions.

In LSQ-RFF, we parameterize the mechanism by the number of random Fourier features the mechanism uses, which (under the setting $J = 1$) coincides with the overall number of repetitions, I, in Algorithm 1.

In LSQ-FGT, there is the added complication that the LSQ family itself has variable computational cost. In order to define the FGT, the user selects an integer parameter $\rho \geq 1$, which determines the properties of the LSQ family as follows:

Proposition E.1. *Let $\rho \geq 1$ be an integer. The Gaussian kernel over points contained in a Euclidean ball of radius Φ in $\mathbb{R}^d$ admits an $e^{-O(\rho)}$-approximate $((1 + \frac{\Phi}{\sqrt{d}}) \cdot \rho)^d, O(1)^d, \rho^{O(d)})$-LSQ family, supported on a single pair of functions (f, g). Furthermore, the evaluation times of f on $x \in \mathbb{R}^d$ and of g on $y \in \mathbb{R}^d$ are both $(d \cdot \rho)^{O(d)}$.*

This is just a restatement of Proposition 3.2, parameterized by ρ instead of α (and it follows from the same proof in Appendix B.1). Note that as ρ increases, the parameters Q and S of the (Q, R, S)-LSQ family grow with it, which increases the computational cost of the LSQ mechanism according to Lemma 2.3. The description of LSQ-FGT in Section 3.2 sets $\rho = O(\log(1/\alpha))$ in order to prove Theorem 1.2, but in practice, when α is not chosen in advance but measured empirically, the user needs to set ρ directly. In our implementation of LSQ-FGT, we set the number of repetitions to $I = 1$, and use ρ as the parameter that governs the efficiency/utility trade-off.

E.2. Experimental Details

Preprocessing. All datasets are available online (download URLs are included in the bibliographic entries).

- Covertype (Blackard & Dean, 1999): No preprocessing.
- GloVe (Pennington et al., 2014): We use the 1M points, 100 dimensions version of the dataset. No preprocessing.
- Diabetes (Strack et al., 2014): we select the "age" and "time in hospital" columns. "time in hospital" is between 1 and 14 (days). "age" is given as a decade-long bracket (e.g., $[40 - 50)$) and we replace it with its midpoint (e.g., 45), and then divide it by 10 to equate the numerical range of both coordinates.[11]
- NYC Taxi (Chavez et al., 2018): We select the "pickup longitude" and "pickup latitude" columns. We filter out points

[11] This is equivalent to choosing the bandwidth as a non-scalar diagonal matrix, namely $\begin{pmatrix} 1 & 0 \\ 0 & 0.1 \end{pmatrix}$. Recall that the bandwidth is, in general, a $d \times d$ positive definite matrix Σ, with which the Gaussian kernel is defined as $k(x, y) = e^{-(x-y)^T \Sigma (x-y)}$.

with "pickup longitude" $\notin (-74.1, -73.15)$ or "pickup latitude" $\notin (40.5, 40.9)$ to eliminate corrupted records (these coordinate ranges are the general geographical vicinity of NYC). We use $100,000$ of the unfiltered points.

Bandwidth selection. For each dataset we tune the bandwidth according to the guidelines in prior work (Jaakkola et al., 1999; Backurs et al., 2019). The values are specified in Table 3. The bandwidth values are tuned are such that mean KDE values are on the order of 10^{-2} and their standard deviation is also on the order of 10^{-2}, yielding a meaningful and non-generate KDE distribution with a range of target values. Note that the performance of the NoisySample baseline in Section 4 (which returns the noisy mean of a sample of query points as the KDE estimate for any query point) corresponds to the standard deviation of KDE values in Table 3.

Table 3. Bandwidth values used in experiments.

Dataset	Bandwidth σ	Est. mean query KDE	Est. standard deviation of query KDE
Covertype	500	0.02	0.01
GloVe	3.33	0.01	0.01
Diabetes	1	0.06	0.03
NYC Taxi	0.01	0.08	0.03

Mechanism parameter selection. As discussed in Section 4.1, DP-KDE mechanisms have an optimal parameter setting for a given combination of error α and privacy ϵ. In our experiments this applies to LSQ-RFF (the parameter is the number of Fourier features), LSQ-FGT (the parameter is ρ, where ρ^d is the number of terms in truncated Hermite expansion) and Bernstein (the parameter is denote by k in (Alda & Rubinstein, 2017), where $(k+1)^d$ is the number of points in the lattice used to construct the Bernstein polynomial approximator, see below). In the error vs. privacy experiments in Section 4.2, we evaluate each mechanism at its optimal parameter for that specific value of ϵ. Due to the existence of the error divergence point (cf. Section 4.1), the optimal parameter setting for each algorithm exists and can be found by a finite parameter search.

For completeness, let us describe the Bernstein mechanism is somewhat more detail. It is parameterized by an integer $k \geq 1$. The mechanism constructs a uniform lattice with $(k+1)^d$ nodes over the unit hypercube $[0,1]^d$. It evaluates the KDE function at each point on the lattice, adds privacy-preserving Laplace noise to these evaluations, and then uses them to construct a Bernstein polynomial approximation of this discretized and privatized version of the true KDE function. As k increases, the mechanism's running time increases too, due to evaluating the KDE on each of the $(k+1)^d$ lattice points. Nonetheless, as shown for LSQ-RFF and LSQ-FGT in Section 4.1, increasing k does not necessarily lead to a smaller error—rather, the error begins to diverge at a certain setting of k, which depends on the desried privacy parameter ϵ. This happens for the same reason discussed in Section 4.1: as k increases, the non-private approximation error of the Bernstein polynomial approximator decays (see Theorem 5 in (Alda & Rubinstein, 2017) for the decay rate, which depends on the smoothness of the KDE function), while the magnitude of the Laplace noise increases like $(k+1)^d/(\epsilon n)$. Therefore, to achieve the optimal error for this mechanism, k needs to be chosen according to the available dataset size n and the desired privacy level ϵ.

E.3. Additional Accuracy Results

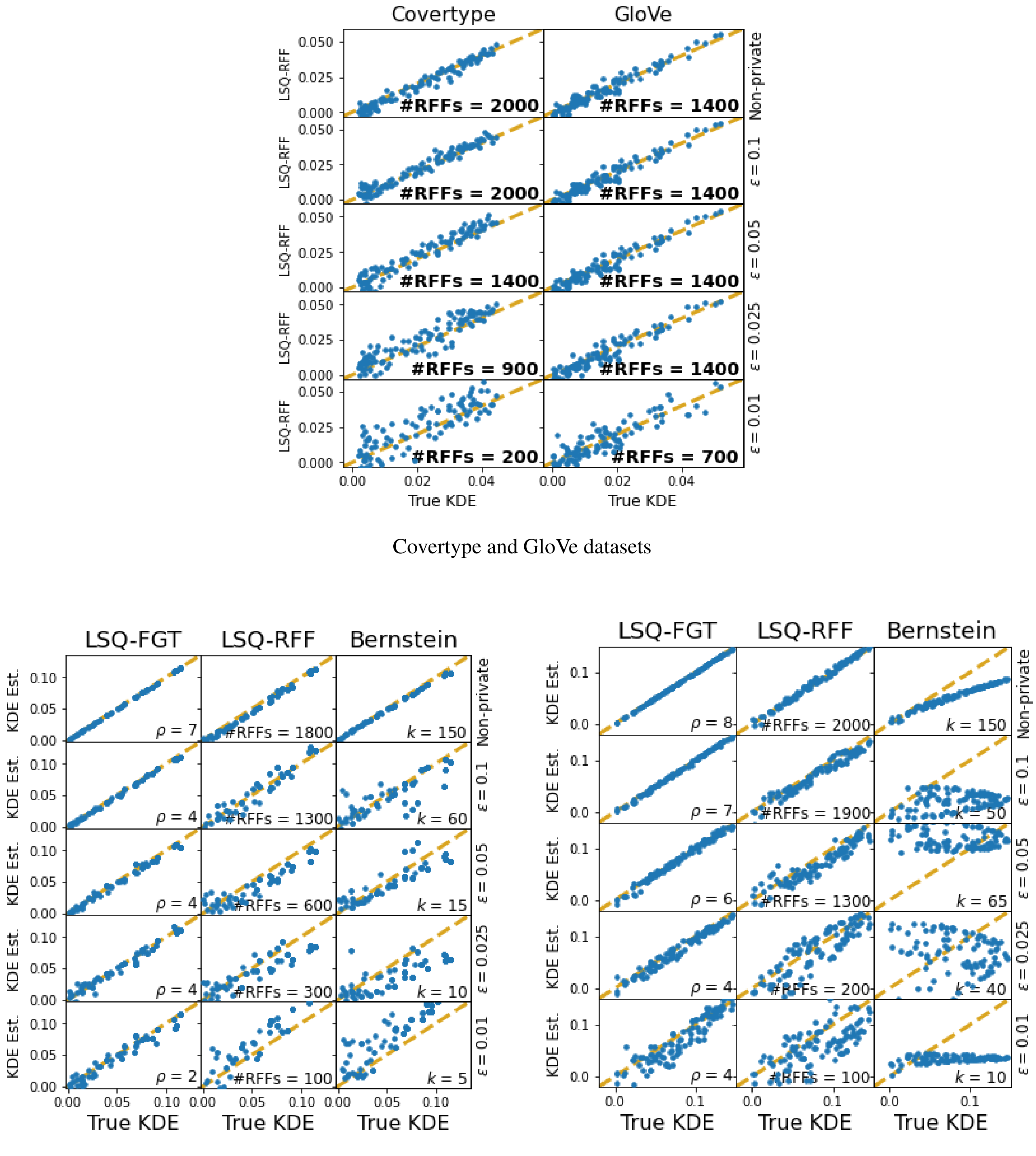
A more visual way to study the privacy-error trade-off of the various DP-KDE mechanisms is by directly comparing the ground-truth KDE values on a held-out test set to the KDE values estimated by the private mechanisms for different values of ϵ. Figure 4 shows the performance of LSQ-RFF under varying privacy budgets for the high-dimensional Covertype and GloVe datasets. Ideally, the estimated values would all lie close to the $y = x$ line, but degradation is inevitable as ϵ decreases. Additionally, Figure 4 compares the performance LSQ-RFF, LSQ-FGT, and the Bernstein mechanism on the low-dimensional Taxi and Diabetes datasets under the same set of privacy budgets. In Section 4.2, we noted that the NYC Taxi dataset poses a challenge for the Bernstein mechanism because of the dependence of sample complexity on $\alpha^{-\Theta(d/\sigma^2)}$. This difficulty manifests itself already in the non-private case, and (as expected) the mechanism output quality degrades further once noise is introduced to preserve privacy.

E.4. Additional Running Time Results

In Figure 3 in Section 4 we plotted the error vs. curator running time plots for all for our datasets, with $\epsilon = 0.05$. Figure 5 below displays the same experiment with $\epsilon = 0.02$.

E.5. Heatmaps

A common use for Kernel Density Estimation for two-dimensional datasets is in the generation of heatmaps showing where the bulk of the samples reside. For both the NYC Taxi and the Diabetes dataset, we use LSQ-RFF and LSQ-FGT to generate differentially private heatmaps for a number of different privacy budgets. The parameters of these algorithms (number of features for RFF, ρ for FGT) are selected to match the optimal values found earlier in Section 4.1. Results are in Figure 6. In all cases, while the heatmap gets increasingly distorted as the privacy budget shrinks, certain aggregate characteristics such as the general shape of the data manifold and the approximate location of its mode remain largely preserved.



Covertype and GloVe datasets

Diabetes dataset

NYC Taxi dataset

Figure 4. Ground truth vs. private estimates

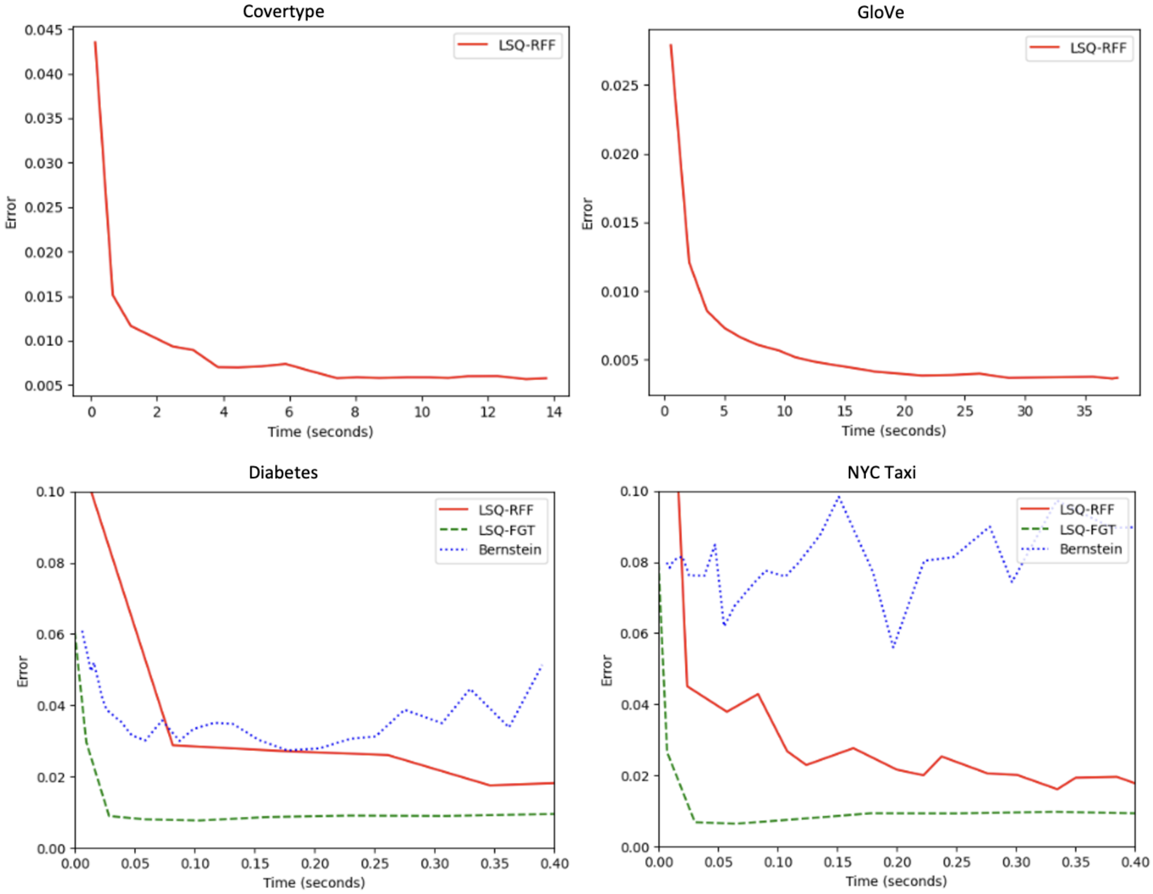
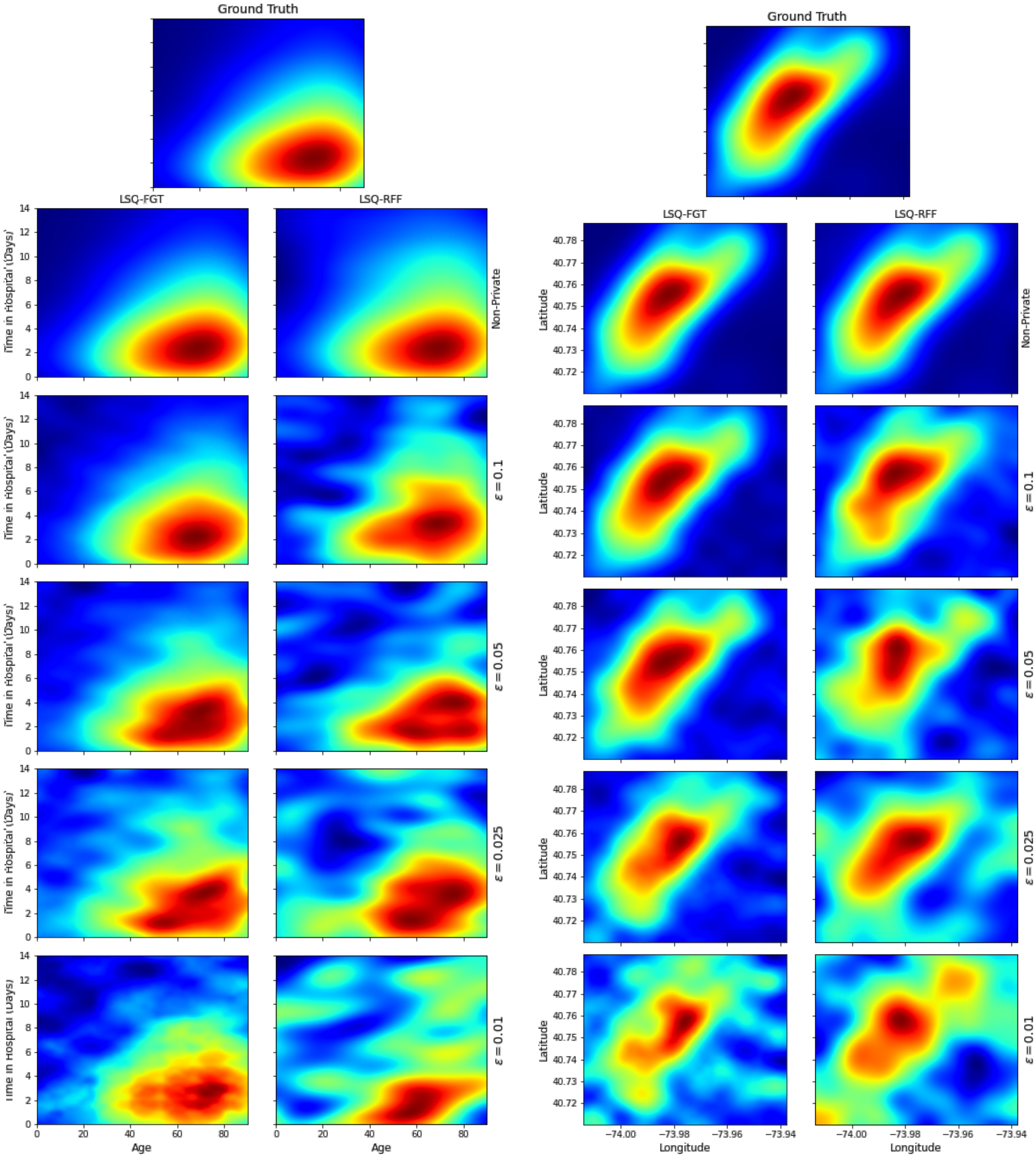


Figure 5. Error vs. curator running times with $\epsilon = 0.02$



Diabetes dataset

NYC Taxi dataset

Figure 6. Impact of privacy budget on the appearance of heatmap plots