Kernel Density Estimation through Density Constrained Near Neighbor Search

Moses Charikar
Computer Science Department
Stanford University
Stanford, USA
Email: moses@cs.stanford.edu

Michael Kapralov
School of Computer and
Communication Sciences
EPFL
Lausanne, Switzerland
Email: michael.kapralov@epfl.ch

Navid Nouri
School of Computer and
Communication Sciences
EPFL
Lausanne, Switzerland
Email: navid.nouri@epfl.ch

Paris Siminelakis
EECS
UC Berkeley
Berkeley, USA
Email: paris@berkeley.edu

***Abstract*—In this paper we revisit the kernel density estimation problem: given a kernel $K(x, y)$ and a dataset of n points in high dimensional Euclidean space, prepare a data structure that can quickly output, given a query q, a $(1+\epsilon)$-approximation to $\mu := \frac{1}{|P|}\sum_{p\in P} K(p, q)$. First, we give a single data structure based on classical near neighbor search techniques that improves upon or essentially matches the query time and space complexity for all radial kernels considered in the literature so far. We then show how to improve both the query complexity and runtime by using recent advances in data-dependent near neighbor search.**

We achieve our results by giving an new implementation of the natural importance sampling scheme. Unlike previous approaches, our algorithm first samples the dataset uniformly (considering a geometric sequence of sampling rates), and then uses existing approximate near neighbor search techniques on the resulting smaller dataset to retrieve the sampled points that lie at an appropriate distance from the query. We show that the resulting sampled dataset has strong geometric structure, making approximate near neighbor search return the required samples much more efficiently than for worst case datasets of the same size. As an example application, we show that this approach yields a data structure that achieves query time $\mu^{-(1+o(1))/4}$ and space complexity $\mu^{-(1+o(1))}$ for the Gaussian kernel. Our data dependent approach achieves query time $\mu^{-0.173-o(1)}$ and space $\mu^{-(1+o(1))}$ for the Gaussian kernel. The data dependent analysis relies on new techniques for tracking the geometric structure of the input datasets in a recursive hashing process that we hope will be of interest in other applications in near neighbor search.

***Keywords*-Kernel Density Estimation, Locality Sensitive Hashing, Near Neighbor Search**

Moses Charikar was supported by a Simons Investigator Award, a Google Faculty Research Award and an Amazon Research Award.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 759471).

Paris Siminelakis was supported by ONR DORECG award N00014-17-1-2127.

I. Introduction

Kernel density estimation is a fundamental problem with numerous applications in machine learning, statistics and data analysis [FG96], [SS01], [JKPV11], [SZK14], [GPPV+14], [ACMP15], [GB17]. Formally, the Kernel Density Estimation (KDE) problem is: preprocess a dataset P of n points $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathbb{R}^d$ into a small space data structure that allows one to quickly approximate, given a query $\mathbf{q} \in \mathbb{R}^d$, the quantity

$$K(P, \mathbf{q}) := \frac{1}{|P|}\sum_{\mathbf{p}\in P} K(\mathbf{p}, \mathbf{q}). \tag{1}$$

where $K(\mathbf{p}, \mathbf{q})$ is the kernel function. The Gaussian kernel

$$K(\mathbf{p}, \mathbf{q}) := \exp(-||\mathbf{p} - \mathbf{q}||_2^2/2)$$

is a prominent example, although many other kernels (e.g., Laplace, exponential, polynomial etc) are the method of choice in many applications [STC+04], [RW06].

In the rest of the paper, we use the notation μ^* defined as $\mu^* := K(P, \mathbf{q})$, and μ is a quantity that satisfies $\mu^* \le \mu \le 4\mu^*$.[1]

The kernel density estimation problem has received a lot of attention over the years, with very strong results available for low dimensional datasets. For example, the celebrated fast multipole method [BG97] and the related Fast Gauss Transform can be used to obtain efficient data structure for KDE (and in fact solves the more general problem of multiplying by a kernel matrix). However, this approach suffers from an exponential dependence on the dimension of the input data points, a deficiency that it shares with other tree-based methods [GM01],

[1]We have replaced μ^* with μ in the abstract for the ease of notation in the abstract.

[GM03], [YDGD03], [LMG06], [RLMG09]. A recent line of work [CS17], [CS19], [BCIS18], [BIW19] designed sublinear query algorithms for kernel density estimation in high dimensions using variants of the Locality Sensitive Hashing [CS17] framework of Indyk and Motwani [IM98].

Most of these works constructed estimators based on locality sensitive hashing, and then bounded the variance of these estimators to show that a small number of repetitions suffices for a good estimate. Bounding the variance of LSH-based estimators is nontrivial due to correlations inherent in sampling processes based on LSH, and the actual variance turns out to be nontrivially high.

In this work we take a different approach to implementing importance sampling for KDE using LSH-based near neighbor search techniques. At a high level, our approach consists of first performing independent sampling on the dataset, and then using using LSH-based near neighbor search primitives to extract relevant data points from this sample[2]. The key observation is that the sampled dataset in the KDE problem has nice geometric structure: the number of data points around a given query cannot grow too fast as a function of distance and the actual KDE value μ (we refer to these constraints as density constraints). The fact that our approach departs from the idea of constructing unbiased estimators of KDE directly from LSH buckets turns out to have two benefits: first, we immediately get a simple algorithm that uses classical LSH-based near neighbor search primitives (Euclidean LSH of Andoni and Indyk [AI06]) to improve on or essentially matches all prior work on kernel density estimation for radial kernels. The result is formally stated as Theorem 1 for the Gaussian kernel below, and its rather compact analysis in a more general form that extends to other kernels is presented in Section III. The second benefit of our approach is that it distills a clean near neighbor search problem, which we think of as near neighbor search under density constraints, and improved algorithms for that problem immediately yield improvements for the KDE problem itself. This clean separation allows us to use the recent exciting data-depending techniques pioneered by [AINR14], [AR15], [ALRW17] in our setting. It turns out that while it seems plausible that data-dependent techniques can improve performance in our setting, actually designing an analyzing a data-dependent algorithm for density constrained near neighbor search is quite nontrivial. The key difficulty here lies in the fact that one needs to design tools for tracking the evolution of the density of the dataset around a given query through a sequence of recursive partitioning steps (such evolution turns out to be quite involved, and in particular governed by a solution to an integral equation involving the log density of the kernel and properties of Spherical LSH). The design of such tools is our main technical contribution and is presented in the full version of this paper. The final result for the Gaussian kernel is given below as Theorem 2, and extensions to other kernels are presented in the full version of this paper.

[2]The approach of [BCIS18] also used near neighbor search techniques, but was only using c-ANN primitives as a black box, which turns out to be constraining – this only leads to strong results for slowly varying kernels (i.e., polynomial kernels). Our data-independent result recovers the results of [BCIS18], up to a $\mu^{-o(1)}$ loss, as a special case.

A. Our results

We instantiate our results for the Gaussian kernel as an illustration, and then discuss extensions to more general settings. We assume that $\mu^* = n^{-\Theta(1)}$, since this is the interesting regime for this problem. For $\mu^* = n^{-\omega(1)}$ under the Orthogonal Vectors Conjecture (e.g. [Rub18]), the problem cannot be solved faster than $n^{1-o(1)}$ using space $n^{2-o(1)}$ [CS19], and for larger values $\mu^* = n^{-o(1)}$ random sampling solves the problem in $n^{o(1)}/\epsilon^2$ time and space.

Data-Independent LSH: Our first result uses data-independent LSH of Andoni-Indyk [AI06] to improve upon the previously best known result [CS17] and follow up works that required query time $\widetilde{O}(\mu^{-0.5-o(1)}/\epsilon^2)$ if only polynomial space in $1/\mu$ is available.[3]

Theorem 1. *Given a kernel $K(\mathbf{p},\mathbf{q}) := e^{-a\|\mathbf{p}-\mathbf{q}\|_2^2}$ for any $a > 0$ and a data set of points P, there exists an algorithm for preprocessing and an algorithm for query procedure such that after receiving query $\mathbf{q}$ one can approximate $\mu^* := K(P,\mathbf{q})$ (see Definition 5) up to $(1\pm\epsilon)$ multiplicative factor, in time $\widetilde{O}\left(\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}\right)$, and the space consumption of the data structure is*

$$\min\left\{\epsilon^{-2}n\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}, \epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{1+o(1)}\right\}.$$

This theorem is stated and proved as Theorem 3 in Section III. To get a sense of the improvement, the result of [CS17] exhibited query time that is roughly a square root of the query time of uniform random sampling (for constant $\epsilon > 0$ and $\mu = O(n^{-o(1)})$). Our result uses the same LSH family as in [CS17] but achieves query time that is itself roughly the square root of that of [CS17]!

Data-Dependent LSH: Our main technical contribution is a collection of techniques for using data dependent hashing introduced by [AINR14], [AR15], [ALRW17] in the context of kernel density estimation. Unlike these works, however, who had no assumptions on the input data set, we show how to obtain refined bounds on the efficiency of near neighbor search under density constraints imposed by assumptions on KDE value as a function of the kernel. This turns out to be significantly more challenging: while in approximate near neighbor search, as in [ALRW17], it essentially suffices to track the size of the dataset in recursive

[3]A version of the results based on data-independent hashing presented here is published in the fourth author's Ph.D. thesis [Sym19].

iterations of locality sensitive hashing and partitioning into spheres, in the case of density constrained range search problems arising from KDE one must keep track of the distribution of points across different distance scales in the hash buckets, i.e. track evolution of functions as opposed to numbers. This leads to a natural linear programming relaxation that bounds the performance of our algorithm that forms the core of our analysis[4]. Our ultimate result for the Gaussian kernel is:

Theorem 2. *For Gaussian kernel K, any data set of points P and any $\epsilon, \mu^* \in (0,1)$, using Algorithm 1 for preprocessing and Algorithm 2 for the query procedure, one can approximate $\mu^* := K(P, \mathbf{q})$ (see Definition 5) up to $(1 \pm \epsilon)$ multiplicative factor, in time $\widetilde{O}(\mu^{-0.173-o(1)}/\epsilon^2)$. The space complexity of the algorithm is also bounded by*

$$\min\left\{O(n \cdot \mu^{-(0.173+o(1))}/\epsilon^2), O\left(\mu^{-(1+c+o(1))}/\epsilon^2\right)\right\},$$

for $c = 10^{-3}$.[5]

The proof of Theorem 2 is given in the full version of this paper.

Our techniques extend to other kernels – the extensions are presented in the full version of this paper.

B. Related Work

For $d \gg 1$, KDE was studied extensively in the 2000's with the works of [GM01], [GM03], [YDGD03], [LMG06], [RLMG09] that employed hierarchical space partitions (e.g. kd-trees, cover-trees) to obtain sub-linear query time for datasets with low intrinsic dimensionality [KR02]. Nevertheless, until recently [CS17], in the regime of $d = \Omega(\log n)$ and under worst case assumptions, the best known algorithm was simple random sampling that for constant $\delta > 0$ requires $O(\min\{1/\epsilon^2\mu, n\})$ evaluations of the kernel function to provably approximate the density at any query point q.

[CS17] revisited the problem and introduced a technique, called Hashing-Based-Estimators (HBE), to implement low-variance Importance Sampling (IS) efficiently for any query through Locality Sensitive Hashing (LSH). For the Gaussian $f(r) = e^{-r^2}$, Exponential $f(r) = e^{-r}$, and t-Student kernels $f(r) = (1 + r^t)^{-1}$ the authors gave the first sub-linear algorithms that require $O(\min\{1/\epsilon^2\sqrt{\mu}, n\})$ kernel evaluations. Using ideas from Harmonic Analysis, the technique was later extended in [CS19], to apply to more general kernels resulting in the first data structures that require $O(\min\{1/\epsilon^2\sqrt{\mu}, n\})$ kernel evaluations to approximate the density for log-convex kernels $e^{\phi(\langle x,y\rangle)}$. Furthermore, under the Orthogonal Vectors Conjecture it was shown that there does not exist a data structure that solves the KDE problem under the Gaussian kernel in time $n^{1-o(1)}/\mu^{o(1)}$ and space $n^{2-o(1)}/\mu^{o(1)}$.

The work most closely related to ours is that of [BCIS18]. [BCIS18] introduced a technique, called Spherical Integration, that uses black-box calls to c-ANN data structures (constructed on sub-sampled versions of the data set) to sample points from "spherical annuli" (r, cr) around the query, for all annuli that had non-negligible contribution to the density of the query. For kernels with polynomial tails of degree t, their approach required $\widetilde{O}(c^{5t})$ calls to such data-structures (without counting the query time required for each such call) to estimate the density. Unfortunately, this approach turns out to be constraining due to its reliance on **black-box** c-ANN calls, and in particular only applies to polynomial kernels. Our techniques in this paper recover the result of [BCIS18] up to $\mu^{-o(1)}$ factors as a special case (see Section III). Furthermore, the $\mu^{-o(1)}$ factor loss that we incur is only due to the fact that we are using the powerful Euclidean LSH family in order to achieve strong bounds for kernels that exhibit fast decay (e.g., Gaussian, exponential and others) using the same algorithm. For polynomial kernels the dependence on μ in our approach can be reduced to polylogarithmic in $1/\mu$ by using an easier hash family (e.g., the hash family of [DIIM04]; see [Sym19, Chapter 10] for details).

Scalable approaches to KDE and Applications: Recent works [SRB$^+$19], [BIW19] also address scalability issues of the original approach of [CS17]. [SRB$^+$19] designed a more efficient adaptive procedure that can be used along with Euclidean LSH [DIIM04] to solve KDE for a variety of power-exponential kernels, most prominently the Gaussian. Their algorithm is the first practical algorithm for Gaussian KDE with worst case guarantees that improve upon random sampling in high dimensions. Experiments in real-world data sets show [SRB$^+$19] that the method of [CS17], yields practical improvements for many real world datasets. [BIW19] introduced a way to sparsify hash tables and showed that in order to estimate densities $\mu^* \geq \tau \geq \frac{1}{n}$ one can reduce the space usage of the data structures [SRB$^+$19] from $O(1/\tau^{3/2}\epsilon^2)$ to $O(1/\tau\epsilon^2)$. The authors also evaluated their approach on real world data for the Exponential $e^{-\|x-y\|_2}$ and Laplace $e^{-\|x-y\|_1}$ kernels showing improvements compared to [CS17] and uniform random sampling. A related approach of Locality Sensitive Samplers [SS17] has also been applied to obtain practical procedures in the contexts of Outlier detection [LS18], Gradient Estimation [CXS19] and Clustering [LS19]. Finally, [WCN18] uses similar ideas to address the problem of approximate range counting on the unit sphere.

Core-sets and Kernel sketching: The problem of KDE is phrased in terms of guarantees for any single query

[4]The actual optimal evolution is described by an integral equation involving the log density of the kernel function and collision probabilities of LSH on the Euclidean sphere, but we do not make the limiting claim formal here since the ultimate integral equation appears to not have a closed form solution, and hence would not be useful for analysis purposes.

[5]This c can be set to any small constant that one desires. For our setting of parameters $c = 10^{-3}$.

$\mathbf{q} \in \mathbb{R}^d$. A related problem is that of Core-sets for kernels [Phi13], where the goal is to find a (small) set $S \subset P$ such that the kernel density estimate on P is close to the one on S. After recent flurry of research efforts [PT18a], [PT18b] has resulted in near optimal [PT18b] unweighted $|S| = O(\sqrt{d \log(1/\epsilon)}/\epsilon)$ and optimal [KL19] weighted core-sets $|S| = O(\sqrt{d}/\epsilon)$ for positive definite kernels. Somewhat related to this problem is the problem of oblivious sub-space embeddings for polynomial kernels [ANW14], [PP13], [AKM+17], [AKK+20].

C. Outline

In this (short) version of the paper, we only present the data independent LSH-based data structure and its analysis. The rest of the discussions are available in the full version of the paper. Preliminary definitions and results are presented in Section II. In Section III, we present our data-independent result for Gaussian KDE and state a general version of our result for other decreasing kernels. We present our data structure based on Data-Dependent LSH for KDE and its analysis in the full version of this paper.

II. Preliminaries

We let $\mu^* \in (0, 1]$ denote the kernel density of a dataset P in $\mathbb{R}^d$ at point $\mathbf{q} \in \mathbb{R}^d$:

$$\mu^* = K(P, \mathbf{q}) := \frac{1}{|P|} \sum_{\mathbf{p} \in P} K(\mathbf{p}, \mathbf{q}).$$

In the rest of the paper, we use the notation μ^* defined as $\mu^* := K(P, \mathbf{q})$, and μ is a quantity that satisfies $\mu^* \leq \mu \leq 4\mu^*$.[6]

A. Basic notation

Throughout the paper we assume that the points lie in a d-dimensional Euclidean space, $\mathbb{R}^d$. We let $\mathcal{S}^{d-1}$ denote the set of points on the unit radius sphere around the origin in $\mathbb{R}^d$. Also, for any $o \in \mathbb{R}^d$ and $R > 0$, we let $\mathcal{S}^{d-1}(o, R)$ to be the set of points on the sphere centered at o and radius R, and for any point $\mathbf{q} \in \mathbb{R}^d \setminus \{o\}$, the projection of $\mathbf{q}$ onto $\mathcal{S}^{d-1}(o, R)$ is defined as the closest point in $\mathcal{S}^{d-1}(o, R)$ to $\mathbf{q}$. For any pair of points $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, we let $||\mathbf{u} - \mathbf{v}||$ to be the Euclidean distance of $\mathbf{u}$ and $\mathbf{v}$.

For any integer J we define $[J] := \{1, 2, \ldots, J\}$. For ease of notation in the rest of the paper, we let $\exp_\mu(a) := \left(\frac{1}{\mu}\right)^a$ and (abusing notation somewhat) let $\exp_2(a) = 2^a$ for any $a \in \mathbb{R}$. We also assume that $d = \widetilde{O}(1)$ (see the full version of this paper for a detailed discussion).

[6]We have replaced μ^* with μ in the abstract for the ease of notation in the abstract.

III. Kernel Density Estimation Using Andoni-Indyk LSH

In this section, we present an algorithm for estimating KDE, using the Andoni-Indyk LSH framework. In order to state the main result of this section for general kernels, we need to define a few notions first. Thus, we state the main result for Gaussian kernel in the following theorem, and then state the general result, Theorem 10, after presenting the necessary definitions.

Theorem 3. *Given a kernel $K(\mathbf{p}, \mathbf{q}) := e^{-a||\mathbf{p}-\mathbf{q}||_2^2}$ for any $a > 0$ and a data set of points P, using Algorithm 1 for preprocessing and Algorithm 2 for the query procedure, one can approximate $\mu^* := K(P, \mathbf{q})$ (see Definition 5) up to $(1 \pm \epsilon)$ multiplicative factor, in time $\widetilde{O}\left(\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}\right)$, for any query point $\mathbf{q}$. Additionally, the space consumption of the data structure is*

$$\min\left\{\epsilon^{-2} n \left(\frac{1}{\mu^*}\right)^{0.25+o(1)}, \epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{1+o(1)}\right\}.$$

Throughout this section, we refer to Andoni-Indyk LSH's main result stated in the following lemma.

Lemma 4 ([AI06])**.** *Let $\mathbf{p}$ and $\mathbf{q}$ be any pair of points in $\mathbb{R}^d$. Then, for any fixed $r > 0$, there exists a hash family $\mathcal{H}$ such that, if $p_{\text{near}} := p_1(r) := \Pr_{h \sim \mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q}) \mid ||\mathbf{p} - \mathbf{q}|| \leq r]$ and $p_{\text{far}} := p_2(r, c) := \Pr_{h \sim \mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q}) \mid ||\mathbf{p} - \mathbf{q}|| \geq cr]$ for any $c \geq 1$, then*

$$\rho := \frac{\log 1/p_{\text{near}}}{\log 1/p_{\text{far}}} \leq \frac{1}{c^2} + O\left(\frac{\log t}{t^{1/2}}\right),$$

for some t, where $p_{\text{near}} \geq e^{-O(\sqrt{t})}$ and each evaluation takes $dt^{O(t)}$ time.

Remark 1. From now on, we use $t = \log^{2/3} n$, which results in $n^{o(1)}$ evaluation time and $\rho = \frac{1}{c^2} + o(1)$. In that case, note that if $c = O\left(\log^{1/7} n\right)$, then

$$\frac{1}{\frac{1}{c^2} + O\left(\frac{\log t}{t^{1/2}}\right)} = c^2(1 - o(1)).$$

Definition 5. For a query $\mathbf{q}$, and dataset $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$, we define

$$\mu^* := K(P, \mathbf{q}) := \frac{1}{|P|} \sum_{\mathbf{p} \in P} K(\mathbf{p}, \mathbf{q})$$

where for any $\mathbf{p} \in P$, $K(\mathbf{p}, \mathbf{q})$ is a monotone decreasing function of $||\mathbf{q} - \mathbf{p}||$. Also, we define

$$w_i := K(\mathbf{p}_i, \mathbf{q}).$$

From now on, we assume that μ is a quantity such that

$$\mu^* \leq \mu \tag{2}$$

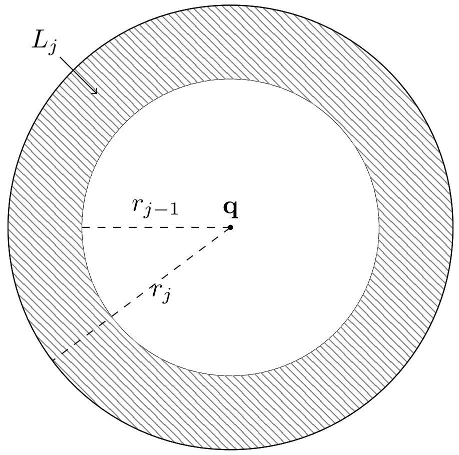

Figure 1: Illustration of definition of r_j's based on L_j's.

We also use variable $J := \left\lceil \log_2 \frac{1}{\mu} \right\rceil$.

Definition 6 (Geometric weight levels)**.** For any $j \in [J]$

$$L_j := \left\{ \mathbf{p}_i \in P : w_i \in \left(2^{-j}, 2^{-j+1}\right] \right\}.$$

This implies corresponding distance levels (see Figure 1), which we define as follows

$$\forall j \in [J] : \; r_j := \max_{\text{s.t. } f(r) \in (2^{-j}, 2^{-j+1}]} r.$$

where $f(r) := K(\mathbf{p}, \mathbf{p}')$ for $r = ||\mathbf{p} - \mathbf{p}'||$. Also define $L_{J+1} := P \setminus \cup_{j \in [J]} L_j$.[7]

We start by stating basic bounds on collision probabilities under the Andoni-Indyk LSH functions in terms of the definition of geometric weight levels L_j (Definition 6):

Claim 7. *Assume that kernel K induces weight level sets, L_j's, and corresponding distance levels, r_j's (as per Definition 6). Also, for any query $\mathbf{q}$, any integers $i \in [J+1], j \in [J]$ such that $i > j$, let $\mathbf{p} \in L_j$ and $\mathbf{p}' \in L_i$. And assume that $\mathcal{H}$ is an Andoni-Indyk LSH family designed for near distance r_j (see Lemma 4). Then, for any integer $k \geq 1$, we have the following conditions:*

1) $\Pr_{h^* \sim \mathcal{H}^k}\left[h^*(\mathbf{p}) = h^*(\mathbf{q})\right] \geq p_{\text{near},j}^k$,
2) $\Pr_{h^* \sim \mathcal{H}^k}\left[h^*(\mathbf{p}') = h^*(\mathbf{q})\right] \leq p_{\text{near},j}^{kc^2(1-o(1))}$,

where $c := c_{i,j} := \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\}$ (see Remark 1) and $p_{\text{near},j} := p_1(r_j)$ in Lemma 4.

Proof: If $\mathbf{p} \in L_j$ by Definition 6, we have

$$||\mathbf{q} - \mathbf{p}|| \leq r_j.$$

Similarly using the fact that the kernel is decaying, for $\mathbf{p}' \in L_i$ we have

$$||\mathbf{q} - \mathbf{p}'|| \geq r_{i-1} \geq c \cdot r_j.$$

[7]One can see that $L_{J+1} = \{\mathbf{p}_i \in P : w_i \leq 2^{-J}\}$.

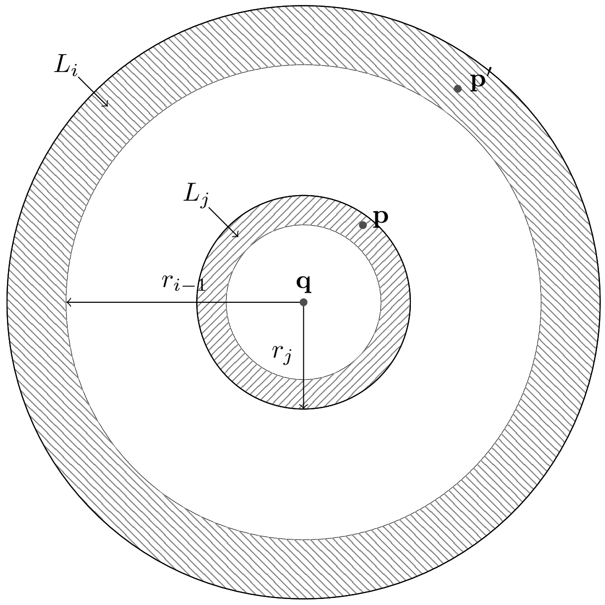

Figure 2: Illustration of r_j and r_{i-1} in terms of L_j and L_i.

So, by Lemma 4 and Remark 1 the claim holds. Figure 2 shows an instance of this claim. ■

Now, we prove an upper-bound on sizes of the geometric weight levels, i.e., L_j's (see Definition 6).

Lemma 8 (Upper bounds on sizes of geometric weight levels)**.** *For any $j \in [J]$, we have*

$$|L_j| \leq 2^j n\mu^* \leq 2^j n\mu.$$

Proof: For any $j \in [J]$ we have

$$\begin{aligned} n\mu \geq n\mu^* &= \sum_{\mathbf{p} \in P} K(\mathbf{p}, \mathbf{q}) && \text{By Definition 5} \\ &\geq \sum_{i \in [J]} \sum_{\mathbf{p} \in L_i} K(\mathbf{p}, \mathbf{q}) \\ &\geq \sum_{\mathbf{p} \in L_j} K(\mathbf{p}, \mathbf{q}) \\ &\geq |L_j| \cdot 2^{-j} \end{aligned}$$

which proves the claim. ■

Definition 9 (Cost of a kernel)**.** Suppose that a kernel K induces geometric weight levels, L_j's, and corresponding distance levels, r_j's (see Definition 6). For any $j \in [J]$ we define *cost* of kernel K for weight level L_j as

$$\text{cost}(K, j) := \exp_2\left(\max_{i = j+1, \ldots, J+1} \left\lceil \frac{i - j}{c_{i,j}^2 (1 - o(1))} \right\rceil \right),$$

where $c_{i,j} := \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\}$. Also, we define the general *cost* of a kernel K as

$$\text{cost}(K) := \max_{j \in [J]} \text{cost}(K, j).$$

Description of algorithm:: The algorithm runs in J phases. For any $j \in [J]$, in the j'th phase, we want to estimate the contribution of points in L_j to $K(P, \mathbf{q})$. We show that it suffices to have an estimation of the number of points in L_j. One can see that if we sub-sample the data set with probability $\min\{\frac{1}{2^j n\mu}, 1\}$, then in expectation we get at most $O(1)$ points from L_i for any $i \leq j$. Now, assume that a point $\mathbf{p} \in L_j$ gets sampled by sub-sampling, then we want to use Andoni-Indyk LSH to distinguish this point from other sub-sampled points, efficiently. Thus, we want to find the appropriate choice of k for the repetitions of Andoni-Indyk LSH (see Claim 7). Suppose that we call Claim 7 with some k (which we calculate later in (4)). Then we have

$$\Pr_{h^* \sim \mathcal{H}^k}[h^*(\mathbf{p}) = h^*(\mathbf{q})] \geq p^k_{\text{near},j},$$

which implies that in order to recover point $\mathbf{p}$ with high probability, we need to repeat the procedure $\widetilde{O}\left(p^{-k}_{\text{near},j}\right)$ times. Another factor that affects the run-time of the algorithm is the number of points that we need to check in order to find $\mathbf{p}$. Basically, we need to calculate the number of points that hash to the same bucket as $\mathbf{q}$ under h^*'s. For this purpose, we use the second part of Claim 7, which bounds the collision probability of far points, i.e., points such as $\mathbf{p}' \in L_i$ for any $i > j$. Intuitively, for any point $\mathbf{p}' \in L_i$ for any $i > j$, by Claim 7 we have

$$\Pr_{h^* \sim \mathcal{H}^k}[h^*(\mathbf{p}') = h^*(\mathbf{q})] \leq p^{kc^2(1-o(1))}$$

where $c := c_{i,j} := \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\}$ and $p := p_{\text{near},j}$[8]. On the other hand, by Lemma 8, for $i = j+1, \ldots, J$ we have

$$|L_i| \leq 2^i n\mu^* \leq 2^i n\mu.$$

Then, one has the following bound,

$$\begin{aligned}
&\mathbb{E}\left[|\{\mathbf{p}' \in L_i : h^*(\mathbf{p}') = h^*(\mathbf{q})\}|\right] \\
&\leq 2^i n\mu \cdot \frac{1}{2^j n\mu} \cdot p^{kc^2(1-o(1))} \\
&= 2^{i-j} \cdot p^{kc^2(1-o(1))}, \qquad (3)
\end{aligned}$$

where the first transition is due to sub-sampling and then applying LSH. Since we have $O\left(\log \frac{1}{\mu}\right)$ geometric weight levels, then the expression in (3) for the worst i, bounds the run-time up to $O\left(\log \frac{1}{\mu}\right)$ multiplicative factor. In order to optimize the run-time up to $\widetilde{O}(1)$ multiplicative factors, we need to set k such that the expression in (3) gets upper-bounded by $O(1)$ for all $i > j$. So, in summary, for any fixed $j \in [J]$, we choose k such that any weight level L_i for $i \geq j$ contributes at most $\widetilde{O}(1)$ points in expectation to the hash bucket of the query, i.e., $h^*(\mathbf{q})$. One can see that we can choose k as follows

$$k := k_j := \frac{-1}{\log p} \cdot \max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c^2_{i,j}(1-o(1))} \right\rceil. \qquad (4)$$

For sampling the points in L_{J+1}, it suffices to sample points in the data set with probability $\frac{1}{n}$ (see line 15 in Algorithm 1), since the size of the sampled data set is small and there is no need to apply LSH. One can basically scan the sub-sampled data set. Now, we present the main result of this section.

Theorem 10 (Query time)**.** *For any kernel K, the expected query-time of the algorithm is equal to $\widetilde{O}\left(\epsilon^{-2} n^{o(1)} \cdot \text{cost}(K)\right)$.*

Assuming Theorem 10, we prove Theorem 3.

Proof of Theorem 3: We first start by proving the query time bound and then we prove the space consumption of the data structure, and the guarantee over the precision of the estimator is given in Claim 13.

Proof of the query time bound:: We calculate the cost of Gaussian kernel $e^{-a\|\mathbf{x}-\mathbf{y}\|_2^2}$. First, we present the weight levels and distance levels induced by this kernel. As per Definition 5, let

$$\mu^* := K(P, \mathbf{q}) = \sum_{\mathbf{p} \in P} e^{-a\|\mathbf{p}-\mathbf{q}\|_2^2}.$$

By Definition 6, one has

$$\begin{aligned}
L_j &:= \left\{\mathbf{p}_i \in P : w_i \in \left(2^{-j}, 2^{-j+1}\right]\right\} \\
&= \left\{\mathbf{p}_i \in P : \|\mathbf{p}_i - \mathbf{q}\|_2 \in \left[\sqrt{\frac{(j-1)\ln 2}{a}}, \sqrt{\frac{j \ln 2}{a}}\right)\right\},
\end{aligned}$$

which immediately translates to $r_j := \sqrt{\frac{j \ln 2}{a}}$ for all $j \in [J]$. Also, we for all $i \in [J+1], j \in [J]$ such that $i > j$, we have

$$\begin{aligned}
c_{i,j} &:= \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\} \\
&= \min\left\{\sqrt{\frac{i-1}{j}}, \log^{1/7} n\right\}
\end{aligned}$$

At this point, one can check that

$$\max_{j \in [J]} \max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c^2_{i,j}(1-o(1))} \right\rceil = (1+o(1))\frac{1}{4} \log \frac{1}{\mu},$$

Therefore, the cost of Gaussian kernel is

$$\text{cost}(K) = \left(\frac{1}{\mu}\right)^{(1+o(1))\frac{1}{4}}.$$

Now, invoking Theorem 10, the statement of the claim about the query time holds.

[8]The indices are dropped for $c_{i,j}$ and $p_{\text{near},j}$ for ease of notation.

Algorithm 1 Preprocessing

```
1: procedure PREPROCESS($(P, \epsilon)$)
2:                                                    ▷ $P$ represents the set of data points
3:                                                    ▷ $\epsilon$ represents the precision of estimation
4:     $K_1 \leftarrow \frac{C \log n}{\epsilon^2} \cdot \mu^{-o(1)}$    ▷ $C$ is a universal constant
5:     $J \leftarrow \left\lceil \log \frac{1}{\mu} \right\rceil$    ▷ We use geometric weight levels with base 2, see Definition 6
6:     for $a = 1, 2, \ldots, K_1$ do    ▷ $O(\log n/\epsilon^2)$ independent repetitions
7:         for $j = 1, 2, \ldots, J$ do    ▷ $J = \left\lceil \log \frac{1}{\mu} \right\rceil$ geometric weight levels
8:             $K_2 \leftarrow 100 \log n \cdot p_{\text{near},j}^{-k_j}$
9:                                                    ▷ See Claim 7 and (4) for definition of $p_{\text{near},j}$ and $k_j$
10:            $p_{\text{sampling}} \leftarrow \min\{\frac{1}{2^j n\mu}, 1\}$
11:            $\widetilde{P} \leftarrow$ sample each element in $P$ with probability $p_{\text{sampling}}$.
12:            for $\ell = 1, 2, \ldots, K_2$ do
13:                Draw a hash function from hash family $\mathcal{H}^{k_j}$ as per Claim 7 and call it $H_{a,j,\ell}$
14:                Run $H_{a,j,\ell}$ on $\widetilde{P}$ and store non-empty buckets
15:        $\widetilde{P}_a \leftarrow$ sample each element in $P$ with probability $\frac{1}{n}$
16:        Store $\widetilde{P}_a$    ▷ Set $\widetilde{P}_a$ will be used to recover points beyond $L_{J+1}$
```

Proof of the space bound:: First, since the query time is bounded by $\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}$, then the number of hash functions used is also bounded by the same quantity. This implies that the expected size of the space needed to store the data structure prepared by the preprocessing algorithm is $\epsilon^{-2} n \left(\frac{1}{\mu^*}\right)^{0.25+o(1)}$, since for each hash function we are hashing at most n points (number of points in the dataset).

For the other bound, we need to consider the effect of sub-sampling the data set. Fix $j \in [J]$. In the phase when we are preparing the data structure to recover points from L_j, we sub-sample the data set with probability $\min\{\frac{1}{2^j n\mu}, 1\}$, and then we apply $\widetilde{O}\left(p_{\text{near},j}^{-k_j}\right)$ hash functions to this sub-sampled data set. Since

$$k_j = \frac{-1}{\log p} \cdot \max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil,$$

by (4), where $p = p_{\text{near},j}$, we have

$$p_{\text{near},j}^{-k_j} = \exp_2\left(\max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil - j \right). \quad (5)$$

At the same time, the expected size of the sampled dataset is bounded by $n \cdot \min\{\frac{1}{2^j n\mu}, 1\} \leq \frac{1}{\mu} \cdot 2^{-j}$. Putting this together with the equation above, we get that the expected size of the dataset constructed for level L_j is upper bounded by

$$\frac{1}{\mu} \exp_2\left(\max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil - j \right). \quad (6)$$

Now for every $i = j+1, \ldots, J$ such that $c_{i,j} = \sqrt{\frac{i-1}{j}}$ one has

$$\max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil - j =$$
$$\max_{i=j+1,\ldots,J+1} \left\lceil j \cdot \frac{i-j}{(i-1)(1-o(1))} \right\rceil - j \leq o(J),$$

and for the other values of i we have $\max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{\log^{1/7} n(1-o(1))} \right\rceil - j \leq o(J)$ as well. Putting this together with (6) and multiplying by $J = O(\log(1/\mu)) = \mu^{-o(1)}$ to account for the number of choices $j \in [J]$, we get the second bound for the expected size of the data structure $\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{1+o(1)}$.

Proof of the precision of the estimator:: First, we prove the following claim, which guarantees high success probability for recovery procedure.

Claim 11 (Lower bound on probability of recovering a sampled point)**.** *Suppose that we invoke Algorithm 1 with (P, ϵ). Suppose that in line 11 of Algorithm 1, when $k = k^*$ and $j = j^*$, we sample some point $\mathbf{p} \in L_{j^*}$. We claim that with probability at least $1 - \frac{1}{n^{10}}$, there exists $\ell^* \in [K_2]$ such that $H_{k^*,j^*,\ell^*}(\mathbf{p}) = H_{k^*,j^*,\ell^*}(\mathbf{q})$.*

Proof: By Claim 7 we have

$$\Pr_{h^* \sim \mathcal{H}^k}\left[h^*(\mathbf{p}) = h^*(\mathbf{q})\right] \geq p_{\text{near},j}^{k_j}.$$

Now note that we repeat this process for $K_2 = 100 \log n \cdot p_{\text{near},j}^{-k_j}$ times. So any point $\mathbf{p}$ which is sampled from band L_{j^*} is recovered in at least one of the repetitions of phase $j = j^*$, with high probability. ■

Now, we argue that the estimators are unbiased (up to small inverse polynomial factors)

Algorithm 2 Query procedure

```
1:  procedure QUERY($P, \mathbf{q}, \epsilon, \mu$)
2:                                            ▷ $P$ represents the set of data points
3:                                            ▷ $\epsilon$ represents the precision of estimation
4:      $K_1 \leftarrow \frac{C \log n}{\epsilon^2} \cdot \mu^{-o(1)}$           ▷ $C$ is a universal constant
5:      $J \leftarrow \left\lceil \log \frac{1}{\mu} \right\rceil$    ▷ We use geometric weight levels with base 2, see Definition 6
6:      for $a = 1, 2, \ldots, K_1$ do                 ▷ $O(\log n/\epsilon^2)$ independent repetitions
7:          for $j = 1, 2, \ldots, J$ do               ▷ $J = \left\lceil \log \frac{1}{\mu} \right\rceil$ geometric weight levels
8:              $K_2 \leftarrow 100 \log n \cdot p_{\text{near},j}^{-k_j}$    ▷ See Claim 7 and (4) for definition of $p_{\text{near},j}$ and $k_j$
9:              for $\ell = 1, 2, \ldots, K_2$ do
10:                 Scan $H_{a,j,\ell}(q)$ and recover points in $L_j$
11:         Recover points from $L_{J+1}$ in the sub-sampled dataset, $\widetilde{P}_a$.
12:         $S \leftarrow$ set of all recovered points in this iteration
13:         for $\mathbf{p}_i \in S$ do
14:             $w_i \leftarrow K(\mathbf{p}_i, \mathbf{q})$
15:             if $\mathbf{p}_i \in L_j$ for some $j \in [J]$ then
16:                 $p_i \leftarrow \min\{\frac{1}{2^j n\mu}, 1\}$,
17:             else if $\mathbf{p}_i \in P \setminus \cup_{j\in[J]} L_j$ then
18:                 $p_i \leftarrow \frac{1}{n}$
19:         $Z_a \leftarrow \sum_{\mathbf{p}_i \in S} \frac{w_i}{p_i}$
```

Claim 12 (Unbiasedness of the estimator)**.** *For every $\mu^* \in (0,1)$, every $\mu \geq \mu^*$, every $\epsilon \in (\mu^{10}, 1)$, every $\mathbf{q} \in \mathbb{R}^d$, estimator Z_a for any $a \in [K_1]$ constructed in* QUERY$(P, \mathbf{q}, \epsilon, \mu)$ *(Algorithm 2) satisfies the following:*

$$(1 - n^{-9})n\mu^* \leq \mathbb{E}[Z_a] \leq n\mu^*$$

Proof: Let $\mathcal{E}$ be the event that every sampled point is recovered and let $Z := Z_a$ (see line 19 in Algorithm 2). By Claim 11 and union bound, we have

$$\Pr[\mathcal{E}] \geq 1 - n^{-9}$$

We have that $\mathbb{E}[Z] = \sum_{i=1}^{n} \frac{\mathbb{E}[\chi_i]}{p_i} w_i$ with $(1 - n^{-9})p_i \leq \mathbb{E}[\chi_i] \leq p_i$, where we now define $\chi_i = 1$ if point $\mathbf{p}_i$ is sampled and recovered in the phase corresponding to its weight level, and $\chi_i = 0$ otherwise. Thus

$$(1 - n^{-9})n\mu^* \leq \mathbb{E}[Z] \leq n\mu^*. \tag{7}$$

■

Remark 2. We proved that our estimator is unbiased[9] for **any choice** of $\mu \geq \mu^*$. Therefore if $\mu \geq 4\mu^*$, by Markov's inequality the estimator outputs a value larger than μ at most with probability $1/4$. We perform $O(\log n)$ independent estimates, and conclude that μ is higher than μ^* if the median of the estimated values is below μ. This estimate is correct with high probability, which suffices to ensure that we find a value of μ that satisfies $\mu/4 < \mu^* \leq \mu$ with high probability by starting with some $\mu = n^{-\Theta(1)}$ (since our analysis assumes $\mu^* = n^{-\Theta(1)}$) and repeatedly halving our estimate (the number of times that we need to halve the estimate is $O(\log n)$ assuming that μ is lower bounded by a polynomial in n, an assumption that we make).

[9]Up to some small inverse polynomial error.

Claim 13 (Variance bounds)**.** *For every $\mu^* \in (0,1)$, every $\epsilon \in (\mu^{10}, 1)$, every $\mathbf{q} \in \mathbb{R}^d$, using estimators Z_a, for $a \in [K_1]$ constructed in* QUERY*$(P, \mathbf{q}, \epsilon, \mu)$ (Algorithm 2), where $\mu/4 \leq \mu^* \leq \mu$, one can output a $(1 \pm \epsilon)$-factor approximation to μ^*.*

Proof: By Claim 12 and noting that $Z \leq n^2\mu^*$, where the worst case (equality) happens when all the points are sampled and all of them are recovered in the phase of their weight levels. Therefore,

$$\mathbb{E}\left[Z|\mathcal{E}\right] \cdot \Pr[\mathcal{E}] + n^2\mu^*(1 - \Pr[\mathcal{E}]) \geq \mathbb{E}[Z].$$

Also, since Z is a non-negative random variable, we have

$$\mathbb{E}\left[Z|\mathcal{E}\right] \leq \frac{\mathbb{E}\left[Z\right]}{\Pr[\mathcal{E}]} \leq \frac{n\mu^*}{\Pr[\mathcal{E}]} = n\mu^*(1 + o(1/n^9))$$

Then, we have

$$\begin{aligned}\mathbb{E}[Z^2] &= \mathbb{E}\left[\left(\sum_{\mathbf{p}_i\in P}\chi_i\frac{w_i}{p_i}\right)^2\right]\\ &= \sum_{i\neq j}\mathbb{E}\left[\chi_i\chi_j\frac{w_iw_j}{p_ip_j}\right]+\sum_{i\in[n]}\mathbb{E}\left[\chi_i\frac{w_i^2}{p_i^2}\right]\\ &\le \sum_{i\neq j}w_iw_j+\sum_{i\in[n]}\frac{w_i^2}{p_i}\mathbb{I}[p_i=1]+\sum_{i\in[n]}\frac{w_i^2}{p_i}\mathbb{I}[p_i\neq 1]\\ &\le \left(\sum_i w_i\right)^2+\sum_{i\in[n]}w_i^2+\max_i\left\{\frac{w_i}{p_i}\mathbb{I}[p_i\neq 1]\right\}\sum_{i\in[n]}w_i\\ &\le 2n^2(\mu^*)^2+\max_{j\in[J],\mathbf{p}_i\in L_j}\{w_i2^{j+1}\}n\mu\cdot n\mu^*\\ &\le 4n^2\mu^2,\end{aligned}$$

where the last transition is due to the fact that $\mu^*\le\mu$. And

$$\mathbb{E}[Z^2|\mathcal{E}]\le\frac{\mathbb{E}[Z^2]}{\Pr[\mathcal{E}]}\le n^2\mu^{2-o(1)}(1+o(1/n^9))$$

Now, since $\mu\le 4\mu^*$, in order to get a $(1\pm\epsilon)$-factor approximation to μ^*, with high probability, it suffices to repeat the whole process $K_1=\frac{C\log n}{\epsilon^2}\cdot\mu^{-o(1)}$ times, where C is a universal constant.

Suppose we repeat this process m times and $\bar{Z}$ be the empirical mean, then:

$$\begin{aligned}&\Pr[|\bar{Z}-\mu^*|\ge\epsilon n\mu^*]\\ &\le\Pr[|\bar{Z}-\mathbb{E}[Z]|\ge\epsilon\mu^*-|\mathbb{E}[Z]-n\mu^*|]\\ &\le\Pr[|\bar{Z}-\mathbb{E}[Z]|\ge(\epsilon-n^{-9})n\mu^*]\\ &\le\frac{\mathbb{E}[\bar{Z}^2]}{(\epsilon-n^{-9})^2(n^2\mu^*)^2}\\ &\le\frac{1}{m}\frac{16n^2(\mu^*)^2}{(\epsilon-n^{-9})^2(n^2\mu^*)^2}\end{aligned}$$

Thus by picking $m=O(\frac{1}{\epsilon^2})$ and taking the median of $O(\log(1/\delta))$ such means we get a $(1\pm\epsilon)$-approximation with probability at least $1-\delta$ per query. ■

All in all, we proved the expected query time bound, the expected space consumption and the precision guarantee in the statement of the theorem. □

Now, we calculate the cost of kernel for t-student kernel.

t-student kernel ($\frac{1}{1+\|\mathbf{x}-\mathbf{y}\|_2^t}$):: We directly calculate distance levels induced by this kernel as follows

$$r_j=\sqrt[t]{2^j-1}$$

which implies that for all $i\in[J+1], j\in[J]$ such that $i>j$,

$$\begin{aligned}c_{i,j}&:=\min\left\{\frac{r_{i-1}}{r_j},\log^{1/7}n\right\}\\ &=\min\left\{\sqrt[t]{\frac{2^{i-1}-1}{2^j-1}},\log^{1/7}n\right\}.\end{aligned}$$

Now, one can check that

$$\max_{j\in[J]}\max_{i=j+1,\dots,J+1}\left\lceil\frac{i-j}{c_{i,j}^2(1-o(1))}\right\rceil=\frac{\log\frac{1}{\mu}}{\log^{2/7}n}(1+o(1)).$$

Thus, we have

$$\mathrm{cost}(K)=\mu^{-o(1)}.$$

We note that this matches the result of [BCIS18] up to the difference between $\mu^{-o(1)}$ and $\log(1/\mu)$ terms. The $\mu^{-o(1)}$ dependence comes from the fact that we used the LSH of [AI06], and the dependence can be improved to $\log(1/\mu)$ by using the hash family of [DIIM04], for instance.

Exponential Kernel ($e^{-\|x-y\|_2}$): The distance levels induced by the kernel are given by $r_j=j\log(2)$ for $j\in[J]$. Hence, we get that $c_{ij}=\min\left\{\frac{r_{i-1}}{r_j},\log^{1/7}n\right\}=\min\{\frac{i-1}{j},\log^{1/7}n\}$. If $i>j\log^{1/7}n+1$ then the cost is increasing in $i>0$ becomes:

$$\begin{aligned}\mathrm{cost}(K,j)&=\exp_2\left(\frac{J+1-j}{\log^{2/7}n}\right)\\ &\le\exp_2\left(\frac{J}{\log^{2/7}n}\right)\\ &=\mu^{-o(1)}.\end{aligned}$$

Thus, for the rest we will assume that $i\le j\log^{1/7}n+1$, and we need to find the maximum over j of

$$(1+o(1))\max_{i=j+1,\dots,J+1}\left\lceil\frac{j^2((i-1)-(j-1))}{(i-1)^2}\right\rceil$$

Setting $x=i-1$ and $A=j-1$, we optimize the function $\frac{(x-A)}{x^2}$ for $x\ge A+1$. We get that the optimal value is attained for $i^*(j)=\max\{\min\{2j-1,J+1\},j+1\}$. We distinguish three cases:

1) $j=1$: then $i^*=2$ and we get $\mathrm{cost}(K,1)=\mu^{-o(1)}$
2) $j>\frac{J+2}{2}$: then the maximum over i is $\frac{j^2(J+1-j)}{J^2}(1+o(1))$, and the optimal choice of j is $j^*=\frac{2(J+1)}{3}$. We thus get
$$\max_{j>\frac{J+2}{2}}\{\mathrm{cost}(K,j)\}=\mu^{-(1+o(1))\frac{4}{27}}$$
3) $j\le\frac{J+2}{2}$: then the maximum over i is $\frac{j^2}{4(j-1)}(1+o(1))$ and the optimal choice for j is $j^*=\frac{J+2}{2}$. We thus get
$$\max_{j\le\frac{J+2}{2}}\{\mathrm{cost}(K,j)\}=\mu^{-(1+o(1))\frac{1}{8}}.$$

Overall, the worst-case cost is attained for $i^*=J$ and $j^*=\frac{2J}{3}$ and yields

$$\mathrm{cost}(K)=\mu^{-(1+o(1))\frac{4}{27}}.$$

Proof of Theorem 10: One should note that the query time of our approach depends on the number of times that we hash the query and the number of points that we check, i.e., the number of points that collide with the query. First, we analyze the number of points colliding with the query. We Fix $j \in [J]$, so, we want to estimate the contribution of points in L_i to $K(P, \mathbf{q})$. We consider 3 cases:

Case 1. $i \leq j$:: Note that we have $|L_i| \leq 2^i n\mu$ and note that in j'th phase, we sample the data set with rate $\min\{\frac{1}{2^j n\mu}, 1\}$. Thus, we have at most $1 = O(1)$ sampled points from L_i in expectation.

Case 2. $i = j+1, \ldots, J$:: Again, note that by Lemma 8, $|L_i| \leq 2^i n\mu$, and the sampling rate is $\min\{\frac{1}{2^j n\mu}, 1\}$. Thus, we have at most 2^{i-j} sampled points from L_i in expectation. Now, we need to analyze the effect of LSH. Note that we choose LSH function such that the near distance is r_j (see Claim 7). Also, note that as per (4), we use

$$k := k_j := \frac{-1}{\log p_{\text{near},j}} \cdot \max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil.$$

as the number of concatenations. Now, we have the following collision probability for $\mathbf{p} \in L_i$ using Claim 7

$$\Pr_{h^* \in \mathcal{H}^k}[h^*(\mathbf{p}) = h^*(\mathbf{q})] \leq p^{kc^2(1-o(1))},$$

where $c := c_{i,j} := \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\}$ and $p := p_{\text{near},j}$ for ease of notation. This implies that the expected number of points from weight level L_i in the query hash bucket is at most

$$2^{i-j} \cdot p^{kc^2(1-o(1))} = \widetilde{O}(1)$$

by the choice of k.

Case 3. points in L_{J+1}:: We know that we have n points, so after sub-sampling, we have at most $\frac{1}{2^j\mu}$ points from this range, remaining in expectation. For any $\mathbf{p} \in L_{J+1}$, note that $||\mathbf{p} - \mathbf{q}|| \geq c \cdot r_j$ for $c := c_{J+1,j} := \min\left\{\frac{r_J}{r_j}, \log^{1/7} n\right\}$. Then,

$$\Pr_{h^* \in \mathcal{H}^k}[h^*(\mathbf{p}) = h^*(\mathbf{q})] \leq p^{kc^2(1-o(1))},$$

which implies that the expected number of points form this range in the query hash bucket is at most

$$\frac{1}{2^j\mu} \cdot p^{kc^2(1-o(1))} = 2^{J-j} \cdot p^{kc^2(1-o(1))} = \widetilde{O}(1)$$

by the choice of k_j.

All in all, we prove that each weight level L_i for $i \in [J+1]$ contribute at most $\widetilde{O}(1)$ points to the hash bucket of query. Now, we need to prove a bound on the number of times we evaluate our hash function. One should note that by the choice of k_j in (4) we have

$$k_j = \widetilde{O}(1)$$

which basically means that we only concatenate $\widetilde{O}(1)$ LSH functions. Thus, we the evaluation time of $h^*(q)$ for any $h^* \in \mathcal{H}^k$ is $\widetilde{O}(n^{o(1)})$, by Remark 1. On the other hand, note that for recovering the points in L_{J+1} we just sub-sampled the data set with probability $\frac{1}{n}$ so in expectation we only scan 1 point. So in total, since we repeat this for all $j \in [J]$ and $J = \lceil \log \frac{1}{\mu} \rceil$, by the choice of K_1 and K_2 assigned in lines 4 and 8 of Algorithm 1, respectively, the claim holds. □

References

[ACMP15] Ery Arias-Castro, David Mason, and Bruno Pelletier. On the estimation of the gradient lines of a density and the consistency of the mean-shift algorithm. *Journal of Machine Learning Research*, 2015.

[AI06] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 459–468. IEEE Computer Society, 2006.

[AINR14] Alexandr Andoni, Piotr Indyk, Huy L. Nguyen, and Ilya P. Razenshteyn. Beyond locality-sensitive hashing. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1018–1028. SIAM, 2014.

[AKK+20] Thomas D Ahle, Michael Kapralov, Jakob BT Knudsen, Rasmus Pagh, Ameya Velingker, David P Woodruff, and Amir Zandieh. Oblivious sketching of high-degree polynomial kernels. In *SODA (to appear)*, 2020.

[AKM+17] Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh. Random fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 253–262. JMLR. org, 2017.

[ALRW17] Alexandr Andoni, Thijs Laarhoven, Ilya P. Razenshteyn, and Erik Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 47–66. SIAM, 2017.

[ANW14] Haim Avron, Huy Nguyen, and David Woodruff. Subspace embeddings for the polynomial kernel. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2258–2266. Curran Associates, Inc., 2014.

[AR15] Alexandr Andoni and Ilya P. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh*

Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015, pages 793–801. ACM, 2015.

[BCIS18] Arturs Backurs, Moses Charikar, Piotr Indyk, and Paris Siminelakis. Efficient density evaluation for smooth kernels. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 615–626. IEEE, 2018.

[BG97] R Beatson and Leslie Greengard. *A short course on fast multipole methods*, pages 1–37. Numerical Mathematics and Scientific Computation. Oxford University Press, 1997.

[BIW19] Arturs Backurs, Piotr Indyk, and Tal Wagner. Space and time efficient kernel density estimation in high dimensions. In *Advances in Neural Information Processing Systems*, 2019.

[CS17] Moses Charikar and Paris Siminelakis. Hashing-based-estimators for kernel density in high dimensions. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1032–1043. IEEE, 2017.

[CS19] Moses Charikar and Paris Siminelakis. Multi-resolution hashing for fast pairwise summations. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2019.

[CXS19] Beidi Chen, Yingchen Xu, and Anshumali Shrivastava. Lsh-sampling breaks the computation chicken-and-egg loop in adaptive stochastic gradient estimation. *arXiv preprint arXiv:1910.14162*, 2019.

[DIIM04] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[FG96] Jianqing Fan and Irene Gijbels. *Local polynomial modelling and its applications: monographs on statistics and applied probability 66*, volume 66. CRC Press, 1996.

[GB17] Edward Gan and Peter Bailis. Scalable kernel density classification via threshold-based pruning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 945–959. ACM, 2017.

[GM01] Alexander G Gray and Andrew W Moore. N-body'problems in statistical learning. In *Advances in neural information processing systems*, pages 521–527, 2001.

[GM03] Alexander G Gray and Andrew W Moore. Nonparametric density estimation: Toward computational tractability. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 203–211. SIAM, 2003.

[GPPV$^+$14] Christopher R Genovese, Marco Perone-Pacifico, Isabella Verdinelli, Larry Wasserman, et al. Nonparametric ridge estimation. *The Annals of Statistics*, 42(4):1511–1545, 2014.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 604–613. ACM, 1998.

[JKPV11] Sarang Joshi, Raj Varma Kommaraji, Jeff M Phillips, and Suresh Venkatasubramanian. Comparing distributions and shapes using the kernel distance. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 47–56. ACM, 2011.

[KL19] Zohar S. Karnin and Edo Liberty. Discrepancy, coresets, and sketches in machine learning. In Alina Beygelzimer and Daniel Hsu, editors, *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, volume 99 of *Proceedings of Machine Learning Research*, pages 1975–1993. PMLR, 2019.

[KR02] David R Karger and Matthias Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 741–750. ACM, 2002.

[LMG06] Dongryeol Lee, Andrew W Moore, and Alexander G Gray. Dual-tree fast gauss transforms. In *Advances in Neural Information Processing Systems*, pages 747–754, 2006.

[LS18] Chen Luo and Anshumali Shrivastava. Arrays of (locality-sensitive) count estimators (ace): Anomaly detection on the edge. In *Proceedings of the 2018 World Wide Web Conference*, pages 1439–1448. International World Wide Web Conferences Steering Committee, 2018.

[LS19] Chen Luo and Anshumali Shrivastava. Scaling-up split-merge mcmc with locality sensitive sampling (lss). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4464–4471, 2019.

[Phi13] Jeff M Phillips. ε-samples for kernels. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1622–1632. SIAM, 2013.

[PP13] Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–247. ACM, 2013.

[PT18a] Jeff M Phillips and Wai Ming Tai. Improved coresets for kernel density estimates. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2718–2727. SIAM, 2018.

[PT18b] Jeff M Phillips and Wai Ming Tai. Near-optimal coresets of kernel density estimates. In *34th International Symposium on Computational Geometry (SoCG 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[RLMG09] Parikshit Ram, Dongryeol Lee, William March, and Alexander G Gray. Linear-time algorithms for pairwise statistical problems. In *Advances in Neural Information Processing Systems*, pages 1527–1535, 2009.

[Rub18] Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1260–1268, 2018.

[RW06] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006.

[SRB$^+$19] Paris Siminelakis, Kexin Rong, Peter Bailis, Moses Charikar, and Philip Levis. Rehashing kernel evaluation in high dimensions. In *International Conference on Machine Learning*, pages 5789–5798, 2019.

[SS01] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[SS17] Ryan Spring and Anshumali Shrivastava. A new unbiased and efficient class of lsh-based samplers and estimators for partition function computation in log-linear models. *arXiv preprint arXiv:1703.05160*, 2017.

[STC$^+$04] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[Sym19] Paraskevas Syminelakis. *Kernel Evaluation in High Dimensions: Importance Sampling and Nearest-Neighbor Search*. PhD thesis, Stanford University, 2019.

[SZK14] Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. Generalized outlier detection with flexible kernel density estimates. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 542–550. SIAM, 2014.

[WCN18] Xian Wu, Moses Charikar, and Vishnu Natchu. Local density estimation in high dimensions. In *International Conference on Machine Learning*, pages 5293–5301, 2018.

[YDGD03] Changjiang Yang, Ramani Duraiswami, Nail A Gumerov, and Larry Davis. Improved fast gauss transform and efficient kernel density estimation. In *Proceedings of the Ninth IEEE International Conference on Computer Vision-Volume 2*, page 464. IEEE Computer Society, 2003.