Efficient Density Evaluation for Smooth Kernels

Arturs Backurs
EECS
MIT
Cambridge, MA, USA
backurs@mit.edu

Moses Charikar
Department of Computer Science
Stanford University
Stanford, CA, USA
moses@cs.stanford.edu

Piotr Indyk
EECS
MIT
Cambridge, MA, USA
indyk@mit.edu

Paris Siminelakis
Department of Electrical Engineering
Stanford University
Stanford, CA, USA
psimin@stanford.edu

***Abstract*—Given a kernel function $k(\cdot,\cdot)$ and a dataset $P \subset \mathbb{R}^d$, the *kernel density function* of P at a point $x \in \mathbb{R}^d$ is equal to $\mathrm{KDF}_P(x) := \frac{1}{|P|}\sum_{y\in P} k(x,y)$. Kernel density evaluation has numerous applications, in scientific computing, statistics, computer vision, machine learning and other fields. In all of them it is necessary to evaluate $\mathrm{KDF}_P(x)$ quickly, often for many inputs x and large point-sets P.**

In this paper we present a collection of algorithms for efficient KDF evaluation under the assumptions that the kernel k is "smooth", i.e. the value changes at most polynomially with the distance. This assumption is satisfied by several well-studied kernels, including the (generalized) t-student kernel and rational quadratic kernel. For smooth kernels, we give a data structure that, after $O(dn\log(\Phi n)/\epsilon^2)$ preprocessing, estimates $\mathrm{KDF}_P(x)$ up to a factor of $1 \pm \epsilon$ in $O(d\log(\Phi n)/\epsilon^2)$ time, where Φ is the aspect ratio. The $\log(\Phi n)$ term can be further replaced by $\log n$ under an additional decay condition on k, which is satisfied by the aforementioned examples.

We further extend the results in two ways. First, we use low-distortion embeddings to extend the results to kernels defined for spaces other than ℓ_2. The key feature of this reduction is that the distortion of the embedding affects only the running time of the algorithm, not the accuracy of the estimation. As a result, we obtain $(1+\epsilon)$-approximate estimation algorithms for kernels over other ℓ_p norms, Earth-Mover Distance, and other metric spaces. Second, for smooth kernels that are decreasing with distance, we present a general reduction from density estimation to approximate near neighbor in the underlying space. This allows us to construct algorithms for general doubling metrics, as well as alternative algorithms for ℓ_p norms and other spaces.

***Keywords*-Heavy-tailed kernels, Dimensionality Reduction, Quad-Trees, Hashing, Approximate Nearest Neighbor Search.**

I. Introduction

Kernel density evaluation is a basic computational task with many applications. Given a kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$ and a dataset $P \subset \mathbb{R}^d$, we define the *kernel density function* of P at a point $x \in \mathbb{R}^d$ as:

$$\mathrm{KDF}_P(x) := \frac{1}{|P|}\sum_{y\in P} k(x,y) \tag{1}$$

The task of computing KDF can be formulated in multiple ways:

- As a data structure problem: given P, build a data structure supporting KDF evaluation for a query x,
- In the batch setting: given two sets P and X, compute $\mathrm{KDF}_P(x)$ for all $x \in X$, or
- In the "all pairs" setting: given two sets P and X, compute
$$k(X,P) = \sum_{x\in X} \mathrm{KDF}_P(x). \tag{2}$$

The batch version of the problem has been studied extensively. In particular, the celebrated Fast Multipole Method gives an efficient approximate algorithm for this problem in low dimensions [1] and has been very influential in scientific computing. Unfortunately, the complexity of this approach scales exponentially with the dimension, while many applications require evaluating kernel densities for *high-dimensional* pointsets. This includes *kernel density estimation*, a classic tool in non-parametric statistics, where the kernel function is used to extend the empirical distribution function over a discrete set of points smoothly to the whole space[1]. This in turn yields algorithms for *mode estimation* [2], *outlier detection* [3], *density based clustering* [4] and other problems. Another class of applications stems from applying kernel methods (e.g., regression [5]) to objects described by point-clouds or distributions, by extending kernel functions to pairs of sets [6]. Computing over such *set kernels* requires many "all-pairs" evaluations as defined in Equation 2. Yet another application is kernel mean estimation using an empirical average (see e.g., [7], section 3.4.2).

A popular method for evaluating kernel densities in high dimensions involves random sampling [8]. Under the assumption that $\mathrm{KDF}_P(x) \geq \mu$ for some $\mu \in (0,1]$, sampling $\Theta(\frac{1}{\mu\epsilon^2}\log(1/\delta))$ points in P suffices to estimate the desired value up to a factor of $1\pm\epsilon$ with probability $1-\delta$. This yields a data structure whose query time is equal to the number of samples times d. The query time can be improved further: in a very recent work, a subset of the authors [9] showed that for several kernels, one can reduce the query time to $\tilde{\Theta}(d\frac{1}{\sqrt{\mu}\epsilon^2}\log(1/\delta))$. Their approach applies to multiple popular kernels, including Gaussian, exponential and t-student for constant t. However, their technique is tailored

[1]In this application, kernel functions are assumed to satisfy certain integrability conditions. We discuss this issue in Section II.

to the specific cases since it requires a hash function family with collision probability matching the kernel in a certain way. It is unclear how to generalize it to handle more general kernels. All of the aforementioned results translate to the batch setting, with the running time essentially equal to the query time bound multiplied by $|X|$.

The main disadvantage of the above high-dimensional results is that the value of μ can be arbitrarily low, leading to high query time. Unfortunately, there is evidence that one cannot obtain fast query times (independent of μ) for arbitrary kernels. Unconditional lower bounds have already been shown for core-sets [10], [11] (with almost matching upper bounds) and in a slightly more general computational model [9]. Moreover, an observation in [12] demonstrates that, in the "all-pairs" setting, approximating KDF values for kernels with exponential tails (such as Gaussian) up to a constant factor requires $n^{2-o(1)}$ time (where $n = |X| = |P|$) unless Strong Exponential Time Hypothesis fails. This holds even if the values are lower bounded by $\mu = \exp(-\log^{O(1)} n)$; the bound can be strengthened further to $1/\text{poly}(n)$ by using the recent result of [13]. This gives evidence that, for kernels that decay fast enough, there is no algorithm with a runtime of $n^{2-o(1)}/\mu^{o(1)}$. This leads to the natural question: *What class of kernels admits efficient kernel density evaluation?*

A. Our results

In this work, we show that quick kernel decay is essentially the *only* obstacle to obtaining more efficient kernel density evaluation algorithms. Specifically, we present a data structure with a logarithmic query time, independent of μ, for kernels whose value changes at most *polynomially* with the distance. Formally:

Definition 1 ((L,t)-smooth function). *We call a function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ (L,t)-smooth if for all $p, p', q \in \mathbb{R}^d$ with $p \neq q$ and $p' \neq q$ we have*

$$\max\left(\frac{k(p,q)}{k(p',q)}, \frac{k(p',q)}{k(p,q)}\right) \leq L \max\left(\frac{\|p-q\|}{\|p'-q\|}, \frac{\|p'-q\|}{\|p-q\|}\right)^t.$$

This general class of kernels includes several well-studied functions [14]–[16]. For example:

- Rational Quadratic kernel $k(p,q) = \frac{1}{(1+\|p-q\|^2)^\beta}$ is $(1,2\beta)$-smooth.[2]
- t-Student kernel $k(p,q) = \frac{1}{1+\|p-q\|^t}$ is $(1,t)$-smooth.

Smooth kernels are frequently used in machine learning and statistics, often yielding results similar to or better than the (more popular) Gaussian kernel (see e.g., [17], Supplementary Material section). For such kernels, we give a data structure with the query time of

$$O(dL2^{O(t)}\log(\Phi n)/\epsilon^2 \log(1/\delta)),$$

where Φ is the aspect ratio of the data and query points (Section II). Furthermore, if we assume a natural kernel decay property (satisfied by the above kernels, see full version), the query time becomes $O(dL2^{O(t)}\log(n)/\epsilon^2\log(1/\delta))$, removing the dependence on the aspect ratio. This improves over the algorithm of [9] for a wide class of kernel functions. In particular, we achieve a query time of

$$O(d\log(n)/\epsilon^2\log(1/\delta))$$

for the Cauchy kernel, and more generally for the Rational Quadratic kernel with $\beta = O(1)$. The above results assume that the distance $\|p-q\|$ is induced by the Euclidean norm.

General metrics: Given a set $\mathcal{M}$, we can consider a slightly more general version of our setting, where the kernel is smooth with respect to a general metric $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}_+$. For example, we can consider the following generalization of the t-student kernel $k(p,q) = \frac{1}{1+d(p,q)^t}$, that is $(1,t)$-smooth with respect to $(\mathcal{M}, d)$. For metrics that are constant power of ℓ_2, as powering changes the smoothness parameters of the kernel only by constant factors, we can immediately apply our algorithm. Using this fact, we can also apply the algorithm for ℓ_q norms with $1 \leq q < 2$ by embedding them into ℓ_2^2 [18]. More generally, it can be extended to other metrics that can be embedded into ℓ_2^2 with some low distortion D. Perhaps surprisingly, this only multiplies the *running time* of the algorithm (by a factor of $D^{O(t)}$), while the approximation factor remains equal to $1+\epsilon$. In particular, this yields an algorithm for the ℓ_∞ norm with the running time multiplied by a factor $d^{O(t)}$; in the specific case of t-student kernel, the running time becomes $O(d^{O(t)}\log(n)/\epsilon^2\log(1/\delta))$. We complement the latter algorithm by showing that, assuming Strong Exponential Time Hypothesis, there is no KDF evaluation algorithm for the t-student kernel under ℓ_∞ norm with a running time of $d^{O(1)}2^{t/10}n^{o(1)}$. We defer the presentation and proofs of these results to the full version of the paper.

Black-Box ANN: For kernels that are decreasing with the distance, recent approaches to speed up kernel evaluations in high dimensions utilize either directly [19] or indirectly [9] methods for *Approximate Nearest Neighbor Search* (ANNS). We investigate this relationship further, by showing that for smooth decreasing[3] kernels the complexity of kernel evaluation is in a very general way closely related to that of ANNS in the underlying space. In fact, we show something stronger:

Theorem 2 (Informal). *Given* black box *access to an oracle that for a finite set S, query q, and numbers $r > 0, c > 1$, is guaranteed to return an* arbitrary *(even adversarial) point from the set $B(q,cr) \cap S$ when a point in $B(q,r) \cap S$ exists, we show that making order $O\left(L^3\left(c^t/\epsilon\right)^5\log^4(\Phi n)\right)$*

[2]For $\beta = 1$, it is often referred to as *Cauchy kernel*.

[3]Actually our requirements are slightly less strict see Definition 17

Table I
KDF EVALUATION FOR DOUBLING METRICS WITH DIMENSION DDIM. $\rho \in (0, 0.5)$ IS ANY CONSTANT.

Technique	Query time (QT)
L_∞ [22], [23]	$\tilde{O}(\epsilon^{-5}(\frac{\log(O(\mathrm{ddim})+\log\log n)}{\rho})^{O(t)}(L\phi)^3)$
ANNS [24]	$O(\epsilon^{-5}\ (\frac{\mathrm{t}}{\mathrm{ddim}})^{O(\mathrm{ddim})}(L\phi)^3)$

queries are sufficient to approximate the kernel density function within accuracy $(1 \pm \epsilon)$ *with probability* $1 - 1/\mathrm{poly}(n)$.

Due to the black-box nature of our result, by utilizing known constructions of ANNS data-structure for different metric spaces we can effortlessly solve the corresponding kernel evaluation problems for smooth-decreasing kernels under such metrics (Table II). Of particular interest, is the fact that we can take advantage of *intrinsic low dimensional structure* in the given instance to obtain kernel evaluation algorithms with complexity dependent on the doubling dimension of the pointset instead on the ambient dimension (Table I). This is relevant as many of the efforts [20], [21] aiming to exploit the intrinsic dimensionality of a dataset for kernel evaluation employ series expansions of kernels whose complexity depends on the ambient dimension.

B. Previous Work

There are two general ideas to improve upon the linear time algorithm and on uniform random sampling [8] for kernel density evaluations:

Well Conditioned Spatial Partitions: Given a query q, the first idea is to produce a (query dependent) partition of the points in P into sets $P_1, \dots, P_\phi$ and separately approximate $\mathrm{KDF}_{P_\ell}(x)$ for all $\ell \in [\phi]$. Within this paradigm, there are two different approaches:

(a) If the diameter of P_ℓ is small enough compared to $d(q, P_\ell)$, then by selecting a single point from P_ℓ and using *series expansions* around it, one can estimate $\mathrm{KDF}_{P_\ell}(q)$. This is the basis of the Fast Multipole Method [1] and related approaches [25]–[29].

(b) If the values of the kernel between points in P_ℓ and q are of similar magnitude, as captured by the ratio between the smallest and larger kernel value (*condition number*), then a few *random samples* suffice to estimate the density in each part [30].

In both cases, in order to keep the number of parts small and to be able to efficiently implement the above primitives (series expansion or random sampling) *upon query time*, hierarchical space decompositions are employed that are typically variants of quad-trees [31], [32]. In *high dimensions*, though, such partitions are *very costly* to compute (exponential in d) and hence this approach so far has only been applied in cases where there is a latent low-dimensional structure [33].

Importance Sampling through Hashing: The second idea is for a query q, instead of performing uniform random sampling, to assign greater probability to points $p \in P$ with larger kernel values $k(q, p)$. This idea is quite simple in principle, but difficult to implement efficiently since such a sampling distribution must be query dependent and can vary significantly for different queries. Very recently, a subset of the authors [9] introduced a technique that uses Locality Sensitive Hashing schemes to construct a sampling distribution with low variance. The crux of the technique is the ability to design a hash function with collision probability that scales like $\Theta(\sqrt{k(q,p)})$. This particular scaling stems from the fact that we need our sampling distribution to be *relatively balanced* along all relevant distance scales and is responsible for the $\frac{1}{\sqrt{\mu}}$ dependence in the number of samples required. Nevertheless, it reduces the high level problem of kernel density evaluation to a low level problem of designing a *hashing scheme* with a *specific collision probability*.

C. Our Techniques

At a high level our work exploits *smoothness of the kernels* to address the natural barriers to both the aforementioned methods (curse of dimensionality and μ-dependence) and overcomes them with new ideas.

Projected quad-trees: Our first estimator involves building *quad-trees on randomly projected* data points, and using partitions induced by the trees to produce an *unbiased estimator* for $\mathrm{KDF}_P(q)$. Unlike earlier papers that combine spatial structures with random projections [34], [35], we use the trees as *estimators*, as opposed to data structures solving (say) the nearest neighbor problem. Specifically, once provided with a query q, we show that the data structure can produce a hierarchical partition of the points depending on the distance from q. We then use this partition to create an unbiased estimator.

Averaged Condition Number: Viewing the projected quad-tree as inducing a query dependent random partition of points, the obvious approach is to analyze the resulting random sampling estimator through the *expected condition number*. This quantity involves computing the expectation of the supremum of the condition number of different parts for this random partition – a global quantity that will likely give a loose bound. We show in a generic way, that for estimators that sample a *random point from each part of a random partition*, a different (local) quantity bounds the variance of such estimators and is much easier to analyze as it results from viewing the random partition as a *hash function* with a *(query-dependent) collision probability*.

For smooth kernels, if we select the dimension $d' = O(t)$ of the projected quad-tree appropriately we show that we can obtain an $O(L2^{O(t)})$ bound on the variance of such estimators and can evaluate them in time $O(dL2^{O(t)} \log(\Phi n/\delta)/\epsilon^2)$. Thus, by viewing projected quad-trees as hash functions and analyzing our estimators through the Averaged Condition Number, our approach combines the *analytical advantages* inspired by Hashing-based Importance Sampling with the *computational benefits* of low-dimensional hierarchical decompositions.

In the abstract setting where we are given an arbitrary (even adversarial) ANNS oracle, we take a different approach: we posit the *existence of a partition* with the desirable properties (bounded condition number), and design tools to generate approximately uniform random samples from parts of the partition given a black-box access to an (even adversarial) ANNS oracle.

Spherical Integration: The partition we consider is defined by concentric exponentially increasing annuli (with constant $c > 1$) around the query so that due to smoothness the condition number in each part is bounded by Lc^t. Our main conceptual contribution is to show how to use the ANNS oracle to construct an approximately-uniform random sampling oracle for each annulus.

First we show that if we have a slightly stronger oracle, that always returns a point from $S \cap B(q,r)$ if it is the unique point in $S \cap B(q,cr)$, and additionally can detect when $|S \cap B(q,r)| > 1$, then by *sub-sampling P for various density levels* and calling the oracle with independently sub-sampled versions of P, we can produce an oracle that returns an *approximately uniform random sample* from $B(q,r) \cap P$. As long as we have an upper bound $|B(q,rc) \cap P| \leq B_{rc}$, the probability that this happens is proportional to $|B(q,r) \cap P|/B_{rc}$. We then show how to construct such an oracle form an arbitrary ANNS oracle using an overhead factor of $O(\log |S|)$. In this way, we can then sample points from $A_r := (B(q,r) \cap P) \setminus B(q,r/c)$ with a rate of $\frac{|B(q,r)\cap P|}{B_{rc}} \frac{|A_r|}{|B(q,r)\cap P|}$.

There are two complications with the above construction (a) the sampling rate can be very small depending on $|A_r|/B_{rc}$, and (b) we always need to maintain an upper bound on $|B(q,r) \cap P| \leq B_r$ even if we don't see points from A_r. When $|B(q,r) \cap P|/B_{cr}$ is larger than a threshold then we may estimate $|B(q,r) \cap P|$ and also have a lower bound on the rate that get a random point from $B(q,r) \cap P$. Further, if $|A_r|/|B(q,r) \cap P|$ is also larger than a threshold then we can also get an estimate of $|A_r|$ and have a lower bound on the rate which we observe samples. The complications arise when either of the two conditions do not hold.

We resolve the above issues by showing that if the rate is small enough then the contribution to the kernel density compared to an outer or inner annulus must be small (this is where we use monotonicity) and by designing an algorithm that *sequential tests the hypotheses* that the respective ratios $\frac{|B(q,r)\cap P|}{B_{rc}}$ or $\frac{|A_r|}{|B(q,r)\cap P|}$ are small enough. The sequential testing of hypothesis, that introduces dependences between our estimates for different annuli, along with the non-uniformity of the sampling oracle, makes the overall analysis of this procedure significantly complicated.

D. Outline of the paper

In the next section we introduce some definitions and basic tools that we require. In Section III, we describe the projected quad-tree data structure and prove our main result. Lastly, we formally define our requirements on ANNS oracles, and present our second main result in Section IV.

II. Preliminaries

Metric spaces: A *metric space* $(\mathcal{M}, d)$ is a set $\mathcal{M}$ equipped with a metric $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}_+$. For a set $\mathcal{M}$ when there is no uncertainty about the metric $d_{\mathcal{M}}$, we will simply refer to $\mathcal{M}$ as a metric space, and implicitly refer to $(\mathcal{M}, d_{\mathcal{M}})$. For a set $P \subseteq \mathcal{M}$ and any potential query $q \in \mathcal{M}$, let $r_0 \leq \min_{x \neq y \in P \cup \{q\}} d_{\mathcal{M}}(x,y)$ and $R \geq \max_{x,y \in P \cup \{q\}} d_{\mathcal{M}}(x,y)$ be bounds on the minimum and maximum distance between points in $P \cup \{q\}$. Given a constant $c > 1$, we define $\Phi = \frac{R}{r_0}$ to be the *aspect ratio* and $\phi_c := \lceil \log_c(\Phi) \rceil$ to be the *log-aspect ratio*. When the constant c is clear from the context we will drop the dependence on c. For a point $q \in \mathcal{M}$, we define the *ball of radius r* in P around q as

$$B_P(q,r) := \{p \in P | d_{\mathcal{M}}(q,p) \leq r\} \tag{3}$$

When there is no uncertainty about which set P we refer to, we will suppress P from the definition and simply write $B(q,r)$. For $c > 1$ and $r > 0$ we also define the *(r,c)-annulus* in P centered at $q \in \mathcal{M}$ as $A_c(q,r) := B(q,r) \setminus B(q, \frac{r}{c})$ and typically suppress c when it is clear from the context. Finally, for any set $C \subset \mathcal{M}$ and $q \in \mathcal{M}$ let $d(q,C) := \min_{p \in C} d(q,p)$.

Kernels: Consider a (kernel) function $k : \mathcal{M} \times \mathcal{M} \to \mathbb{R}_+$. For a set $P \subseteq \mathcal{M}$ and point $q \in \mathcal{M}$, let $\mathrm{KDF}_P(q) := \frac{1}{|P|} \sum_{p \in P} k(q,p)$. We will often denote $\mathrm{KDF}_P(q)$ simply by $\mathrm{KDF}(q)$ suppressing P when it is clear from the context. In this paper we focus on *smooth kernels*, as defined in the introduction. Note that kernel functions do not need to be non-increasing or non-decreasing in the distance to be smooth. In general, the definition allows for somewhat unpredictable behavior as a function of the distance, which leads to the (logarithmic) dependence of the query time on the aspect ratio.

Integrability and Density Estimation: In the specific context of non-parametric *kernel density estimation*, typically kernels refer to integrable translation invariant functions $k(x,y) = k(x-y)$.

Definition 3. *A function $k : \mathbb{R}^d \to \mathbb{R}_+$ with $\int_{\mathbb{R}^d} |k| < \infty$ and $\int_{\mathbb{R}^d} k(x)dx = 1$ is called a* kernel.

Such functions are used in non-parametric density estimation [36] over $\mathbb{R}^d$. Given a bandwidth parameter $\sigma > 0$ it is easy to see that for the kernel $k_\sigma(x-y) := \frac{1}{\sigma^d} k(\frac{x-y}{\sigma})$, KDF_P defines a valid probability density function. Smooth kernels, like the t-student, kernel are not integrable per se unless $t > d$. Nevertheless, given an smooth and decreasing kernel (that are typically employed for density estimation) we can produce an integrable kernel by truncating after $d(x,y) \geq \frac{1}{\tau^{1/t}}$ and normalizing so that the integral is equal to one. This introduces an additive error of $O(\tau)$ in the estimates which can be made arbitrarily small.

A. Geometric tools

We state the following bound (e.g. [37]) on the volume of a d-dimensional Euclidean ball.

Proposition 4. *The volume $V_d(r)$ of a d-dimensional Euclidean ball of radius $r > 0$ satisfies $V_d(r) \leq \frac{1}{e}\left(\frac{2\pi e}{d}\right)^{\frac{d}{2}} \cdot r^d$.*

We also require a *dimensionality reduction* lemma that is of similar nature to [38, Lemma 2.2].

Lemma 5. *Let $d > d' \in \mathbb{N}$ and $A_{ij} \overset{iid}{\sim} N(0, \frac{1}{d'})$ for $ij \in [d'] \times [d]$. For all $T \geq \sqrt{17}$, $x \neq 0$, we have*

$$\max\left\{\mathbb{P}\big[\|Ax\|_2 \leq \frac{1}{T}\|x\|_2\big],\ \mathbb{P}\big[\|Ax\|_2 \geq T\|x\|_2\big]\right\} \leq T^{-\frac{2d'}{3}}$$

Finally, in order to extend our results from ℓ_2 norm to ℓ_q norms with $1 \leq q < 2$ we use the a variant of Theorem 116 from [18] to get *low-distortion embeddings* in ℓ_2^2.

Theorem 6. *Let $2 \geq q \geq 1$, $R > 0$ and $\Phi \geq 2$ be real numbers. There exists a mapping $f : \mathbb{R}^d \to \mathbb{R}^{O(d\log(\Phi d/\varepsilon)/\varepsilon)}$ such that for any $x, y \in \mathbb{R}^d$ with $R \leq \|x-y\|_q \leq \Phi R$, we have*

$$(1-\varepsilon)\|x-y\|_q^q \leq \|f(x) - f(y)\|_2^2 \leq (1+\varepsilon)\|x-y\|_q^q.$$

B. Partition Estimators

Let $w_1, \ldots, w_n$ be a set of n positive weights (e.g $w_i = k(q, x_i)$ for $i \in [n]$). We are interested in providing a $(1\pm\epsilon)$-multiplicative approximation to the sum of weights $\mu = \sum_{i=1}^n w_i$. Given any partition $\mathcal{P}$ of $[n]$ and set of weights we may define the following randomized estimator for μ.

Definition 7. *Given a partition $\mathcal{P}$ of $[n]$ with parts $P_1, \ldots, P_\phi$, for $\ell \in [\phi]$ let $I_\ell \sim P_\ell$ be a uniform random point from P_ℓ and set $N_\ell = |P_\ell|$. We define the* partition estimator $Z_{\mathcal{P}} = \sum_{\ell=1}^{\phi} N_\ell w_{I_\ell}$.

Definition 8. *Given a set $S \subseteq [n]$ we define the condition number of S as $\kappa_S := \sup_{i,j\in S} \frac{w_i}{w_j}$. The* condition number *of a partition $\mathcal{P} = \{P_1, \ldots, P_\phi\}$ is given by $\kappa(\mathcal{P}) := \max_{\ell\in[\phi]} \kappa_{P_\ell}$.*

For a fixed partition $\mathcal{P}$, it is not hard to see that the estimator is unbiased $\mathbb{E}_I[Z_{\mathcal{P}}] = \mu$ and $\mathbb{E}_I[Z_{\mathcal{P}}^2] \leq 2\kappa(\mathcal{P})\cdot\mu^2$. If the partition $\mathcal{P}$ is random, a simple calculation shows that $\mathbb{E}_{\mathcal{P},I}[Z_{\mathcal{P}}^2] \leq 2\,\mathbb{E}_{\mathcal{P}}[\kappa(\mathcal{P})]\cdot\mu^2$. This upper bound depends on the *expected condition number*:

$$\mathbb{E}[\kappa(\mathcal{P})] := \mathbb{E}\left[\sup_{\ell\in[\phi]}\sup_{i,i'\in P_\ell}\left\{\frac{w_i}{w_{i'}}\right\}\right], \tag{4}$$

a global quantity where the supremum is taken inside the expectation and thus complicated to compute. Next we introduce an alternative viewpoint of random partitions through hashing that allow us to obtain a better upper bound that depends only on local events. Let $h \sim \mathcal{H}$ be a hash function selected from some distribution $\mathcal{H}$. The hash function h induces a (random) partition $\mathcal{P}_h$ through the equivalence $i \sim i' \Leftrightarrow h(i) = h(i')$.

Definition 9. *For a distribution $\mathcal{H}$ over hash functions, define the* averaged condition number *as:*

$$\alpha(\mathcal{H}) := \sup_{i,i'\in P}\ \mathop{\mathbb{E}}_{h\sim\mathcal{H}}\left[\frac{w_i}{w_{i'}}\mathbb{I}[h(i) = h(i')]\right] \tag{5}$$

This quantity allows us to obtain a tighter analysis of the variance of *partition estimators*.

Lemma 10 (Variance Bound)**.** *Let $h \sim \mathcal{H}$, then $\mathbb{E}[Z_{\mathcal{P}_h}] = \mu$ and $\mathbb{E}[Z_{\mathcal{P}_h}^2] \leq 2\alpha(\mathcal{H})\mu^2 \leq 2\,\mathbb{E}[\kappa(\mathcal{P}_h)]\mu^2$.*

Proof: We start by analyzing the second moment:

$$\begin{aligned}
\mathop{\mathbb{E}}_{I,h}[Z_{\mathcal{P}_h}^2] &= \mathop{\mathbb{E}}_{h}\left[\sum_{\ell,\ell'=1}^{\phi} N_\ell N_{\ell'} \mathop{\mathbb{E}}_{I}[w_{I_\ell} w_{I_{\ell'}}]\right] \\
&= \mathop{\mathbb{E}}_{h}\left[\sum_{\ell\neq\ell'}\left(\sum_{i\in P_\ell} w_i \sum_{i'\in P_{\ell'}} w_{i'}\right)\right] + \mathop{\mathbb{E}}_{h}\left[\sum_{\ell=1}^{\phi} N_\ell \sum_{i\in P_\ell} w_i^2\right] \\
&\leq \mu^2 + \mathop{\mathbb{E}}_{h}\left[\sum_{\ell=1}^{\phi}\sum_{i'=1}^{n}\sum_{i=1}^{n}\mathbb{I}[h(i) = h(i') = \ell] w_i^2\right] \\
&= \mu^2 + \mathop{\mathbb{E}}_{h}\left[\sum_{i'=1}^{n}\sum_{i=1}^{n}\left(\sum_{\ell=1}^{\phi}\mathbb{I}[h(i) = h(i') = \ell]\right) w_i^2\right] \\
&= \mu^2 + \mathop{\mathbb{E}}_{h}\left[\sum_{i,i'=1}^{n}\mathbb{I}[h(i) = h(i')] w_i^2\right] \\
&= \mu^2 + \sum_{i,i'=1}^{n}\mathop{\mathbb{E}}_{h}\left[\mathbb{I}[h(i) = h(i')]\frac{w_i}{w_{i'}}\right] w_i w_{i'} \\
&\leq \mu^2 + \alpha(\mathcal{H})\sum_{i,i'=1}^{n} w_i w_{i'} \\
&\leq (2\alpha(\mathcal{H}))\cdot\mu^2
\end{aligned}$$

We complete the proof by showing:

$$\begin{aligned}
\mathbb{E}[\kappa(\mathcal{P}_h)] &= \mathbb{E}\left[\sup_{\ell\in[\phi]}\sup_{i,i'\in P}\left\{\frac{w_i}{w_{i'}}\mathbb{I}[i,i'\in P_\ell]\right\}\right] \\
&= \mathbb{E}\left[\sup_{i,i'\in P}\left\{\frac{w_i}{w_{i'}}\sum_{\ell\in[\phi]}\mathbb{I}[i,i'\in P_\ell]\right\}\right] \\
&\geq \sup_{i,i'\in P}\mathbb{E}\left[\frac{w_i}{w_{i'}}\mathbb{I}[h(i) = h(i')]\right] = \alpha(\mathcal{H})
\end{aligned}$$

■

III. Scale-sensitive Hashing using Projected Quad-trees

Our goal is to construct a hashing scheme such that given a query q it approximately partitions the points in P depending on the scale of their distance away from the query q. The principle behind our data-structure is quite simple: randomly projecting into $d' < d$ dimension tends to preserve pairwise distances, then by pretending the distances are preserved we use a quad-tree data structure to create such a partition in low-dimensions. If the dimension d' is picked appropriately, we show that we can bound the *averaged condition number* of the resulting *partition estimator* for smooth kernels. This leads to the following theorem.

Theorem 11. *There exists a data structure that given a data set $P \subset \mathbb{R}^d$, using $O(dL2^{O(t)} \log(\frac{\Phi n}{\delta})\frac{1}{\epsilon^2} \cdot n)$ space and pre-processing time, for any (L, t)-nice kernel and a query $q \in \mathbb{R}^d$, estimates $\mathrm{KDF}_P(q)$ with accuracy $(1 \pm \epsilon)$ in time $O(dL2^{O(t)} \log(\frac{\Phi n}{\delta})\frac{1}{\epsilon^2})$ with probability at least $1 - 1/\mathrm{poly}(n) - \delta$.*

A. Projected quad-trees

To build the above data structure we start by reducing the dimensionality of the pointset to $d' = O(t)$. Let A be a $d' \times d$ random projection matrix, with i.i.d. entries chosen from the normal distribution with variance $1/d'$. We map every point $p \in P$ to Ap to get the projected data set P'.

Given a number $\phi \in \mathbb{Z}$, we build a quad-tree in the d'-dimensional space for the resulting pointset. First, we choose an integer $h_+ \in \mathbb{Z}$ such that the resulting point set P' as well as Aq for all future queries q are contained inside the d'-dimensional cell $[-2^{h_+}/2, 2^{h_+}/2]^{d'}$. For any $h = h_+ - 1, h_+ - 2, \ldots, h_+ - \phi$ we subdivide every cell with side-length 2^{h+1} that contains at least two projected points from P into $2^{d'}$ non-overlapping cells with side-length 2^h, keeping only cells that contain at least one point.

In this way we build a tree where each node corresponds to a certain cell and store for each cell the number of points, the indices as well as the points themselves. Let $h_0 = h_-$ and define $h_\ell = h_0 + \ell$ for $\ell \in [\phi]$. For any $\ell = 1, \ldots, \phi$, we define *level* ℓ to consist of all cells with side length 2^{h_ℓ}. In what follows $W_\ell := 2^{h_\ell}\sqrt{d'}$ for all $\ell \in [\phi]$.

B. Querying Algorithm

Having a quad-tree as above, for a given query $q \in \mathbb{R}^d$ we use it to define a *partition estimator* as follows:

1) **Input:** $P \subset \mathbb{R}^d$, quad-tree $\mathcal{T}$ on $P' \subset \mathbb{R}^{d'}$, $q \in \mathbb{R}^d$.
2) **Initialization:** initialize two arrays $I \in [n]^\phi$ and $N \in \mathbb{N}^\phi$ where for all $\ell \in [\phi]$, I_ℓ and N_ℓ respectively denote the current random index and number of points from the ℓ-th part of the partition $\mathcal{P}_\mathcal{T}$ defined by the tree for the given query. Let $\ell = \phi$ and $q' = Aq$ be the projection of query q. Create an empty queue Q, enqueue the root of $\mathcal{T}$, and set $M_\ell = 1$.

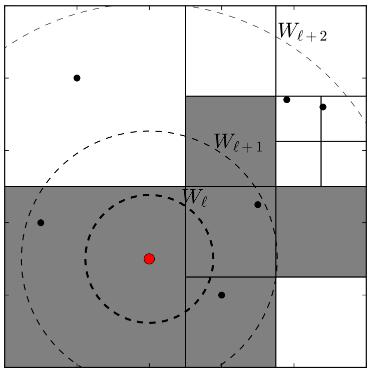

Figure 1. Illustration of the query algorithm for a specific query (red). The gray cells correspond to the cells of the tree that were used to sample at level ℓ and the white cells to the ones used to sample at level $\ell + 1$.

3) **Main loop:** as long as Q is not empty and $\ell \geq 1$:
   a) for $i = 1$ to M_ℓ:
      i) remove the top element C from Q.
      ii) if C contains only one point $p' \in P'$ then let $s \in [\phi]$ be such that $W_s \leq d(q', p') \leq 4 \cdot W_s$ then $\text{ReservoirSampling}_{I,N}(s, C)$.
      iii) else if $d(q', C) \geq W_\ell$, then $\text{ReservoirSampling}_{I,N}(\ell, C)$
      iv) else add the children of C to the queue.
   b) Set $\ell \leftarrow \ell - 1$ and M_ℓ equal to the current length of the queue.
4) **Output:** $Z_{\mathcal{P}_\mathcal{T}} = \frac{1}{n}\sum_{\ell \in [\phi]} N_\ell \cdot k(q, x_{I_\ell})$

The procedure $\text{ReservoirSampling}_{I,N}(\ell, C)$ when called for a level $\ell \in [\phi]$ and a cell C, with probability $\frac{|C|}{N_\ell + |C|}$ updates I_ℓ to be a random index from C, and updates $N_\ell \leftarrow N_\ell + |C|$. See Figure 1 for an illustration of the querying algorithm for a projected point set P'.

C. Analysis

For all $\ell \in [\phi]$, let P_ℓ be the union of indices of points belonging in cells for which the procedure ReservoirSampling was called with first argument equal to ℓ. It is a well-known fact that I_ℓ will be uniformly distributed in P_ℓ when $Z_{\mathcal{P}_\mathcal{T}}$ is computed. Thus, $Z_{\mathcal{P}_\mathcal{T}}$ is a valid *partition estimator*.

1) Running Time: The running time of our algorithm is at most proportional to the number of nodes that we add in the queue times the time it takes to sample a random point from each such cell. We assume that we can sample a random point in constant time, thus we only care about the number of cells that we process.

Lemma 12. *Let C' be a child of any cell C that we expanded at level $\ell \in [\phi]$, then $d(q', C') < \frac{3}{2}W_\ell$ and for all $x \in C'$ we have $d(q', x) \leq 2W_\ell = 4W_{\ell-1}$.*

Proof: Since we expanded C at level ℓ, it must be the case that $d(q', C) < 2^{h_\ell}\sqrt{d'}$. Hence, there exists a child C'' (even if it was deleted because it was empty) such that $d(q', C'') < 2^{h_\ell}\sqrt{d'}$. Then:

$$d(q', C') \leq d(q', C'') + \sup_{x \in C''} \min_{y \in C'} d(x, y) \tag{6}$$

$$\leq 2^{h_\ell}\sqrt{d'} + 2^{h_\ell - 1}\sqrt{d'} \leq \frac{3}{2} 2^{h_\ell}\sqrt{d'}. \tag{7}$$

Observing that the diameter of C' is at most $W_{\ell-1} = \frac{W_\ell}{2}$ completes the proof. ■

By the above lemma all cells that we process at level ℓ are contained in a ball of diameter at most $4W_\ell$ around the query and by construction all such cells are disjoint and have volume $2^{h_\ell d'}$. By Proposition 4, the number of cells that we process at level ℓ are at most:

$$\frac{V_{d'}(4W_\ell)}{2^{d' h_\ell}} \leq \frac{1}{e}\left(\frac{\sqrt{2\pi e}}{\sqrt{d'}} 4\sqrt{d'} \frac{2^{h_\ell}}{2^{h_\ell}}\right)^{d'} \leq 2^{4.05 d'} \tag{8}$$

Summing up over all levels we require time at most $O(d'\phi 2^{4.05d'} + \phi d)$ to produce the output.

2) Collision probability: We define the hash function $h_q : P \to [\phi]$ by setting $h_q(x_i) = \ell$ for all $i \in P_\ell$ and $\ell \in [\phi]$. We denote the resulting distribution (randomness in the partition is due to random matrix A) over hash functions as $h_q \sim \text{kdTree}_{d',P}(q)$.

Lemma 13. *Given $t > 0$, let $d' = \max\{\lceil 3t \rceil, 2\}$ and $h_q \sim \text{kdTree}_{d',P}(q)$, then*

$$\mathbb{P}[h_q(x) = h_q(y)] \leq 8 \cdot 2^{6.09t} \cdot \left(\min\left\{\frac{\|x-q\|}{\|y-q\|}, \frac{\|y-q\|}{\|x-q\|}\right\}\right)^t$$

Proof: We first observe that for any point $x \in P$ to have $h_q(x) = \ell$, it is necessary due to Lemma 12 that $W_\ell \leq d(x', q') \leq 4W_\ell$. Without loss of generality assume that $r = \|x - q\| \leq \|y - q\| = r'$ and let as before $W_\ell = 2^{h_\ell}\sqrt{d'}$. Define, $\mathcal{L}_{xy} = \left\{\ell \in [\phi] \,\middle|\, \max\left\{\frac{W_\ell}{r}, \frac{r'}{4W_\ell}\right\} < \sqrt{17}\right\}$ to be the set of levels for which we cannot apply the bounds from Lemma 5 to analyze the probability of the event $\mathbb{P}[h_q(x) = h_q(y) = \ell]$. Using this notation we may write:

$$\mathbb{P}[h_q(x) = h_q(y)] \leq \mathbb{I}[\mathcal{L}_{xy} \neq \emptyset] + (1 - \mathbb{I}[\mathcal{L}_{xy} \neq \emptyset]) \cdot \sum_{\ell=1}^{\phi} \min\{\mathbb{P}[h_q(x) = \ell], \mathbb{P}[h_q(x) = \ell]\} \tag{9}$$

A necessary condition for $\mathcal{L}_{xy} \neq \emptyset$ is that $r' < 68r$. We distinguish two cases

- Case 1: if $r' < 68r$ then we use

$$\mathbb{P}[h_q(x) = h_q(y)] \leq 1 < \left(68\frac{r}{r'}\right)^{\frac{d'}{3}}. \tag{10}$$

- Case 2: if $r' \geq 68r$ then $\min\left\{\frac{r}{W_\ell}, \frac{4W_\ell}{r'}\right\} \leq \sqrt{4\frac{r}{r'}} \leq \frac{1}{\sqrt{17}}$ and consequently for $z \in \{x, y\}$

$$\mathbb{P}[h_q(z) = \ell] \leq \min\left\{\frac{r}{W_\ell}, \frac{4W_\ell}{r'}\right\}^{\frac{2d'}{3}} \tag{11}$$

Substituting this bound in 9 gives

$$\mathbb{P}[h_q(x) = h_q(y)] \leq \sum_{\ell=1}^{\Phi} \min\left\{\frac{r}{W_\ell}, \frac{4W_\ell}{r'}\right\}^{\frac{2d'}{3}} \tag{12}$$

$$\leq 4 \frac{1}{1 - 1/17^{-\frac{d'}{6}}} \left(4\frac{r}{r'}\right)^{\frac{d'}{3}} \tag{13}$$

where the last inequality follows from the fact that the sum can be written as a sum of two geometrically decreasing series.

Setting, $d' = \max\{\lceil 3t \rceil, 2\}$ we arrive at the bound $\mathbb{P}[h_q(x) = h_q(y)] \leq \max\{8 \cdot 4^t, 68^t\}\left(\frac{r}{r'}\right)^t$. ■

3) Averaged Condition Number: We next use our bounds on the colision probability to bound the condition number of the resulting partition estimator $Z_{\mathcal{P}_T}$. Let $h_q \sim \mathcal{H} = \text{kdTree}_{d',P}(q)$, and k an (L, t)-nice kernel, by Lemma 13

$$\alpha(\mathcal{H}) = \sup_{x,y \in P}\left\{\frac{k(y,q)}{k(x,q)} \mathbb{P}[h_q(x) = h_q(y)]\right\}$$

$$\leq \sup_{x,y \in P}\left\{L\left(\max\left\{\frac{\|x-q\|}{\|y-q\|}, \frac{\|y-q\|}{\|x-q\|}\right\}\right)^t \cdot 8 \cdot 2^{6.09t}\left(\min\left\{\frac{\|x-q\|}{\|y-q\|}, \frac{\|y-q\|}{\|x-q\|}\right\}\right)^t\right\}$$

$$\leq 8L \cdot 2^{6.09t}$$

D. Proof of Theorem 1

Our data structure is formed by sampling m projected quad-trees for our dataset P. Each one takes time $O(n \times d' \times d)$ (random projection) plus $O(d'\phi 2^{d'} n)$ since, a point belongs to at most $O(\phi)$ cells in the tree, and it can be part of a split only once at every level, producing possibly $2^{d'}$ cells that some of them might be deleted in the process. Hence, overall we require space/time $O\left((\phi 2^{d'} + d)d'mn\right)$.

Assuming that the aspect ratio of the point set and possible queries is bounded by Φ, then if we set $\phi = O(\log_2(d\Phi nm))$ by union bound and standard Gaussian concentration we know that for all of the m projected data sets the *log-aspect ratio* is less than ϕ with probability $1 - \chi = 1 - 1/\text{poly}(n)$.

Let $\alpha = O(L 2^{6.09t})$ be the averaged condition number of the quad-tree. Using Lemma 10 and the median-of-means technique, we see that if we pick $m = O(\frac{1}{\epsilon^2} L 2^{6.09t} \log(\frac{1}{\delta}))$, we can estimate any query with accuracy $(1 \pm \epsilon)$ with probability at least $1 - \chi - \delta$. The query time is bounded by $O([d'\phi 2^{4.05d'} + \phi d] \cdot m)$. Substituting $d' = \max\{\lceil 3t \rceil, 2\}$ the resulting query time is $O(L\frac{1}{\epsilon^2}(d' 2^{O(t)} + d 2^{O(t)}) \log(\frac{n\Phi}{\delta}))$ and the space usage $O(L\frac{1}{\epsilon^2}(d' 2^{O(t)} + d 2^{O(t)}) \log(\frac{n\Phi}{\delta}) \cdot n)$.

IV. Spherical Integration

The Projected Quadtree algorithm was based on randomized partitions that behave favorably under random sampling. In this Section we take a different more abstract approach: (i) we postulate the *existence of a partition* with the desirable properties, and (ii) design tools to *simulate access* to that partition given black-box access to (r,c)-ANN oracle.

A. Spherical Partitions

In order to define a partition that would allow efficient estimation of the kernel density, we rely on *smoothness*. For fixed $c>1$, let $r_\ell := r_0 c^\ell$ for $\ell = 0,\dots,\phi$ be a geometrically increasing sequence of radii. For a point $q \in \mathcal{M}$, define $A_\ell = A_\ell(q) := A_c(q, r_\ell)$ to be the corresponding sequence of *spherical annuli*, where we omit q and c if they are clear from the context. Finally, for any such sequence of radii, set P, and point $q \in \mathcal{M}$, we define the *ℓ-th annulus density* as $\rho_\ell := \rho_\ell(q,P) = \frac{|A(q,r_\ell)|}{|P|}$ and $\mu_\ell := \mathrm{KDF}_{A_\ell}(q)$ to be the corresponding kernel density. Using the notation above we have the following basic identity:

$$\mathrm{KDF}_P(q) = \sum_{\ell=0}^{\phi} \frac{|A_\ell|}{|P|}\mathrm{KDF}_{A_\ell}(q) = \sum_{\ell=0}^{\phi} \rho_\ell \mu_\ell \tag{14}$$

For convenience we define $Z_\ell := \rho_\ell\mu_\ell$ to be the *contribution of the ℓ-th annulus* to $\mathrm{KDF}(q)$. Observe that (L,t)-smoothness implies that within each annulus the condition number $\max_{p,p' \in A_\ell}\{k(q,p)/k(q,p')\}$ is bounded by Lc^t. Hence, if we had access to a random sample from A_ℓ we could estimate both ρ_ℓ as well as μ_ℓ using $O(\epsilon^{-2}Lc^t\log(\frac{1}{\chi}))$ random samples for all $\ell \in [\phi]$. We call this abstract approach *Spherical Integration*. The main obstacle to carrying it out, is efficiently implementing the primitive that returns a *random sample from a specified annulus* $A_\ell = A(q, r_\ell)$.

B. Random Sampling Oracles via Black-box ANN

Our main conceptual contribution is that if one is given *black-box* access to an *c-approximate nearest neighbor search* oracle, it is possible to implement the meta-algorithm of Spherical Integration through a more involved procedure. We start by stating our requirements on the ANNS oracle.

Definition 14 ((r,c,χ')-ANNSO)**.** *Given a set $S \subseteq \mathcal{M}$ and numbers $r>0$, $c>1$, $\chi' < 1$, an (r,c,χ')-approximate nearest neighbor search oracle $\mathcal{O}$ for $\mathcal{M}$ returns a data-structure $\mathrm{ANN}_{S,r,c} \leftarrow \mathcal{O}(S,r,c)$ such that for any query $q \in \mathcal{M}$, its output $\mathrm{ANN}_{S,r,c}(q)$ satisfies:*

(O1) $\mathrm{ANN}_{S,r,c}(q) \in B_S(q,rc) \cup \{\perp\}$
(O2) $|B_S(q,r)| > 1 \Rightarrow \mathrm{ANN}_{S,r,c}(q) = \perp$
(O3) $B_S(q,cr) = B_S(q,r) = \{p\} \Rightarrow \mathrm{ANN}_S(q,r,c) = p$

with probability at least $1-\chi'$ over the internal randomness of the data structure. Furthermore, different calls to the oracle are independent.

Properties (O1) and (O3) are inherent in the definition of any (r,c)-ANN data structure. The only extra requirement is property (O2) that asserts that the data structure can detect when there is not a unique point in $B_S(q,r)$. For most data structures, this can be easily implemented by outputting $\perp$ whenever $|B_S(q,rc)| > 1$. In the full version of the paper, we devise a black-box reduction for any ANNS data structure to an ANNSO oracle using only a logarithmic increase in resources. One technical novelty of our work is that one can use such an oracle to sample a point from $B_P(q,r)$ "near" uniformly at random.

The main idea is that, property (O2) (resp. (O3)) implies that a necessary (resp. sufficient) condition for the oracle to return a point $p \in B_P(q,r)$ is for p to be the *only* point in $B_S(q,r)$ (resp. $B_S(q,rc)$). We exploit this property by calling the oracle on random subsets S of our point set P by sub-sampling. We show below that if the sub-sampling probability is small enough compared to $|B_P(q,rc)|^{-1}$ then whenever we get a point $p \in B_P(q,r)$ it is approximately uniform in $B_P(q,r)$.

Lemma 15 (Sampling Lemma)**.** *For $u \in [0,1]$ and $P \subset \mathcal{M}$, let $S_u \sim P[u]$ denote a random subset of P where each point has been included independently with probability u. Given an (r,c,χ')-ANNO $\mathcal{O}$ for $\mathcal{M}$, let $\mathrm{ANN}_{S_u,r,c} \leftarrow \mathcal{O}(S_u,r,c)$. For a query $q \in \mathcal{M}$ and point $p \in B_P(q,r)$, define $G_q(p) := \{\mathrm{ANN}_{S_u,r,c}(q) = p\}$ to be the event that the data structure returns p when queried with q. For all $\delta \in (0,1]$ such that $\frac{2\chi'}{\delta} \le u \le \frac{\delta}{2|B_P(q,rc)|}$, we have:*

$$\left|\mathbb{P}[G_q(p)] - u\right| \le \delta \cdot u, \forall p \in B_P(q,r) \tag{15}$$

Proof: Let F be the event that any of the three conditions are violated for $\mathrm{ANN}_{S_u,r,c}(q)$, then by definition $\mathbb{P}[F] \le \chi'$. Let $k = |B_P(q,rc)|$ and $|B_P(q,c)| = \alpha k$ with $\alpha \in [0,1]$. We have:

$$\begin{aligned}\mathbb{P}[G_q(p)] &= \mathbb{P}[G_q(p)|F]\mathbb{P}[F] + \mathbb{P}[G_q(p)|F^c]\mathbb{P}[F^c] \\ &\ge \mathbb{P}[G_q(p)|F^c](1-\chi') \\ &\ge u(1-u)^{k-1}(1-\chi') \\ &\ge u(1-uk-\chi')\end{aligned} \tag{16}$$

$$\begin{aligned}\mathbb{P}[G_q(p)] &\le \mathbb{P}[F] + \mathbb{P}[G_q(p)|F^c] \\ &\le \chi' + u(1-u)^{\alpha k - 1} \\ &\le u(1+\frac{\chi'}{u})\end{aligned} \tag{17}$$

where in (16) and (17) we used respectively (O3) and (O2). ■

Thus, using an upper bound $B \ge |B_P(q,rc)|$ and the ANNS oracle we can implement a data structure that returns approximately uniform random points from the set $B_P(q,r)$ at a rate (number of calls) that depends on the ratio between our upper bound B and $|B_P(q,r)|$. Our approach is to use the oracle to obtain random samples from a given annulus $A(q,r)$ and then use these samples to estimate $\mathrm{KDF}_A(q)$ through the Median-of-means technique. The fact that we don't have exact knowledge of and control over the sampling probabilities, requires a slightly modified analysis of the Median-of-means technique.

Procedure 1 MEDIANOFMEANS ($\mathrm{MoM}_{\epsilon,\delta,\chi}(Q,V)$)

```
1: Input: $V \geq 0$, set $Q$, accuracy $\epsilon \in (0,1)$, deviation $\delta \in [0,\epsilon)$, failure prob. $\chi \in (0,1)$.
2: $m(\epsilon,\delta,V) \leftarrow \lceil \frac{6(1+\delta)}{(\epsilon-\delta)^2} V \rceil$, $L(\chi) \leftarrow \lceil 9\log(1/\chi) \rceil$.
3: Partition $Q \to Q_1 \uplus Q_2 \uplus \ldots \uplus Q_L \uplus Q_{L+1}$ such that $|Q_i| = m$, $\forall i = 1,\ldots,L$.
4: $Z^{(i)} \leftarrow \frac{1}{m}\sum_{j\in Q_i} Z_j$, $i = 1,\ldots L$
5: Output: $Z_{\epsilon,\delta,\chi} \leftarrow \mathrm{median}\{Z^{(1)},\ldots,Z^{(L)}\}$
```

Lemma 16 (Non-uniform MoM)**.** *Let $Q = \{j_1,\ldots,j_T\}$ be a set of indices such that $Z_{j_1},\ldots,Z_{j_T}$ are independent random variables with $|\mathbb{E}[Z_i] - \mu| \leq \delta\mu$ and $\mathbb{E}[Z_i^2] \leq (1+\delta)V\mu^2$. For $\epsilon,\chi > \delta \geq 0$, if $T \geq \lceil \frac{6(1+\delta)}{(\epsilon-\delta)^2} V \rceil \lceil 9\log(\frac{1}{\chi}) \rceil$, then* $\mathbb{P}\left[\left|\mathrm{MoM}_{\epsilon,\delta,\chi}(Q,V) - \mu\right| \geq \epsilon\mu\right] \leq \chi$

In order to build a data structure that can generate sufficient number of random samples from different annuli, we make calls to the oracle for different values of r but with fixed $c > 1$ and random subsets of P of varying size. In particular, for a given $\epsilon \in (0,1)$ let $\delta = \epsilon/9$ and $I_\delta := \lfloor \log_{1+\delta} n \rfloor$. For $i = 0,\ldots,I_\delta$ we define sampling probabilities $u_i := \frac{\delta}{2n}(1+\delta)^i$ and set $T_{\max} = 1728\frac{(1+\frac{\epsilon}{9})(1+\frac{\epsilon}{4})}{(1-\frac{\epsilon}{9})(1-\epsilon)^2}\frac{1}{\epsilon^3}\frac{1}{\alpha_{\bar{t}}^2}\bar{L}c^{\bar{t}}\log(\frac{1}{\chi})$, where $\alpha_t = \frac{\epsilon}{\phi\bar{L}c^{2\bar{t}}}$. We build a data structure RSO (Random Sampling Oracle) by storing references to the data structures produced by calls to the oracle $\mathcal{O}$ for all the combinations of radii, sampling probabilities and for sufficient number $T_{\max}$ of random subsets of P. The total number of calls to is $I_\delta \cdot \phi_c \cdot T_{\max}$.

Procedure 2 ANN2RSO (Preprocessing)

```
Input: $P \subset M$, oracle $\mathcal{O}$, $c > 1$, $\{u_i\}$, $\{r_\ell\}, \phi_c, I_\delta, T_{\max}$.
1: for $i = 0,\ldots,I_\delta$ do
2:     for $j = 1,\ldots,T_{\max}$ do
3:         $S_{u_i}^{(j)} \sim P[u_i]$
4:         for $\ell = 0,\ldots,\phi_c$ do
5:             $\mathrm{RSO}_{\ell,i,j} \leftarrow \mathcal{O}(S_{u_i}^{(j)}, r_\ell, c)$
Output: RSO
```

C. Spherical Inegration via Random Sampling Oracles

We next show how to implement the meta-algorithm of Spherical integration using the Random Sampling Oracle. The algorithm will be slightly more complicated in that we won't be able to estimate accurately the contribution of each annulus (by getting enough random samples from the annulus). Instead we will prove that we can estimate accurately the contribution of *all annuli* that have *non-negligible contribution to the kernel density*. This is achieved by showing, that our failure to get enough random samples from an annulus is attributed with high probability to the fact that the annulus in question has a negligible contribution *compared to the contribution of an outer annulus or inner ball*. This is possible for the following family of kernels.

Table II
APPLICATIONS OF THEOREM 20 FOR KDF UNDER (L,t)-SMOOTH AND DECREASING KERNELS FOR $\chi = \mathrm{poly}(\frac{1}{n})$.

Metric space	Query time (QT) / $\epsilon^{-5}(L\phi)^3$
L_2 [39]	$\tilde{O}(d(\frac{\log n}{5t})^{10t(1+o(1))})$
L_∞ [23]	$\tilde{O}(d^2(\frac{\log n}{5t}\log\log d)^{5t(1+o(1))})$
L_p ($p > 2$) [40], [41]	$\tilde{O}(d^2\frac{\log^{1+\frac{1}{p}} n}{3t}p)^{5t(1+o(1))})$
Sym. norms [42]	$\tilde{O}(d^{O(1)}(O(\log^5 n\frac{\log\log d}{t^5\log^5\log n}))^{10t})$
Ulam [40], [43]	$\tilde{O}(d^{O(1)}(O(\log^4 n\frac{\log\log d}{t^4}))^{10t})$
Edit ($\ell > 1$) [44]	$\tilde{O}(d3^{5\ell t})$

Definition 17 ((L,t)-Radial Kernels)**.** *Fix $L > 0, t > 0$ and a metric space $\mathcal{M}$. A function $k : \mathcal{M}\times\mathcal{M} \to \mathbb{R}_+$ is called (L,t)-radial if $\forall s \geq 0, \forall q \in \mathcal{M}$,*

$$\max\left\{\frac{\sup_{p\in A_\ell} k(q,p)}{\inf\limits_{p'\in B(q,r_{\ell-s})} k(q,p')}, \frac{\sup_{p\in A_\ell} k(q,p)}{\inf\limits_{p'\in A_{\ell+s}} k(q,p')}\right\} \leq L\cdot c^{(s+1)t}$$

For this family of kernels we have the following structural properties, whose proofs we present in Section IV-E.

Proposition 18. *If a kernel is decreasing in $d_{\mathcal{M}}$ and (L,t)-smooth then it is also (L,t)-radial.*

Lemma 19. *For (L,t)-radial kernels and $\alpha \in (0,\frac{1}{11}]$, if $\frac{|A(q,r_\ell)|}{|A(q,r_{\ell+s})|} < \alpha$ or $\frac{|A(q,r_\ell)|}{|B(q,r_{\ell-s+1})|} < \alpha$ it holds that $Z_\ell < \frac{11}{10}\alpha\cdot Lc^{(s+1)t}\cdot\mathrm{KDF}(q)$.*

We were able to obtain the following result.

Theorem 20. *Let $P \subseteq \mathcal{M}$ be a set of n points with* log*-aspect ratio ϕ and $\bar{L},\bar{t} > 0, \epsilon \in (0,0.99)$ be fixed numbers. Given an (r,c,χ')-ANNSO $\mathcal{O}$ for $\mathcal{M}$ with $\chi' \leq \epsilon^2/(324n)$, there exists a data-structure constructed using $O\left(\epsilon^{-6}c^{5\bar{t}}[\bar{L}\phi]^3\log(n)\log(\frac{\phi}{\chi})\right)$ calls to the oracle $\mathcal{O}$, that for any query $q \in \mathcal{M}$ and (L,t)-radial kernel with $1 \leq L \leq \bar{L}$, $0 \leq t \leq \bar{t}$, can provide an $(1+O(\epsilon))$-approximation to $\mathrm{KDF}_P(q)$ with probability at least $1-\chi$ using $O\left(\epsilon^{-5}c^{5t}[L\phi]^3\log(\frac{\phi}{\chi})\right)$ total queries to the data-structures created by $\mathcal{O}$.*

In Table II, we provide applications of our result to kernel density evaluation algorithms for various metrics using known ANNS data-structures.

Our algorithm (Procedure 3) proceeds in $\phi + 1$ stages, where for each stage $\ell \in \{0,\ldots,\phi\}$ there are three goals:

1) **Sampling invariant:** maintaining an upper bound $B_{\ell+1} \geq |B(q, r_{\ell+1})|$ that allows us to set the sampling probability u appropriately (Lemma 15) and ensure that any point $p \in B(q, r_\ell)$ returned by the RSO data-structure is close to a uniformly random point.
2) **Estimating** $|B(q, r_\ell)|$: during this phase we either get an upper bound on $|B(q, r_\ell)|$ that either satisfies the condition of Lemma 19 and allows to maintain the invariant when we decrement ℓ, or we get a multiplicative approximation $\hat{N}_\ell$ to $|B(q, r_\ell)|$ that is used both to maintain the invariant and to possibly estimate the contribution of the ℓ-th annulus to $\mathrm{KDF}(q)$.
3) **Estimating** $|A(q, r_\ell)|$ **and annulus contribution:** this phase is executed only if $|B(q, r_\ell)|$ is significant in which case we have a lower bound on the probability of getting a sample in $B(q, r_\ell)$. This means that after a certain number of trials we can either get an upper bound $A(q, r_\ell)$ or will be able to get a multiplicative approximation to $\rho_\ell = |A(q, r_\ell)|/|P|$ and have enough samples to estimate Z_ℓ through median of means.

A subtle point that we deal with in the analysis is that from step 2 above we *only get an upper bound* B_ℓ on the size of $|B(q, r_\ell)|$. However, it is entirely possible that this upper bound is quite far from the true value. Complications arise because we use this upper bound to set the sampling probability in the next stage $\ell - 1$, resulting in seeing points from $B(q, r_{\ell-1})$ in a reduced rate, as well as due to the fact that in cases where we fail to get enough samples from $|B(q, r_\ell)|$ for consecutive values of ℓ, the upper bounds that we get depend on previous stages. In this way long range interaction between stages exist and we have to account for.

D. Analysis of Spherical Integration

To prove correctness of our algorithm we break up the execution of our algorithm in stages depending on the value of $\ell = \phi, \ldots, 0$. For any ℓ for which the invariant $|B(q, r_{\ell+1})| \leq B_{\ell+1}$ is true, we say that the *ℓ-stage succeeds* if: (a) the invariant is true for $\ell - 1$, (b) whenever $\hat{Z}_\ell = 0$, the contribution of the annulus is less than $C_\delta = \frac{121}{40}\left(\frac{1+\delta}{1-\delta}\right)^2 \frac{\epsilon}{\phi}\mathrm{KDF}(q)$, and (c) whenever $\hat{Z}_\ell \neq 0$, then $|\hat{\rho}_\ell - \rho_\ell| < \epsilon(2+\epsilon)\rho_\ell$ and $|\hat{\mu}_\ell - \mu_\ell| < \epsilon\mu_\ell$.

The main lemma we have to show is the following.

Lemma 21 (Main lemma). *For $\ell \in \{0, \ldots, \phi\}$, let U_ℓ be the event that the ℓ-th stage succeeds and define $U_{>\ell} := \bigwedge_{\ell'=\ell+1}^{\phi} U_{\ell'}$. Then $\mathbb{P}[U_\ell | U_{>\ell}] \geq 1 - 19\chi$*

The proof of this lemma is long and technical and is included in the full version of the paper. Having this lemma at hand we can readily prove the theorem.

Proof of Theorem 20: We first prove a lower bound on the probability that all stages succeed $\mathbb{P}[\bigwedge_{\ell=0}^{\phi} U_\ell] = \prod_{\ell=0}^{\phi} \mathbb{P}\left[U_{\phi-\ell} \middle| U_{>\phi-\ell}\right] \geq (1-19\chi)^{\phi+1} \geq 1 - 19(\phi+1)\chi$.

Procedure 3 SPHERICAL INTEGRATION

```
Input: $t, L, c, \epsilon, \phi_c, \chi, n, \mathrm{RSO}$
 1: $\delta \leftarrow \epsilon/9$                                   ▷ Distance from uniformity
 2: $I_\delta \leftarrow \lfloor \log_{(1+\delta)} n \rfloor$          ▷ Number of sampling levels
 3: $\alpha_t \leftarrow \frac{\epsilon}{\phi}\frac{1}{Lc^{2t}}$          ▷ Threshold for significant annuli
 4: $T_1 \leftarrow \lceil 4(1+\delta)^2 \frac{1}{\delta^3}\frac{1}{\alpha_t}\log(\frac{1}{\chi})\rceil$
 5: $T_2 \leftarrow \lceil 145(1+\delta)(1+\frac{\epsilon}{4})\frac{1}{\alpha_t^2}\frac{1}{\epsilon^2}\frac{1}{\delta}Lc^t\log(\frac{1}{\chi})\rceil$
 6: $u_i \leftarrow \frac{\delta}{2n}(1+\delta)^i$ for $i = 0, \ldots, I_\delta$.   ▷ Sampling levels
 7: $r_\ell \leftarrow r_0 c^\ell$ for $\ell = 0, \ldots, \phi_c$.          ▷ Sequence of radii
 8: $B_{\phi_c+1} \leftarrow n$   ▷ Initialize invariant $|B(q, r_{\ell+1})| \leq B_{\ell+1}$
 9: for $\ell = \phi_c, \ldots, 0$ do
10:     $i_* \leftarrow \arg\max_{i \leq I_\delta}\{u_i B_{\ell+1} \leq \frac{\delta}{2}\}$   ▷ under-sampling
11:     $Q_+ \leftarrow \emptyset$                    ▷ Start with an empty multi-set
12:     for $j = 1, \ldots, T_1$ do
13:         if $\mathrm{RSO}_{\ell,i_*,j}(q) \in B(q, r_\ell)$ then
14:             add $\mathrm{RSO}_{\ell,i_*,j}(q)$ to $Q_+$
15:     $\zeta_+ \leftarrow (1+\delta)(\alpha_t B_{\ell+1} u_{i_*})$   ▷ significance threshold
16:     if $|Q_+| < \zeta_+ T_1$ then
17:         $B_\ell \leftarrow \frac{(1+\delta)^2}{1-\delta}\alpha_+ B_{\ell+1}$        ▷ Maintain the invariant
18:         $\hat{Z}_\ell \leftarrow 0$                       ▷ annulus “insignificant”
19:     else
20:         $\hat{N}_\ell \leftarrow \frac{|Q_+|}{u_{i_*}}\frac{1}{T}$            ▷ $(1+\epsilon)$-approx to $|B(q, r_\ell)|$
21:         $B_\ell \leftarrow \frac{1}{1-\epsilon}\hat{N}_\ell$                   ▷ Maintain the invariant
22:         $Q_- \leftarrow Q_+$
23:         for $j = T_1 + 1, \ldots, T_2$ do
24:             if $\mathrm{RSO}_{\ell,i_*,j}(q) \in B(q, r_\ell)$ then
25:                 add $\mathrm{RSO}_{\ell,i_*,j}(q)$ to $Q_-$
26:         $\zeta_- \leftarrow 2\frac{1+\delta}{1-\delta}\alpha_t$      ▷ threshold for significant annuli
27:         $Q_\ell \leftarrow Q_- \cap A(q, r_\ell)$
28:         if $|Q_\ell| < \zeta_- |Q_-|$ then
29:             $\hat{Z}_\ell \leftarrow 0$                   ▷ annulus “insignificant”
30:         else
31:             $\hat{\rho}_\ell \leftarrow \frac{|Q_\ell|}{|Q_-|}\frac{\hat{N}_\ell}{n}$             ▷ Estimate $|A(q, r_\ell)|/N$
32:             $V_t \leftarrow Lc^t$            ▷ “condition number” $A(q, r_\ell)$
33:             $\hat{\mu}_\ell \leftarrow \mathrm{MoM}_{\epsilon,\frac{\epsilon}{4},\chi}(Q_\ell, V_t)$   ▷ estim. $\mathrm{KDF}_{A_{r_\ell}}(q)$
34:             $\hat{Z}_\ell \leftarrow \hat{\rho}_\ell \hat{\mu}_\ell$
Output: $\sum_{\ell=0}^{L} \hat{Z}_\ell$
```

Next we show that conditional on the event $\bigwedge_{\ell=0}^{\phi} U_\ell$ the algorithm outputs a multiplicative approximation to $\mu = \mathrm{KDF}(q)$. Let $J := \{\ell \in \{0, \ldots, \phi\} | \hat{Z}_\ell \neq 0\}$, then

$$
\begin{aligned}
|\sum_{\ell=0}^{\phi} \hat{Z}_\ell - \mu| &\leq \sum_{\ell \in J} |\hat{\rho}_\ell\hat{\mu}_\ell - \rho_\ell\mu_\ell| + \sum_{\ell' \notin J} \rho_{\ell'}\mu_{\ell'} \\
&\leq \sum_{\ell \in J} \{\hat{\mu}_\ell|\hat{\rho}_\ell - \rho_\ell| + \rho_\ell|\hat{\mu}_\ell - \mu_\ell|\} + C_\delta\epsilon\mu \\
&\leq \sum_{\ell \in J} \{\epsilon(2+\epsilon)\hat{\mu}_\ell\rho_\ell + \rho_\ell\epsilon\mu_\ell\} + C_\delta\epsilon\mu \\
&\leq [(2+\epsilon)(1+\epsilon) + 1]\epsilon\mu + C_\delta\epsilon\mu
\end{aligned}
$$

where for the first two inequalities we used triangle inequality twice and the definition of J along with part (b) of the definition of U_ℓ. The last two inequalities are due to part (c) and (14). The total number of calls to the RSO data structure for each query is bounded by $T_3 = T_2(\phi+1) = \Theta\left(\epsilon^{-5}c^{5t}(L\phi)^3\log(\frac{1}{\chi})\right)$. The number of calls required to build the RSO data structure is $T_3 \cdot I_\delta = O(T_3 \cdot \frac{\log n}{\delta})$. To make the success probability at least $1-\nu$ we simply need to substitute χ with $\frac{\nu}{19(\phi+1)}$ and increase the number of samples accordingly. ■

E. Proofs of results for (L,t)-Radial Kernels

Proof of Proposition 18: For any point $q \in \mathcal{M}$, let $p_0, p_+, p_- \in \mathcal{M}$ such that $d_{\mathcal{M}}(q,p_0) \geq r_{\ell-1}$, $d_{\mathcal{M}}(q,p_-) \leq r_{\ell-s}$ and $d_{\mathcal{M}}(q,p_+) \leq r_{\ell+s}$. By the fact that the kernel is decreasing in $d_{\mathcal{M}}$ we have:

$$\max\left\{\frac{\sup_{p\in A_\ell} k(q,p)}{\inf\limits_{p'\in B(q,r_{\ell-s})} k(q,p')}, \frac{\sup_{p\in A_\ell} k(q,p)}{\inf\limits_{p'\in A_{\ell+s}} k(q,p')}\right\} \leq \tag{18}$$

$$\max\left\{\frac{k(q,p_0)}{k(q,p_-)}, \frac{k(q,p_0)}{k(q,p_+)}\right\} \leq Lc^{(s+1)t} \tag{19}$$

where in the last inequality we used the fact that the kernel is (L,t)-smooth. ■

Proof of Lemma 19: For brevity, we will denote here $A(q,r_\ell)$ as A_ℓ. The first condition implies that $|A_\ell| < \alpha|A_{\ell+s}|$, while the second condition implies that $|A_\ell| < \frac{\alpha}{1-\alpha}|B(q,r_{\ell-s})|$. In both cases we proceed by obtaining an upper bound on the contribution of the set A_ℓ in terms of the contribution of an outer annulus or inner ball respectively. Starting from the basic inequality $\sum_{p\in A_\ell} k(q,p) \leq |A_\ell|\sup_{p\in A_\ell} k(q,p)$, we get:

$$|A_\ell| \sup_{p\in A_\ell} k(q,p) \leq \alpha \frac{\sup_{p\in A_\ell} k(q,p)}{\inf\limits_{p'\in A_{\ell+s}} k(q,p')} |A_{\ell+s}| \inf_{p'\in A_{\ell+s}} k(q,p')$$

and

$$|A_\ell| \sup_{p\in A_\ell} k(q,p) \leq \frac{\alpha}{1-\alpha} \frac{\sup_{p\in A_\ell} k(q,p)}{\inf\limits_{p'\in B(q,r_{\ell-s})} k(q,p')} \cdot |B(q,r_{\ell-s})| \inf_{p'\in B(q,r_{\ell-s})} k(q,p')$$

In both, cases by the fact that the kernel is (L,t)-*radial* we get that:

$$\max\left\{\frac{\sup_{p\in A_\ell} k(q,p)}{\inf\limits_{p'\in B(q,r_{\ell-s})} k(q,p')}, \frac{\sup_{p\in A_\ell} k(q,p)}{\inf\limits_{p'\in A_{\ell+s}} k(q,p')}\right\} \leq Lc^{(s+1)t}$$

Using the observation that for any subset of $S \subseteq P$, $|S|\inf_{p\in S} k(q,p) \leq |P|\mathrm{KDF}_P(q)$ we get the statement. ■

ACKNOWLEDGMENTS

Arturs Backurs and Piotr Indyk were supported by the NSF, the Simons Foundation and an IBM PhD Fellowship. Moses Charikar and Paris Siminelakis were supported by NSF grant CCF-1617577 and a Simons Investigator Award. Paris Siminelakis is partially supported by an Onassis Foundations Scholarship.

REFERENCES

[1] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *Journal of computational physics*, vol. 73, no. 2, pp. 325–348, 1987.

[2] B. Georgescu, I. Shimshoni, and P. Meer, "Mean shift based clustering in high dimensions: A texture classification example." in *ICCV*, vol. 3, 2003, p. 456.

[3] E. Schubert, A. Zimek, and H.-P. Kriegel, "Generalized outlier detection with flexible kernel density estimates," in *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 2014.

[4] A. Rinaldo and L. Wasserman, "Generalized density clustering," *The Annals of Statistics*, pp. 2678–2722, 2010.

[5] Z. Szabó, B. K. Sriperumbudur, B. Póczos, and A. Gretton, "Learning theory for distribution regression," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 5272–5311, 2016.

[6] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola, "Multi-instance kernels," in *ICML*, vol. 2, 2002, pp. 179–186.

[7] K. Muandet, K. Fukumizu, B. Sriperumbudur, B. Schölkopf *et al.*, "Kernel mean embedding of distributions: A review and beyond," *Foundations and Trends® in Machine Learning*, vol. 10, no. 1-2, pp. 1–141, 2017.

[8] M. P. Holmes, A. G. Gray, and C. L. Isbell, "Ultrafast Monte Carlo for Kernel Estimators and Generalized Statistical Summations," in *Advances in Neural Information Processing Systems*, 2008.

[9] M. Charikar and P. Siminelakis, "Hashing-based-estimators for kernel density in high dimensions," in *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2017.

[10] J. M. Phillips and W. M. Tai, "Improved coresets for kernel density estimates," in *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2018.

[11] ——, "Near-optimal coresets of kernel density estimates," in *34th International Symposium on Computational Geometry (SoCG)*, 2018.

[12] A. Backurs, P. Indyk, and L. Schmidt, "On the fine-grained complexity of empirical risk minimization: Kernel methods and neural networks," in *Advances in Neural Information Processing Systems*, 2017.

[13] A. Rubinstein, "Hardness of approximate nearest neighbor search," in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. ACM, 2018.

[14] M. G. Genton, "Classes of kernels for machine learning: a statistics perspective," *Journal of Machine Learning Research*, vol. 2, no. Dec, pp. 299–312, 2001.

[15] R. Krasny and L. Wang, "Fast evaluation of multiquadric RBF sums by a Cartesian treecode," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2341–2355, 2011.

[16] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. ONeil, "Fast Direct Methods for Gaussian Processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 252–265, 2016.

[17] Z. Szabó, A. Gretton, B. Póczos, and B. Sriperumbudur, "Two-stage sampled learning theory on distributions," in *Artificial Intelligence and Statistics*, 2015, pp. 948–957.

[18] H. Lê Nguyên, "Algorithms for high dimensional data," Ph.D. dissertation, Princeton University, 2014.

[19] W. B. March, B. Xiao, and G. Biros, "ASKIT: Approximate skeletonization kernel-independent treecode in high dimensions," *SIAM Journal on Scientific Computing*, vol. 37, no. 2, pp. A1089–A1110, 2015.

[20] C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis, "Improved fast gauss transform and efficient kernel density estimation," in *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2003.

[21] M. Griebel and D. Wissel, "Fast approximation of the discrete gauss transform in higher dimensions," *Journal of Scientific Computing*, vol. 55, no. 1, pp. 149–172, 2013.

[22] O. Neiman, "Low dimensional embeddings of doubling metrics," *Theory of Computing Systems*, vol. 58, no. 1, pp. 133–152, 2016.

[23] P. Indyk, "On approximate nearest neighbors under ℓ_∞ norm," *Journal of Computer and System Sciences*, vol. 63, no. 4, pp. 627–638, 2001.

[24] S. Har-Peled and M. Mendel, "Fast construction of nets in low-dimensional metrics and their applications," *SIAM Journal on Computing*, vol. 35, no. 5, pp. 1148–1184, 2006.

[25] J. Barnes and P. Hut, "A hierarchical $o(n \log n)$ force-calculation algorithm," *Nature*, vol. 324, no. 6096, pp. 446–449, 1986.

[26] P. B. Callahan and S. R. Kosaraju, "A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields," *Journal of the ACM (JACM)*, vol. 42, no. 1, pp. 67–90, 1995.

[27] A. G. Gray and A. W. Moore, "N-body'problems in statistical learning," in *Advances in neural information processing systems*, 2001.

[28] ——, "Nonparametric density estimation: Toward computational tractability," in *Proceedings of the 2003 SIAM International Conference on Data Mining*. SIAM, 2003.

[29] D. Lee, A. W. Moore, and A. G. Gray, "Dual-tree fast gauss transforms," in *Advances in Neural Information Processing Systems*, 2006.

[30] D. Lee and A. G. Gray, "Fast high-dimensional kernel summations using the monte carlo multipole method," in *Advances in Neural Information Processing Systems*, 2009.

[31] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[32] R. F. Sproull, "Refinements to nearest-neighbor searching ink-dimensional trees," *Algorithmica*, vol. 6, no. 1-6, pp. 579–589, 1991.

[33] P. Ram, D. Lee, W. March, and A. G. Gray, "Linear-time algorithms for pairwise statistical problems," in *Advances in Neural Information Processing Systems*, 2009.

[34] S. Dasgupta and Y. Freund, "Random projection trees and low dimensional manifolds," in *Proceedings of the 40th annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2008.

[35] S. S. Vempala, "Randomly-oriented kd trees adapt to intrinsic dimension," in *LIPIcs-Leibniz International Proceedings in Informatics*, vol. 18. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[36] L. Devroye and G. Lugosi, *Combinatorial methods in density estimation*. Springer Science & Business Media, 2012.

[37] K. Ball *et al.*, "An elementary introduction to modern convex geometry," *Flavors of geometry*, vol. 31, pp. 1–58, 1997.

[38] S. Dasgupta and A. Gupta, "An elementary proof of a theorem of Johnson and Lindenstrauss," *Random Structures & Algorithms*, vol. 22, no. 1, pp. 60–65, 2003.

[39] S. Har-Peled, P. Indyk, and R. Motwani, "Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality." *Theory of computing*, vol. 8, no. 1, pp. 321–350, 2012.

[40] A. Andoni, "Nearest neighbor search: the old, the new, and the impossible," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[41] Y. Bartal and L.-A. Gottlieb, "Approximate nearest neighbor search for ℓ_p-spaces $(2 < p < \infty)$ via embeddings," in *Latin American Symposium on Theoretical Informatics*. Springer, 2018, pp. 120–133.

[42] A. Andoni, H. L. Nguyen, A. Nikolov, I. P. Razenshteyn, and E. Waingarten, "Approximate near neighbors for general symmetric norms," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2017.

[43] M. Charikar and R. Krauthgamer, "Embedding the ulam metric into l1." *Theory of Computing*, vol. 2, no. 11, pp. 207–224, 2006.

[44] P. Indyk, "Approximate nearest neighbor under edit distance via product metrics," in *Proceedings of the 15th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 2004.