A Quasi-Monte Carlo Data Structure for Smooth Kernel Evaluations

Moses Charikar* Michael Kapralov† Erik Waingarten‡

Abstract

In the kernel density estimation (KDE) problem one is given a kernel $K(x, y)$ and a dataset P of points in a high dimensional Euclidean space, and must prepare a small space data structure that can quickly answer density queries: given a point q, output a $(1+\epsilon)$-approximation to $\mu := \frac{1}{|P|}\sum_{p\in P} K(p, q)$. The classical approach to KDE (and the more general problem of matrix vector multiplication for kernel matrices) is the celebrated fast multipole method of Greengard and Rokhlin [1983]. The fast multipole method combines a basic space partitioning approach with a multidimensional Taylor expansion, which yields a $\approx \log^d(n/\epsilon)$ query time (exponential in the dimension d). A recent line of work initiated by Charikar and Siminelakis [2017] achieved polynomial dependence on d via a combination of random sampling and randomized space partitioning, with Backurs et al. [2018] giving an efficient data structure with query time $\approx \mathrm{polylog}(1/\mu)/\epsilon^2$ for smooth kernels.

Quadratic dependence on ϵ, inherent to the sampling (i.e., Monte Carlo) methods above, is prohibitively expensive for small ϵ. This is a classical issue addressed by quasi-Monte Carlo methods in numerical analysis. The high level idea in quasi-Monte Carlo methods is to replace random sampling with a discrepancy based approach – an idea recently applied to coresets for KDE by Phillips and Tai [2020]. The work of Phillips and Tai gives a space efficient data structure with query complexity $\approx 1/(\epsilon\mu)$. This is polynomially better in $1/\epsilon$, but exponentially worse in $1/\mu$. In this work we show how to get the best of both worlds: we give a data structure with $\approx \mathrm{polylog}(1/\mu)/\epsilon$ query time for smooth kernel KDE. Our main insight is a new way to combine discrepancy theory with randomized space partitioning inspired by, but significantly more efficient than, that of the fast multipole methods. We hope that our techniques will find further applications to linear algebra for kernel matrices.

1 Introduction

In the *kernel evaluation* problem, an algorithm receives as input a dataset P of n points $p_1, \ldots, p_n \in \mathbb{R}^d$ and must preprocess it into a small-space data structure that allows one to quickly approximate, given a query $q \in \mathbb{R}^d$, the quantity

$$\mathsf{K}(P, q) := \sum_{p\in P} \mathsf{K}(p, q). \tag{1.1}$$

where $\mathsf{K}(p, q)$ is the kernel function. In this paper, we will study positive definite (p.d) radial kernels K. In particular, we consider kernel functions $\mathsf{K}\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ given by a decreasing function of the distance, i.e., there exists a function $G\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that $\mathsf{K}(p, q) = G(\|p - q\|_2^2)$. The restriction that K is positive definite means that for any collection of points $x_1, \ldots, x_m \in \mathbb{R}^d$, the $m \times m$ kernel matrix whose (i, j)-entry is $\mathsf{K}(x_i, x_j)$ is positive definite. Some example of prominent p.d radial kernels include[1] the Cauchy kernel (or, more generally, the rational quadratic kernel) for any $\beta \geq 1$, and the Student-t kernel, respectively,

$$\mathsf{K}(p, q) := \left(\frac{1}{1 + \|p - q\|_2^2}\right)^\beta \quad \text{and} \quad \mathsf{K}(p, q) := \frac{1}{1 + \|p - q\|_2^t}. \tag{1.2}$$

A variety of kernels are used in applications [STC+04, RW06], and kernel methods are a fundamental approach with numerous applications in machine learning, statistics and data analysis [FG96, SS01, JKPV11, SZK14, GPPV+14, ACMP15, GB17].

*Stanford University. Supported by a Simons Investigator award.

†EPFL. Supported in part by ERC Starting Grant 759471

‡University of Pennsylvania. Part of this work was done as a postdoc at Stanford University, supported by an NSF postdoctoral fellowship and Moses Charikar's Simons Investigator award.

[1]The Gaussian kernel $\mathsf{K}(p, q) = \exp(-\|p - q\|_2^2/(2\sigma^2))$ is a widely-used p.d kernel, but not "smooth" (as per the definition in [BCIS18a]).

For example, kernel density estimation is a classic tool in non-parametric statistics, where a kernel function is applied to extrapolate a function specified on a discrete set of points to the entire space. This is used in algorithms for mode estimation [GSM03], outlier detection [SZK14], density based clustering [RW10] and other problems. Kernel methods (e.g., regression [SSPG16]) have also been applied to objects specified by point-clouds or distributions [GFKS02], requiring summing up the kernel function between all pairs of points across two sets. Another application is kernel mean estimation using an empirical average (see [MFS+17], section 3.4.2).

We are interested in fast and space efficient approximations algorithms for kernel evaluations. An estimator for $\hat{\mathsf{K}}(P,q)$ is a $(1+\epsilon)$-approximation to $\mathsf{K}(P,q)$ if

$$(1-\epsilon)\cdot \mathsf{K}(P,q) \leq \hat{\mathsf{K}}(P,q) \leq (1+\epsilon)\cdot \mathsf{K}(P,q).$$

The kernel density estimation problem has received a lot of attention over the years, with a number of powerful algorithmic ideas leading to different precision/space/query time tradeoffs. The focus of this work is algorithms for the so-called "high-dimensional" regime: we will study algorithms whose complexity will depend at most polynomially, as opposed to exponentially in the underlying dimension.

Prior Work: Fast Multipole Methods. The celebrated fast multipole method [BG97] can be used to obtain efficient data structure for kernel evaluation (see also the Barnes-Hut algorithm [BH86]). However, this approach suffers from an exponential dependence on the dimensionality of the input data points: it provides a $(1+\epsilon)$-approximation to kernel evaluation value using space $\approx n\log^d(n/\epsilon)$ and query time $\approx \log^d(n/\epsilon)$.

In particular, the exponential dependence on the dimensionality is due to a (deterministic) space partitioning procedure (essentially building a quadtree) which is central to the fast-multipole method. More generally, this deficiency is shared by other tree-based methods for kernel density estimation [GM01, GM03, YDGD03, LMG06, RLMG09]. Methods based on polynomial approximations have recently been used to obtain fast algorithms for kernel graphs [ACSS20] as well as attention mechanisms in transformers [AS23] – all these approaches are only efficient in relatively low dimensions.

Prior Work: Sampling-based approaches (Monte-Carlo methods). A recent line of work [CS17a, CS19, BCIS18b, BIW19, CKNS20a] sought to design sublinear query-time algorithms for kernel density estimation while avoiding exponential dependencies in the dimension (thereby allowing these methods to be scaled to high-dimensional spaces). These works parametrize the query time of the data structure in terms of the value $\mu = \mathsf{K}(P,q)/n$, and the goal is to achieve query times which are significantly faster than $O(d/(\epsilon^2\mu))$, which is what one achieves via uniform random sampling. Surprisingly, [CS17b] showed how, for several kernels, one can reduce the query time to $O(d/(\epsilon^2\sqrt{\mu}))$ by using the Locality-Sensitive Hashing framework of Indyk and Motwani [IM98]. Furthermore, [BCIS18b] (and also [CKNS20a]) developed the approach further, and showed that for "smooth" kernels, i.e. kernels with polynomial decay, a multiplicative approximation can be obtained with just a *polylogarithmic dependence on* $1/\mu$. Specifically, they achieve a $(1+\epsilon)$-approximation using space $\approx n\cdot \mathrm{polylog}(1/\mu)/\epsilon^2$ space and

$$\text{query time} \approx \mathrm{polylog}(1/\mu)/\epsilon^2. \tag{1.3}$$

Prior Work: Quasi-Monte Carlo (i.e., discrepancy-based) methods. In the approaches above, the focus has always been on the dependence on μ, and the additional factor of $1/\epsilon^2$ a consequence of the sampling-based approach. More broadly, this quadratic dependence on $1/\epsilon$ is unsurprising. It is common, in approaches via random sampling, to design an estimator and utilize Chebyshev's inequality to upper bound the probability that the estimator deviates from its expectation. On the other hand, in applications where the $1/\epsilon^2$ dependence (coming from random sampling) is prohibitively expensive, a common approach in numerical analysis is to use quasi-Monte Carlo methods.

At a high level, these techniques are based on discrepancy theory. They seek to find a "random-like" set of samples (such that the estimator will approximate the expectation), and at the same time, minimize the "random deviations" that one expects from random sampling. In the context of kernel density estimation, this idea was used in the beautiful work of Phillips and Tai [PT20], who used discrepancy theory to build a small coreset for kernel density estimation. As we detail in Section 2, standard subsampling approaches correspond to chosing the colors uniformly at random, but [PT20] show that a coloring with much lower discrepancy can be constructed using Banaszcyk's theorem. Repeatedly 'subsampling' using these low-discrepancy colorings, they give a data structure with space $\approx \mathrm{poly}d/(\epsilon\mu)$ and

$$\text{query time} \approx \mathrm{poly}d/(\epsilon\mu). \tag{1.4}$$

This gives yet another way to improve on the $O(d/(\epsilon^2\mu))$ query time that one achieves via uniform random sampling.[2] In the work of [PT20], the focus is on the dependence on ϵ and the dependence on $1/\mu$ remains linear.[3]

Our contribution. In this work, we show how to achieve the best of both (1.3) and (1.4) quantitatively. Namely, we show how one may use randomized partitioning techniques (like those in [BCIS18a, CKNS20a]) to obtain a poly-logarithmic dependence on $1/\mu$ for smooth kernels, and at the same time, a linear dependence on $1/\epsilon$. The resulting data structure will obtain space complexity $\approx n \cdot \mathrm{poly}d\log(1/\mu)/\epsilon$ and

$$\text{query time } \approx \mathrm{poly}d\log(1/\mu)/\epsilon.$$

At a qualitative level, our work gives new structural results and algorithmic techniques for both the sampling-based and quasi-Monte Carlo-based approaches, and we are hopeful that these techniques will prove useful for fast algorithms in related tasks for kernels in high-dimensional spaces.

From the perspective of the sampling-based methods, our work shows that the quadratic dependence on $1/\epsilon$ is not intrinsic to the randomized approaches for high-dimensions, and that quasi-Monte Carlo (discrepancy-based) techniques can be used to design kernel density estimators in high-dimensions. From the perspective of the quasi-Monte Carlo methods, our work shows that, if one allows randomized data structures, then randomized space partitioning can give exponential improvements on the μ-dependence of discrepancy for smooth kernels; from $\mathrm{poly}d/(\epsilon\mu)$ to roughly $\mathrm{poly}d\log(1/\mu)/\epsilon$. In what follows, we will formally state our results and some open problems, and give an outline of our techniques.

Our results. Our focus is on developing fast data structures for "smooth" p.d radial kernels. We reproduce the definition of "smooth" kernel from [BCIS18a] below, and then we state our main result. As we will soon see, the smoothness condition will become important in the technical details. After discussing the main result, we will give a few open problems and highlight a few concrete challenges involved in obtaining improved ϵ-dependencies for non-smooth kernels (like the Gaussian kernel).

Definition 1.1. (Smooth Function [BCIS18a]) *A kernel $\mathsf{K}\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is (L,t)-smooth if for any three points $p_1, p_2, q \in \mathbb{R}^d$ with $p_1 \neq q \neq p_2$,*

$$\max\left\{\frac{\mathsf{K}(p_1,q)}{\mathsf{K}(p_2,q)}, \frac{\mathsf{K}(p_2,q)}{\mathsf{K}(p_1,q)}\right\} \leq L \cdot \left(\max\left\{\frac{\|p_1-q\|_2}{\|p_2-q\|_2}, \frac{\|p_2-q\|_2}{\|p_1-q\|_2}\right\}\right)^t.$$

Remark 1.1. *We remark that the definition above encompasses kernels with a polynomial decay, such as, for example, the rational quadratic kernel and its variants with polynomial decay (1.2). While certainly less popular than the Gaussian kernel, the rational quadratic kernel is commonly listed among the standard covariance functions (i.e. kernels) in the literature on Gaussian processes. See, e.g., Section 4.2.1 of [RW06], as well as Section 5.4 of the same book, which in particular presents settings where the rational quadratic covariance assumption leads to improvements of the Gaussian.*

Theorem 1.1. (Main result (Informal – see Theorem 4.1)) *For every $\epsilon \in (0,1)$ and $\mu > 0$ there exists a randomized data structure for kernel evaluation of (L,t)-smooth p.d radial kernels K with polynomial preprocessing time,*

$$\textit{space complexity: } n \cdot L \cdot (d\log(n\Phi/(\epsilon\mu)))^{O(t)}/\epsilon \qquad \textit{and} \qquad \textit{query time: } L \cdot (d\log(n\Phi/(\epsilon\mu)))^{O(t)}/\epsilon$$

which outputs a multiplicative $(1\pm\epsilon)$-approximation to kernel evaluations whenever the kernel evaluation is at least μn, where Φ is the aspect ratio of the points.[4]

[2] One should interpret the dimensionality d as being $\omega(\log n) \leq d \leq n^{o(1)}$. This means that exponential dependencies on d should be avoided (as they incur super-polynomial factors in n), but arbitrary polynomial dependencies are allowed.

[3] In fact, the data structure of [PT20] has a Las Vegas guarantee; it uses randomness to build the data structure, but then guarantees that all queries are correct.

[4] The dependence on Φ seems necessary when making no assumptions on the smooth kernels. This dependence can be removed under a mild assumption on the decay of the kernel (which holds for the Cauchy and Student-t kernels listed as examples).

The main conceptual contribution behind the result is a framework for combining discrepancy techniques with randomized space partitioning. To the best of our knowledge, ours is the first work that succeeds in combining these lines of work. At a technical level the two main innovations that we introduce are **(a)** a strong bound on the discrepancy of the kernel matrix for smooth kernels under a geometric separability assumption akin to the one used in Fast Multipole Methods and **(b)** a way to partition a dataset (and potential queries) into a small number of pieces that ensure separation.

Dependence on Φ. We note that a dependence on the aspect ratio Φ of the dataset is also present in [BCIS18a], and seems necessary for general smooth kernels. For instance, a natural approach to removing it would be to **(a)** remove data points that are poly$1/(\epsilon\mu)$ far from the query point from consideration and **(b)** discretize points without significantly changing the kernel values but ensure that non-equal points have a minimum distance—**(a)** and **(b)** effectively upper bounds the aspect ratio Φ by upper bounding the maximum distance and lower bounding the minimum distance. However, this does not work without a closer look at the specific kernel function K. For example, the counting kernel $\mathsf{K}(x,y) = 1$ for all x, y would mean one cannot do **(a)**. For **(b)**, consider the kernel $\mathsf{K}(x,y) = \frac{1}{1+\sigma^2\|x-y\|^2}$, where we let σ go to infinity. This kernel is smooth as per our definition for every σ, independent of L. At the same time, the value of the kernel starts dropping sharply at $\|x-y\|_2 \approx 1/\sigma$, so one should not give a discretization independent of σ. At the same time, the dependence on Φ can be removed for specific kernels such as the Cauchy or the t-Student kernel using the approach outlined above.

Dependence on $(d\log(n\Phi/(\epsilon\mu)))^{O(t)}$. Our techniques will naturally incur a poly-logarithmic dependence on $d\log(n\Phi/(\epsilon\mu))$, where the specific power of the exponent will be $O(t)$ for (L,t)-smooth kernels K. For the Cauchy kernel or Student-t kernel with β and t being $O(1)$, the additional factors are poly-logarithmic. Interestingly, the dependence on the smoothness t in [BCIS18a] is $2^{O(t)}$, which becomes important once $t = \Theta(\log n)$.

1.1 Future Directions and Open Problems

We hope that our techniques, which allow one to combine discrepancy methods with randomized space partitioning, will prove instrumental in resolving other exciting questions in numerical linear algebra with kernel matrices. We mention two prominent questions here.

- Kernel density estimation for Gaussian Kernel. Does there exist a data structure for kernel density estimation for the Gaussian kernel (i.e., $\mathsf{K}(p,q) = \exp(-\|p-q\|_2^2/2)$) with polynomial space complexity and query time poly$d/(\epsilon\cdot\mu^{0.99})$? The Gaussian kernel is not "smooth" so the approach from this paper would degrade to $(d\log(1/\mu))^{\Omega(d)}/\epsilon$. A specific hurdle is that our partitioning techniques only ensure an $\Omega(1/\sqrt{d})$ relative separation of the query and dataset (here 'relative' refers to the size of a bounding Euclidean ball - see Section 2 for more details). However, a natural extension of our data-dependent embeddings to the Gaussian kernel incurs an exponential dependence on the inverse of this relative separation – see Remark 3.1 for more details.

- Fast Multipole Methods with a polynomial dependence on dimension. In fast multipole methods one approximates a kernel matrix by first using a crude (ℓ_∞ ball carving) space partitioning to partition space into bounded regions, and then Taylor expands the kernel arounds centers of these regions to approximate interactions between well-separated data points (see, e.g., [BG97]). Both steps incur an exponential in the dimension loss: the former because a constant factor separation is ensured in the crude ℓ_∞ ball-carving used, which requires breaking an ℓ_∞ ball into $d^{\Omega(d)}$ balls of factor $\sqrt{d}$ smaller radius. The latter because a Taylor expansion has an exponential number of terms in the dimension of the dataset. The latter exponential loss was recently overcome by the work of [AKK+20a], who showed how to *sketch* the polynomial kernel (i.e., the Taylor expansion) with only a polynomial in d dependence. However, the approach of [AKK+20a] suffers from a polynomial dependence on the radius of the dataset, as they do not supply a corresponding space partitioning primitive. Our space partitioning methods are able to exploit only a $\Omega(\frac{1}{\sqrt{d}})$ relative separation, and only result in a polynomial in d dependence: can our techniques be used together with [AKK+20a] to optimally sketch kernel matrices?

2 Technical Overview

In this section, we give an overview of the techniques involved. In order to highlight our main contributions, it will be useful to consider the case of the 2-Student kernel $\mathsf{K}\colon \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$ for concreteness. This kernel is

given by

$$\mathsf{K}(x,y) = \frac{1}{1+\|x-y\|_2^2}.$$

One receives a dataset $P \subset \mathbb{R}^d$ of points and seeks to process them into a data structure to support kernel evaluation queries. Namely, a query $q \in \mathbb{R}^d$ will come, and we want to estimate $\sum_{p\in P} \mathsf{K}(p,q)$ up to a factor of $1 \pm \epsilon$, and similarly to [CS17a, CS19, BCIS18b, BIW19, CKNS20a] we will parametrize our time and space complexity in terms of $\mu = (1/n)\sum_{p\in P} \mathsf{K}(p,q)$.

We start by explaining two prior approaches which will be important ingredients in our scheme. First, a simple random sampling approach, and then the prior work of [PT20] which shows how to use discrepancy theory to obtain an improved dependence on ϵ. Then, we overview our approach. First, we show how we may improve on the discrepancy bounds when datasets are "well-separated" from a query, and then how to algorithmically utilize the improved discrepancy bounds in the worst-case.

Random sampling as repeated coloring. It is not hard to see that, since kernel values are always between 0 and 1, a uniformly random sample from P of size $O(1/(\epsilon^2\mu))$ will approximate the kernel evaluation of any q with probability at least 0.9. More generally, one may take samples from an unbiased estimator of $(1/n)\sum_{p\in P} \mathsf{K}(p,q)$, and show that the estimate is good via bounding the variance of the estimator. If proceeding with this plan, the quadratic dependence on $1/\epsilon$ is a consequence of using Chebyshev's inequality; since the probability that the estimate is off by more than $\epsilon\mu$ (the quantity we want to minimize by taking more samples) becomes at most the variance divided by $\epsilon^2\mu^2$. To facilitate comparison with discrepancy-based approaches, we now sketch an alternative derivation of this result. And for this purpose it will be useful to instead think of repeatedly sampling data points in P with probability $1/2$ and analyzing how errors in the corresponding KDE estimates accumulate.

Suppose that we would like to subsample the dataset P to a subset P' containing about half of the points in P while preserving KDE value. It is convenient to think of this process as coloring the points in P

$$\chi\colon P \to \{-1,1\},$$

and then letting the 'subsampled' dataset P' contain points that were colored 1, say:

$$P' = \{p \in P : \chi(p) = 1\}.$$

We thus would like to find a coloring χ of P such that

$$\left| \frac{1}{|P|}\sum_{p\in P} \mathsf{K}(p,q) - \frac{2}{|P|}\sum_{p\in P'} \mathsf{K}(p,q) \right| \le \epsilon\mu. \tag{2.5}$$

The left hand side of (2.5) can be expressed as

$$\begin{aligned}\left| \sum_{p\in P} \mathsf{K}(p,q) - 2\sum_{p\in P'} \mathsf{K}(p,q) \right| &= \left| \sum_{p\in P} \mathsf{K}(p,q) - 2\sum_{p\in P:\chi(p)=1} \mathsf{K}(p,q) \right| \\ &= \left| \sum_{p\in P:\chi(p)=-1} \mathsf{K}(p,q) - \sum_{p\in P:\chi(p)=1} \mathsf{K}(p,q) \right| \\ &=: \operatorname{disc}_{\mathsf{K}}(P,\chi,q),\end{aligned}$$

the discrepancy of coloring χ with respect to query q. Choosing P' to be a uniformly random subset of P containing every data point independently with probability $1/2$ amounts to a uniformly random coloring χ of P, and a simple calculation shows that

$$\operatorname{disc}_{\mathsf{K}}(P,\chi,q) = O\left(\sqrt{\sum_{p\in P} \mathsf{K}(p,q)^2}\right) = O\left(\sqrt{\sum_{p\in P} \mathsf{K}(p,q)}\right) = O\left(\sqrt{\mu\cdot|P|}\right)$$

for every kernel bounded by 1, with constant probability. Substituting back into (2.5) and taking the normalizing factor of $\frac{1}{|P|}$ into account, we get that the error introduced by subsampling is below $\epsilon\mu$ as long as

$$\frac{1}{|P|} \cdot O\left(\sqrt{\mu \cdot |P|}\right) = O\left(\sqrt{\frac{\mu}{|P|}}\right) \ll \epsilon\mu.$$

This means that we can keep subsampling[5] while $|P| \gg 1/(\epsilon^2\mu)$. This recovers the bound from Chebyshev's inequality[6].

Better colorings via Banaszczyk's theorem [PT20]. Instead of selecting the coloring randomly, [PT20] note that since K is a p.d kernel with $\mathsf{K}(p,q) \leq 1$ for all p, q, there exists an embedding ϕ such that for every $p, q \in \mathbb{R}^d$ one has

$$\|\phi(p)\|_2 \leq 1, \|\phi(q)\|_2 \leq 1$$

as well as

$$\mathsf{K}(p,q) = \langle \phi(p), \phi(q) \rangle.$$

The existence of a coloring χ such that

$$\text{disc}_{\mathsf{K}}(P, \chi, q) \leq O(\gamma_2(\mathsf{K})) = O(1) \cdot \left(\max_{p \in \mathbb{R}^d} \|\phi(p)\|_2^2\right) = O(1) \tag{2.6}$$

for all q then follows by Banaszczyk's theorem – see Theorem 3.1.[7] Here $\gamma_2(\mathsf{K})$ is the γ_2-norm of the kernel matrix K, which provides a commonly used route for upper bounding discrepancy – see Section 3 for more details. Substituting this bound into (2.5), we get that the error introduced by subsampling is below $\epsilon\mu$ as long as

$$\frac{1}{|P|} \cdot O(1) \geq 1/(\epsilon\mu). \tag{2.7}$$

This means that we can keep subsampling while $|P| \gg 1/(\epsilon\mu)$. This is a quadratic improvement on the ϵ-dependence from random sampling, but far short of our goal of $\text{polylog}(1/\mu)/\epsilon$. Furthermore, the bound on discrepancy provided by (2.6) is **tight** in general.

Our approach: discrepancy bounds for well-separated datasets. In order to get around the tightness of the above bound for general datasets, we show that every dataset P can be decomposed into a small number of datasets that are 'nice' with respect to any fixed query $q \in \mathbb{R}^d$. This decomposition is independent of the query, and relies on randomized space partitioning in high dimensions akin to locality sensitive hashing. We then design specialized feature embeddings for each element in this decomposition to establish a significantly stronger upper bound on the discrepancy of the corresponding sub-dataset with respect to q.

Our geometric assumptions are inspired by those in Fast Multipole Methods [BG97]. In the Fast Multipole Methods, the space $\mathbb{R}^d$ is *deterministically* recursively partitioned with ℓ_∞-balls of geometrically decreasing diameter. The benefit of using ℓ_∞-balls is that they tile $\mathbb{R}^d$, and whenever p and q belong to two different ℓ_∞-balls which are not adjacent (share a corner) at a particular radius, p and q are separated by at least the diameter. The downside is that, since Euclidean separation ultimately matters, one must decrease the radius by at least a constant (and, in fact, at least $\sqrt{d}$) factor at every level – and this leads to an exponential dependence in the dimension as the tree encoding the recursive partition has degree $\exp(\Omega(d))$. Our recursive partitioning will be *randomized* and use ℓ_2-balls of randomly chosen radii. With such a partitioning scheme one can ensure a weaker separation, namely a relative $\Omega(\frac{1}{\sqrt{d}})$ separation. We show, however, that this suffices! Specifically, we show strong discrepancy bounds when either

(1) the dataset P lies in a spherical shell and the query q lies in a slightly larger spherical shell (see Fig. 1)

or

[5] One can verify that the total error induced by a sequence of recolorings is dominated by the error introduced in the last step – see Section 3 for more details.

[6] Of course, this derivation also uses Chebyshev's inequality in bounding the discrepancy of a random coloring; however, as we show next, it readily generalizes to settings when the coloring is obtained by a more careful method than uniformly random choice.

[7] Banaszczyk's theorem gives a distribution over (random) colorings which achieves a $O(1)$ discrepancy for each q with very high probability, so in general, there is a $O(\sqrt{\log m})$-factor, where m denotes the number of rows, i.e., queries, of the kernel matrix which one wishes to support.

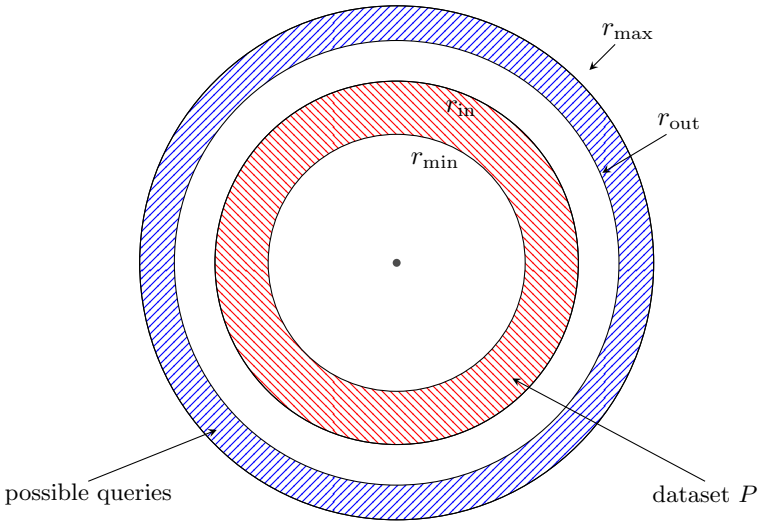

Figure 1: Illustration of the well-separated setup. The point $c \in \mathbb{R}^d$ is the center of ball, which may be assumed to be the origin after a translation, and we consider two well-separated shells at radii between $[r_{\min}, r_{\text{in}}]$ and $[r_{\text{out}}, r_{\max}]$ centered at c, where $r_{\text{in}} < r_{\text{out}}$. We consider dataset points which lie in the inner shell, with distance between $r_{\min}$ and r_{in} from c, and queries which lie in the outer shell, with distance between r_{out} and $r_{\max}$ from c. This is the case we consider throughout the technical overview; the symmetric case when the dataset is inside a low radius shell and query is outside will be analogous.

(2) the query q lies inside a spherical shell and dataset P lies in a slightly larger spherical shell (the opposite of **(1)**).

This in particular is where we crucially use the assumption that the kernel is smooth – see Remark 3.1 in Section 3 for more details.

In what follows we give a more detailed outline of how our feature embeddings are constructed, using the 2-Student kernel as a running example. We first define a basic feature embedding, then explain how to apply discrepancy theory to achieve good colorings via the γ_2 norm of the embedding and then talk about our modified feature embeddings that achieve strong discrepancy bounds in settings **(1)** and **(2)** above.

Feature Embeddings for Smooth Kernels. The crucial property of positive definite kernels K is that they may be represented as inner products in a (potentially infinite dimensional) feature space. Consider the following explicit construction for the 2-Student kernel, which proceeds by taking the inverse Laplace transform of $1/(1+\|x-y\|_2^2)$ and a Taylor expansion of e^x:

$$\begin{aligned}\mathsf{K}(x,y) = \frac{1}{1+\|x-y\|_2^2} &= \int_{t:0}^{\infty} e^{-t(\|x-y\|_2^2+1)}dt = \int_{t:0}^{\infty} e^{-t\|x\|_2^2}e^{-t\|y\|_2^2}e^{2t\langle x,y\rangle}e^{-t}dt \\ &= \int_{t:0}^{\infty} e^{-t\|x\|_2^2}e^{-t\|y\|_2^2}\cdot e^{-t}\sum_{k=0}^{\infty}\frac{(2t)^k}{k!}\cdot\langle x^{\otimes k}, y^{\otimes k}\rangle dt.\end{aligned}$$

We may now consider an embedding ϕ which takes as input a vector $x \in \mathbb{R}^d$ and outputs the (infinite-dimensional) function $\phi(x)$ whose inputs are a number $t \in [0,\infty)$, an index $k \in \mathbb{Z}_{\geq 0}$, and sets

$$\phi(x)(t,k) := e^{-t\|x\|_2^2}\sqrt{\frac{(2t)^k}{k!}}\cdot e^{-t/2}\cdot x^{\otimes k} \in \mathbb{R}^{d^k}. \tag{2.8}$$

This representation has the benefit that for all $x, y \in \mathbb{R}^d$

$$\mathsf{K}(x,y) = \langle\phi(x),\phi(y)\rangle,$$

where $\langle\phi(x),\phi(y)\rangle = \int_{t:0}^{\infty}\sum_{k=0}^{\infty}(\phi(x)(t,k))^{\mathsf{T}}(\phi(y)(t,k))dt$.

Discrepancy on the Feature Space. The key to applying discrepancy minimization algorithms is understanding the so-called γ_2-norm of the kernel matrix, since this will govern the discrepancy that we may achieve (and hence the number of times that we may halve the dataset). Consider the kernel matrix $A = (\mathsf{K}(q,p))$ where rows are indexed by a set of possible queries Q and columns are indexed by dataset points P. The γ_2-norm of A is the minimum, over all factorization of A into UV, where U is a $|Q| \times d'$ matrix and V is a $d' \times |P|$ matrix, of the maximum row norm of Q times the maximum column norm of P. By placing $\phi(q)$ as the rows of U for each potential query in Q and placing $\phi(p)$ as the columns of V for each dataset point in P, we obtain that

$$\gamma_2(A) \leq \max_{q \in Q} \|\phi(q)\|_2 \cdot \max_{p \in P} \|\phi(p)\|_2.$$

By construction, $\|\phi(q)\|_2^2 = \langle \phi(q), \phi(q) \rangle = \mathsf{K}(q,q)$ which equals 1 for any q (and similarly p) – this gives the bound on the γ_2-norm used by [PT20]. As we show below, however, significantly stronger upper bounds on the γ_2-norm can be obtained if the dataset is well-separated from the query in an appropriate way – this, coupled with a new hashing-based procedure for reducing to the well-separated setting, will lead to our improved bounds.

Modifying the Feature Embedding. Suppose that every dataset point p in P was contained within a shell of inner radius $r_{\min}$ and outer radius r_{in}, and that every query q in Q which we will consider was contained within a shell of inner radius r_{out} (which is larger than r_{in}) and outer radius $r_{\max}$ (the symmetric case when the query is inside a shell close to the origin and the dataset points are in a shell far from the origin will be analogous). See Fig. 1 for an illustration.

In Section 3.1, we show that a configuration gives an improved bound on the γ_2 norm of the kernel matrix. Indeed, the fact that q has norm which is at least r_{out} and every p in P has norm which is at most r_{in} guarantees that the points p and q are not too close to each other, i.e.,

$$\|p - q\|_2 \geq r_{\text{out}} - r_{\text{in}}. \tag{2.9}$$

For example, if we could support a small additive error $\xi > 0$ (which we will later incur a logarithmic dependence, so we will set ξ to $\epsilon\mu/n$), then, it suffices to "cut off" the feature embedding at

$$t_0 := O\left(\frac{\ln(1/\xi)}{(r_{\text{out}} - r_{\text{in}})^2}\right),$$

because for any such "well-separated" pair of points $p, q \in \mathbb{R}^d$,

$$\int_{t:t_0}^{\infty} e^{-t(\|p-q\|_2^2+1)} dt \leq e^{-t_0(r_{\text{out}}-r_{\text{in}})^2} \int_{t:t_0}^{\infty} e^{-t} dt \leq \xi.$$

Once we introduce this change, we will exploit the fact that $\|p\|_2 \in [r_{\min}, r_{\text{in}}]$ and $\|q\|_2 \in [r_{\text{out}}, r_{\max}]$ in order to modify the embedding to ϕ' such that the norm of $\phi'(p)$ will not increase too much, but the norm of $\phi'(q)$ will decrease a significant amount. Overall, we will show that $\|\phi'(p)\|_2 \|\phi'(q)\|_2$, which gives an upper bound on the γ_2-norm of the matrix, will be much smaller. In particular, for a setting of $\rho > 1$, we introduce the change

$$\phi'(p)(t,k) = \phi(p)(t,k) \cdot \rho^k \quad \text{and} \quad \phi'(q)(t,k) = \frac{\phi(q)(t,k)}{\rho^k}, \tag{2.10}$$

and ϕ' certifies an improved bound on the γ_2-norm of an additive ξ-perturbation of kernel matrices. In particular, we upper bound the product of $\|\phi'(p)\|_2 \|\phi'(q)\|_2$ while using the fact that $\|p\|_2 \in [r_{\min}, r_{\text{in}}]$ and $\|q\|_2 \in [r_{\text{out}}, r_{\max}]$. We point the reader to Section 3.1 with $G(t) = (1+t)^{-1}$, where we show that this product can be at most,

$$\|\phi'(p)\|_2 \|\phi'(q)\|_2 \lesssim \frac{\ln(1/\xi) \cdot r_{\text{in}}^2}{(r_{\text{out}} - r_{\text{in}})^2 \cdot r_{\text{out}} \cdot r_{\min}} \leq \frac{\ln(1/\xi) \cdot r_{\text{in}}}{(r_{\text{out}} - r_{\text{in}})^2 \cdot r_{\min}}. \tag{2.11}$$

The above bound gives us the upper bound on the discrepancy that we will achieve, and this will dictate how many times we may half the dataset and incur at most $\epsilon\mu$ error. Importantly, since the query q and every dataset point p considered (inside the shell) satisfies by (2.9), we have a lower bound on what each point p from the shell contributes to the kernel evaluation to a query $q \in Q$,

$$\mu \geq \min_{p \in P, q \in Q} \mathsf{K}(p,q) \geq \frac{1}{1 + (r_{\text{in}} + r_{\max})^2},$$

since $r_{\text{in}} + r_{\max}$ is an upper bound on the maximum distance from q to P. We will be able to ensure that $r_{\max} = O(r_{\text{out}})$, $r_{\text{in}}/r_{\min} = O(1)$, and that $r_{\text{out}} - r_{\text{in}} \geq \Omega(r_{\text{out}}/\sqrt{d})$ (we expand on why in the next subsection). Overall, this means that

$$\mu \geq \Omega\left(\frac{1}{1 + r_{\text{out}}^2}\right).$$

Since we had $r_{\text{in}}/r_{\min} = O(1)$ and $(r_{\text{out}} - r_{\text{in}})^2 \geq r_{\text{out}}^2/d$, when we use the discrepancy bound in (2.11), each step of halving incurs error $O(\ln(1/\xi) \cdot r_{\text{out}}^2/d)$, and similarly to the discussion in (2.7), we may continue decreasing the dataset until

$$\frac{1}{|P|} \cdot O\left(\frac{\ln(1/\xi) \cdot d}{r_{\text{out}}^2}\right) \lesssim \frac{\epsilon}{1 + r_{\text{out}}^2} \qquad \Longrightarrow \qquad |P| \gtrsim \frac{d \ln(1/\xi)}{\epsilon}.$$

Even though we have specialized the discussion to the 2-Student kernel, we use a characterization of positive definite radial kernels due to Schoenberg in order to use the above embeddings in general. We note that the smoothness assumption come in the following way. Note that the γ_2-norm bound depends on how $r_{\min}, r_{\text{in}}$ and r_{out} relate to each other, which will factor into the number of times we may halve the dataset while incurring at most $\epsilon\mu$ in the error. This must be compared to $\mathsf{K}(p, q)$ (which depends on $\|p - q\|_2$) so that the additive error can be absorbed into $\epsilon \cdot \mathsf{K}(P, q)/|P|$. (See also, Remark 3.1.)

REMARK 2.1. *Taking a broader perspective on kernel methods for high-dimensions, we are not aware of any prior work which adapt the feature embeddings to the specific dataset for improved algorithms. As an example, sampling techniques for the Nyström method [MM17], random Fourier features [RR08, AKM+17], and sketching methods [ANW14, ACW17a, ACW17b, AKK+20b] always consider the feature embedding $\phi\colon \mathbb{R}^d \to L_2$ which gives $\langle \phi(p), \phi(q) \rangle = \mathsf{K}(p, q)$ for all $p, q \in \mathbb{R}^d$. Our approach, of dividing the dataset and adapting the feature embeddings to the various parts of the dataset, fits nicely within a recent line-of-work on "data-dependent" techniques for high-dimensional datasets [AINR14, AR15, ALRW17, ANN+18, CKNS20b, CJLW22], and we are hopeful that such techniques are applicable in other algorithmic contexts*

Maintaining Separation for Coresets. It remains to show that we can build a data structure which always constructs coresets while guaranteeing a separation between queries and dataset points. The idea is to proceed via ball carving of randomly chosen radii. Suppose, for instance, that all dataset points P lie within a ball of radius $R > 0$, and let c denote the center of that ball. We consider three cases, corresponding to where the (unknown) query q may be in comparison to the ball. The first two cases are relatively simple to handle, and most of the work involves the last case.

Case 1 (easy): q is much farther than R/ϵ from c. Then, since any p lies within distance R from c, one may use the triangle inequality to conclude that the kernel evaluation of $\mathsf{K}(q, p)$ and $\mathsf{K}(q, c)$ is the same up to $1 \pm \epsilon$. In this case, a data structure just needs to remember the center c and the size of the dataset $|P|$, so that one can output $|P| \cdot \mathsf{K}(q, c)$ to approximate $\mathsf{K}(P, q)$. See Definition 4.1 and Claim 4.1.

Case 2 (relatively easy): q is farther than $3R$ but within R/ϵ of c. In this case, we can utilize the coreset construction. Indeed, we "guess" the distance from q to c (for which there are at most $O(\log(1/\epsilon))$ many choices). Let $R' \geq 3R$ a guessed distance such that $\|q - c\|_2 \approx 3R'$. Pick a randomly chosen point c' drawn from $B_2(c, R'/2)$ uniformly and use that as our new "center." Every point in P will be within a ball of radius $(R'/2 + R) \leq 3R'/2$ from c' (by the triangle inequality), and at distance at least $R'/100$ from c' (with high probability for $d = \omega(\log n)$). In addition, the query is within distance at least $5R'/2$ from c'. Setting $r_{\min} = R'/100$, $r_{\text{in}} = 3R'/2$, and $r_{\text{out}} = 5R'/2$, we can always guarantee an upper bound on the γ_2-norm of $O(1/(R')^2)$. In addition, the $\mathsf{K}(q, p) \geq O(1/(R')^2)$, so the repeated halving technique of [PT20] will give the desired coreset. This is handled in Section 4.3.

Case 3 (the difficult case). This case occurs when q is within $3R$ from c – see Fig. 2 for an illustration. Here, we want to partition the space such that, as in the fast multipole method, we can guarantee some amount of separation, without the exponential dependency obtained from partitioning into cubes (as is usually done in the fast multipole methods). Consider a point p from P and a query q such that $\|p - q\|_2 = \Omega(R)$. We will consider the following randomized partition. First, we sample a random point c' within $O(R)$ from c, and then we sample a (random) radius r on the order of R. The hope is that p falls inside the ball around c' of radius r, and that q falls outside the ball. In order to apply the improved discrepancy bound for well-separated datasets, we must also fit a shell inner radius r_{in} and outer radius r_{out}, where $r_{\text{out}} - r_{\text{in}} = \Omega(R/\sqrt{d})$. We point the reader to Lemma 4.1.

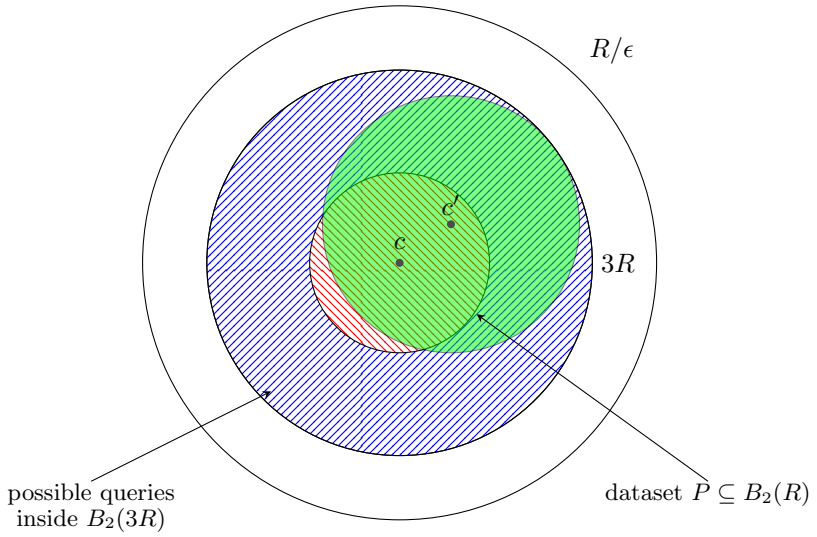

Figure 2: Illustration of ball-carving with balls of radius $O(R)$ when query is at distance at most $3R$ from c.

where we show this partitioning procedure separates each $p \in P$ with $\|p - q\|_2 = \Omega(R)$ with probability at least $\Omega(1/\sqrt{d})$, so after repeating $O(\sqrt{d} \log n)$ times, we are guaranteed to have separated every dataset point p at distance $\Omega(R)$ from q.

In the data structure, we sample a ball and consider the points within the inner-shell and the points outside the outer shell. We construct a coreset for the points inside the inner-shell, and we will use these coresets to evaluate queries which come outside the outer shell (to guarantee separation). Whenever queries come outside the outer shell, the points inside the inner shell are "captured," and the coreset approximates their contribution. For these queries, it remains to recurse on the dataset points which were not captured by the coreset. Similarly, we construct a coreset of the dataset points which evaluate queries which fall inside the inner shell (again, to guarantee separation), and we must recurse on the dataset points outside the outer shell. This recursive partitioning scheme is done in Algorithm 4.2.

Note that there is a small technical issue arising, since the points which fall within radius r_{in} and r_{out} from c' are replicated. While this may naively blow up the space of the data structure, the probability that any dataset point falls inside this shell can be made small. The is done by decreasing $r_{\text{out}} - r_{\text{in}}$ to $\alpha R/\sqrt{d}$, at a cost of an increase in the coreset size.

Organization. In the rest of the paper we first present the formal construction and analysis of our improved coresets for well-separated points in Section 3. We then present the details of our reduction to the case of well-separated datasets and the final algorithm in Section 4.

3 Structural Result: Improved Coresets for Well-Separated Shells

Given a p.d kernel K, the "kernel trick" refers to the fact there exists a feature embedding ϕ mapping $\mathbb{R}^d$ into a much larger and possibly infinite-dimensional space $\mathbb{R}^{d'}$ where $\mathsf{K}(x, y) = \langle \phi(x), \phi(y) \rangle$ for any $x, y \in \mathbb{R}^d$. As [PT20] show, whenever $\mathsf{K}(x, x) = 1$ for all x, the existence of such a feature embedding ϕ, as well as discrepancy minimization algorithms are useful for constructing coresets for kernel evaluation.

The approach proceeds as follows: one receives a dataset $P \subset \mathbb{R}^d$ and wants to support kernel evaluation queries for a finite set of queries $Q \subset \mathbb{R}^d$.[8] Then, consider the $|Q| \times |P|$ kernel matrix $A = (a_{qp})$ where the (q, p)-entry is $\mathsf{K}(q, p)$. Notice that $A \cdot \mathbf{1} \in \mathbb{R}^{|Q|}$ (where $\mathbf{1} \in \mathbb{R}^{|P|}$ is the all-1's vector) is the vector of kernel evaluations at each of the queries in Q. If one finds a vector $\chi \in \{-1, 1\}^{|P|}$ where $\|A \cdot \chi\|_\infty \leq \alpha$, then considering the partition of P into P_+ and P_- according to whether a $p \in P$ has $\chi_p = 1$ (in which case it belongs to P_+, and P_- otherwise) at least one of the subsets is smaller than $|P|/2$ and for any $q \in Q$, one may approximate the

[8] [PT20] show how to discretize Q to only need to consider $\exp(d)$-sized query sets, but since we will allow queries to fail with a small probability, the discretization will not be an issue in this work.

kernel evaluation by only considering that subset. Formally, for both sets S being P_+ and P_-,

$$\left|2\sum_{p\in S}\mathsf{K}(p,q)-\sum_{p\in P}\mathsf{K}(p,q)\right| = \left|\sum_{p\in P_+}\mathsf{K}(p,q)-\sum_{p\in P_-}\mathsf{K}(p,q)\right| = (A\chi)_q \le \alpha.$$

In essence, one decreases the size of the dataset by a factor of 2 and incurs an additive error of α on the kernel evaluation. One can bound α using discrepancy theory.

DEFINITION 3.1. (γ_2-NORM OF A MATRIX) *For a matrix $A \in \mathbb{R}^{m\times n}$, the γ_2-norm of A is given by*

$$\gamma_2(A) = \inf\left\{\|u_i\|_2 \cdot \|v_j\|_2 : UV = A,\ u_1,\dots,u_m \text{ are rows of } U \text{ and } v_1,\dots,v_n \text{ columns of } V\right\}.$$

THEOREM 3.1. (BANASZCZYK'S THEOREM) *For any matrix $A \in \mathbb{R}^{m\times n}$,*

$$\min_{\chi\in\{-1,1\}^n} \|A\cdot\chi\|_\infty \lesssim \gamma_2(A)\cdot\sqrt{\log m}.$$

The final ingredient is showing that $\gamma_2(A) \le 1$, which is a simple consequence of the existence of ϕ. In particular, consider the $|Q| \times d'$ matrix U whose rows correspond to $\phi(q)$ for $q \in Q$, and the $d' \times |P|$ matrix whose columns correspond to $\phi(p)$ for $p \in P$. Then, $A = UV$ since ϕ is a feature embedding of $\mathsf{K}(\cdot,\cdot)$, and the maximum row norm of U and column norm of V is at most 1, because $\|\phi(q)\|_2^2 = \mathsf{K}(q,q) = 1$ and similarly for p.

In this section, we will do two things.

1. First, we explicitly construct new feature embeddings for p.d radial kernels $\mathsf{K}\colon \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$. For parameters $0 < r_{\text{in}} < r_{\text{out}}$, as well as an additive error $\xi > 0$ (think of ξ as extremely small, since we will depend logarithmically on $1/\xi$), our new feature embedding ϕ will only map vectors $z \in \mathbb{R}^d$ which satisfy $\|z\|_2 \notin [r_{\text{in}}, r_{\text{out}}]$, i.e., they avoid a shell of outer radius r_{out} and inner radius r_{in} around the origin. Whenever $\|y\|_2 \le r_{\text{in}}$ and $\|x\|_2 \ge r_{\text{out}}$, then $\langle\phi(x),\phi(y)\rangle$ will be up to $\pm\xi$ the same as $\mathsf{K}(x,y)$, yet $\|\phi(x)\|_2 \cdot \|\phi(y)\|_2$ will be smaller than 1.

2. Second, we use the technique of [PT20] to repeatedly halve the dataset and construct the coreset whenever a query and dataset will be separated by a shell of inner radius r_{in} and outer radius r_{out}, and in addition between $r_{\min}$ and $r_{\max}$ from the origin. One (minor) difference is that we utilize the self-balancing walk of [ALS21] instead of algorithmic versions of Banaszczyk's theorem. This will lose a logarithmic factor in the size of the coreset, but has the benefit of being very simple.

3.1 New Feature Embeddings In what follows, for any two thresholds $0 < r_{\text{in}} < r_{\text{out}}$, we let $\text{Shell}(r_{\text{in}}, r_{\text{out}}) \subset \mathbb{R}^d$ be the set of vectors $x \in \mathbb{R}^d$ with $r_{\text{in}} < \|x\|_2 < r_{\text{out}}$. The main theorem of this section is the following.

THEOREM 3.2. *Let $G\colon \mathbb{R}_{\ge 0} \to [0,1]$ be such that $\mathsf{K}(x,y) = G(\|x-y\|_2^2)$ is a p.d kernel for every $\mathbb{R}^d$. For any two thresholds $0 < r_{\text{in}} < r_{\text{out}}$ and $\xi > 0$, there exists a map $\phi\colon \mathbb{R}^d \setminus \text{Shell}(r_{\text{in}}, r_{\text{out}}) \to L_2$ such that every $x, y \in \mathbb{R}^d$ where $0 < \|y\|_2 \le r_{\text{in}} < r_{\text{out}} \le \|x\|_2$ satisfy:*

- *The inner product $|\langle\phi(x),\phi(y)\rangle - \mathsf{K}(x,y)| \le \xi$.*
- *For any $z \in \mathbb{R}^d \setminus \text{Shell}(r_{\text{in}}, r_{\text{out}})$,*

$$\|\phi(z)\|_2^2 \lesssim G\left(\frac{(r_{\text{out}}-r_{\text{in}})^2}{\ln(1/\xi)}\cdot\frac{\|z\|_2^2}{r_{\text{in}}^2}\right).$$

We will prove Theorem 3.2 in the next subsection, but we note below that it directly implies an improvement on the γ_2-norm of kernel matrices of p.d kernels (up to a small additive error ξ). In particular, Theorem 3.2 implies that for any sets $P, Q \subset \mathbb{R}^d$ where $P \subset B_2(0, r_{\text{in}}) \setminus B_2(0, r_{\min})$ and $Q \not\subset B_2(0, r_{\text{out}})$ (or vice-versa), there exists a $|Q| \times |P|$ matrix $\tilde{A}$ which is entry-wise ξ-close to the $|Q| \times |P|$ kernel matrix $(\mathsf{K}(q,p))_{q\in Q, p\in P}$, and

$$\gamma_2(\tilde{A}) \lesssim \left(G\left(\frac{(r_{\text{out}}-r_{\text{in}})^2}{\ln(1/\xi)}\cdot\frac{r_{\text{out}}^2}{r_{\text{in}}^2}\right)\cdot G\left(\frac{(r_{\text{out}}-r_{\text{in}})^2}{\ln(1/\xi)}\cdot\frac{r_{\min}^2}{r_{\text{in}}^2}\right)\right)^{1/2}.$$

REMARK 3.1. (USING SMOOTHNESS TO RELATE $\gamma_2(\tilde{A})$ TO $\mathsf{K}(P,q)/|P|$) *Here, one can see where smoothness of the kernel becomes essential. As per (2.7), one can decrease the size of P while the error incurred from discrepancy, which is given by $\gamma_2(\tilde{A})/|P|$, is smaller than $\epsilon \cdot \mathsf{K}(P,q)/|P|$. Note that when $p \in P$ and $q \in Q$ are always at distance at most $2r_{\max}$, we know $\mathsf{K}(p,q) \geq G(4r_{\max}^2)$. Thus, we can continue decreasing the dataset while $(1/|P|) \cdot \gamma_2(\tilde{A})$ is significantly smaller than $\epsilon \cdot G(4r_{\max}^2)$, which occurs while*

$$|P| \gtrsim \frac{1}{\epsilon}\left(G\left(\frac{(r_{\text{out}}-r_{\text{in}})^2}{\ln(1/\xi)} \cdot \frac{r_{\text{out}}^2}{r_{\text{in}}^2}\right) \cdot G\left(\frac{(r_{\text{out}}-r_{\text{in}})^2}{\ln(1/\xi)} \cdot \frac{r_{\min}^2}{r_{\text{in}}^2}\right)\right)^{1/2} \frac{1}{G(4r_{\max}^2)}.$$

The smoothness comes in when computing the ratio of $G(\cdot)$'s, since the smoothness allows us to relate the $G(r_1^2)/G(r_2^2)$ in terms of r_2/r_1.

In particular, this is why extending our approach to kernels with faster decay requires interesting new ideas: for the Gaussian kernel, for example, the right hand side of the equation above is significantly larger than $\mathsf{K}(P,q)/|P|$, as the $\Theta(1/\sqrt{d})$ factor stemming from the separation that our partitioning scheme ensures affects the exponent.

3.2 Proof of Theorem 3.2 We will construct the map ϕ explicitly. In order to do so, we first recall Schoenberg's characterization of p.d radial kernels, as well as the Haussdorf-Bernstein-Widder theorem, which we will use in constructing ϕ. We point to Chapter 7 of [Wen04] for an extensive treatment of these topics.

THEOREM 3.3. (SCHOENBERG'S CHARACTERIZATION) *A kernel $\mathsf{K}(x,y) = G(\|x-y\|_2^2)$ is p.d if and only if the function G is completely monotone on $\mathbb{R}_{\geq 0}$, i.e., $G \in C^\infty(\mathbb{R}_{\geq 0})$, and*

$$(-1)^\ell \cdot G^{(\ell)}(t) \geq 0 \qquad \text{for all } \ell \in \{0\} \cup \mathbb{N} \text{ and } t \geq 0.$$

THEOREM 3.4. (HAUSSDORF-BERNSTEIN-WIDDER) *A function $G\colon \mathbb{R}_{\geq 0} \to \mathbb{R}$ is completely monotone if and only if there exists a non-negative finite Borel measure μ where*

$$G(\lambda) = \int_{t:0}^{\infty} e^{-t\lambda}\mu(dt).$$

We introduce the following notation:

$$t_0 = \frac{\log(1/\xi)}{(r_{\text{out}}-r_{\text{in}})^2} \qquad \text{and} \qquad \rho = 1 - \min\left\{\frac{1}{t_0 \cdot r_{\text{in}}^2}, \frac{1}{2}\right\}.$$

We show how to map each point $\phi(x)$ and $\phi(y)$ when $\|x\|_2 \geq r_{\text{out}}$ and $\|y\|_2 \leq r_{\text{in}}$. First, we let $u_x, v_y\colon [0,t_0] \times (\{0\} \cup \mathbb{N}) \to \mathbb{R}^*$ (where $\mathbb{R}^*$ consists of the union of all finite length tuples $\cup_{j\geq 1}\mathbb{R}^j$) be the functions given by

$$u_x(t,k) = e^{-t\|x\|_2^2} \cdot \frac{1}{\sqrt{k!}}\left(\sqrt{2t}\cdot\rho\right)^k \cdot \sqrt{\mu(t)} \cdot x^{\otimes k} \in \mathbb{R}^{d^k},$$
$$v_y(t,k) = e^{-t\|y\|_2^2} \cdot \frac{1}{\sqrt{k!}}\left(\sqrt{2t}\cdot\frac{1}{\rho}\right)^k \cdot \sqrt{\mu(t)} \cdot y^{\otimes k} \in \mathbb{R}^{d^k}.$$

The map $\phi(x)$ will consider the collection of functions $\{u_x(\cdot,k)\colon [0,t_0] \to \mathbb{R}^{d^k} : k \geq 1\}$ and "concatenate them." In particular, we let H denote the Hilbert space over functions $g\colon [0,t_0] \times \left(\cup_{k\geq 1}[d]^k\right) \to \mathbb{R}$, where $g_1, g_2 \in H$ have

$$\langle g_1, g_2\rangle := \int_{t:0}^{t_0} \sum_{k=0}^{\infty}\sum_{i=1}^{d^k} g_1(t,k,i)\cdot g_2(t,k,i)dt.$$

Then, $\phi(x)$ and $\phi(y)$ are the functions where $\phi(x)(t,k,i) = u_x(t,k)_i$ and $\phi(y)(t,k,i) = v_y(t,k)_i$. We note that the novelty in the above definitions is introducing the ρ and $1/\rho$ factors in u_x and v_y. In the absence of the ρ- and $1/\rho$-factors, the proof that we produce would recover features embeddings of unit norm. By introducing these

factors, we are able to exploit the fact that x and y have different norms. In particular, these terms cancel out when computing $\langle u_x, v_y\rangle$, but affect the norms $\|u_x\|_2^2$ and $\|v_y\|_2^2$. For instance, we have

$$\begin{aligned}\langle \phi(x), \phi(y)\rangle &= \sum_{k=0}^{\infty} \int_{t:0}^{t_0} \langle u_x(t,k), v_y(t,k)\rangle dt \\ &= \int_{t:0}^{t_0} e^{-t\|x\|_2^2 - t\|y\|_2^2} \sum_{k=0}^{\infty} \frac{1}{k!} \left(2t\langle x, y\rangle\right)^k \mu(dt) \\ &= \int_{t:0}^{t_0} e^{-t\|x\|_2^2 - t\|y\|_2^2 + 2t\langle x,y\rangle} \mu(dt) = \int_{t:0}^{t_0} e^{-t\|x-y\|_2^2} \mu(dt).\end{aligned} \tag{3.12}$$

By Theorem 3.4, we have

$$G(\|x-y\|_2^2) - \langle \phi(x), \phi(y)\rangle = \int_{t:t_0}^{\infty} e^{-t\|x-y\|_2^2} \mu(dt) \le \exp\left(-\frac{\ln(1/\xi)\|x-y\|_2^2}{(r_{\text{out}} - r_{\text{in}})^2}\right) \le \xi,$$

where we used the fact that $\|x-y\|_2 \ge (r_{\text{out}} - r_{\text{in}})$ by the triangle inequality and the fact that $\int_{t:t_0}^{\infty} \mu(dt) \le \int_{t:0}^{\infty} \mu(dt) = \mathsf{K}(0,0) \le 1$. In particular, $\mathsf{K}(x,y) - \xi \le \langle \phi(x), \phi(y)\rangle \le \mathsf{K}(x,y)$.

The norms $\|\phi(x)\|_2^2$ and $\|\phi(y)\|_2^2$ satisfy

$$\|\phi(x)\|_2^2 = \int_{t:0}^{t_0} e^{-2t\|x\|_2^2} \sum_{k=0}^{\infty} \frac{1}{k!} (2t\rho^2\|x\|_2^2)^k \mu(dt) \le \int_{t:0}^{t_0} e^{-2t\|x\|_2^2(1-\rho^2)} \mu(dt) \le G\left(2(1-\rho^2)\|x\|_2^2\right)$$

and

$$\begin{aligned}\|\phi(y)\|_2^2 &= \int_{t:0}^{t_0} e^{2t(1/\rho^2-1)\|y\|_2^2} \mu(dt) = \int_{t:0}^{t_0} e^{3t(1/\rho^2-1)\|y\|_2^2} \cdot e^{-t(1/\rho^2-1)\|y\|_2^2} \mu(dt) \\ &\le e^{3t_0(1-\rho^2)\|y\|_2^2/\rho^2} \int_{t:0}^{\infty} e^{-t(1-\rho^2)\|y\|_2^2} \mu(dt) = e^{3t_0(1-\rho^2)\|y\|_2^2/\rho^2} \cdot G\left(\frac{(1-\rho^2)\|y\|_2^2}{\rho^2}\right).\end{aligned}$$

Since $G(\cdot)$ is decreasing (because G is total monotone by assumption, so the derivative of G is always non-positive) and $\rho < 1$, we have

$$G(2(1-\rho^2)\|x\|_2^2) \le G\left(\frac{\|x\|_2^2}{t_0 \cdot r_{\text{in}}^2}\right) \qquad \text{and} \qquad G\left(\frac{(1-\rho^2)\|y\|_2^2}{\rho^2}\right) \le G\left(\frac{\|y\|_2^2}{t_0 \cdot r_{\text{in}}^2}\right)$$

The final upper bound on $\|\phi(y)\|_2^2$ comes from the fact

$$\frac{3t_0(1-\rho^2)\|y\|_2^2}{\rho^2} \lesssim t_0 \cdot \frac{1}{t_0 \cdot r_{\text{in}}^2} \cdot \|y\|_2^2 \le 1,$$

since $\|y\|_2 \le r_{\text{in}}$.

3.3 Coresets from New Feature Embeddings

DEFINITION 3.2. (SMOOTH FUNCTION [BCIS18A]) *For $L, t \ge 1$, a kernel $\mathsf{K}\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is (L,t)-smooth if for any three points $p_1, p_2, q \in \mathbb{R}^d$ with $p_1 \ne q \ne p_2$,*

$$\max\left\{\frac{\mathsf{K}(p_1,q)}{\mathsf{K}(p_2,q)}, \frac{\mathsf{K}(p_2,q)}{\mathsf{K}(p_1,q)}\right\} \le L \cdot \left(\max\left\{\frac{\|p_1-q\|_2}{\|p_2-q\|_2}, \frac{\|p_2-q\|_2}{\|p_1-q\|_2}\right\}\right)^t.$$

Suppose the kernel $\mathsf{K}(x,y) = G(\|x-y\|_2^2)$ is p.d for every $\mathbb{R}^d$, and in addition, is (L,t)-smooth. We will need one preliminary theorem which will follow from the (online) discrepancy minimization algorithm of [ALS21]. Then, we state and prove Lemma 3.1, which is the main lemma of this section.

THEOREM 3.5. (SELF-BALANCING WALK [ALS21]) *For any $n, d \in \mathbb{N}$, there exists a randomized algorithm which receives as input a set of vectors $V = \{v_1, \ldots, v_n\} \in \mathbb{R}^d$ and a parameter $\delta > 0$. The algorithm outputs a (random) subset $\mathbf{V}' \subset V$ such that, for any vector $u \in \mathbb{R}^d$, with probability at least $1 - \delta$,*

$$\left| \sum_{i \in \mathbf{V}'} \langle v_i, u \rangle - \sum_{i \notin \mathbf{V}'} \langle v_i, u \rangle \right| \lesssim \log(n/\delta) \cdot \|u\|_2 \cdot \max_{i \in [n]} \|v_i\|_2.$$

Furthermore, the algorithm does not require explicit access to V; it only requires oracle access to $\{\langle v_i, v_j \rangle\}_{i,j \in [n]}$.

Proof. The statement of Theorem 3.5 above does not explicitly appear in [ALS21], but readily follows from the proof of Theorem 1.1 in their paper. In particular, [ALS21] give a randomized algorithm, BALANCE, which receives as input a sequence of vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ of norm at most 1, and a failure probability δ.[9] The algorithm produces a sequence of (random) vectors $\boldsymbol{w}_0 = 0, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_n \in \mathbb{R}^d$ such that for any vector $u \in \mathbb{R}^d$ with $\|u\|_2 \le 1$,[10]

$$\mathop{\mathbb{E}}_{\boldsymbol{w}_i} \left[\exp\left(\frac{\langle \boldsymbol{w}_i, u \rangle^2}{240\pi \log(n/\delta)} \right) \right] \le \sqrt{2}.$$

In addition, as long as $|\langle \boldsymbol{w}_i, v_{i+1} \rangle| \le c = 30 \log(n/\delta)$ for all $i \in [n]$, then there is a setting of signs $\boldsymbol{\sigma} \in \{-1, 1\}^n$ such that every $i \in [n]$ satisfies

$$\boldsymbol{w}_i = \sum_{\ell=1}^{i} \boldsymbol{\sigma}_i v_i.$$

As in their proof of Theorem 1.1, one may take a union bound over the n steps and conclude that $|\langle \boldsymbol{w}_n, u \rangle| \le c$ except with probability at most δ. Theorem 3.5 stated above simply performs the above argument with $u' = u/\|u\|_2$ and $v'_i = v_i / \max_i \|v_i\|_2$ to obtain the setting signs $\boldsymbol{\sigma} \in \{-1, 1\}^n$, and sets $\mathbf{V}'$ to be those indices $i \in [n]$ where $\boldsymbol{\sigma}_i = 1$. ∎

The main result of this section is

LEMMA 3.1. *For $L, t \ge 1$, let $\mathsf{K} \colon \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$ be a p.d radial kernel which is (L, t)-smooth. There exists a randomized algorithm which receives as input a subset $X \subset \mathbb{R}^d$, four thresholds $0 < r_{\min} \le r_{\text{in}} < r_{\text{out}} \le r_{\max}$, and three parameters $\epsilon, \xi, \delta \in (0, 1)$. The algorithm outputs a random subset $\mathbf{S} \subset X$ and a number $T \ge 0$ which satisfy the following conditions.*

- *For any $q \in \mathbb{R}^d$, if $\|q\|_2 \in [r_{\text{out}}, r_{\max}]$ and $\|x\|_2 \in [r_{\min}, r_{\text{in}}]$ for every $x \in X$, or $\|q\|_2 \in [r_{\min}, r_{\text{in}}]$ and $\|x\|_2 \in [r_{\text{out}}, r_{\max}]$ for every $x \in X$, the following holds with probability at least $1 - \delta$,*

$$\left| 2^T \sum_{x \in \mathbf{S}} \mathsf{K}(x, q) - \sum_{x \in X} \mathsf{K}(x, q) \right| \le \epsilon \sum_{x \in X} \mathsf{K}(x, q) + 2|X|\xi.$$

- *The size of the subset $\mathbf{S}$ is bounded by*

$$|\mathbf{S}| \le \frac{\log^2(|X|/\delta)}{\epsilon} \cdot L \cdot \left(\frac{2 \cdot r_{\max}}{r_{\text{out}} - r_{\text{in}}} \cdot \frac{r_{\text{in}}}{r_{\min}} \sqrt{\ln(1/\xi)} \right)^t$$

Proof. Let $\phi \colon \mathbb{R}^d \setminus \text{Shell}(r_{\min}, r_{\text{out}}) \to L_2$ be the map of Theorem 3.2 instantiated with the parameter ξ. The algorithm will proceed in iterations and, go through $\ell \in \{0, \ldots, T\}$ specifying the sets

- $\mathbf{V}_0 = \{\phi(x) : x \in X\}$.

[9] We consider the minor modification to their algorithm, where the second condition of Line 4, that $\|w_{i-1}\|_\infty \le c$ is dropped, which allows us to set $c = 30 \log(n/\delta)$, as they do in their proof of Theorem 1.2. We note that the reason the check $\|w_{i-1}\|_\infty \le c$ is present in their algorithm is because in *online* discrepancy, one wants to ensure that every coordinate of any partial sum of vectors is small, whereas we will only care about the final vector.

[10] The specific constant of $240\pi \log(n/\delta)$ follows their bound of $4Lc$, since $L = 2\pi$ and $c = 30 \log(n/\delta)$.

- For $\ell \in [T]$, the set $\mathbf{V}_\ell$ is set to the smallest of $\mathbf{V}'_{\ell-1}$ or $\mathbf{V}_{\ell-1} \setminus \mathbf{V}'_{\ell-1}$, where $\mathbf{V}'_{\ell-1}$ is the output of algorithm from Theorem 3.5 with input $\mathbf{V}_{\ell-1}$ and failure probability set to δ/T.[11]

By definition of the procedure above, its simple to see that $|\mathbf{V}_T| \le |X|/2^T$. In order to show the approximation bound, we first note that

$$\left|2^T \sum_{x\in\mathbf{V}_T} \mathsf{K}(x,q) - \sum_{x\in X} \mathsf{K}(x,q)\right| \le \left|2^T \sum_{x\in\mathbf{V}_T} \langle\phi(x),\phi(q)\rangle - \sum_{x\in X}\langle\phi(x),\phi(q)\rangle\right| + (|X| + 2^T|\mathbf{V}_T|)\cdot\xi, \tag{3.13}$$

and because of the fact $|\mathbf{V}_T| \le |X|/2^T$, we have $|X| + 2^T|\mathbf{V}_T| \le 2|X|$. This handles the claimed additive error. Then, we also have

$$\begin{aligned}\left|2^T \sum_{x\in\mathbf{V}_t} \langle\phi(x),\phi(q)\rangle - \sum_{x\in X}\langle\phi(x),\phi(q)\rangle\right| &\le \sum_{\ell=0}^{T} 2^{\ell-1}\left|2\sum_{x\in\mathbf{V}_\ell}\langle\phi(x),\phi(q)\rangle - \sum_{x\in\mathbf{V}_{\ell-1}}\langle\phi(x),\phi(q)\rangle\right| \\ &= \sum_{\ell=0}^{T} 2^{\ell-1}\left|\sum_{x\in\mathbf{V}'_{\ell-1}}\langle\phi(x),\phi(q)\rangle - \sum_{x\in\mathbf{V}_{\ell-1}\setminus\mathbf{V}'_{\ell-1}}\langle\phi(x),\phi(q)\rangle\right| \\ &\lesssim \sum_{\ell=0}^{T} 2^{\ell-1}\cdot\log(|X|T/\delta)\cdot G\left(\frac{(r_{\text{out}}-r_{\text{in}})^2}{\ln(1/\xi)}\cdot\frac{r_{\min}^2}{r_{\text{in}}^2}\right), \end{aligned} \tag{3.14}$$

where the final line applies the upper bound of Theorem 3.5, as well as the fact that

$$\|\phi(q)\|_2 \cdot \max_{x\in X}\|\phi(x)\|_2 \lesssim G\left(\frac{(r_{\text{out}}-r_{\text{in}})^2}{\ln(1/\xi)}\cdot\frac{r_{\min}^2}{r_{\text{in}}^2}\right).$$

from Theorem 3.2. Furthermore, we notice that $\|x-q\|_2 \le 2r_{\max}$ for every $x \in X$, which means $\mathsf{K}(x,q) \ge G(4r_{\max}^2)$. Using the fact that K is (L,t)-smooth, we may always upper bound (3.14) by

$$\left(\frac{2^T\log\left(\frac{|X|T}{\delta}\right)}{|X|}\right)\cdot L\cdot\left(\frac{2\cdot r_{\max}}{r_{\text{out}}-r_{\text{in}}}\cdot\frac{r_{\text{in}}}{r_{\min}}\cdot\sqrt{\ln(1/\xi)}\right)^t\sum_{x\in X}\mathsf{K}(x,q),$$

and the above bound is smaller than ϵ as long as we set

$$T = \log_2\left(\frac{\epsilon\cdot|X|}{\log^2(|X|/\delta)}\cdot\frac{1}{L}\cdot\left(\frac{r_{\text{out}}-r_{\text{in}}}{2\cdot r_{\max}}\cdot\frac{r_{\min}}{r_{\text{in}}}\cdot\frac{1}{\sqrt{\ln(1/\xi)}}\right)^t\right),$$

and results in a set $\mathbf{S}$ of size at most

$$|\mathbf{S}| \le \frac{|X|}{2^T} \le \frac{\log^2(|X|/\delta)}{\epsilon}\cdot L\cdot\left(\frac{2\cdot r_{\max}}{r_{\text{out}}-r_{\text{in}}}\cdot\frac{r_{\text{in}}}{r_{\min}}\cdot\sqrt{\ln(1/\xi)}\right)^t.$$

□

The above lemma readily implies the following theorem, which we state so that we can directly invoke this in the subsequent sections. Specifically, Lemma 3.1 can produce a coreset $\mathbf{S}$, so if we store the coreset points, the parameter T, and a translation vector $c \in \mathbb{R}^d$.

LEMMA 3.2. *Let $n, d \in \mathbb{N}$, and suppose that for $L, t \ge 1$, $\mathsf{K}\colon \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$ is a p.d radial kernel which is (L,t)-smooth. There exist two randomized algorithms with the following guarantees:*

[11] Even though the vectors $\phi(x)$ are infinite dimensional, there are only a finite set of vectors. In addition, Theorem 3.5 has no dependence on the dimensionality d. Thus, it suffices to implicitly work with the subspace spanned by $\{\phi(x) : x \in X\}$ and $\phi(q)$.

- ProcessCaptured$(X, c, r_{\min}, r_{\text{in}}, r_{\text{out}}, r_{\max}, \epsilon, \xi, \delta)$ *receives as input a set* $X \subset \mathbb{R}^d$ *of size at most* n*, a point* $c \in \mathbb{R}^d$*, thresholds* $r_{\min} < r_{\text{in}} < r_{\text{out}} \leq r_{\max} \in \mathbb{R}_{\geq 0}$*, error parameters* $\epsilon, \xi \in (0,1)$*, and failure probability* $\delta \in (0,1)$*. We are promised that one of the following two hold*

$$\|x - c\|_2 \in [r_{\min}, r_{\text{in}}] \quad : \quad \forall x \in X \tag{3.15}$$

$$\|x - c\|_2 \in [r_{\text{out}}, r_{\max}] \quad : \quad \forall x \in X. \tag{3.16}$$

The algorithm outputs a pointer to a data structure v*.*

- QueryCaptured$(q, v, c, r_{\min}, r_{\text{in}}, r_{\text{out}}, r_{\max}, \epsilon, \xi, \delta)$ *receives as input a query* $q \in \mathbb{R}^d$*, a pointer to a data structure* v*, a point* $c \in \mathbb{R}^d$*, thresholds* $r_{\min} < r_{\text{in}} < r_{\text{out}} < r_{\max} \in \mathbb{R}_{\geq 0}$*, error parameters* $\epsilon, \xi \in (0,1)$*, and failure probability* $\delta \in (0,1)$*. The algorithm outputs a value* $\boldsymbol{\eta} \in \mathbb{R}_{\geq 0}$*.*

For any query $q \in \mathbb{R}^d$*, if* $\|q - c\|_2 \in [r_{\text{out}}, r_{\max}]$ *and* (3.15) *holds, or* $\|q - c\|_2 \in [r_{\min}, r_{\text{in}}]$ *and* (3.16) *holds, the following occurs with probability at least* $1 - \delta$ *over the randomness in the algorithm. We execute* ProcessCaptured$(X, c, r_{\min}, r_{\text{in}}, r_{\text{out}}, r_{\max}, \epsilon, \xi, \delta)$ *and we let* $\boldsymbol{v}$ *denote the pointer to the data structure it outputs, and we let* $\boldsymbol{\eta}$ *be the output of* QueryCaptured$(q, \boldsymbol{v}, c, r_{\min}, r_{\text{in}}, r_{\text{out}}, r_{\max}, \epsilon, \xi, \delta)$*. Then,*

- **Correctness**: *The estimate* $\boldsymbol{\eta} \in \mathbb{R}_{\geq 0}$ *that we output satisfies*

$$(1-\epsilon)\sum_{x \in X} \mathsf{K}(q,x) - 2\xi|X| \leq \boldsymbol{\eta} \leq (1+\epsilon)\sum_{x \in X} \mathsf{K}(q,x) + 2\xi|X|.$$

- **Time and Space Complexity**: *The algorithm* ProcessCaptured$(X, c, r_{\min}, r_{\text{in}}, r_{\text{out}}, r_{\max}, \epsilon, \xi, \delta)$ *takes time at most* polynd *time to output a data structure* $\boldsymbol{v}$*.*[12] *The total space of* $\boldsymbol{v}$*, as well as the running time of* QueryCaptured$(q, \boldsymbol{v}, c, r_{\min}, r_{\text{in}}, r_{\text{out}}, r_{\max}, \epsilon, \xi, \delta)$ *is, up to a constant factor, at most*

$$d \cdot \frac{\log^2(|X|/\delta)}{\epsilon} \cdot L \cdot \left(\frac{2 \cdot r_{\max}}{r_{\text{out}} - r_{\text{in}}} \cdot \frac{r_{\text{in}}}{r_{\min}} \cdot \sqrt{\ln(1/\xi)} \right)^t.$$

Remark 3.2. *Lemma* 3.2 *assumes black-box access to dot products in the embedded space. This can typically be achieved by obtaining an analytic expression for the measure* μ *and integrating as per* (3.12)*. For example, if* $\mathsf{K}(x,y) = 1/(1+\|x-y\|^2)$*, then* $G(\lambda) = 1/(1+\lambda)$ *in Theorem* 3.4 *and one has* $\mu(t) = e^{-t}$*. Then* $\langle \phi(x), \phi(y) \rangle$ *can be evaluated per* (3.12) *at polylogarithmic cost.*

4 Algorithmic Result: Data Structure for Evaluating the Coresets

4.1 A Ball Carving Hash For this section, we present a ball carving hash function. We let $d \in \mathbb{N}$ will be the dimensionality of the space (which we view as a non-constant parameter), as well as a dataset $P \subset \mathbb{R}^d$ of n points. The goal will be to provide a randomized partition of the dataset, such that whenever we use the coreset data structure of [PT20], we are doing so for points P (and potential queries Q) whose kernel matrix has a smaller γ_2-norm. Hence, this section does not concern the specific kernel K, and will simply be a ball-carving hash function.

Lemma 4.1. *There exists absolute constants* $c_1, c_2 > 0$ *such that, for any* $\alpha \in (0, c_2)$ *and* $R > 0$*, we have the following. There exists a distribution* $\mathcal{D}$ *supported on pairs* $(c, r) \in \mathbb{R}^d \times \mathbb{R}_{\geq 0}$ *which specify a function* $h_{c,r} \colon \mathbb{R}^d \to \{0, 1, *\}$ *given by*

$$h_{c,r}(x) = \begin{cases} 0 & x \in B_2\left(c, r - \frac{\alpha R}{\sqrt{d}}\right) \\ 1 & x \notin B_2\left(c, r + \frac{\alpha R}{\sqrt{d}}\right) \\ * & o.w. \end{cases}. \tag{4.17}$$

The distribution satisfies the three guarantees:

[12] It is important here that the algorithm can efficiently compute $\langle \phi(x), \phi(y) \rangle$ for any $x, y \in X$ for the feature embeddings of Theorem 3.2. See Remark 3.2 for more details.

- **Separate Far Points:** *For any two points $x, y \in B_2(0,R)$ where $\|x-y\|_2 \geq R/100$, the probability over $(\mathbf{c},\mathbf{r}) \sim \mathcal{D}$ that $h_{\mathbf{c},\mathbf{r}}(x), h_{\mathbf{c},\mathbf{r}}(y) \in \{0,1\}$ and $h_{\mathbf{c},\mathbf{r}}(x) \neq h_{\mathbf{c},\mathbf{r}}(y)$ is at least $c_1/\sqrt{d}$.*
- **Avoid Boundary:** *For any point $x \in B_2(0,R)$, the probability over $(\mathbf{c},\mathbf{r}) \sim \mathcal{D}$ that $h_{\mathbf{c},\mathbf{r}}(x) = *$ is at most $\alpha/\sqrt{d}$.*
- **Far From Center**: *For any point $x \in \mathbb{R}^d$, with probability at least $1 - 2^{-\Omega(d)}$ over the draw of $(\mathbf{c},\mathbf{r}) \sim \mathcal{D}$, $\|x-\mathbf{c}\|_2 \geq R/100$, $\|\mathbf{c}\|_2 \leq 2R$.*

Proof. The distribution $\mathcal{D}$ samples a point $\mathbf{c}$ and a threshold $\mathbf{r}$ by letting $\mathbf{c} \sim \mathcal{N}(0, R^2 \cdot I_d/d)$ and $\mathbf{r} \sim [0,3R]$. The third item is the simplest: (i) by anti-concentration of $\mathcal{N}(0, R^2 \cdot I_d/d)$, the probability mass on any ball of radius $R/100$ is at most $2^{-\Omega(d)}$, and (ii) by concentration of $\mathcal{N}(0, R^2 \cdot I_d/d)$, $\|\mathbf{c}\|_2 \leq 2R$ except with probability $2^{-\Omega(d)}$. The second item is also simple to argue: for any fixed $c \in \mathbb{R}^d$, $h_{c,\mathbf{r}}(x) = *$ whenever $|\mathbf{r} - \|x-c\|_2| \leq \alpha R/\sqrt{d}$. Therefore, the probability over the draw of $\mathbf{r} \sim [0,3R]$ that the above occurs is at most $\alpha/\sqrt{d}$.

We now argue the first item, that $h_{\mathbf{c},\mathbf{r}}$ tends to separate far points. Consider a fixed setting of $p, q \in B_2(0,R)$, and for a sample $(\mathbf{c},\mathbf{r}) \sim \mathcal{D}$, let

$$\boldsymbol{\gamma} := \big|\|p-\mathbf{c}\|_2 - \|q-\mathbf{c}\|_2\big|.$$

Then, the event that $h_{\mathbf{c},\mathbf{r}}(p) \neq h_{\mathbf{c},\mathbf{r}}(q)$ occurs whenever the following two events hold:

- Event $\boldsymbol{A}$: For $\delta = 1/2$, we have $p, q \in B_2(\mathbf{c}, (2+\delta)R)$, and
- Event $\boldsymbol{B}$: The threshold $\mathbf{r}$ lies within an interval of $3R$ of length $\boldsymbol{\gamma} - 2\alpha R/\sqrt{d}$.

Notice that Event $\boldsymbol{A}$ holds with high probability because we may apply the triangle inequality and the fact $p, q \in B_2(0,R)$. If event $\boldsymbol{A}$ fails, then the center point $\mathbf{c}$ must have Euclidean norm larger than $(1+\delta)R$, so

$$\Pr_{\mathbf{c}}[\text{Event } \boldsymbol{A} \text{ fails}] \leq \Pr_{\mathbf{c}}[\|\mathbf{c}\|_2 \geq (1+\delta)\cdot R] \leq \Pr_{\mathbf{x}\sim\chi^2(d)}[\mathbf{x} - d \geq \delta \cdot d] \leq e^{-d\delta^2/8},$$

using a standard concentration inequality on $\chi^2(d)$ random variables (see, Example 2.5 in [Wai19]). Therefore, we have

$$\begin{aligned} \Pr[\text{Events } \boldsymbol{A} \text{ and } \boldsymbol{B} \text{ hold}] &\geq \Pr[\text{Event } \boldsymbol{B} \text{ holds}] - \Pr[\text{Event } \boldsymbol{A} \text{ fails}] \\ &\geq \Pr[\text{Event } \boldsymbol{B} \text{ holds}] - e^{-d\delta^2/8}. \end{aligned} \tag{4.18}$$

So it remains to lower bound the probability over $(\mathbf{c},\mathbf{r}) \sim \mathcal{D}$ that $h_{\mathbf{c},\mathbf{r}}(p) \neq h_{\mathbf{c},\mathbf{r}}(q)$, which is at least

$$\Pr[\text{Event } \boldsymbol{B} \text{ holds}] = \mathop{\mathbb{E}}_{\mathbf{c}}\left[\frac{\boldsymbol{\gamma} - 2\alpha R/\sqrt{d}}{3R}\right] = \frac{1}{3R}\cdot\mathop{\mathbb{E}}_{\mathbf{c}}[\boldsymbol{\gamma}] - \frac{\alpha}{\sqrt{d}}. \tag{4.19}$$

We may lower bound the expectation of $\boldsymbol{\gamma}$ by considering whether or not event $\boldsymbol{A}$ holds. In particular, we may always lower bound, for any setting of the randomness of $\boldsymbol{\gamma}$,

$$\boldsymbol{\gamma} = \frac{\left|\|p-\mathbf{c}\|_2^2 - \|q-\mathbf{c}\|_2^2\right|}{\|p-\mathbf{c}\|_2 + \|q-c\|_2} \geq \frac{\left|\|p-\mathbf{c}\|_2^2 - \|q-\mathbf{c}\|_2^2\right|}{2(2+\delta)R} - \mathbf{1}\{\text{Event } \boldsymbol{A} \text{ fails}\}\cdot(R + \|\mathbf{c}\|_2). \tag{4.20}$$

The reason is the following: if event $\boldsymbol{A}$ holds, then $\|p-\mathbf{c}\|_2 + \|q-\mathbf{c}\|_2 \leq 2(2+\delta)R$ and we obtain the desired lower bound. If event $\boldsymbol{A}$ fails, then since $\|p\|_2, \|q\|_2 \leq R$,

$$\frac{\left|\|p-\mathbf{c}\|_2^2 - \|q-\mathbf{c}\|_2^2\right|}{2(2+\delta)R} = \frac{\left|\|p\|_2^2 - \|q\|_2^2 - 2\langle p-q, \mathbf{c}\rangle\right|}{2(2+\delta)R} \leq \frac{2R^2}{2(2+\delta)R} + \frac{2\|p-q\|_2\|\mathbf{c}\|_2}{2(2+\delta)R} \leq R + \|\mathbf{c}\|_2,$$

so subtracting $\mathbf{1}\{\text{Event } \boldsymbol{A} \text{ fails}\}\cdot(R+\|\mathbf{c}\|_2)$ ensures that the right-hand side of (4.20) is negative. Hence,

$$\mathop{\mathbb{E}}_{\mathbf{c}}[\boldsymbol{\gamma}] \geq \mathop{\mathbb{E}}_{\mathbf{c}}\left[\frac{\left|\|p-\mathbf{c}\|_2^2 - \|q-\mathbf{c}\|_2^2\right|}{2(2+\delta)R}\right] - R\cdot\Pr_{\mathbf{c}}[\text{Event } \boldsymbol{A} \text{ fails}] - \mathop{\mathbb{E}}_{\mathbf{c}}[\|\mathbf{c}\|_2\cdot\mathbf{1}\{\|\mathbf{c}\|_2 \geq (1+\delta)R\}]$$

We bound each term:

$$\mathop{\mathbb{E}}_{\mathbf{c}}\left[\frac{\left|\|p-\mathbf{c}\|_2^2-\|q-\mathbf{c}\|_2^2\right|}{2(2+\delta)R}\right]=\frac{1}{2(2+\delta)R}\cdot\mathop{\mathbb{E}}_{\mathbf{c}}\left[\left|\|p\|_2^2-\|q\|_2^2-2\langle p-q,\mathbf{c}\rangle\right|\right]=\Omega\left(\frac{\|p-q\|_2}{\sqrt{d}}\right),$$

where by 2-stability of the Gaussian distribution, $2\langle p-q,\mathbf{c}\rangle$ is distributed like $\mathcal{N}(0,4\|p-q\|_2^2\cdot R^2/d)$, and anti-concentration of the Gaussian distribution. Recall that we have already upper bounded the probability that event $\boldsymbol{A}$ fails, and finally, we can similarly bound the final expectation,

$$\mathop{\mathbb{E}}_{\mathbf{c}}\left[\|\mathbf{c}\|_2\cdot\mathbf{1}\{\|\mathbf{c}\|_2\geq(1+\delta)R\}\right]\leq\int_{\zeta:\delta}^{\infty}R\cdot\Pr_{\mathbf{c}}\left[\|\mathbf{c}\|_2\geq(1+\zeta)R\right]d\zeta\leq R\int_{\zeta:\delta}^{\infty}e^{-d\zeta^2/8}d\zeta\leq R\cdot e^{-\Omega(d)},$$

using the fact $\delta=1/2$. Substituting into (4.19) into (4.18), we obtain

$$\begin{aligned}\Pr_{(\mathbf{c},\mathbf{r})\sim\mathcal{D}}\left[\begin{array}{c}h_{\mathbf{c},\mathbf{r}}(p)\neq h_{\mathbf{c},\mathbf{r}}(q)\\ h_{\mathbf{c},\mathbf{r}}(p),h_{\mathbf{c},\mathbf{r}}(q)\in\{0,1\}\end{array}\right]&\geq\frac{1}{3R}\left(\Omega\left(\frac{\|p-q\|_2}{\sqrt{d}}\right)-R\cdot e^{-\Omega(d)}\right)-\frac{\alpha}{\sqrt{d}}-e^{-\Omega(d)}\\&\geq\Omega\left(\frac{1}{\sqrt{d}}\right).\end{aligned}$$

since $\|p-q\|_2\geq R/100$, α is a small enough constant, and d is at least a large enough constant. ∎

4.2 Data Structure In this section, we will use Lemma 4.1 to preprocess the dataset by recursively hashing. We will consider a dataset $P\subset\mathbb{R}^d$ which consists of n points, an unknown query $q\in\mathbb{R}^d$, and the parameter $\Phi>1$ which denotes the maximum aspect ratio of $P\cup\{q\}$, i.e.,

$$\Phi=\frac{\max_{x,y\in P\cup\{q\}}\|x-y\|_2}{\min_{x\neq y\in P\cup\{q\}}\|x-y\|_2}.$$

We assume that the dimensionality d is $\omega(\log n\log\log\Phi)\leq d$; the assumption is without loss of generality, as we may add coordinates which are set to 0.

THEOREM 4.1. *For $n,d\in\mathbb{N}$, and $L,t\geq1$, consider any p.d (L,t)-smooth kernel $\mathsf{K}\colon\mathbb{R}^d\times\mathbb{R}^d\to[0,1]$. There exists two randomized algorithms* PREPROCESS *(Algorithm 4.2) and* QUERY *(Algorithm 2) such that,*

- PREPROCESS(P,ϵ,ξ) *receives as input a dataset $P\subset\mathbb{R}^d$ of at most n points, and two error parameters $\epsilon,\xi\in(0,1)$. The algorithm outputs a pointer to a data structure u.*
- QUERY(q,u,ϵ,ξ) *receives as input a query $q\in\mathbb{R}^d$, a pointer to a data structure u generated by* PREPROCESS*, and two error parameters $\epsilon,\xi\in(0,1)$.*

Suppose that $P\cup\{q\}$ has aspect ratio at most Φ, then we satisfy the following guarantees with probability at least 0.9*,*

- **Correctness***: If $\mathbf{u}$ is the output of* PREPROCESS(P,ϵ,σ) *and $\boldsymbol{\eta}\in\mathbb{R}_{\geq0}$ is the output of* QUERY$(q,\mathbf{u},\epsilon,\sigma)$*. Then,*
$$(1-\epsilon)\sum_{p\in P}\mathsf{K}(p,q)-2\xi n\leq\boldsymbol{\eta}\leq(1+\epsilon)\sum_{p\in P}\mathsf{K}(p,q)+2\xi n.$$
- **Space Complexity***: The total space of a data structure $\mathbf{u}$ produced by* PREPROCESS(P,ϵ,ξ) *is at most, up to a constant factor,*
$$\frac{nd}{\epsilon}\cdot L\cdot\left(\sqrt{d\cdot\mathrm{polylog}(nd\Phi\ln(1/\xi)/\epsilon)\cdot\ln(1/\xi)}\right)^t.$$
- **Query Time***:* QUERY$(q,\mathbf{u},\epsilon,\xi)$ *takes time at most, up to a constant factor,*
$$\frac{d}{\epsilon}\cdot L\cdot\left(\sqrt{d\cdot\mathrm{polylog}(nd\Phi/\epsilon)\cdot\ln(1/\xi)}\right)^t.$$

- **Preprocessing Time**: *Assuming access to oracles for computing* $\langle \phi(x), \phi(y) \rangle$ *for* $x, y \in \mathbb{R}^d$ *and* ϕ *constructed from Theorem 3.2, the algorithm* PREPROCESS(P, ϵ, ξ) *runs in time* poly$nd \log \Phi \ln(1/\xi)/\epsilon$.

REMARK 4.1. (SIMPLIFICATIONS ON THE NOTATION AND ERROR PROBABILITY) *In the description of the algorithms* PREPROCESS(P, ϵ, ξ) *and* QUERY(q, u, ϵ, ξ), *we will make multiple calls to* PROCESSCAPTURED *and* QUERYCAPTURED *with various parameter settings and we will make the following simplifications. First, we will drop the dependence on* ϵ *and* ξ *in the description below, as these will remain constant throughout the algorithm. Second, we will, at the forefront, set the error probability* δ *in calls to* PROCESSCAPTURED *and* QUERYCAPTURED *to* $1/\text{poly}dn\Phi/\epsilon$. *Thus, in the presentation below, we (i) simplify the notation by dropping the dependence on* δ *in calls to* PROCESSCAPTURED *and* QUERYCAPTURED, *and (ii) essentially assume that* PROCESSCAPTURED *and* QUERYCAPTURED *are deterministic algorithms. Indeed, the theorem that we seek to prove, Theorem 4.1, allows for* polylog$(dn\Phi/\epsilon)$ *dependencies, and Lemma 3.2 has poly-logarithmic dependence on* $1/\delta$. *Since the preprocessing time of* PREPROCESS(P, ϵ, ξ) *and query time* QUERY(q, u, ϵ, ξ) *which is at most polynomial in* $nd \log \Phi$, *setting* $\delta \leq 1/\text{poly}dn\Phi$ *allows one to union bound over all executions of* PROCESSCAPTURED *and* QUERYCAPTURED *so that they are all correct with high probability.*

The remainder of this section gives the proof of Theorem 4.1. The preprocessing algorithm that we present below can be thought of as consisting of two main parts, where we prepare for an (unknown) query $q \in \mathbb{R}^d$. The data structure will be stored as a binary tree where nodes will hold additional information. Specifically, a call to PREPROCESS(P) instantiates a node u, and will have certain attributes stored in the node. The notation for these will be $u.\mathsf{attr}$, for some "attribute" attr. Each non-leaf node u will have at most two children, which will be stored in $u.\mathsf{Child}(0)$ and $u.\mathsf{Child}(1)$. The formal description appears in Algorithm 4.2.

High Level Structure of Algorithm 4.2 The first thing the algorithm does is enclose the dataset P with an (approximately) minimum enclosing ball around a center point $u.\mathsf{cen}$ of radius $u.\mathsf{rad}$. This is done in Line 3, and can be done in $O(nd)$ since it will suffice to obtain a constant-factor approximation (for instance, picking an arbitrary point $p \in P$ and letting $u.\mathsf{cen} = p$ and $u.\mathsf{rad} = \max_{p' \in P} \|p - p'\|_2$ suffices). If Line 5 is triggered, then there is at most 1 distinct point (which is stored as $u.\mathsf{cen}$) and since the algorithm stores $|P|$ in $u.\mathtt{size}$, it will be able to compute the kernel contribution of P. The parameter $u.\mathsf{R}$ is set to $2 \cdot u.\mathsf{rad}$ and the algorithm will prepare for two cases: when the (unknown) query $q \in \mathbb{R}^d$ is "close" to $u.\mathsf{cen}$ (within distance at most $u.\mathsf{R}$), and when the query $q \in \mathbb{R}^d$ is "far" from $u.\mathsf{cen}$.

- **Query Close**: In this case, we will be preparing for a query q satisfying $\|q - u.\mathsf{cen}\|_2 \leq u.\mathsf{R}$, so that we may instantiate Lemma 4.1 with the origin as $u.\mathsf{cen}$ and R as $u.\mathsf{R}$. In this case, we can consider the hash function as dividing $\mathbb{R}^d$ into three parts: (i) within distance $u.\mathsf{r}_{\text{in}}$ of $u.\mathsf{newcen}$ (with high probability, we also have the distance will be at least $u.\mathsf{r}_{\text{min}}$ as well), (ii) within the shell around $u.\mathsf{newcen}$ of inner-radius $u.\mathsf{r}_{\text{in}}$ and outer-radius $u.\mathsf{r}_{\text{out}}$, and (iii) outside the ball around $u.\mathsf{newcen}$ of radius $u.\mathsf{r}_{\text{out}}$ (and we will also have an upper bound on the distance to $u.\mathsf{newcen}$ of $u.\mathsf{r}_{\text{max}}$). In this case, we build two coresets with PROCESSCAPTURED for regions (i) and (iii). Then, we recursively preprocess the points which are not captured by the coresets and store these data structures as the children, $u.\mathsf{Child}(0)$ and $u.\mathsf{Child}(1)$.

- **Query Far**: In this case, we prepare for a query q which satisfies $\|q - u.\mathsf{cen}\|_2 \geq u.\mathsf{R}$. Since every point $p \in P$ lies within distance $u.\mathsf{rad}$ of $u.\mathsf{cen}$ and $u.\mathsf{R} = 2 \cdot u.\mathsf{rad}$, we can already guarantee a separation of least $u.\mathsf{rad}$ between q and any $p \in P$. Formally, we will handle this in Line 15 with the sub-routine PREPROCESSFAR (which we specify shortly). This case will not recursively call PREPROCESS.

As we will formally show below, recursively applying the ball-carving hash function result in calls to PREPROCESS with datasets of decreasing radii. This is because Lemma 4.1 guarantees that we separate far points with at least some probability. Eventually the radius becomes 0 and Line 5 ends the recursion.

Algorithm 1 Preprocessing of a dataset $P \subset \mathbb{R}^d$ into a data structure.

procedure PREPROCESS(P)
  Initialize a node u.
  Let $\texttt{MEB}(P) \subset \mathbb{R}^d$ denote an (approximately) minimum enclosing ball of P.
  Store the parameters $u.\mathsf{rad}, u.\mathsf{cen}, u.\mathsf{R}, u.\texttt{size}$ as

$$\begin{aligned} u.\mathsf{rad} &= \text{radius of } \texttt{MEB}(P) \in \mathbb{R}_{\geq 0} \\ u.\mathsf{cen} &= \text{center of } \texttt{MEB}(P) \in \mathbb{R}^d \\ u.\mathsf{R} &= 2 \cdot u.\mathsf{rad} \\ u.\texttt{size} &= |P| \in \mathbb{N}. \end{aligned}$$

  if $u.\mathsf{rad} = 0$ **then**
    return u.
  Sample $(\mathbf{c}, \mathbf{r}) \sim \mathcal{D}$ from Lemma 4.1 with R set to $u.\mathsf{R}$ and $\alpha = c_1/(4 \log n \log \Phi)^2$.
  Store $u.\mathsf{c} = \mathbf{c}$ and $u.\mathsf{r} = \mathbf{r}$ and define the following quantities

$$\begin{aligned} u.\mathsf{r}_{\min} &= \frac{u.\mathsf{R}}{100} \\ u.\mathsf{r}_{\text{in}} &= \mathbf{r} - \frac{\alpha \cdot u.\mathsf{R}}{\sqrt{d}} \\ u.\mathsf{r}_{\text{out}} &= \mathbf{r} + \frac{\alpha \cdot u.\mathsf{R}}{\sqrt{d}} \\ u.\mathsf{r}_{\max} &= 3 \cdot u.\mathsf{R} \\ u.\mathsf{newcen} &= u.\mathsf{c} + u.\mathsf{cen}. \end{aligned}$$

  We defined the following subsets of P (which do not necessarily partition P):

$$\text{Cap}(b) = \{p \in P : h_{\mathbf{c},\mathbf{r}}(p - u.\mathsf{cen}) = 1 - b\}, \qquad \text{for } b \in \{0, 1\}$$

  for $b \in \{0, 1\}$ **do**
    Run PROCESSCAPTURED($\text{Cap}(b), u.\mathsf{newcen}, u.\mathsf{r}_{\min}, u.\mathsf{r}_{\text{in}}, u.\mathsf{r}_{\text{out}}, u.\mathsf{r}_{\max}$).
    Store the output in $u.\mathsf{InnerBall}(b)$.
    if $P \setminus \text{Cap}(b) \neq \emptyset$ **then**
      Execute PREPROCESS($P \setminus \text{Cap}(b)$) and store the output in $u.\mathsf{Child}(b)$.
  Run PREPROCESSFAR($P, u.\mathsf{cen}, u.\mathsf{rad}, u.\mathsf{R}$) and store in $u.\texttt{FarDS}$.
  return u.

Algorithm 2 Querying a data structure rooted at u with a point $q \in \mathbb{R}^d$.

procedure QUERY(q, u)
  if $\|q - u.\mathsf{cen}\|_2 \leq u.\mathsf{R}$ **then**
    We evaluate the hash function $h_{u.\mathsf{c},u.\mathsf{r}}(q - u.\mathsf{cen})$, and set $b \in \{0, 1, *\}$ to be its output.
    if $b = *$ **then**
      return "fail."
    Execute QUERYCAPTURED($q, u.\mathsf{InnerBall}(b), u.\mathsf{newcen}, u.\mathsf{r}_{\min}, u.r_{\text{in}}, u.r_{\text{out}}, u.\mathsf{r}_{\max}$)
    Execute QUERY($q, u.\mathsf{Child}(b)$).
    return the sum of the outputs of Line 6 and Line 7.
  if $\|q - u.\mathsf{cen}\|_2 > u.\mathsf{R}$ **then**
    return QUERYFAR($q, u.\texttt{FarDS}, u.\mathsf{cen}, u.\mathsf{R}$).

4.3 The PreprocessFar and QueryFar Algorithms In this section, we give the descriptions of PreprocessFar (Algorithm 3) and QueryFar (Algorithm 4). These are meant to capture cases where the query lies substantially far from the dataset. The approach will be straight-forward: we will partition the (query) space $\mathbb{R}^d$ into geometrically increasing balls (up to a certain point), and maintain a coreset for each of these balls. The remaining piece is to bound how many balls we need.

Definition 4.1. *Let $\mathsf{K}\colon \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$ be a p.d radial kernel. For any $r \geq 0$ and $\epsilon, \xi \in (0,1)$, let $R(r,\epsilon,\xi) > 0$ denote the minimum over all $R \geq 0$ such that, for all datasets $P \subset \mathbb{R}^d$ where $P \subset B_2(0,r)$, and for all queries $q \in \mathbb{R}^d$ with $\|q\|_2 \geq R$,*

$$(1-\epsilon)\sum_{p\in P}\mathsf{K}(p,q) - 2\xi|P| \leq |P|\cdot\mathsf{K}(0,q) \leq (1+\epsilon)\sum_{p\in P}\mathsf{K}(p,q) + 2\xi|P|.$$

Claim 4.1. *We always have $R(r,\epsilon,\xi) = O(r\ln(1/\xi)/\epsilon)$.*

Proof. By Theorem 3.4 there exists a non-negative finite Borel measure μ such that

$$\mathsf{K}(p,q) = \int_{t:0}^{\infty} e^{-t\|p-q\|_2^2}\mu(dt) \tag{4.21}$$

for all $p, q \in \mathbb{R}^d$. We consider the parameter $R = Cr\ln(1/\xi)/\epsilon$, for large enough constant C, and we show that enforcing $\|q\|_2 \geq R$ suffices. Note that $R \geq 2r$, and that for any $p \in B_2(0,r)$, the following occurs. If we consider a query $q \in \mathbb{R}^d$ with $\|q\|_2 \geq R$, we always have $\|p-q\|_2 \geq \|q\|_2 - \|p\|_2 \geq \|q\|_2/2$. Consider first the case that $t \geq 0$ satisfies, $t\|q\|_2^2 \geq 4\ln(1/\xi)$. Then, we will have

$$e^{-t\|p-q\|_2^2} \leq e^{-t\|q\|_2^2/4} \leq \xi. \tag{4.22}$$

On the other hand, if $t \geq 0$ is such that $t\|q\|_2^2 \leq 4\ln(1/\xi)$, then

$$t \leq \frac{4\ln(1/\xi)}{\|q\|_2^2} \leq \frac{4\epsilon^2}{C^2r^2\ln(1/\xi)} \qquad \text{and} \qquad t\|q\|_2 \leq \frac{4\epsilon}{Cr},$$

which implies the following two inequalities:

$$\frac{e^{-t\|p-q\|_2^2}}{e^{-t\|q\|_2^2}} = e^{-t\|p\|_2^2+2t\langle p,q\rangle} \leq e^{2tr\|q\|_2} \leq e^{8\epsilon/C} \leq 1+\epsilon/2$$
$$e^{-t\|p\|_2^2+2t\langle p,q\rangle} \geq e^{-tr^2-2tr\|q\|_2} \geq e^{-4\epsilon^2/(C^2\ln(1/\xi))-8\epsilon/C} \geq 1-\epsilon/2.$$

Putting both cases together, whenever $\|q\|_2 \geq R$, we have

$$\sum_{p\in P}\mathsf{K}(p,q) = \sum_{p\in P}\int_{t:0}^{\infty} e^{-t\|p-q\|_2^2}\mu(dt) \leq \sum_{p\in P}\int_{t:0}^{4\ln(1/\xi)/\|q\|_2^2} e^{-t\|p-q\|_2^2}\mu(dt) + |P|\xi$$
$$\leq (1+\epsilon/2)\sum_{p\in P}\int_{t:0}^{4\ln(1/\xi)/\|q\|_2^2} e^{-t\|q\|_2^2}\mu(dt) + |P|\xi \leq (1+\epsilon/2)|P|\cdot\mathsf{K}(0,q) + |P|\xi,$$

and analogously,

$$\sum_{p\in P}\mathsf{K}(p,q) \geq \sum_{p\in P}\int_{t:0}^{4\ln(1/\xi)/\|q\|_2^2} e^{-t\|p-q\|_2^2}\mu(dt) \geq (1-\epsilon)\sum_{p\in P}\int_{t:0}^{4\ln(1/\xi)/\|q\|_2^2} e^{-t\|q\|_2^2}\mu(dt)$$
$$\geq (1-\epsilon/2)\sum_{p\in P}\int_{t:0}^{\infty} e^{-t\|q\|_2^2}\mu(dt) - |P|\xi \geq (1-\epsilon/2)|P|\cdot\mathsf{K}(0,q) - |P|\xi.$$

Re-arranging terms gives the desired inequality. ∎

Algorithm 3 Preprocessing of a dataset $P \subset \mathbb{R}^d$ within a ball when the query will be far.

procedure PREPROCESSFAR($P, c, r_{\text{in}}, \mathsf{r}$)

Initialize a node u, and sample a point $\mathbf{c}_0 \sim B_2(0, r_{\text{in}}/3)$ and let

$$\begin{aligned} u.\mathsf{cen} &= c + \mathbf{c}_0 \in \mathbb{R}^d \\ u.\mathsf{r}_{\min} &= \frac{r_{\text{in}}}{100} \\ u.\mathsf{r}_{\text{in}} &= 4 \cdot r_{\text{in}}/3 \\ u.\mathsf{r} &= \mathsf{r} - r_{\text{in}}/3 \geq \frac{5}{4} \cdot u.\mathsf{r}_{\min} \quad (\text{when } \mathsf{r} \geq 2 \cdot r_{\min}) \\ u.\mathtt{size} &= |P|. \end{aligned}$$

for $h \in \{0, \dots, \lceil \log_2 (R(u.\mathsf{r}, \epsilon, \xi)/u.\mathsf{r}) \rceil\}$ **do**

Execute PROCESSCAPTURED($P, u.\mathsf{cen}, u.\mathsf{r}_{\min}, u.\mathsf{r}_{\text{in}}, 2^h \cdot u.\mathsf{r}, 2^{h+1} \cdot u.\mathsf{r}$),

Store the output in $u.\mathsf{OuterBall}(h)$.

return u.

Algorithm 4 Querying of a dataset $P \subset \mathbb{R}^d$ within a ball when the query is far.

procedure QUERYFAR(q, u, c, r)

Compute $r = \|q - u.\mathsf{cen}\|_2$ (note we will always have $r \geq u.\mathsf{r}$).

Let $h \in \{0, 1, 2, \dots\}$ such that $2^h \cdot u.\mathsf{r} \leq r \leq 2^{h+1} \cdot u.\mathsf{r}$.

if $h \leq \lceil \log_2(R(u.\mathsf{r}, \epsilon, \xi)/u.\mathsf{r}) \rceil$ **then**

return QUERYCAPTURED($q, u.\mathsf{OuterBall}(h), u.\mathsf{r}_{\min}, u.\mathsf{r}_{\text{in}}, 2^h \cdot u.\mathsf{r}, 2^{h+1} \cdot u.\mathsf{r}$).

if $h > \lceil \log_2(R(u.\mathsf{r}, \epsilon, \xi)/u.\mathsf{r}) \rceil$ **then**

return $u.\mathtt{size} \cdot \mathsf{K}(q, u.\mathsf{cen})$.

LEMMA 4.2. *Let $n, d \in \mathbb{N}$, and suppose that, for $L, t \geq 1$, $\mathsf{K}\colon \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$ is a p.d radial kernel which is (L,t)-smooth. There are two randomized algorithms with the following guarantees:*

- PREPROCESSFAR($P, c, r_{\text{in}}, \mathsf{r}$) *receives as input a set $P \subset \mathbb{R}^d$ of size at most n, a point c, and two thresholds r_{in} and r, where $\mathsf{r} \geq 2 \cdot r_{\text{in}}$. We are promised that every $p \in P$ satisfies $\|p - c\|_2 \leq r_{\text{in}}$. The algorithm outputs a pointer to a data structure $\mathbf{u}$.*
- QUERYFAR(q, u, c, r) *receives as input a query $q \in \mathbb{R}^d$, and a pointer to a data structure (generated from* PREPROCESSFAR*). We are promised that the query q satisfies $\|q - c\|_2 \geq \mathsf{r}$, and the algorithm returns a value $\boldsymbol{\eta} \in \mathbb{R}_{\geq 0}$.*

We satisfy the following guarantees with probability at least $1 - 1/\text{poly}nd\Phi/\epsilon$:

- **Correctness***: If $\mathbf{u}$ is the output of* PREPROCESSFAR($P, c, r_{\text{in}}, \mathsf{r}$) *and $\boldsymbol{\eta}$ is the output of* QUERYFAR($q, \mathbf{u}, c, \mathsf{r}$), *then*
$$(1 - \epsilon) \sum_{p \in P} \mathsf{K}(p, q) - 2\xi|P| \leq \boldsymbol{\eta} \leq (1 + \epsilon) \sum_{p \in P} \mathsf{K}(p, q) + 2\xi|P|.$$
- **Preprocessing Time and Space Complexity***: The algorithm* PREPROCESSFAR *preprocesses inputs in time at most* $\text{poly}nd \log \Phi \ln(1/\xi)/\epsilon$ *to output the data structure $\mathbf{u}$. The total space of $\mathbf{u}$ is, up to a constant factor, at most*
$$\frac{d}{\epsilon} \cdot \text{polylog}(dn\Phi \ln(1/\xi)/\epsilon) \cdot L \cdot \left(O(\sqrt{\ln(1/\xi)})\right)^t$$
- **Query Time***: The algorithm* QUERYFAR *outputs an estimate $\boldsymbol{\eta}$ in time, up to a constant factor, at most*
$$\frac{d}{\epsilon} \cdot \log^2(nd\Phi/\epsilon) \cdot L \cdot \left(O(\sqrt{\ln(1/\xi)})\right)^t$$

Proof. The purpose of Line 2 is to find an appropriate center which will guarantee some separation between the query and the dataset, and to ensure that no dataset point is too close to the center. In particular, since $\mathbf{c}_0 \sim B_2(0, r_{\text{in}}/2)$ is drawn randomly and $d = \omega(\log n \log\log \Phi))$, it is not too hard to check that with high probability, the new center $u.\mathsf{newcen}$ and thresholds $u.\mathsf{r}_{\min} < u.\mathsf{r}_{\text{in}} < u.\mathsf{r}$ satisfy that (i) $u.\mathsf{r} - u.\mathsf{r}_{\text{in}} \geq u.\mathsf{r}_{\text{in}}/4$, (ii) every $p \in P$ satisfies $\|p - u.\mathsf{cen}\|_2 \in [u.\mathsf{r}_{\min}, u.\mathsf{r}_{\text{in}}]$, and (iii) every $q \in \mathbb{R}^d$ with $\|q - c\|_2 \geq \mathsf{r}$ will also satisfy $\|q - u.\mathsf{cen}\|_2 \geq u.\mathsf{r}$. Thus, the calls to ProcessCaptured$(P, u.\mathsf{cen}, u.\mathsf{r}_{\min}, u.\mathsf{r}_{\text{in}}, 2^h \cdot u.\mathsf{r}, 2^{h+1} \cdot u.\mathsf{r})$ and QueryCaptured$(q, u.\mathsf{OuterBall}(h), u.\mathsf{r}_{\min}, u.\mathsf{r}_{\text{in}}, 2^h \cdot u.\mathsf{r}, 2^{h+1} \cdot u.\mathsf{r})$ satisfy the correctness guarantees for queries q with $\|q - u.\mathsf{cen}\|_2 \in [2^h \cdot u.\mathsf{r}, 2^{h+1} \cdot u.\mathsf{r}]$. If $\|q - u.\mathsf{cen}\|_2 \geq R(u.\mathsf{r}, \epsilon, \xi)$, then by Definition 4.1, the entire kernel contribution of P is approximated by $\mathsf{K}(q, u.\mathsf{cen}) \cdot u.\mathtt{size}$. The bound on the space and query time follow from Lemma 3.2, plugging in $r_{\max} = 2^{h+1} \cdot u.\mathsf{r}$, $r_{\text{out}} = 2^h \cdot u.\mathsf{r}$, $r_{\text{in}} = u.\mathsf{r}_{\text{in}}$ and $r_{\min} = u.\mathsf{r}_{\min}$

$$\frac{2 \cdot 2^{h+1} \cdot u.\mathsf{r}}{2^h \cdot \mathsf{r} - u.\mathsf{r}_{\text{in}}} \cdot \frac{u.\mathsf{r}_{\text{in}}}{u.\mathsf{r}_{\min}} = O(1).$$

□

4.4 Analysis of Preprocess and Query Before we begin to analyze Preprocess(P), it is useful to check that the invocations of ProcessCaptured satisfy the requirements specified in Lemma 3.2. We check both cases of $b \in \{0, 1\}$ individually.

- **Case** $b = 1$. By definition of $h_{c,r}$, we have the following guarantees. The set Cap(1) consists of all points $p \in P$ where p is within $B_2(u.\mathsf{newcen}, u.\mathsf{r}_{\text{in}})$. Furthermore, by a union bound over $|P|$ points, with probability at least $1 - |P|/2^{\Omega(d)}$, every $p \in \text{Cap}(1)$ also satisfies $\|p - u.\mathsf{newcen}\|_2 \geq u.\mathsf{R}/100 = u.\mathsf{r}_{\min}$. Thus, the call to ProcessCaptured with $b = 1$ satisfies Inequality 3.15 in Lemma 3.2, and will handle cases where $\|q - u.\mathsf{newcen}\|_2 \in [u.\mathsf{r}_{\text{out}}, u.\mathsf{r}_{\max}]$.
- **Case** $b = 0$. In a similar vein, Cap(0) consists of points whose distance to $u.\mathsf{newcen}$ is at least $u.\mathsf{r}_{\text{out}}$. Furthermore, since $P \subset B_2(u.\mathsf{cen}, u.\mathsf{rad})$ and by the sampling procedure (recall $u.\mathsf{c} \sim \mathcal{N}(0, u.\mathsf{R}^2 I_d/d)$,) we have $\|u.\mathsf{c}\|_2 \leq 2u.\mathsf{R}$ with probability $1 - 2^{-\Omega(d)}$, we have $P \subset B_2(u.\mathsf{newcen}, 3u.\mathsf{R})$, so $\|p - u.\mathsf{newcen}\|_2 \leq u.\mathsf{r}_{\max}$. Thus, the call to ProcessCaptured with $b = 1$ satisfies Inequality 3.16 and $\|q - u.\mathsf{newcen}\|_2 \in [u.\mathsf{r}_{\min}, u.\mathsf{r}_{\text{in}}]$ in Lemma 3.2 with probability at least $1 - 2^{-\Omega(d)}$. Hence, in both $b \in \{0, 1\}$ of Line 11, we apply Lemma 3.2 with

$$\frac{2 \cdot r_{\max}}{r_{\text{out}} - r_{\text{in}}} \cdot \frac{r_{\text{in}}}{r_{\min}} = O\left(\frac{\sqrt{d}}{\alpha}\right).$$

With those remarks set, we now analyze the space complexity Preprocess(P). Since we will later bound the space complexity of ProcessCaptured, we will mostly be concerned with the size of the binary tree rooted at u produced by Preprocess(P). The one challenging aspect in bounding the size of the tree is that the two sets $P \setminus \text{Cap}(0)$ and $P \setminus \text{Cap}(1)$ are not necessarily disjoint. In particular, if $p \in P$ happens to satisfy $h_{\mathbf{c},\mathbf{r}}(p - u.\mathsf{cen}) = *$, then p is contained in both sets. This will mean that the tree may be super-linear in size (if we are not careful about the setting of α). We will first bound the depth of the tree.

Lemma 4.3. *The total depth of an execution of* Preprocess(P) *is* $O(\sqrt{d}\log(n\log\Phi)\log\Phi)$ *with high probability.*

Proof. We will show that the following occurs with high probability, which in turn implies the desired bound on the depth. Let u be any node of the tree and $P_u \subset P$ such that u was the output of Preprocess(P_u). Then, over the randomness of the next s levels down the tree, we consider any path $u = \mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_s$ down the tree where $\mathbf{u}_\ell$ is the child of $\mathbf{u}_{\ell-1}$, and we have $\mathbf{u}_s.\mathsf{rad} \leq u.\mathsf{rad}/2$. Suppose that we set s such that the above event occurs with probability at least $1 - 1/o(\log\Phi)$ (so we can union bound over $O(\log\Phi)$ such events). Then, every s levels of the tree, the values of $\mathbf{u}.\mathsf{rad}$ decrease by a factor of 2, and by the definition of the aspect ratio Φ, there are at most $O(s\log\Phi)$ recursive levels before Line 6 always returns.

We now show that we may set $s = \Theta(\sqrt{d}\log(n\log\Phi))$ in the above argument. Consider fixing any two points $p_1, p_2 \in P_u$. Let $\mathbf{T}$ denote the random sub-tree rooted u which recursively contains children $\boldsymbol{v}$ generated from calls Preprocess$(P_{\boldsymbol{v}})$ in Line 14 where $p_1, p_2 \in P_{\boldsymbol{v}}$. We notice that $\boldsymbol{v}.\mathsf{rad}$ is always at most $2 \cdot u.\mathsf{rad}$ (because $P_{\boldsymbol{v}} \subset P_u$ and the factor of 2 occurs because we may choose a different center of $\mathtt{MEB}(P_u)$ and $\mathtt{MEB}(P_{\boldsymbol{v}})$), so that in

Line 7, the distribution $\mathcal{D}$ has $\boldsymbol{v}.\mathsf{R} \le 4u.\mathsf{rad}$. Then, the number of nodes at depth ℓ of $\mathbf{T}$ is a (simple) branching process (known as a Galton-Watson process), where in a node which contains p_1 and p_2 undergoes the following process:

- By Lemma 4.1, the fact $R \le 4u.\mathsf{rad}$, and $\|p_1 - p_2\|_2 \ge u.\mathsf{rad}/2$, we have that with probability at least $c_1/\sqrt{d}$, the hash $h_{\mathbf{c},\mathbf{r}}$ satisfies $h_{\mathbf{c},\mathbf{r}}(p_1 - \boldsymbol{v}.\mathsf{cen}), h(p_2 - \boldsymbol{v}.\mathsf{cen}) \in \{0,1\}$ and $h_{\mathbf{c},\mathbf{r}}(p_1 - \boldsymbol{v}.\mathsf{cen}) \ne h(p_2 - \boldsymbol{v}.\mathsf{cen})$. Thus, the node $\boldsymbol{v}$ has *no descendants in* $\mathbf{T}$ (since we've separated p_1, p_2).
- If $h_{\mathbf{c},\mathbf{r}}(p_1 - \boldsymbol{v}.\mathsf{cen}) = h_{\mathbf{c},\mathbf{r}}(p_2 - \boldsymbol{v}.\mathsf{cen}) = *$, then p_1 and p_2 are included in both calls to PREPROCESS($P_{\boldsymbol{v}} \setminus$ Cap(0)) and PREPROCESS($P_{\boldsymbol{v}} \setminus$ Cap(1)) in Line 14. In this case, $\boldsymbol{v}$ has *two descendants in* $\mathbf{T}$, but by Lemma 4.1, this occurs with probability at most $\alpha/\sqrt{d}$.
- Finally, if neither of the above two holds, then $\boldsymbol{v}$ has one descendant in $\mathbf{T}$, since p_1 and p_2 are not separated, but they are also not both replicated.

Suppose that we let $d(s)$ denote the probability that a fixed node u has some ancestor at depth s in $\mathbf{T}$. Then, we may give a recursive upper bound for $d(s)$, since either (i) a node has one child (with probability at most $1 - c_1/\sqrt{d}$) and that child must have some ancestor at depth $s-1$ in $\mathbf{T}$, or (ii) a node has two children (with probability at most $\alpha/\sqrt{d}$), and then at least one of them must have an ancestor at depth $s-1$. Thus,

$$d(s) \le \left(1 - \frac{c_1}{\sqrt{d}}\right) \cdot d(s-1) + \frac{\alpha}{\sqrt{d}}\left(1 - (1 - d(s-1))^2\right) \le \left(1 - \frac{c_1}{\sqrt{d}} + \frac{2\alpha}{\sqrt{d}}\right) \cdot d(s-1),$$

implying

$$\Pr\left[\mathbf{T} \text{ has depth } s\right] \le \left(1 - \frac{c_1}{\sqrt{d}} + \frac{2\alpha}{\sqrt{d}}\right)^{s-1}.$$

Since $\alpha \le c_1/4$ and we let $s = \Theta(\sqrt{d}\log(n\log\Phi))$, this probability is at most $(n\log\Phi)^{-10}$, so we can union bound over n^2 potential pairs p_1, p_2 at distance $\|p_1 - p_2\|_2 \ge u.\mathsf{rad}/2$, and we get the desired bound. ∎

LEMMA 4.4. *With probability at least* $1 - o(1)$*, an execution of* PREPROCESS(P) *results in a rooted tree of* $O(n\sqrt{d}\log(n\log\Phi)\log\Phi)$ *nodes.*

Proof. Suppose that we consider executing PREPROCESS(P) while allowing at most t depths of recursion. Then, we bound the expected number of leaf nodes resulting from such a tree, and the total number of nodes will be at most t times that. Since each leaf node contains at least one point, we bound the total number of leaf nodes containing any one point $p \in P$. Any node u generated from call PREPROCESS(P_u) where $p \in P_u$ has that either (i) u is a leaf, (ii) the point p satisfies $h_{\mathbf{c},\mathbf{r}}(p - u.\mathsf{cen}) \in \{0,1\}$ in Line 7 and p lies in a single child of u, or (iii) p satisfies $h_{\mathbf{c},\mathbf{r}}(p - u.\mathsf{cen}) = *$ in Line 7, and then p lies in both children of u. Again, this length-t branching process is a simple process, and it is not hard to see that the expected number of leaves containing p within depth t of u is at most $2^{\alpha t/\sqrt{d}}$, and thus the total expected number of leaves at most $n2^{\alpha t/\sqrt{d}}$.

By the Markov inequality at a union bound over the depth of PREPROCESS(P, ϵ, σ), it suffices to consider $t = \Theta(\sqrt{d}\log(n\log\Phi)\log\Phi)$, and since $\alpha \le c_1/(4\log n\log\Phi)^2$, $2^{\alpha t/\sqrt{d}} = O(1)$ and $t \cdot O(n)$ gives us the desired bound. ∎

LEMMA 4.5. *For any fixed query* $q \in \mathbb{R}^d$*, the following occurs with high probability. Suppose* $\mathbf{u}$ *is the output of an execution of* PREPROCESS(P)*, and* $\boldsymbol{\eta} \in \mathbb{R}$ *is the output of* QUERY($q, \mathbf{u}$)*. Then, we have:*

$$(1-\epsilon)\sum_{p\in P}\mathsf{K}(p,q) - \sigma|P| \le \boldsymbol{\eta} \le (1+\epsilon)\sum_{p\in P}\mathsf{K}(p,q) + \sigma|P|.$$

Proof. First, the query q generates a root-to-leaf path given by the executions QUERY($q, \boldsymbol{v}$). In order for the algorithm to avoid outputting "fail," it cannot be the case $h_{u.\mathbf{c},u.\mathbf{r}}(q - u.\mathsf{cen}) = *$. For any fixed node v, the probability that this occurs is always at most $\alpha/\sqrt{d}$. Therefore, we note that the probability over the execution of PREPROCESS(P) that QUERY($q, \mathbf{u}$) outputs "fail" sometime within the first $o(\sqrt{d}/\alpha)$ recursive calls is at most

$o(1)$. At the same time, the depth of the tree is at most $O(\sqrt{d}\log(n\log\Phi)\log\Phi) = o(\sqrt{d}/\alpha)$ with high probability. Taking a union bound, we have that we reach the end of the tree and we do not output "fail" with high probability.

The other necessary event to check is that, if v is a node on the root-to-leaf path in an execution of $\textsc{Query}(q, u)$, and the call which initialized v was $\textsc{Preprocess}(P_v)$, then we always have every $p \in P_v$ satisfies $\|p - v.\mathsf{newcen}\|_2 \geq v.\mathsf{R}/100$ and $\|q - v.\mathsf{newcen}\|_2 \geq v.\mathsf{R}/100$. For any fixed node, the probability that this occurs is at least $1 - (n+1)2^{-\Omega(d)}$, so we can again union bound over the first $O(\sqrt{d}\log(n\Phi)\log\Phi)$ levels of the tree when $d = \omega(\log n \log\log\Phi)$.

Finally, we note that in any execution of $\textsc{Query}(q, \mathbf{u})$, the output is given by a sum of at most $|P|$ executions of $\textsc{QueryCaptured}$, so that the additive errors σ accumulate to at most $\sigma|P|$. ∎

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.

[ACSS20] Josh Alman, Timothy Chu, Aaron Schild, and Zhao Song. Algorithms and hardness for linear algebra on geometric graphs. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 541–552. IEEE, 2020.

[ACW17a] Haim Avron, Kenneth L. Clarkson, and David P. Woodruff. Faster kernel ridge regressionusing sketching and preconditioning. *SIAM Journal of Matrix Analysis and Applications*, 38(4):1116–1138, 2017.

[ACW17b] Haim Avron, Kenneth L. Clarkson, and David P. Woodruff. Sharper bounds for regularized data fitting. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, 2017.

[AINR14] Alexandr Andoni, Piotr Indyk, Huy L. Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA '2014)*, pages 1018–1028, 2014. Available as arXiv:1306.1547.

[AKK+20a] 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.

[AKK+20b] Thomas D. Ahle, Michel Kapralov, Jakob B. T. Knudsen, Rasmus Pagh, Ameya Velingker, David P. Woodruff, and Amir Zandieh. Oblivious sketching of high-degree polynomial kernels. In *Proceedings of the 31st ACM-SIAM Symposium on Discrete Algorithms (SODA '2020)*, 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 (ICML '2017)*, 2017.

[ALRW17] Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Optimal hashing-based time–space trade-offs for approximate near neighbors. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA '2017)*, 2017. Available as arXiv:1608.03580.

[ALS21] Ryan Alweiss, Yang P. Liu, and Mehtaab Sawhney. Discrepancy minimization via a self-balancing walk. In *Proceedings of the 53rd ACM Symposium on the Theory of Computing (STOC '2021)*, 2021.

[ANN+18] Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Data-dependent hashing via non-linear spectral gaps. In *Proceedings of the 50th ACM Symposium on the Theory of Computing (STOC '2018)*, 2018.

[ANW14] Haim Avron, Huy L. Nguyen, and David P. Woodruff. Subspace embeddings for the polynomial kernel. In *Proceedings of Advances in Neural Information Processing Systems 25 (NIPS '2014)*, 2014.

[AR15] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC '2015)*, pages 793–801, 2015. Available as arXiv:1501.01062.

[AS23] Josh Alman and Zhao Song. Fast attention requires bounded entries. *CoRR*, abs/2302.13214, 2023.

[BCIS18a] Arturs Backurs, Moses Charikar, Piotr Indyk, and Paris Siminelakis. Efficient density evaluation for smooth kernels. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2018)*, 2018.

[BCIS18b] 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.

[BH86] J. Barnes and P. Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324(4):446–449, 1986.

[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.

[CJLW22] Xi Chen, Rajesh Jayaram, Amit Levi, and Erik Waingarten. New streaming algorithms for high dimensional emd and mst. In *Proceedings of the 54th ACM Symposium on the Theory of Computing (STOC '2022)*, 2022.

[CKNS20a] Moses Charikar, Michael Kapralov, Navid Nouri, and Paris Siminelakis. Kernel density estimation through density constrained near neighbor search. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 172–183. IEEE, 2020.

[CKNS20b] Moses Charikar, Michael Kapralov, Navid Nouri, and Paris Siminelakis. Kernel density estimation through density constrained near neighbor search. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS '2020)*, 2020.

[CS17a] 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.

[CS17b] Moses Charikar and Paris Siminelakis. Hashing-based-estimators for kernel density in high dimensions. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2017)*, 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.

[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.

[GFKS02] Thomas Gärtner, Peter A Flach, Adam Kowalczyk, and Alexander J Smola. Multi-instance kernels. In *ICML*, volume 2, pages 179–186, 2002.

[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.

[GSM03] Georgescu, Shimshoni, and Meer. Mean shift based clustering in high dimensions: a texture classification example. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 456–463 vol.1, 2003.

[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.

[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.

[MFS+17] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, Bernhard Schölkopf, et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017.

[MM17] Cameron Musco and Christopher Musco. Recursive sampling for the nyström method. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS '2017)*, 2017.

[PT20] Jeff M. Phillips and Wai Ming Tai. Near-optimal coresets for kernel density estimates. *Discrete and Computational Geometry*, 63(4):867–887, 2020.

[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.

[RR08] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of Advances in Neural Information Processing Systems 21 (NIPS '2008)*, 2008.

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

[RW10] Alessandro Rinaldo and Larry Wasserman. Generalized density clustering. *The Annals of Statistics*, pages 2678–2722, 2010.

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

[SSPG16] Zoltán Szabó, Bharath K Sriperumbudur, Barnabás Póczos, and Arthur Gretton. Learning theory for distribution regression. *The Journal of Machine Learning Research*, 17(1):5272–5311, 2016.

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

2004.

[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.

[Wai19] Martin J. Wainwright. *High-dimensional statistics: a non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.

[Wen04] Holger Wendland. *Scattered Data Approximation*. Cambridge University Press, 2004.

[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.