Kernel Density Estimation through Density Constrained Near Neighbor Search

Moses Charikar
Stanford University

Michael Kapralov
EPFL

Navid Nouri
EPFL

Paris Siminelakis
UC Berkeley

January 12, 2021

Abstract

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

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

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

Contents

1 Introduction

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

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

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

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

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

In the rest of the paper, we use the notation μ^* defined as $\mu^* := K(P, \mathbf{q})$, and μ is a quantity that satisfies $\mu^* \leq \mu \leq 4\mu^*$.[1] Moreover, in the statement of the main results, we assume that a constant factor lower bound to the actual kernel density, μ^*, is known. In general, if we only know that $\mu^* \geq \tau$ for some τ, then the μ^* terms in the space should be replaced by τ (similar to prior results in the literature). However, the query time can always be stated in terms of μ^*.

The kernel density estimation problem has received a lot of attention over the years, with very strong results available for low dimensional datasets. For example, the celebrated fast multipole method [BG97] and the related Fast Gauss Transform can be used to obtain efficient data structure for KDE (and in fact solves the more general problem of multiplying by a kernel matrix). However, this approach suffers from an exponential dependence on the dimension of the input data points, a deficiency that it shares with other tree-based methods [GM01, GM03, YDGD03, LMG06, RLMG09]. A recent line of work [CS17, CS19, BCIS18, BIW19a] designed sublinear query algorithms for kernel density estimation in high dimensions using variants of the Locality Sensitive Hashing [CS17] framework of Indyk and Motwani [IM98].

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

In this work we take a different approach to implementing importance sampling for KDE using LSH-based near neighbor search techniques. At a high level, our approach consists of first performing independent sampling on the dataset, and then using using LSH-based near neighbor search primitives to extract relevant data points from this sample[2]. The key observation is that the sampled dataset in the KDE problem has nice geometric structure: the number of data points around a given query cannot grow too fast as a function of distance and the actual KDE value μ (we refer to these constraints as density constraints – see Section 2 for more details). The fact that our approach departs from the idea of constructing unbiased estimators of KDE directly from

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

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

LSH buckets turns out to have two benefits: first, we immediately get a simple algorithm that uses classical LSH-based near neighbor search primitives (Euclidean LSH of Andoni and Indyk [AI06]) to improve on or essentially matches all prior work on kernel density estimation for radial kernels. The result is formally stated as Theorem 1 for the Gaussian kernel below, and its rather compact analysis in a more general form that extends to other kernels is presented in Section 4. The second benefit of our approach is that it distills a clean near neighbor search problem, which we think of as near neighbor search under density constraints, and improved algorithms for that problem immediately yield improvements for the KDE problem itself. This clean separation allows us to use the recent exciting data-depending techniques pioneered by [?, ?, ALRW17] in our setting. It turns out that while it seems plausible that data-dependent techniques can improve performance in our setting, actually designing an analyzing a data-dependent algorithm for density constrained near neighbor search is quite nontrivial. The key difficulty here lies in the fact that one needs to design tools for tracking the evolution of the density of the dataset around a given query through a sequence of recursive partitioning steps (such evolution turns out to be quite involved, and in particular governed by a solution to an integral equation involving the log density of the kernel and properties of Spherical LSH). The design of such tools is our main technical contribution and is presented in Section 5. The final result for the Gaussian kernel is given below as Theorem 2, and extensions to other kernels are presented in Section 5.

1.1 Our results

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

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

Theorem 1. *Given a kernel $K(\mathbf{p},\mathbf{q}) := e^{-a\|\mathbf{p}-\mathbf{q}\|_2^2}$ for any $a > 0$, $\epsilon = \Omega\left(\frac{1}{\text{polylog} n}\right)$, $\mu^* = n^{-\Theta(1)}$ and a data set of points P, there exists an algorithm for preprocessing and an algorithm for query procedure such that after receiving query $\mathbf{q}$ one can approximate $\mu^* := K(P,\mathbf{q})$ (see Definition 17) up to $(1\pm\epsilon)$ multiplicative factor, in time $\widetilde{O}\left(\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}\right)$, and the space consumption of the data structure is*

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

Remark 1. In Theorem 1 (and similar theorems in the rest of the paper), we assumed that $\epsilon = \Omega\left(\frac{1}{\text{polylog} n}\right)$ and $\mu^* = n^{-\Theta(1)}$, so that we can assume $d = \widetilde{O}(1)$ (and ignore the dependencies on dimension in the statements). The reason (for $d = \widetilde{O}(1)$) is that in this case the contribution of far points (points at distance $\Omega(\log n)$) is negligible and for close points, we can use Johnson-Lindenstrauss (JL) lemma to reduce the dimension to $O(\text{polylog} n)$, without distorting the kernel value by a more than $1 \pm o(1)$ multiplicative factor. If we remove these assumptions, we need to multiply the query-time and space bounds by dimension d.

This theorem is stated and proved as Theorem 15 in Section 4. To get a sense of the improvement, the result of [CS17] exhibited query time that is roughly a square root of the query time of uniform random sampling. Our result uses the same LSH family as in [CS17] but achieves query time that is itself roughly the square root of that of [CS17]!

Data-Dependent LSH Our main technical contribution is a collection of techniques for using data dependent hashing introduced by [**?**, **?**, ALRW17] in the context of kernel density estimation. Unlike these works, however, who had no assumptions on the input data set, we show how to obtain refined bounds on the efficiency of near neighbor search under density constraints imposed by assumptions on KDE value as a function of the kernel. This turns out to be significantly more challenging: while in approximate near neighbor search, as in [ALRW17], it essentially suffices to track the size of the dataset in recursive iterations of locality sensitive hashing and partitioning into spheres, in the case of density constrained range search problems arising from KDE one must keep track of the distribution of points across different distance scales in the hash buckets, i.e. track evolution of functions as opposed to numbers. This leads to a natural linear programming relaxation that bounds the performance of our algorithm that forms the core of our analysis[3]. Our ultimate result for the Gaussian kernel is:

Theorem 2. *For Gaussian kernel K, any data set of points P and any $\epsilon = \Omega\left(\frac{1}{\text{polylog} n}\right)$, $\mu^* = n^{-\Theta(1)}$, using Algorithm 1 for preprocessing and Algorithm 2 for the query procedure, one can approximate $\mu^* := K(P, \mathbf{q})$ (see Definition 17) up to $(1\pm\epsilon)$ multiplicative factor, in time $\widetilde{O}(\mu^{-0.173-o(1)}/\epsilon^2)$. The space complexity of the algorithm is also bounded by*

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

for $c = 10^{-3}$.[4]

The proof of Theorem 2 is given in Section 5.

Our techniques extend to other kernels – the extensions are presented in Section 5.

1.2 Related Work

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

[CS17] revisited the problem and introduced a technique, called Hashing-Based-Estimators (HBE), to implement low-variance Importance Sampling (IS) efficiently for any query through Locality Sensitive Hashing (LSH). For the Gaussian $f(r) = e^{-r^2}$, Exponential $f(r) = e^{-r}$, and t-Student kernels $f(r) = (1 + r^t)^{-1}$ the authors gave the first sub-linear algorithms that require $O(\min\{1/\epsilon^2\sqrt{\mu}, n\})$ kernel evaluations. Using ideas from Harmonic Analysis, the technique was later extended in [CS19], to apply to more general kernels resulting in the first data structures that

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

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

require $O(\min\{1/\epsilon^2\sqrt{\mu}, n\})$ kernel evaluations to approximate the density for log-convex kernels $e^{\phi(\langle x,y\rangle)}$. Furthermore, under the Orthogonal Vectors Conjecture it was shown that there does not exist a data structure that solves the KDE problem under the Gaussian kernel in time $n^{1-o(1)}/\mu^{o(1)}$ and space $n^{2-o(1)}/\mu^{o(1)}$.

The work most closely related to ours is that of [BCIS18]. [BCIS18] introduced a technique, called Spherical Integration, that uses black-box calls to c-ANN data structures (constructed on sub-sampled versions of the data set) to sample points from "spherical annuli" (r, cr) around the query, for all annuli that had non-negligible contribution to the density of the query. For kernels with polynomial tails of degree t, their approach required $\widetilde{O}(c^{5t})$ calls to such data-structures (without counting the query time required for each such call) to estimate the density. Unfortunately, this approach turns out to be constraining due to its reliance on **black-box** c-ANN calls, and in particular only applies to polynomial kernels. Our techniques in this paper recover the result of [BCIS18] up to $\mu^{-o(1)}$ factors as a special case (see Section 4). Furthermore, the $\mu^{-o(1)}$ factor loss that we incur is only due to the fact that we are using the powerful Euclidean LSH family in order to achieve strong bounds for kernels that exhibit fast decay (e.g., Gaussian, exponential and others) using the same algorithm. For polynomial kernels the dependence on μ in our approach can be reduced to polylogarithmic in $1/\mu$ by using an easier hash family (e.g., the hash family of [DIIM04]; see [Sym19, Chapter 10] for details).

Scalable approaches to KDE and Applications Recent works [SRB+19, BIW19a] also address scalability issues of the original approach of [CS17]. [SRB+19] designed a more efficient adaptive procedure that can be used along with Euclidean LSH [DIIM04] to solve KDE for a variety of power-exponential kernels, most prominently the Gaussian. Their algorithm is the first practical algorithm for Gaussian KDE with worst case guarantees that improve upon random sampling in high dimensions. Experiments in real-world data sets show [SRB+19] that the method of [CS17], yields practical improvements for many real world datasets. [BIW19a] introduced a way to sparsify hash tables and showed that in order to estimate densities $\mu^* \geq \tau \geq \frac{1}{n}$ one can reduce the space usage of the data structures [SRB+19] from $O(1/\tau^{3/2}\epsilon^2)$ to $O(1/\tau\epsilon^2)$. The authors also evaluated their approach on real world data for the Exponential $e^{-\|x-y\|_2}$ and Laplace $e^{-\|x-y\|_1}$ kernels showing improvements compared to [CS17] and uniform random sampling. A related approach of Locality Sensitive Samplers [SS17] has also been applied to obtain practical procedures in the contexts of Outlier detection [LS18], Gradient Estimation [CXS19] and Clustering [LS19]. Finally, [WCN18] uses similar ideas to address the problem of approximate range counting on the unit sphere.

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

1.3 Outline

We start by giving a technical overview of the paper in Section 2. Preliminary definitions and results are presented in Section 3. In Section 4, we present our data-independent result for Gaussian KDE and state a general version of our result for other decreasing kernels. We present our data structure based on Data-Dependent LSH for Gaussian KDE in Section 5 and its analysis in Sections 6 (Query time), 7 (Valid execution path analysis), 8 (Linear Program analysis), and 9 (Primal-Dual solution).

2 Technical overview

In this section we give an overview of our results and the main ideas behind them. For simplicity we use the Gaussian kernel, even though both our results extend to more general settings. Thus, for the purposes of this overview our problem is: preprocess a dataset P of n points $\mathbf{p}_1, \dots, \mathbf{p}_n \in \mathbb{R}^d$ into a small space data structure that allows fast KDE queries, i.e. can quickly approximate, given $\mathbf{q} \in \mathbb{R}^d$, the quantity

$$K(P, \mathbf{q}) := \frac{1}{|P|} \sum_{\mathbf{p} \in P} K(\mathbf{p}, \mathbf{q}), \tag{2}$$

where

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

We present two schemes based on ideas from data independent and data dependent LSH schemes. Both schemes employ the strategy of first sampling the dataset at a sequence of geometric levels, and then using near neighbor search algorithms to retrieve all points at an appropriate distance from the query from the sample. The difference between the two approaches lies in the implementation and analysis of the near neighbor search primitive used for this retrieval. In what follows we first overview our approach to implementing importance sampling for KDE using near neighbor search primitives, and then instantiate this scheme with data-independent (Section 2.1) and data-dependent (Section 2.2) schemes.

2.1 Data-independent algorithm (Section 4)

We start by showing a new application of data-independent locality sensitive hashing to KDE that results in a simple scheme that provides the following result.

Theorem 3 (Informal version of Theorem 15)**.** *If $\mu^* := K(P, \mathbf{q})$, then there exists an algorithm that can approximate μ^* up to $(1 \pm \epsilon)$ multiplicative factor, in time $\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}$, using a data structure of size*

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

We remark that the actual non-adaptive algorithm that we present in Section 4 is more general than the above and applies to a wide class of kernels. In particular, it simultaneously improves upon all prior work on radial kernels that exhibit fast tail decay (such as the exponential and the Gaussian kernels) [CS17] as well as matches the result of [BCIS18] on kernels with only inverse polynomial rate of decay up to $\mu^{-o(1)}$ factors.

We now outline the algorithm and the analysis. The main idea is simple: we note that in order to approximate the sum on the right hand side of (2), ideally we would like to do importance sampling, i.e. pick every point with probability proportional to its contribution to the KDE value. It is of course not immediate how to do this, since the contribution depends on the query, which we do not know at the preprocessing stage. However, we show that it is possible to simply prepare sampled versions of the input dataset using a fixed geometric sequence of sampling rates, and then use locality sensitive hashing to retrieve the points relevant to the given query from this sample efficiently. Below, we present an overview of our algorithm.

Geometric weight levels: Let $J := \lceil \log \frac{1}{\mu} \rceil$ and partition the points in the data set into J sets, such that the contribution of any point in the j'th set to the kernel density is $\approx 2^{-j}$. If $w_i := K(\mathbf{p}_i, \mathbf{q})$, then we define (see Definition 18) level sets

$$L_j := \left\{ \mathbf{p}_i \in P : w_i \approx 2^{-j} \right\}.$$

The kernel density can be expressed in terms of the level sets as

$$K(P, \mathbf{q}) \approx \frac{1}{n} \sum_{j=1}^{J} |L_j| \cdot 2^{-j},$$

which implies size upper bounds for L_j, namely:

$$|L_j| \lesssim 2^j n\mu. \tag{3}$$

This means that for every query $\mathbf{q}$ such that the KDE value at $\mathbf{q}$ equals μ to within constant factors one can place an upper bound of $2^j n\mu$ on the number of points at distance corresponding to level L_j – these are exactly the geometric weight constraints that make our near neighbor search primitives very efficient. Note that we are only considering level sets L_j for j at most $J = \lceil \log \frac{1}{\mu} \rceil$. We describe our implementation of importance sampling now.

Importance sampling: Suppose that one designs a sampling procedure that samples each point $\mathbf{p}_i$ with probability p_i and calculates the following estimator

$$Z = \sum_i \frac{\chi_i}{p_i} w_i$$

where $\chi_i = 1$ if $\mathbf{p}_i$ is sampled and $\chi_i = 0$ otherwise. Obviously, this estimator is an unbiased estimator for $n\mu^*$. So, if we can prove that this estimator has a relatively low variance, then by known techniques (repeating many times, averaging and taking the median) one can approximate μ^*, efficiently. It can be shown (see Claim 25) that if p_i's are proportional to w_i's (more specifically, we set $p_i \approx \frac{w_i}{n\mu}$) then the variance is low. This approach is known as **importance sampling**. In other words, we need to sample points with higher contribution, with higher probability.

If L_j's were known to the algorithm in the preprocessing phase, then for each j, one could have sampled points in L_j with probability $\approx \frac{1}{2^j n\mu}$. However, the query is not known in the preprocessing phase and hence geometric weight levels are not known beforehand.

Our approach is the following: for each j we sample the data set P with probability $\frac{1}{2^j n\mu}$. Then, we prepare a data structure (for this sampled data set) that can **recover** any sampled point with contribution $\approx 2^{-j}$ in the query procedure, efficiently and with high probability. Note that the number of points with contribution $\geq 2^{-j}$ is upper bounded by $2^j n\mu$. So, on average after the sub-sampling we expect to have at most $O(1)$ point from $L_1 \cup \ldots \cup L_j$. On the other hand, since Gaussian kernel is a decreasing function of distance, points in $L_{j+1} \cup \ldots \cup L_J$ are actually further than the query. Thus, our **recovery** problem can be seen as an instance of **near neighbor** problem. Therefore, we use the **locality sensitive hashing (LSH)** approach, which has been used in the literature for solving the approximate near neighbor problem.

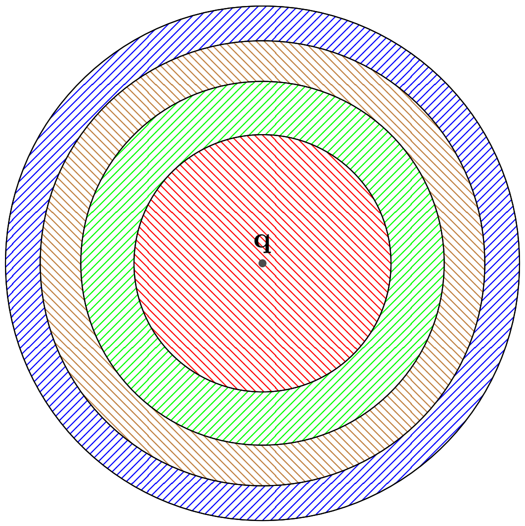


Figure 1: Illustration of distance levels induced by geometric weight levels. Areas marked with colors red, green, brown, blue and so on correspond to geometric weight levels $L_1, \dots, L_4$ and so on.

Using Euclidean LSH for recovery: Now, we explain how one can use Euclidean LSH scheme to design a data structure to recover points from L_j in the corresponding sub-sampled data set.

We first present an informal and over-simplified version of LSH function used in [AI06]. Roughly speaking [AI06] presents the following result (see Lemma 16 for the formal statement):

Lemma 4 (Informal version of Lemma 16)**.** *For every r there exists a (locality sensitive) hash family such that, if $\mathbf{p}$ (a 'close' point) and $\mathbf{p}'$ (a 'far' point) are at distance r and $\geq c \cdot r$ (for some $c \geq 1$) of some point $\mathbf{q}$, respectively, then if*

$$p := \Pr[h(\mathbf{p}) = h(\mathbf{q})],$$

then

$$\Pr[h(\mathbf{p}') = h(\mathbf{q})] \leq p^{(1-o(1))c^2}.$$

Now given a query $\mathbf{q}$, for every j we use Euclidean LSH to retrieve the points in L_j from a sample of the dataset where every point is included with probability $\frac{1}{2^j n\mu}$. We repeat the hashing process multiple times to ensure high probability of recovery overall, as in the original approach of [IM98]. However, the parameter setting and the analysis are different, since in the context of KDE we can exploit the geometric structure of the sampled dataset, namely upper bounds on the sizes of level sets L_j given in (3) above – we outline the parameter setting and analysis now.

On the other hand, geometric weight levels induce distance levels (see Definition 18 and Figure 1). Roughly speaking, for the Gaussian kernel if $\mathbf{p} \in L_j$ and $\mathbf{p}' \in L_i$, then

$$\frac{\|\mathbf{p}' - \mathbf{q}\|_2}{\|\mathbf{p} - \mathbf{q}\|_2} \approx \sqrt{\frac{i}{j}} =: c_{i,j}.$$

Recall that since we sampled the data set with probability $\frac{1}{2^j n\mu}$ then for every i we will have at most $\approx 2^{i-j}$ points from L_i in the sampled set, in expectation. In particular, most likely the sample does not contain points from level sets $i < j$. We instantiate Euclidean LSH from Lemma 4 with the 'near' distance r being the distance to the target level set L_j. Let p denote the probability that the query collides with a point in L_j. Now by Lemma 4 we upper bound the expected number of points from level sets $L_i, i > j$, in the bucket of the query:

$$\sum_{i>j} 2^{i-j} \cdot p^{c_{i,j}^2}$$

We now select p (note that Lemma 4 allows flexibility in selecting p, which is achieved by concatenating hash functions; see Section 4 for the detailed analysis). We set p such that the number of points from each L_i in the bucket of the query is at most 1 for all $i > j$. For every such i, $2^{i-j} \cdot p^{\frac{i}{j}} \leq 1$ implies $p \leq \left(\frac{1}{2}\right)^{j-\frac{j^2}{i}}$, and hence we let

$$p = p_j = \min_{i>j} \left(\frac{1}{2}\right)^{j-\frac{j^2}{i}},$$

where we give the probability a subscript j to underscore that this is the setting for level set L_j.

On the other hand, note that since the point that we want to recover will be present in the query's bucket with probability p_j, we need to repeat this procedure $\widetilde{O}\left(\frac{1}{p_j}\right)$ times, to recover the point with high probability. This means that for every j the contribution of level set L_j to the query time will be $\widetilde{O}\left(\frac{1}{p_j}\right)$. Now, note that

$$\begin{aligned}\max_{j\in[J]} \log_2 \frac{1}{p_j} &= \max_{j\in[J]} \max_{i\in(j,J]} \left(j - \frac{j^2}{i}\right) \\ &= J \cdot \max_{j\in[J]} \frac{j}{J} \cdot \left(1 - \frac{j}{J}\right) \\ &= \frac{J}{4} \\ &= \frac{1}{4} \log \frac{1}{\mu},\end{aligned} \tag{4}$$

implying a $(1/\mu)^{0.25}$ upper bound on the query time. This (informally) recovers the result mentioned in Theorem 3. Note that the space complexity of our data structure is no larger than the number of data points times the query time, i.e., $\approx n(1/\mu)^{0.25}$, since at every sampling rate we hash at most the entire dataset about $(1/\mu)^{0.25}$ times independently. The space complexity can also be bounded by $\widetilde{O}(1/\mu)$ by noting that the datasets for which we have the highest query time and hence many repetitions are in fact heavily subsampled versions of the input dataset. These bounds are incomparable, and the latter is preferable for large values of KDE value μ.

We used the Gaussian kernel in the informal description above to illustrate our main ideas, but the approach extends to a very general class of kernels. In particular, it gives improvements over all prior work on the KDE problem for shift invariant kernels (with the only exception that our results essentially match the results of [BCIS18], where an already very efficient algorithm with a polylogarithmic dependence on $1/\mu$ is presented). We present the detailed analysis of this approach in Section 4.

2.2 Data dependent algorithm (Section 5)

We note that the efficiency of our implementation of importance sampling relies heavily on the efficiency of near neighbor search primitive under density constraints. In this section we show

how to use data-dependent techniques, i.e. data partitioning followed by the use of the more efficient Spherical LSH, to achieve significantly better results. Our approach builds on the exciting recent line of work on data-dependent near neighbor search [**?**, **?**, ALRW17], but the fact that we would like to optimally use the assumptions on the density of various spherical ranges that follow from assumptions on KDE value, the analysis turns out to be significantly more challenging. In particular, the core of our approach is a linear program that allows one to analyze the worst case evolution of densities during the hashing process. The analysis is presented in Section 5, Section 8 and Section 9. Since the analysis is somewhat involved, we present it for the case of the Gaussian kernel to simplify notation. We then provide a version of the key lemma for other kernels and state the corresponding results.

Theorem 5 (Informal version of Theorem 26)**.** *There exists an algorithm that, when K is the Gaussian kernel and $\mu^* := K(P, \mathbf{q})$, for $\epsilon \in (0,1)$ approximates μ^* to within a $(1 \pm \epsilon)$ multiplicative factor, in expected time $\left(\frac{1}{\mu^*}\right)^{0.173+o(1)}$ and space $\min\{n\left(\frac{1}{\mu^*}\right)^{0.173+o(1)}, \left(\frac{1}{\mu^*}\right)^{1+o(1)}\}$.*

Our techniques extend to kernels beyond the Gaussian kernel (e.g., the exponential kernel, for which we obtain query time $\left(\frac{1}{\mu^*}\right)^{0.1+o(1)}$ and space $n\left(\frac{1}{\mu^*}\right)^{0.1+o(1)}$). We outline the extension in Section 5.

Recall that we need to preprocess a dataset P of n points $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathbb{R}^d$ into a small space data structure that allows fast KDE queries, i.e., can quickly approximate, given $\mathbf{q} \in \mathbb{R}^d$, the quantity

$$\mu^* = K(P, \mathbf{q}) = \frac{1}{|P|} \sum_{\mathbf{p} \in P} \exp(-||\mathbf{p} - \mathbf{q}||_2^2/2). \tag{5}$$

Recall also that we assume knowledge of a quantity μ such that

$$\mu^* \leq \mu \leq 4\mu^*. \tag{6}$$

This is without loss of generality by a standard reduction – see Section 5, Remark 2. For simplicity of presentation, in this section we use a convenient rescaling of points so that

$$\mu^* = K(P, \mathbf{q}) = \frac{1}{|P|} \sum_{\mathbf{p} \in P} (1/\mu)^{-||\mathbf{p}-\mathbf{q}||_2^2/2}. \tag{7}$$

Note that this is simply a rescaling of the input points, namely multiplying every coordinate by $(\log(1/\mu))^{-1/2}$. This is for analysis purposes only, and the algorithm does not need to perform such a rescaling explicitly. We fix the query $\mathbf{q}$ for the rest of this section.

Densities of balls around query. Upper bounds on the number of points at various distances from the query point in dataset (i.e., densities of balls around the query) play a central part in our analysis. For any $x \in (0, \sqrt{2})$ let

$$D_x(\mathbf{q}) := \{||\mathbf{p} - \mathbf{q}|| : \ \mathbf{p} \in P, ||\mathbf{p} - \mathbf{q}|| \gtrsim x\}, \tag{8}$$

denote the set of possible distances from $\mathbf{q}$ to points in the dataset P. Note that we are ignoring distances that are too close to x – this is for technical reasons that let us introduce some simplifications with respect to the analysis of [ALRW17] at the expense of a small constant loss in the exponent of the ultimate query time (see Section 5.3 for more discussion of this). When there is no ambiguity we drop $\mathbf{q}$ and x and we simply call it D. For any $y \in D$ we let

$$P_y(\mathbf{q}) := \{\mathbf{p} \in P : ||\mathbf{p} - \mathbf{q}|| \le y\} \tag{9}$$

be the set of points at distance y from $\mathbf{q}$. Since for every $y > 0$

$$\begin{aligned} \mu^* = K(P, \mathbf{q}) &= \frac{1}{n} \sum_{\mathbf{p} \in P} \mu^{||\mathbf{p}-\mathbf{q}||_2^2/2} \\ &\ge \frac{\mu^{y^2/2}}{n} |P_y(\mathbf{q})| \end{aligned}$$

we get

$$|P_y(\mathbf{q})| \le n\mu^* \cdot \left(\frac{1}{\mu}\right)^{\frac{y^2}{2}} \le n \cdot \left(\frac{1}{\mu}\right)^{\frac{y^2}{2}-1}, \tag{10}$$

since $\mu^* \le \mu$ by assumption.

We implement the same importance sampling strategy as in Section 2.1: sample the dataset at a geometric sequence of sampling rates, and for each such sampling rate use approximate near neighbor search primitives (in this case data dependent ones) to retrieve the relevant points (which are generally a few closest points to the query) from the sample. The rescaling of the input space (7) together with the assumption (6) implies that one essentially only needs to care about points $\mathbf{p} \in P$ such that

$$||\mathbf{p} - \mathbf{q}||_2 \approx x \text{ for some } x \in [0, \sqrt{2}).$$

This is because every $\mathbf{p} \in P$ such that $||\mathbf{p}-\mathbf{q}||_2 \ge \sqrt{2}$ contributes at most $(1/\mu)^{-||\mathbf{p}-\mathbf{q}||_2^2/2} \le \mu \le 4\mu^*$ by (6). This means that the contribution of such points can be approximated well by simply sampling every point with probability $\approx 1/n = 1/|P|$ and examining the entire sample – see Section C for details. Therefore in the rest of this section (and similarly in its formal version, namely Section 5) we focus on the following single scale recovery problem:

> Given $x \in (0, \sqrt{2})$ and a sample $\widetilde{P}$ of the dataset P that includes every point with probability $\frac{1}{n} \cdot \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}}$, recover all sampled points at distance at most $\approx x$ from the query.

Fix $x \in (0, \sqrt{2})$, and recall that $\widetilde{P}$ contains every point in P independently with probability $\frac{1}{n} \cdot \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}}$. Note that by (10) for every $y \in (x, \sqrt{2}]$ the expected number of points at distance at most y from q that are included in $\widetilde{P}$ is upper bounded by

$$n \cdot \left(\frac{1}{\mu}\right)^{\frac{y^2}{2}-1} \cdot \frac{1}{n} \cdot \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}} \approx \left(\frac{1}{\mu}\right)^{\frac{y^2-x^2}{2}}. \tag{11}$$

What we defined so far is of course just a reformulation of our approach from Section 2.1, and indeed our data-dependent result follows the overall uniform sampling scheme. The difference comes in a much more powerful primitive for recovering data points at distance $\approx x$ from the query from the uniform sample. We describe this primitive now. In this development we start with the observation that underlies the work of [ALRW17] on data-dependent near neighbor search. Namely, one first observes that if the points in the sampled dataset $\widetilde{P}$ were uniformly random on the sphere (except of course for the actual points at distance $\approx x$ from the query $\mathbf{q}$), then instead of Euclidean LSH one could use random spherical caps to partition the dataset, leading to significantly improved performance. In order to leverage this observation, the work of [ALRW17] introduces the definition of a pseudo-random dataset (see Definition 13 below), gives an efficient procedure for decomposing any dataset into pseudorandom components and shows that the pseudorandom property is sufficiently strong to allow for about the same improvements as a random dataset does. Then their algorithm is a recursive process that partitions a given input dataset using random spherical caps, decomposes the resulting smaller datasets into pseudorandom components and recurses. Our algorithm follows this recipe, but the analysis turns out to be significantly more challenging due to the fact that we need to track the evolution of the densities of balls around the query during this recursive process. In what follows we state the necessary definitions and outline our algorithm.

The work of [ALRW17] introduces a key definition of a pseudorandom dataset (see Definition 13), which we reuse in our analysis and state here for convenience of the reader:

Definition 13 *(Restated) Let P be a set of points lying on $\mathcal{S}^{d-1}(o,r)$ for some $o \in \mathbb{R}^d$ and $r \in \mathbb{R}_+$. We call this sphere a* pseudo-random *sphere*[5]*, if $\nexists \mathbf{u}^* \in \mathcal{S}^{d-1}(o,r)$ such that*

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u}-\mathbf{u}^*|| \le r(\sqrt{2}-\gamma)\right\}\right| \ge \tau \cdot |P|.$$

In other words, a dataset is pseudorandom on a sphere if at most a small fraction of this dataset can be captured by a spherical cap of nontrivially small volume. It turns out [ALRW17] that every dataset can be partitioned into pseudorandom components efficiently, so one can assume that the input dataset is pseudorandom. The significance of this lies in the fact that the power of Spherical LSH manifests itself on the points $\mathbf{p}$ at distance $\sqrt{2}-\gamma$ from the query essentially as well as on uniformly random points. Thus, if the fraction τ of 'violating' points is small, one now use Spherical LSH to partition the dataset into hash buckets and then recursive on the hash buckets, partition them into pseudorandom components and proceed recursively in this manner. Our algorithms follows this recipe, but the analysis introduces new techniques, as we describe below. We start by fixing some notation. Our algorithm (Algorithm 3) recursively constructs a tree $\mathcal{T}$ with alternating levels of SphericalLSH nodes and Pseudorandomify nodes, which correspond to partitioning the dataset using locality sensitive hashing and extraction of dense components as per Definition 13 respectively. At every SphericalLSH node (Algorithm 4) we repeatedly generate subsets P' of the dataset P by sampling a Gaussian vector $g \sim N(0,1)^d$ and letting

$$P' \leftarrow \left\{\mathbf{p} \in P : \left\langle \frac{p-o}{R}, g \right\rangle \ge \eta \right\},$$

where R and o are the radius and center of the sphere that dataset P resides on, and $\eta = \omega(1)$ is an appropriately chosen parameter – we choose η to ensure that the collision probability of the query with a point at distance x from it is exactly $\mu^{1/T}$ for a parameter T (see line 16 of Algorithm 4). Crucially, we chose the parameter η to ensure that the size of the spherical cap is not too large. Specifically, for a parameter $T = \omega(1)$ that governs the depth of our recursive process we choose η to ensure that for every $\mathbf{p} \in P$ such that $||\mathbf{p}-\mathbf{q}||_2 \approx x$ one has

$$\Pr_{g\sim N(0,I)^d}\left[\left\langle \frac{\mathbf{p}-o}{R}, g \right\rangle \ge \eta \middle| \left\langle \frac{\mathbf{q}-o}{R}, g \right\rangle \ge \eta \right] \approx \mu^{1/T},$$

[5]Whenever we say *pseudo-random sphere*, we implicitly associate it with parameter τ, γ which are fixed throughout the paper.

where we assume for simplicity of presentation here that the query is on the sphere. The number of datasets $\widetilde{P}$ is chosen to be such that the query $\mathbf{q}$ collides with any given point $\mathbf{p}$ at distance $\approx x$ with high constant probability over all $O(T)$ levels of the tree $\mathcal{T}$. This means (see Section 6) that the expected number of datasets that the query $\mathbf{q}$ will be exploring is $(1/\mu)^{1/T}$. We limit the depth of the exploration process to $\approx 0.172 \cdot T$ (see line 27 of Algorithm 4), so that the Query algorithm (see Algorithm 6) explores at most $((1/\mu)^{1/T})^{0.172 \cdot T} = (1/\mu)^{0.172}$ leaf datasets in the tree $\mathcal{T}$. The main challenge lies in showing that these leaf datasets have small (nearly constant) expected size. In other words, we need to bound the effect of such a filtering process on the density of balls of various radius y around the query $\mathbf{q}$. Generally, the densities along any root to leaf path are decreasing because of two effects:

Truncation due to pseudorandom spheres: First effect that we consider is the condition that pseudo-randomness of spheres imply over the densities. Consider any query $\mathbf{q}$ and any pseudo-random sphere with radius r, and let ℓ be the distance from $\mathbf{q}$ to the center of the sphere. Let $\mathbf{q}'$ be the projection of the query on the sphere. Then, by pseudo-randomness of the sphere, we know that most of the points are orthogonal to $\mathbf{q}'$, i.e., have distance $\approx \sqrt{2}r$ from $\mathbf{q}'$ (see Lemma 14). However, we are interested in the condition that implies over the densities. Roughly speaking, the orthogonal points are at distance $c := \sqrt{\ell^2 + r^2}$. So we expect that the number of points at distance $\approx c$ will dominate the densities.

Claim 6 (Informal version of Claim 34)**.** *Suppose that a sphere with center o and radius r is pseudo-random. Then, if $\ell \approx ||\mathbf{q} - o||$, $c := \sqrt{\ell^2 + r^2}$ and for all y we let B_y be the number of points at distance y from $\mathbf{q}$ in the sphere. Then, the following conditions hold.*

$$\sum_{y \le c - r\psi} B_y \le \frac{\tau}{1 - 2\tau} \cdot \sum_{y \in (c - r\psi, c + r\psi)} B_y,$$

and

$$\sum_{y \ge c + r\psi} B_y \le \frac{\tau}{1 - 2\tau} \cdot \sum_{y \in (c - r\psi, c + r\psi)} B_y,$$

where $\psi = o(1)$ is small factor.

Removing points due to Spherical LSH: The second phenomenon that reduces the densities is spherical LSH rounds. We set the size of the spherical cap as described above. Under this setting of size of spherical cap, the probability that a spherical cap conditioned on capturing the query, captures $\mathbf{p}$, which is at distance y from $\mathbf{q}$, is given by Claim 35, which is restated informally below.

Claim 7 (Informal version of Claim 35)**.** *Consider a sphere of radius r around point o, and let $\ell \approx ||\mathbf{q} - o||$. Also let $\mathbf{p}$ be a point on the sphere such that $y = ||\mathbf{p} - \mathbf{q}||$. Now, suppose that one generates a Gaussian vector g as in Algorithm 4. Then, we have*

$$\Pr_{g \sim N(0,1)^d}\left[\langle g, \frac{\mathbf{p} - o}{||\mathbf{p} - o||}\rangle \ge \eta | \langle g, \frac{\mathbf{q} - o}{||\mathbf{q} - o||}\rangle \ge \eta\right] \lesssim \exp_\mu\left(-\frac{4(r/x')^2 - 1}{4(r/y')^2 - 1} \cdot \frac{1}{T}\right).$$

where

- *η is such that $\frac{F(\eta)}{G(x'/r,\eta)} \approx \left(\frac{1}{\mu}\right)^{\frac{1}{T}}$ (see line 16 of Algorithm 4).*
- *$x' :=$ Project(x, ℓ, r) (see Definition 11).*

- $y' := \textsc{Project}(y, \ell, r)$.

We use Claim 6 and Claim 7 to bound the evolution of the density of various balls around the query $\mathbf{q}$ in the datasets constructed on the way from the root of the tree $\mathcal{T}$ down to a leaf.

Formally, we gather all necessary information about such a path in the definition of a *valid execution path* below:

Definition 36 (Valid execution path; slightly informal version) Let $R := (r_j)_{j=1}^J$ and $L := (\ell_j)_{j=1}^J$ for some positive values r_j's and ℓ_j's such that for all $j \in [J]$, $x \gtrsim |\ell_j - r_j|$. Also let D be as defined in (8). Then, for

$$\begin{aligned} A &:= (a_{y,j}), \quad y \in D, j \in [J] \cup \{0\} && \text{(Intermediate densities)} \\ B &:= (b_{y,j}), \quad y \in D, j \in [J+1] \cup \{0\} && \text{(Truncated intermediate densities)} \end{aligned}$$

(L, R, A, B) is called a *valid execution path*, if the conditions below are satisfied for $\psi := o(1)$ and $c_j := \sqrt{r_j^2 + \ell_j^2}$ for convenience.

(1) Initial densities condition. The $a_{y,0}$ and $b_{y,0}$ variables are upper-bounded by the initial expected densities in the sampled dataset: for all $y \in D$

$$\sum_{y' \in [0,y] \cap D} a_{y',0} \leq \min\left\{ \exp_\mu\left(\frac{y^2 - x^2}{2}\right), \exp_\mu\left(\frac{1 - x^2}{2}\right) \right\}$$

and

$$\sum_{y' \in [0,y] \cap D} b_{y',0} \leq \min\left\{ \exp_\mu\left(\frac{y^2 - x^2}{2}\right), \exp_\mu\left(\frac{1 - x^2}{2}\right) \right\}$$

(2) Truncation conditions (effect of PseudoRandomify). For any $j \in [J]$, for all $y \in D \setminus [\ell_j - r_j, \ell_j + r_j]$ one has $b_{y,j} = 0$ (density is zero outside of the range corresponding to the j-th sphere on the path; condition **(2a)**), for all $y \in D \cap [\ell_j - r_j, \ell_j + r_j]$ one has $b_{y,j} \leq a_{y,j-1}$ (removing points arbitrarily **(2b)**) and

$$\sum_{y \in [0, c_j - \psi r_j] \cap D} b_{y,j} \leq \frac{\tau}{1 - 2\tau} \cdot \sum_{y \in (c_j - \psi r_j, c_j + \psi r_j) \cap D} b_{y,j} \qquad \text{(condition } \mathbf{(2c)}\text{)}$$

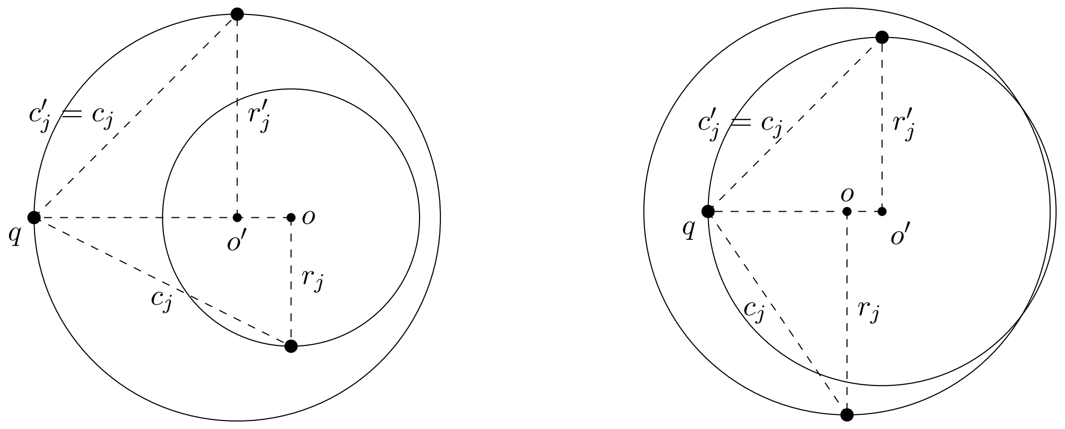
(3) LSH conditions. For every $j \in [J]$ and all $y \in [\ell_j - r_j, \ell_j + r_j] \cap D$

$$a_{y,j} \leq b_{y,j} \cdot \exp_\mu\left(-\frac{4\left(\frac{r_j}{x'}\right)^2 - 1}{4\left(\frac{r_j}{y'}\right)^2 - 1} \cdot \frac{1}{T} \right)$$

where $x' := \textsc{Project}(x + \Delta, \ell_j, r_j)$ and $y' := \textsc{Project}(y - \Delta/2, \ell_j, r_j)$. See Remark 4 below for a discussion about Δ factors.

(4) Terminal density condition. For any y such that $a_{y,J}$ is defined, $b_{y,J+1} \leq a_{y,J}$.

Thus, our main goal is to show that



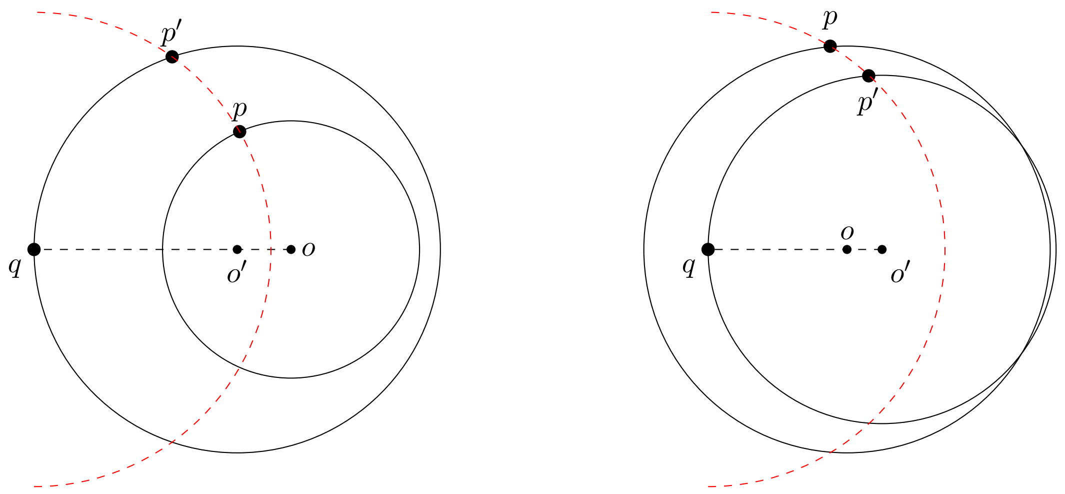
(a) When the query is outside of the sphere. (b) When the query is inside the sphere

Figure 2: Converting a (non-zero-distance) sphere to its corresponding zero-distance sphere

For every valid execution path (L, R, A, B) one has $\sum_y b_{y,J+1} = n^{o(1)}$.

The main challenge here is optimizing over sequences $(\ell_j, r_j)_{j=1}^J$ (distance to center of the sphere from $\mathbf{q}$ and the radius of the sphere). We perform this optimization in two steps, which we describe below.

Step 1. Suppose that there are two spheres such that the distance from the query to the orthogonal points for these spheres are the same. Also, assume that for the first sphere the query is not on the sphere, but for the second sphere the query is on the sphere (see Figure 2). Now, let $\mathbf{p}$ and $\mathbf{p}'$ lie on the first and the second spheres, respectively. Moreover, assume that they have the same distance from the query (see Figure 3). Comparing the spherical LSH effect on these two spheres, we prove that $\mathbf{p}$ is removed with higher probability compared to $\mathbf{p}'$ (see Claim 48). So, when the query is on the sphere, the densities are shrinking with a lower rate.



(a) When the query is outside of the sphere (b) When the query is inside the sphere

Figure 3: Mapping points from a sphere to its corresponding zero-distance sphere.

Step 2. Now consider two spheres with different radii, and assume that query $\mathbf{q}$ lies on them at the same time. Due to less curvature on the larger sphere, after one round of spherical LSH on these two spheres, the densities are shrinking with a lower rate on this sphere compared to the smaller sphere (see Claim 49).

The following definition enables us to state our claims more efficiently.

Definition 38 *(Zero-distance and monotone path) Let (L, R, A, B) be an execution path defined in Definition 36. If for $R = (r_j)_{j=1}^J$, r_j's are non-increasing in j, and $L = R$, then we say that (L, R, A, B) is a* zero-distance and monotone *execution path. When $L = R$, we usually drop L, and simply write (R, A, B).*

Now, using the two steps above, we can argue that for any valid execution path, we can find a zero-distance monotone execution path, with the same terminal densities and the same length (see Lemma 39 restated below for the convenience of the reader).

Lemma 39 *(Zero-distance and monotone path) For every valid execution path (L, R, A, B) (see Definition 36), there exists a* zero-distance and monotone *valid execution path (R', A', B') (see Definition 38) such that $b'_{y,J+1} = b_{y,J+1}$ for all $y \in D$[6] and $|R'| = |R|$ (i.e., the length of the paths are equal).*

The proof of the lemma (the formal version of the two steps mentioned above) is given in Section 7.

As mentioned before, we analyze the evolution of density of points in various distances. First, we define a grid of distances around query, which we use to properly round the distances of real spheres in the execution of algorithm. Second, instead of analyzing continuous densities, we define a new notion, called *discretized log-densities* (see Definition 41 below), for which we round densities to the discretized distances in a natural way, and for simplicity of calculations we take the log of these densities.

Definition 40 *(x-centered grid Z_x; restated) For every $x \in (0, R_{max})$ define the grid $Z_x = \{z_I, z_{I-1}, \ldots, z_0\}$ by letting $z_I = x$, letting $z_{I-i} := (1+\delta_z)^i \cdot z_I$ for all $i \in [I]$ and choosing the smallest integer I such that $z_0 \geq R_{max}\sqrt{2}$.*

Definition 41 *(Discretized log-densities $f_{z_i,j}$; restated) For any zero-distance monotone valid execution path (R, A, B) (as per Definition 36) with radii bounded by R_{max} and $J = |R|$, for all $j \in [J]$ let k_j be the index of the largest grid element which is not bigger than $r_j \cdot (\sqrt{2} + \psi)$, i.e.,*

$$r_j \cdot (\sqrt{2} + \psi) \in [z_{k_j}, z_{k_j - 1}) \tag{12}$$

and for every integer $i \in \{k_j, \ldots, I\}$ define

$$f_{z_i,j} := \log_{1/\mu}\left(\sum_{y \in D \cap [z_{i+1}, z_{i-1})} b_{y,j}\right) \tag{13}$$

Note that the variables $b_{y,j}$ on the right hand side of (13) are the $b_{y,j}$ variables of the execution path (R, A, B).

These two steps, allow us to analyze the evolution of densities over the course of time. In this section, we present an LP (see (33)) that its optimal cost bounds the query time of our algorithm. The main idea behind the linear program is to relax the notion of a zero-distance monotone path, which may involve only a small number of decreasing sphere radii, to a process that uses a grid $Z = Z_x$ of decreasing radii and possibly applies locality sensitive hashing at every such point (see the spherical LSH constraint in (14) below), and applies pseudorandmification, i.e. ensures that the dataset is dominated by points at distance $z_j \approx \sqrt{2} r_j$ from the query (see the truncation constraints

[6] We need the final condition to argue that we have the same number of points remaining at the end.

in (14) below). We note that the grid Z_x represents distances to points on the j-the sphere that are nearly orthogonal to the query, i.e. whose at distance $\approx \sqrt{2}r_j$, as opposed to the radii themselves. It is also important to note that the linear program is parameterized by two quantities: the target distance $x \in [0, \sqrt{2}]$ and a parameter j^* that indexes a point z_{j^*} in the grid Z_x. The quantity z_{j^*} should be thought of as the distance scale that contributes the most to query time, i.e. the band that the has the most number of points in the final densities (see non-empty range constraints in the linear program (14), as well as the similar calculation (4) in Section 2.1). To obtain our final bound on the query time, we enumerate over all x and $j^* \in Z_x$, upper bound the value of the corresponding $\mathrm{LP}(x, j^*)$ and take the maximum. Finally, we note that the intended LP solution is as follows. Consider a root to leaf path in the tree $\mathcal{T}$ constructed by $\textsc{PreProcess}(\widetilde{P}, x, \mu)$ that an invocation of $\textsc{Query}(\mathbf{q}, \mathcal{T}, x)$, and suppose that the sequence of radii of spheres traversed by $\textsc{Query}$ is exactly Z_x. Then letting α_j denote the number of LSH nodes that correspond to sphere with radius $r_j = z_j/\sqrt{2}$, divided by T, should intuitively give a feasible solution[7].

In Section 8 we will show in details why this LP formulation is enough to analyze the query time. Informally, this LP considers all possible root to leaf paths, and applies corresponding *truncation* and *spherical LSH* functions on the density and its cost is related to the length of root to leaf paths. We show in Section 8 that any execution path with large enough final densities gives a feasible solution to the linear program whose cost is (almost) equal to the length of the path divided by T. Thus, if we take any path with length more than $T \cdot \mathrm{OPT(LP)}$, the final densities are small.

Letting $Z := Z_x$ to simplify notation, we will consider I linear programs defined below in (33), enumerating over all $j^* \in [I]$, where we let $x' = x + \Delta$:

$$
\begin{aligned}
\mathrm{LP}(x, j^*): \quad & \max_{\alpha \geq 0} \sum_{j=1}^{j^*-1} \alpha_j && && (14)\\
& \forall y \in Z : g_{y,1} \leq \min\left\{\frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\} && \text{Density constraints}\\
& \text{for all } j < j^*, y \in Z, y < z_j : && \\
& \qquad g_{y,j} \leq g_{z_j,j} && \text{Truncation}\\
& \qquad g_{y,j+1} \leq g_{y,j} - \frac{2\left(z_j/x\right)^2 - 1}{2\left(z_j/y\right)^2 - 1} \cdot \alpha_j && \text{Spherical LSH}\\
& \qquad g_{z_{j^*},j^*} \geq 0 && \text{Non-empty range constraint}
\end{aligned}
$$

The following claim is the main technical claim relating zero-distance monotone execution paths and the linear program (14):

Claim 55 *(Feasible LP solution from an execution path; Restated) If integer J is such that $J > \frac{T}{1-10^{-4}}\mathrm{OPT(LP)}$ then, for all $y \leq z_{j^*-1}$, $f_{y,J+1} < 7\delta_z$ for $j^* = k_J + 1$ (see Definition 41 for the definition of k_J).*

The proof of Claim 55 is somewhat delicate, and exploits specific properties of the (negative) log-density of the Gaussian kernel. In fact, one can construct rather simple kernels with non-decreasing log-density for which Claim 55 is false – we give an example in Figure 4a. Informally, we

[7]This statement is somewhat imprecise, and in fact is quite nontrivial to make fully formal – this is exactly what our algorithm achieves by introducing the notion of valid execution paths.

call a kernel well-behaved, if the log-densities after applying a few rounds of LSH (and corresponding truncations), are increasing up to some point and then they are decreasing. More formal description is given after the following paragraph.

Intuitively, the reason is the difference between how the LP works and how the algorithm works. In the algorithm if we are running LSH on some sphere z we apply truncations based on distance z after each round of LSH (except the last step, for the intuition we can ignore this fact) and when we move to the next sphere z', the algorithm applies truncation to log-densities with respect to log-density at z'. However, the LP applies all the LSH rounds at once and then does truncation with respect to all bands from z to z'. Now, if some kernel is not well-behaved, say like the kernel depicted in Figure 4a then when the LP wants to move from z to z' it also truncates the log-densities with respect to the log-density at any $\eta \in (z', z)$. Then, for some η as shown in Figure 4a the log-density at some $\eta \in (z', z)$ is lower compared to the density at z and z'. Thus the log-densities in the LP shrink faster than the algorithm, which makes this approach not applicable to these set of kernels. However, for instance in the case of Gaussian kernel, the truncation with respect to log-densities at $\eta \in (z', z)$, do not impose a problem since the log-density at any $\eta \in (z', z)$ is larger than the minimum of densities at z and z'. This informally suggests that the evolution of the LP, can be seen as evolution of log-densities for well-behaved kernels, and thus can be used to analyze the run-time of the algorithm.

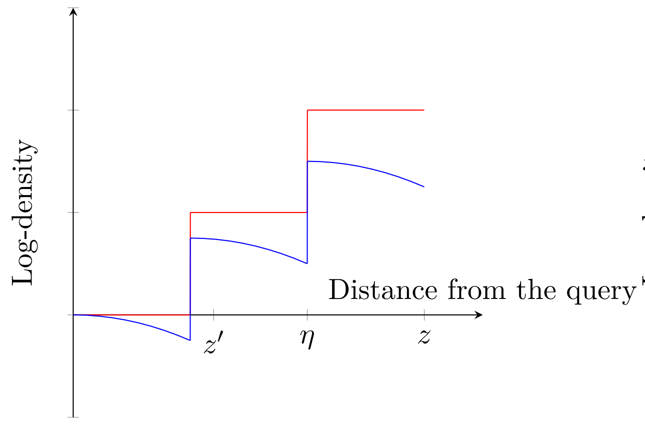
Now, we present a relatively more formally definition of well-behaved kernels. We say that a kernel $k(\mathbf{p}, \mathbf{q}) = \exp(-h(||\mathbf{p} - \mathbf{q}||_2))$ with the input space scaled so that $\exp(-h(\sqrt{2})) = \mu$ is well-behaved if for every integer $t \geq 1$, $x \in (0, \sqrt{2})$ and any sequence $c_1 \geq c_2 \geq \ldots \geq c_t \geq x$, such that

$$f(y) = \frac{y^2 - x^2}{2} - \sum_{s=1}^{t} \frac{2(c_s/x)^2 - 1}{2(c_s/y)^2 - 1} \cdot \frac{1}{T}$$

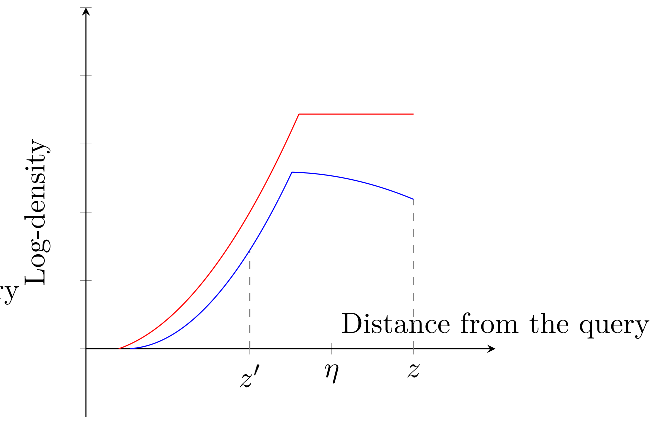
satisfies $f(\sqrt{2}c_t) > 0$, the following conditions hold. There exists $y^* \in (x, \sqrt{2}c_t]$ such that the function satisfies $f(y^*) = 0$ is monotone increasing on the interval $[y^*, \eta]$, where η is where the (unique) maximum of f on $(y^*, \sqrt{2}c_t]$ happens. See Fig. 4b for an illustration. Intuitively, a log-density h is well-behaved if the result of applying any amount of LSH on any collection of spheres to h results in a function with at most one maximum. This lets us control the structure of log-densities that arise after several iterations of LSH and truncation primitives in a valid execution path (and thus in a root to leaf path in $\mathcal{T}$ that a query $\mathbf{q}$ traverses).

We show in Section 8 (see Claim 53) that the Gaussian kernel is well behaved, and use this fact that prove Claim 55. We also show a similar claim for the class of kernels whose negative log density is concave (the exponential kernel is one example). This lets us extend our result to kernels beyond Gaussian (see Remark 3 in Section 5).

On the other hand, we show numerically that the solution of the LP in (14) is upper bounded



(a) A kernel, which is not well-behaved



(b) A well-behaved kernel

Figure 4: In both figures, the red curve and the blue curve represents the densities before and after running LSH rounds on sphere z, respectively. In the case of well-behaved kernels the density at any $\eta \in (z', z)$ is lower-bounded by the minimum of densities at z and z'. However, for a kernel which is not well-behaved, for instance for the η shown in the left figure, the density is lower than the density at z and z'.

by 0.1718 for the Gaussian kernel. This is done in Section 9 by formulating the dual LP

$$\min \sum_{y \in Z} \left\{ \frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2} \right\} r_{y,0} \tag{15}$$

$$\begin{aligned}
&\text{such that :} && \\
\forall j \in [j^*-1], y \in Z, y < z_j &: r_{y,j-1} - r_{y,j} + q_{y,j} = 0 && (g_{y,j}) \quad \text{Mass transportation} \\
\forall j \in [j^*-1] &: r_{z_j,j-1} - \sum_{x \in Z, x < z_j} q_{x,j} = 0 && (g_{z_j,j}) \quad \text{Max tracking} \\
\forall y \in Z, y < z_{j^*} &: r_{y,j^*-1} = 0 && (g_{y,j^*}) \quad \text{Sink} \\
&\quad -\eta + r_{z_{j^*},j^*-1} = 0 && (g_{z_{j^*},j^*}) \quad \text{Terminal flow} \\
j \in [j^*-1] &: \sum_{y \in Z : y < z_j} \frac{2\left(z_j/x\right)^2 - 1}{2\left(z_j/y\right)^2 - 1} r_{y,j} \geq 1 && (\alpha_j) \\
&\quad r_{y,j}, q_{y,j} \geq 0 && \\
&\quad \eta \geq 0 &&
\end{aligned}$$

and exhibiting a dual feasible solution of value ≈ 0.1716 for a fine grid of points Z_x and every x in a fine grid over $[0, \sqrt{2}]$. We also give an analytic upper bound of $\frac{x^2}{2}(1 - \frac{x^2}{2}) + 0.001$ on the value of the LP (14).

3 Preliminaries

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

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

3.1 Basic notation

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

For any integer J we define $[J] := \{1, 2, \ldots, J\}$. For ease of notation in the rest of the paper, we let $\exp_\mu(a) := \left(\frac{1}{\mu}\right)^a$ and (abusing notation somewhat) let $\exp_2(a) = 2^a$ for any $a \in \mathbb{R}$.

3.2 $F(\eta)$ and $G(s, \eta, \sigma)$

In this section, we define notations and present results, which we later use to analyze the collision probability of spherical-LSH.

Lemma 8 (Lemma 3.1, [ALRW17])**.** *If for any $u \in \mathcal{S}^{d-1}$ we define*

$$F(\eta) := \Pr_{z \sim N(0,1)^d} \left[\langle z, u \rangle \geq \eta\right],$$

then, for $\eta \to \infty$

$$F(\eta) = e^{-(1+o(1)) \cdot \frac{\eta^2}{2}}.$$

Lemma 9 (Lemma 3.2, [ALRW17])**.** *If for any $u, v \in \mathcal{S}^{d-1}$ such that $s := ||u - v||$, we define*

$$G(s, \eta, \sigma) := \Pr_{z \sim N(0,1)^d} \left[\langle z, u \rangle \geq \eta \text{ and } \langle z, v \rangle \geq \sigma\right],$$

then if $\sigma, \eta \to \infty$, and $\frac{\max\{\sigma,\eta\}}{\min\{\sigma,\eta\}} \geq \alpha(s)$, then one has

$$G(s, \eta, \sigma) = e^{-(1+o(1)) \cdot \frac{\eta^2 + \sigma^2 - 2\alpha(s)\eta\sigma}{2\beta^2(s)}},$$

where $\alpha(s) := 1 - \frac{s^2}{2}$ and $\beta(s) := \sqrt{1 - \alpha^2(s)}$.

Definition 10. For ease of notation we also define

$$G(s, \eta) := G(s, \eta, \eta).$$

3.3 Projection

Definition 11. Let $\mathbf{q}$ be a point on $\mathcal{S}^{d-1}(o, R_1)$ and $\mathbf{p}$ be a point on $\mathcal{S}^{d-1}(o, R_2)$, such that $y := ||q - p||$. Now, if we define $\mathbf{q}'$ as the projection of $\mathbf{q}$ on $\mathcal{S}^{d-1}(o, R_2)$. Then, we define the following

$$\textsc{Project}(y, R_1, R_2) := ||\mathbf{q}' - \mathbf{p}||.$$

Lemma 12. *For any $R_1, R_2 \in \mathbb{R}_+$ and $o \in \mathbb{R}^d$ assume that we have points $\mathbf{q}, \mathbf{p}$ on spheres $\mathcal{S}_1 := \mathcal{S}^{d-1}(o, R_1)$ and $\mathcal{S}_2 := \mathcal{S}^{d-1}(o, R_2)$, respectively. Also, let $x := ||\mathbf{p} - \mathbf{q}||$ and let $\mathbf{q}'$ be the projection of point $\mathbf{q}$ on $\mathcal{S}_2$. Then we have the following*

$$\textsc{Project}(x, R_1, R_2) = ||\mathbf{q}' - \mathbf{p}|| = \sqrt{\frac{R_2}{R_1}\left(x^2 - (R_2 - R_1)^2\right)}.$$

The proof is deferred to Appendix A.

3.4 Pseudo-Random Spheres

Definition 13. (Pseudo-random spheres) Let P be a set of points lying on $\mathcal{S}^{d-1}(o, r)$ for some $o \in \mathbb{R}^d$ and $r \in \mathbb{R}_+$. We call this sphere a *pseudo-random* sphere[8], if $\nexists \mathbf{u}^* \in \mathcal{S}^{d-1}(o, r)$ such that

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{u}^*|| \le r(\sqrt{2} - \gamma)\right\}\right| \ge \tau \cdot |P|.$$

As shown in [?, Section 6], it is possible to decompose a dataset P into pseudo-random components in time poly $(d, \gamma^{-1}, \tau^{-1}, \log |P|) \cdot |P|$. We present a slight modification of their argument using our notation in Appendix B. The following claim summarizes the properties of a pseudo-random sphere that we use later:

Claim 14. *If P is a set of points lying on $\mathcal{S}^{d-1}(o, r)$ for some $o \in \mathbb{R}^d$ and $r \in \mathbb{R}_+$, and P is a pseudo-random sphere (see Definition 13) then for any point $\mathbf{q}'$ on the sphere we have the following property*

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}'|| \le r(\sqrt{2} - \gamma)\right\}\right| \le \frac{\tau}{1 - 2\tau} \cdot \left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}'|| \in \left(r\left(\sqrt{2} - \gamma\right), r(\sqrt{2} + \gamma)\right)\right\}\right|,$$

and consequently,

$$\left|\left\{\mathbf{w} \in P : ||\mathbf{w} - \mathbf{q}'|| \in \left(r\left(\sqrt{2} - \gamma\right), r\left(\sqrt{2} + \gamma\right)\right)\right\}\right| = \Omega(|P|).$$

The proof is deferred to Appendix A.

4 Kernel Density Estimation Using Andoni-Indyk LSH

In this section, we present an algorithm for estimating KDE, using the Andoni-Indyk LSH framework. In order to state the main result of this section for general kernels, we need to define a few notions first. Thus, we state the main result for Gaussian kernel in the following theorem, and then state the general result, Theorem 22, after presenting the necessary definitions.

[8]Whenever we say *pseudo-random sphere*, we implicitly associate it with parameter τ, γ which are fixed throughout the paper.

Theorem 15. *Given a kernel $K(\mathbf{p},\mathbf{q}) := e^{-a\|\mathbf{p}-\mathbf{q}\|_2^2}$ for any $a>0$, $\epsilon = \Omega\left(\frac{1}{\text{polylogn}}\right)$, $\mu^* = n^{-\Theta(1)}$ and a data set of points P, using Algorithm 1 for preprocessing and Algorithm 2 for the query procedure, one can approximate $\mu^* := K(P,\mathbf{q})$ (see Definition 17) up to $(1\pm\epsilon)$ multiplicative factor, in time $\widetilde{O}\left(\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}\right)$, for any query point $\mathbf{q}$. Additionally, the space consumption of the data structure is*

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

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

Lemma 16 ([AI06])**.** *Let $\mathbf{p}$ and $\mathbf{q}$ be any pair of points in $\mathbb{R}^d$. Then, for any fixed $r>0$, there exists a hash family $\mathcal{H}$ such that, if $p_{\text{near}} := p_1(r) := \Pr_{h\sim\mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q}) \mid \|\mathbf{p}-\mathbf{q}\| \le r]$ and $p_{\text{far}} := p_2(r,c) := \Pr_{h\sim\mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q}) \mid \|\mathbf{p}-\mathbf{q}\| \ge cr]$ for any $c \ge 1$, then*

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

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

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

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

Definition 17. For a query $\mathbf{q}$, and dataset $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$, we define

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

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

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

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

$$\mu^* \le \mu \tag{16}$$

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

Definition 18 (Geometric weight levels)**.** For any $j \in [J]$

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

This implies corresponding distance levels (see Figure 1 and Figure 5), which we define as follows

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

where $f(r) := K(\mathbf{p},\mathbf{p}')$ for $r = \|\mathbf{p}-\mathbf{p}'\|$. Also define $L_{J+1} := P \setminus \cup_{j\in[J]} L_j$.[9]

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

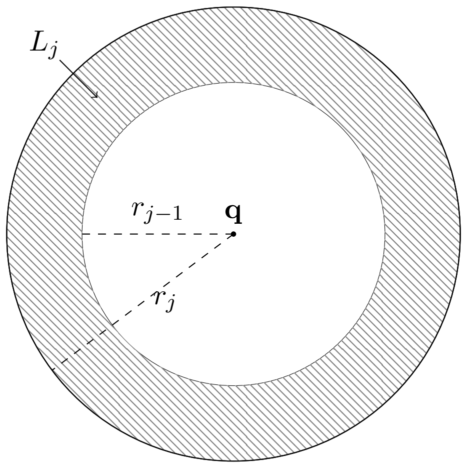


Figure 5: Illustration of definition of r_j's based on L_j's.

We start by stating basic bounds on collision probabilities under the Andoni-Indyk LSH functions in terms of the definition of geometric weight levels L_j (Definition 18):

Claim 19. *Assume that kernel K induces weight level sets, L_j's, and corresponding distance levels, r_j's (as per Definition 18). Also, for any query $\mathbf{q}$, any integers $i \in [J+1], j \in [J]$ such that $i > j$, let $\mathbf{p} \in L_j$ and $\mathbf{p}' \in L_i$. And assume that $\mathcal{H}$ is an Andoni-Indyk LSH family designed for near distance r_j (see Lemma 16). Then, for any integer $k \geq 1$, we have the following conditions:*

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

where $c := c_{i,j} := \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\}$ (see Remark 1) and $p_{\text{near},j} := p_1(r_j)$ in Lemma 16.

Proof. If $\mathbf{p} \in L_j$ by Definition 18, we have

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

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

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

So, by Lemma 16 and Remark 1 the claim holds. Figure 6 shows an instance of this claim. □

Now, we prove an upper-bound on sizes of the geometric weight levels, i.e., L_j's (see Definition 18).

Lemma 20 (Upper bounds on sizes of geometric weight levels)**.** *For any $j \in [J]$, we have*

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

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

$$\begin{aligned} n\mu \geq n\mu^* &= \sum_{\mathbf{p} \in P} K(\mathbf{p}, \mathbf{q}) && \text{By Definition 17} \\ &\geq \sum_{i \in [J]} \sum_{\mathbf{p} \in L_i} K(\mathbf{p}, \mathbf{q}) \\ &\geq \sum_{\mathbf{p} \in L_j} K(\mathbf{p}, \mathbf{q}) \\ &\geq |L_j| \cdot 2^{-j} \end{aligned}$$

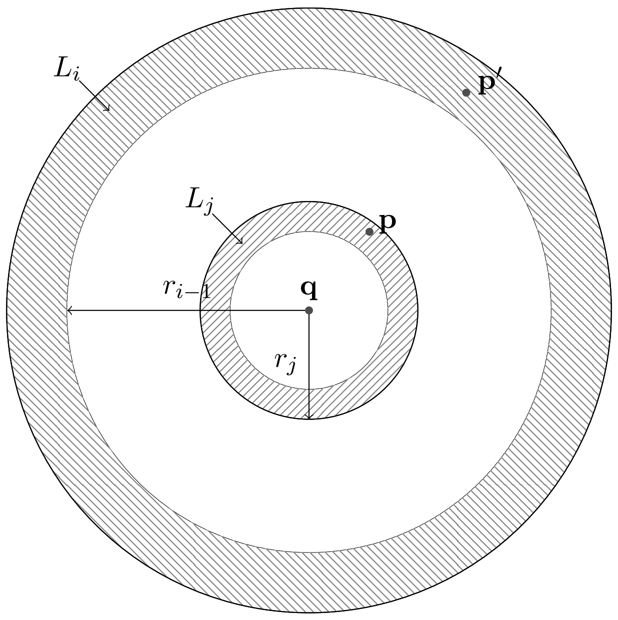


Figure 6: Illustration of r_j and r_{i-1} in terms of L_j and L_i.

which proves the claim. □

Definition 21 (Cost of a kernel)**.** Suppose that a kernel K induces geometric weight levels, L_j's, and corresponding distance levels, r_j's (see Definition 18). For any $j \in [J]$ we define *cost* of kernel K for weight level L_j as

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

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

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

Description of algorithm: The algorithm runs in J phases. For any $j \in [J]$, in the j'th phase, we want to estimate the contribution of points in L_j to $K(P, \mathbf{q})$. We show that it suffices to have an estimation of the number of points in L_j. One can see that if we sub-sample the data set with probability $\min\{\frac{1}{2^j n\mu}, 1\}$, then in expectation we get at most $O(1)$ points from L_i for any $i \le j$. Now, assume that a point $\mathbf{p} \in L_j$ gets sampled by sub-sampling, then we want to use Andoni-Indyk LSH to distinguish this point from other sub-sampled points, efficiently. Thus, we want to find the appropriate choice of k for the repetitions of Andoni-Indyk LSH (see Claim 19). Suppose that we call Claim 19 with some k (which we calculate later in (18)). Then we have

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

which implies that in order to recover point $\mathbf{p}$ with high probability, we need to repeat the procedure $\widetilde{O}\left(p_{\mathrm{near},j}^{-k}\right)$ times. Another factor that affects the run-time of the algorithm is the number of points that we need to check in order to find $\mathbf{p}$. Basically, we need to calculate the number of points that hash to the same bucket as $\mathbf{q}$ under h^*'s. For this purpose, we use the second part of Claim 19, which bounds the collision probability of far points, i.e., points such as $\mathbf{p}' \in L_i$ for any $i > j$. Intuitively, for any point $\mathbf{p}' \in L_i$ for any $i > j$, by Claim 19 we have

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

where $c := c_{i,j} := \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\}$ and $p := p_{\text{near},j}$[10]. On the other hand, by Lemma 20, for $i = j+1, \dots, J$ we have

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

Then, one has the following bound,

$$\begin{aligned}
&\mathbb{E}\left[\left|\{\mathbf{p}' \in L_i : h^*(\mathbf{p}') = h^*(\mathbf{q})\}\right|\right] \\
&\leq 2^i n\mu \cdot \frac{1}{2^j n\mu} \cdot p^{kc^2(1-o(1))} && \text{Sub-sampling and then applying LSH} \\
&= 2^{i-j} \cdot p^{kc^2(1-o(1))}. && (17)
\end{aligned}$$

Since we have $O\left(\log\frac{1}{\mu}\right)$ geometric weight levels, then the expression in (17) for the worst i, bounds the run-time up to $O\left(\log\frac{1}{\mu}\right)$ multiplicative factor. In order to optimize the run-time up to $\widetilde{O}(1)$ multiplicative factors, we need to set k such that the expression in (17) gets upper-bounded by $O(1)$ for all $i > j$. So, in summary, for any fixed $j \in [J]$, we choose k such that any weight level L_i for $i \geq j$ contributes at most $\widetilde{O}(1)$ points in expectation to the hash bucket of the query, i.e., $h^*(\mathbf{q})$. One can see that we can choose k as follows

$$k := k_j := \frac{-1}{\log p} \cdot \max_{i=j+1,\dots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil. \tag{18}$$

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

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

Algorithm 1 Preprocessing

```
1: procedure PREPROCESS($P, \epsilon$)
2:                                                  ▷ $P$ represents the set of data points
3:                                                  ▷ $\epsilon$ represents the precision of estimation
4:     $K_1 \leftarrow \frac{C \log n}{\epsilon^2} \cdot \mu^{-o(1)}$          ▷ $C$ is a universal constant
5:     $J \leftarrow \lceil \log \frac{1}{\mu} \rceil$          ▷ We use geometric weight levels with base 2, see Definition 18
6:     for $a = 1, 2, \ldots, K_1$ do                ▷ $O(\log n/\epsilon^2)$ independent repetitions
7:         for $j = 1, 2, \ldots, J$ do              ▷ $J = \lceil \log \frac{1}{\mu} \rceil$ geometric weight levels
8:             $K_2 \leftarrow 100 \log n \cdot p_{\text{near},j}^{-k_j}$
9:                                 ▷ See Claim 19 and (18) for definition of $p_{\text{near},j}$ and $k_j$
10:            $p_{\text{sampling}} \leftarrow \min\{\frac{1}{2^j n\mu}, 1\}$
11:            $\widetilde{P} \leftarrow$ sample each element in $P$ with probability $p_{\text{sampling}}$.
12:            for $\ell = 1, 2, \ldots, K_2$ do
13:                Draw a hash function from hash family $\mathcal{H}^{k_j}$ as per Claim 19 and call it $H_{a,j,\ell}$
14:                Run $H_{a,j,\ell}$ on $\widetilde{P}$ and store non-empty buckets
15:        $\widetilde{P}_a \leftarrow$ sample each element in $P$ with probability $\frac{1}{n}$
16:        Store $\widetilde{P}_a$                   ▷ Set $\widetilde{P}_a$ will be used to recover points beyond $L_{J+1}$
```

Algorithm 2 Query procedure

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

Now, we present the main result of this section.

Theorem 22 (Query time)**.** *For any kernel K, the expected query-time of the algorithm is equal to $\widetilde{O}\left(\epsilon^{-2} n^{o(1)} \cdot \text{cost}(K)\right)$.*

Assuming Theorem 22, we prove Theorem 15.

Proof of Theorem 15: We first start by proving the query time bound and then we prove the space consumption of the data structure, and the guarantee over the precision of the estimator is given in Claim 25.

Proof of the query time bound: We calculate the cost of Gaussian kernel $e^{-a||\mathbf{x}-\mathbf{y}||_2^2}$. First, we present the weight levels and distance levels induced by this kernel. As per Definition 17, let

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

By Definition 18, one has

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

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

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

At this point, one can check that

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

Therefore, the cost of Gaussian kernel is

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

Now, invoking Theorem 22, the statement of the claim about the query time holds.

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

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

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

by (18), where $p = p_{\text{near},j}$, we have

$$p_{\text{near},j}^{-k_j} = \exp_2\left(\max_{i=j+1,\ldots,J+1}\left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))}\right\rceil - j\right). \tag{19}$$

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

$$\frac{1}{\mu}\exp_2\left(\max_{i=j+1,\ldots,J+1}\left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))}\right\rceil - j\right). \tag{20}$$

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

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

and for the other values of i we have $\max_{i=j+1,\ldots,J+1}\left\lceil\frac{i-j}{\log^{1/7} n(1-o(1))}\right\rceil - j \leq o(J)$ as well. Putting this together with (20) and multiplying by $J = O(\log(1/\mu)) = \mu^{-o(1)}$ to account for the number of choices $j \in [J]$, we get the second bound for the expected size of the data structure $\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{1+o(1)}$.

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

Claim 23 (Lower bound on probability of recovering a sampled point)**.** *Suppose that we invoke Algorithm 1 with (P, ϵ). Suppose that in line 11 of Algorithm 1, when $k = k^*$ and $j = j^*$, we sample some point $\mathbf{p} \in L_{j^*}$. We claim that with probability at least $1 - \frac{1}{n^{10}}$, there exists $\ell^* \in [K_2]$ such that $H_{k^*,j^*,\ell^*}(\mathbf{p}) = H_{k^*,j^*,\ell^*}(\mathbf{q})$.*

Proof. By Claim 19 we have

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

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

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

Claim 24 (Unbiasedness of the estimator)**.** *For every $\mu^* \in (0,1)$, every $\mu \geq \mu^*$, every $\epsilon \in (\mu^{10}, 1)$, every $\mathbf{q} \in \mathbb{R}^d$, estimator Z_a for any $a \in [K_1]$ constructed in* Query$(P, \mathbf{q}, \epsilon, \mu)$ *(Algorithm 2) satisfies the following:*

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

Proof. Let $\mathcal{E}$ be the event that every sampled point is recovered and let $Z := Z_a$ (see line 19 in Algorithm 2). By Claim 23 and union bound, we have

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

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

$$(1-n^{-9})n\mu^* \le \mathbb{E}[Z] \le n\mu^*. \tag{21}$$

□

Remark 3. We proved that our estimator is unbiased[11] for **any choice** of $\mu \ge \mu^*$. Therefore if $\mu \ge 4\mu^*$, by Markov's inequality the estimator outputs a value larger than μ at most with probability 1/4. We perform $O(\log n)$ independent estimates, and conclude that μ is higher than μ^* if the median of the estimated values is below μ. This estimate is correct with high probability, which suffices to ensure that we find a value of μ that satisfies $\mu/4 < \mu^* \le \mu$ with high probability by starting with some $\mu = n^{-\Theta(1)}$ (since our analysis assumes $\mu^* = n^{-\Theta(1)}$) and repeatedly halving our estimate (the number of times that we need to halve the estimate is $O(\log n)$ assuming that μ is lower bounded by a polynomial in n, an assumption that we make).

Claim 25 (Variance bounds)**.** *For every $\mu^* \in (0,1)$, every $\epsilon \in (\mu^{10}, 1)$, every $\mathbf{q} \in \mathbb{R}^d$, using estimators Z_a, for $a \in [K_1]$ constructed in* QUERY$(P, \mathbf{q}, \epsilon, \mu)$ *(Algorithm 2), where $\mu/4 \le \mu^* \le \mu$, one can output a $(1 \pm \epsilon)$-factor approximation to μ^*.*

Proof. By Claim 24 and noting that $Z \le n^2\mu^*$, where the worst case (equality) happens when all the points are sampled and all of them are recovered in the phase of their weight levels. Therefore,

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

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

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

Then, we have

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

and

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

[11] Up to some small inverse polynomial error.

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

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

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

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

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

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

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

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

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

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

Now, one can check that

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

Thus, we have

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

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

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

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

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

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

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

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

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

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

Proof of Theorem 22: One should note that the query time of our approach depends on the number of times that we hash the query and the number of points that we check, i.e., the number of points that collide with the query. First, we analyze the number of points colliding with the query. We Fix $j \in [J]$, so, we want to estimate the contribution of points in L_i to $K(P, \mathbf{q})$. We consider 3 cases:

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

Case 2. $i = j+1, \dots, J$**:** Again, note that by Lemma 20, $|L_i| \leq 2^i n\mu$, and the sampling rate is $\min\{\frac{1}{2^j n\mu}, 1\}$. Thus, we have at most 2^{i-j} sampled points from L_i in expectation. Now, we need to analyze the effect of LSH. Note that we choose LSH function such that the near distance is r_j (see Claim 19). Also, note that as per (18), we use

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

as the number of concatenations. Now, we have the following collision probability for $\mathbf{p} \in L_i$ using Claim 19

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

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

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

by the choice of k.

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

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

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

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

by the choice of k_j.

All in all, we prove that each weight level L_i for $i \in [J+1]$ contribute at most $\widetilde{O}(1)$ points to the hash bucket of query. Now, we need to prove a bound on the number of times we evaluate our hash function. One should note that by the choice of k_j in (18) we have

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

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

5 Improved algorithm via data dependent LSH

In this section, we improve the algorithm presented in the previous section using data dependent LSH approach for the Gaussian kernel. Consider a data set $P \subset \mathbb{R}^d$, a positive real number a, and a query $\mathbf{q} \in \mathbb{R}^d$. Let

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

denote the KDE value at the query $\mathbf{q} \in \mathbb{R}^d$ of interest, and for the rest of the paper suppose that the algorithm is given a parameter μ that satisfies the following property

$$\mu^* \le \mu. \tag{22}$$

We prove the following main result in the rest of the paper.

Theorem 26. *Given a kernel $K(\mathbf{p}, \mathbf{q}) := e^{-a||\mathbf{p}-\mathbf{q}||_2^2}$ for any $a > 0$, $\epsilon = \Omega\left(\frac{1}{\text{polylog}n}\right)$, $\mu^* = n^{-\Theta(1)}$ and a data set of points P, there exists a preprocessing algorithm and a corresponding query algorithm that one can approximate $\mu^* := K(P, \mathbf{q})$ (see Definition 17) up to $(1 \pm \epsilon)$ multiplicative factor, in time $\widetilde{O}\left(\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.173+o(1)}\right)$, for any query point $\mathbf{q}$. Additionally, the space consumption of the data structure is*

$$\min\left\{\epsilon^{-2} n \left(\frac{1}{\mu^*}\right)^{0.173+o(1)}, \epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{1+c+o(1)}\right\}.$$

for a small constant $c = 10^{-3}$.

Proof. First, in Section 5.1 we present the main primitives in the preprocessing phase (Algorithms 3, 4 and 5) and prove the space bound in Lemma 30. The standard outer algorithm is presented in Appendix C for completeness. The main query primitive in query algorithm is presented in Section 5.3, and the query time is proved in Section 6 in Lemma 31. The correctness proof (precision of the estimator) is rather standard and similar to the correctness proof in Section 4 and is given in Appendix C for completeness. □

Remark 4. Although we present the analysis for the Gaussian kernel, our techniques can be used for other kernels such as the exponential kernel as well. We do not present the full analysis to simplify presentation of our main result for the Gaussian kernel, but provide proofs of key lemmas in Appendix F. Specifically, we present the equivalent of Claims 53 and 54, which underly our LP analysis, for kernels whose negative log density is concave (this includes the exponential kernel $\exp(-||x||_2)$). Our dual solution presented in Section 9 gives an upper bound of ≈ 0.1 on the value of the corresponding LP. Replacing the parameter α^* in the algorithms presented in this section with 0.1 thus yield an data structure for KDE with the exponential kernel with query time $\widetilde{O}\left(\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.1+o(1)}\right)$ and space consumption $\epsilon^{-2} n \left(\frac{1}{\mu^*}\right)^{0.1+o(1)}$ for the exponential kernel.

In order to simplify notation we apply the following normalization without loss of generality: For any point in $\mathbf{p} \in P \cup \{\mathbf{q}\}$, let $\mathbf{p}' := \sigma\mathbf{p}$ and $\sigma := \sqrt{\frac{2a}{\log(1/\mu)}}$ such that a point $\mathbf{p}'$ at distance $\sqrt{2}$ from the query $\mathbf{q}'$ contributes exactly μ to the kernel. In other words, we assume by convenient scaling that

$$K(P, \mathbf{q}) = \frac{1}{n}\sum_{\mathbf{p}'} (\mu)^{||\mathbf{p}'-\mathbf{q}'||_2^2/2}.$$

and to lighten notation we will assume that $\sigma = 1$, i.e. points are already properly scaled. For $x \in (0, \sqrt{2})$, let $\widetilde{P}$ be the dataset obtained from P by including every point independently with probability $\min\left\{\frac{1}{n} \cdot \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}}, 1\right\}$. We state these conditions in a compact way as follows and use them in the rest of the paper.

Assumption 1. We have the followings

- $P \subset \mathbb{R}^d$ and $|P| = n$.
- $\mathbf{q} \in \mathbb{R}^d$.
- $\mu^* := K(P, \mathbf{q})$
- $\frac{1}{\mu^*} = n^{\Omega(1)}$
- μ is such that $\mu^* \leq \mu$.
- $\mu = n^{-\Theta(1)}$.
- The points are scaled so that $K(\mathbf{p}, \mathbf{q}) = \mu^{\frac{||\mathbf{p}-\mathbf{q}||^2}{2}}$.
- $\widetilde{P}$ is obtained by independently sub-sampling elements of P with probability $\min\left\{\frac{1}{n} \cdot \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}}, 1\right\}$, for some $x \in (0, \sqrt{2})$, which is clear from the context.

In this section we design a data structure that allows preprocessing $\widetilde{P}$ as above using small space such that every point at distance at most x from any query $\mathbf{q}$ is recovered with probability at least 0.8 (see Lemma 60).

In what follows we present our preprocessing algorithm (Algorithm 3) in Section 5.1, the query algorithm (Algorithm 6) in Section 5.3 as well as proof of basic bounds on their performance in the same sections. Our main technical contribution is the proof of the query time bound. This proof relies on a novel linear programming formulation that lets us bound the evolution of the density of points around the query q as the query percolates does the tree $\mathcal{T}$ of hash buckets produced by PREPROCESS. This analysis is given in Section 6, with the main supporting technical claims presented in Section 8.

5.1 Preprocessing algorithm and its analysis

Our preprocessing algorithm is recursive. At the outer level, given the sampled dataset $\widetilde{P}$ as input, the algorithm hashes $\widetilde{P}$ into buckets using Andoni-Indyk Locality sensitive hashing. The goal of this is to ensure that with high probability all hash buckets that a given query explores are of bounded diameter, while at the same time ensuring that any close point $\mathbf{p}$ hashes together with $\mathbf{q}$ in at least one of the hash buckets with high constant probability. The corresponding analysis is presented in Sections 5.3 and 6.

Our main tool in partitioning the data set into (mostly) low diameter subsets is an Andoni-Indyk Locality Sensitive Hash family. Such a family is provided by Lemma 16, which was our main tool in obtaining the non-adaptive KDE primitives in Section 4, and Corollary 27 below. We restate the lemma below for convenience of the reader:

Lemma 16 *([AI06]) (Restated) Let* $\mathbf{p}$ *and* $\mathbf{q}$ *be any pair of points in* $\mathbb{R}^d$*. Then, for any fixed* $r > 0$*, there exists a hash family* $\mathcal{H}$ *such that, if* $p_{\text{near}} := p_1(r) := \Pr_{h\sim\mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q}) \mid ||\mathbf{p} - \mathbf{q}|| \le r]$ *and* $p_{\text{far}} := p_2(r, c) := \Pr_{h\sim\mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q}) \mid ||\mathbf{p} - \mathbf{q}|| \ge cr]$ *for any* $c > 1$*, then*

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

for some t*, where* $p_{\text{near}} \ge e^{-O(\sqrt{t})}$ *and each evaluation takes* $dt^{O(t)}$ *time.* One should also recall Remark 1, which ensures $n^{o(1)}$ evaluation time, with appropriate choice of t in Lemma 16.

Corollary 27. *Let* α *be a constant, and let* $x \in (0, \sqrt{2})$ *and* y *be such that* $y \ge x$*. Then, there exists a hash family* $\mathcal{H}$ *such that for any points* $\mathbf{q} \in \mathbb{R}^d$*,* $\mathbf{p}$ *and* $\mathbf{p}'$*, where* $||\mathbf{p} - \mathbf{q}|| \le x$ *and* $||\mathbf{p}' - \mathbf{q}|| \ge y$*, we have the following conditions*

- $\Pr_{h\sim\mathcal{H}}\left[h(\mathbf{q}) = h(\mathbf{p})\right] \ge \mu^{\alpha}$
- $\Pr_{h\sim\mathcal{H}}\left[h(\mathbf{q}) = h(\mathbf{p}')\right] \le \mu^{\alpha c^2(1-o(1))}$

where $c := \min\left\{\frac{y}{x}, \log^{1/7} n\right\}$*, and we call such a hash family a* (α, x, μ)*-AI hash family.*

Our preprocessing algorithm is given below. It simply hashes the dataset several times independently using an Andoni-Indyk LSH family and calls Spherical-LSH (Algorithm 4 below) on the buckets. The hashing is repeated several times to ensure that the query collides with any given close point with high probability in at least one of the hashings. Overall PreProcess simply reduces the the diameter of the dataset, whereas most of the work is done by Spherical-LSH, defined below.

Algorithm 3 PreProcess: $\widetilde{P}$ is the subsampled data-set, x is the target distance to recover

```
1: procedure PreProcess($\widetilde{P}, x, \mu$)
2:     Add a root $w_0$ to the recursion tree $\mathcal{T}$
3:     $w_0.P \leftarrow \widetilde{P}$, $w_0.level \leftarrow 0$, $w_0.g \leftarrow 0$        ▷ $g = 0$ since this node uses Euclidean LSH
4:     if $x > \sqrt{2}$ then return $\widetilde{P}$        ▷ In that case the expected size of $\widetilde{P}$ is small
5:     $\alpha \leftarrow 10^{-4}$        ▷ Choice of $\alpha$ affects hash bucket diameter, see Lemma 28
6:     $K_1 = 100\left(\frac{1}{\mu}\right)^{\alpha}$        ▷ The number of repetitions of the first round of hashing
7:     for $j = 1, 2, \ldots, \lceil K_1 \rceil$ do
8:         Pick $h_j$ from a $(\alpha, x, \mu)$-AI hash family, $\mathcal{H}$        ▷ See Definition 27
9:         $B \leftarrow$ set of non-empty hash buckets by hashing points in $P$ using $h_j$.
10:        for each $b \in B$ do
11:            Add a node $v$ as a child of $w_0$ in recursion tree $\mathcal{T}$
12:            $v.P \leftarrow b$, $v.level \leftarrow 0$, $v.g \leftarrow 0$
13:            $v.o \leftarrow$ any point in bucket $b$
14:            Spherical-LSH($v, x, \mu$)
15:    return $\mathcal{T}$
```

We will use the following basic upper bound on the Euclidean diameter of LSH buckets:

Lemma 28 (Diameter bound for Andoni-Indyk LSH buckets)**.** *Under Assumption 1, suppose that* $\mathcal{H}$ *is a* (α, x, μ)*-AI hash family (see Corollary 27), for some constant* α *and let* $c := \min\left\{\frac{R_{\text{diam}}}{x}, \log^{1/7} n\right\}$ *for some* $R_{\text{diam}} \ge \sqrt{2}$*, then if* $\alpha c^2 = 2 + \Omega(1)$ *then one has*

(a) $\mathbb{E}_{h\sim\mathcal{H},\widetilde{P}}\left[\left|\left\{\mathbf{p}\in\widetilde{P} : ||\mathbf{p}-\mathbf{q}|| \geq R_{\text{diam}} \textit{ and } h(\mathbf{p})=h(\mathbf{q})\right\}\right|\right] \leq \left(\frac{1}{\mu}\right)^{2-\alpha c^2(1-o(1))}$

(b) *and consequently*

$$\Pr_{h\sim\mathcal{H},\widetilde{P}}[\textit{diameter of } h^{-1}(\mathbf{q})\cap\widetilde{P} \textit{ is larger than } R_{\text{diam}}] \leq \left(\frac{1}{\mu}\right)^{2-\alpha c^2(1-o(1))}.$$

Proof. One has for every $R_{\text{diam}} \geq \sqrt{2}$

$$\mathbb{E}_{\widetilde{P}}\left[\left|\left\{\mathbf{p}\in\widetilde{P} : ||\mathbf{p}-\mathbf{q}|| \geq R_{\text{diam}}\right\}\right|\right] \leq \mathbb{E}_{\widetilde{P}}\left[\left|\widetilde{P}\right|\right] = n\cdot\frac{1}{n}\left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}} = \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}}$$

Taking the expectation with respect to the hash function h, we get,

$$\begin{aligned}\mathbb{E}_{h\sim\mathcal{H},\widetilde{P}}\left[\left|\left\{\mathbf{p}\in\widetilde{P} : ||\mathbf{p}-\mathbf{q}|| \geq R_{\text{diam}} \text{ and } h(\mathbf{p})=h(\mathbf{q})\right\}\right|\right] &\leq \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}-\alpha c^2(1-o(1))}\\ &\leq \left(\frac{1}{\mu}\right)^{2-\alpha c^2(1-o(1))},\end{aligned}$$

establishing **(a)**. Claim **(b)** now follows by applying Markov's inequality since $\alpha c^2 = 2+\Omega(1)$. □

Now, we establish a constant upper bound on the diameter of data set after the Andoni-Indyk LSH round. Since we have $\mu = n^{-\Theta(1)}$, and $\alpha = 10^{-4}$ as per line 5 of Algorithm 3, by Lemma 28 if we let R_{diam} be a large enough constant, then one has

$$\Pr_h[\text{diameter of } h^{-1}(\mathbf{q})\cap\widetilde{P} \text{ is larger than } R_{diam}] \leq n^{-20} \tag{23}$$

Let $\mathcal{E}_{diam}$ denote the event that all Andoni-Indyk hash buckets that the query hashes to have diameter bounded by R_{diam}. We have, combining the failure event over sampling of $\widetilde{P}$ (over-sampling by a factor more than $O(\log n)$) with (23) that $\Pr[\bar{\mathcal{E}}] \leq 2n^{-20} \leq n^{-19}$. Conditioned on $\mathcal{E}$ buckets that the query hashes have diameter bounded by R_{diam}. Now, if we take any point in the data set and consider a ball of radius $R_{\max} := 2R_{\text{diam}}$, using the triangle inequality, it contains all the points of this hash bucket. This ensures that all the spheres in the recursion tree have radius bounded by $R_{\max} = \Theta(1)$.

Corollary 29 (Bounded diameter spheres)**.** *All the spheres that the query scan in the algorithm have radius bounded by* $R_{max} = \Theta(1)$.[12]

We are now ready to present our main preprocessing primitive SPHERICAL-LSH, given as Algorithm 4 below. The input to the algorithm is a node in the recursion tree $\mathcal{T}$ created by recursive invocations of SPHERICAL-LSH. Every such node v is annotated with a dataset $v.P$, a radius $v.r$ of a ball enclosing the dataset, the center $v.o$ of that ball and a level, $v.level$, initially set to 0 for the root of the tree $\mathcal{T}$ that is created by PREPROCESS. SPHERICAL-LSH then proceeds as follows. First it calls the PSEUDORANDOMIFY procedure (Algorithm 5 below). This procedure partitions the input dataset $v.P$ into subsets that are pseudorandom as per Definition 13. A similar procedure was used in the work of [ALRW17] on space/query time tradeoffs for nearest neighbor search. Intuitively, a dataset is pseudorandom if the points belong to a thin spherical shell and

[12]Since we did not use any density constraints other than the upper bound of n on the number of points, this corollary applies for all spheres in the Algorithm.

furthermore do not concentrate on any spherical cap in this shell (appropriately defined). These pseudorandom datasets are added to the recursion tree $\mathcal{T}$ as children of v. Spherical-LSH then generates random subsets of these pseudorandom spheres defined by random spherical caps, adds these datasets to the recursion tree $\mathcal{T}$ and recursively calls itself until a depth budget T (see line 3 below) is exhausted. Note that the radius of spherical caps generated depends on the distance x' from the projected query point to the target near point (which is assumed to be at distance x from the query). Note that since the query is not available at the preprocessing stage, the algorithm prepares data structures for all possible values of x' (see line 14 in Algorithm 4 below). Note that the value of x' is passed down the recursion tree. In the section below, we set the parameters that we use in the algorithms.

5.2 Parameter settings

- $\gamma = \frac{1}{\log\log\log n}$ and $\tau = \frac{1}{10}$ are the parameters used for pseudo-random spheres (see Definition 13) in Algorithm 5.
- $\alpha^* = 0.172$ (see Section 9), $T = \sqrt{\log n}$ and $J = \min\left\{\alpha^* \cdot T, \left(\frac{x^2}{2}\left(1 - \frac{x^2}{2}\right) + 10^{-4}\right) \cdot T\right\}$ are parameters to bound the depth of the recursion tree.
- $\delta = \exp(-(\log\log n)^C)$ for some large enough constant C, is a parameter used for partitioning point in a ball to discrete spheres of radii multiplies of δ (see Algorithm 5)
- $\delta' = \exp(-(\log\log n)^C)$ for some large enough constant C, is a parameter for rounding x''s to x'''s (see lines 18 and 19 in Algorithm 6)
- $R_{\min} = 10^{-5}$ is a lower bound on the radius of spheres that we process further, i.e., we stop whenever the radius becomes less than $R_{\min}$.
- $\Delta = 10^{-20}$ is a tiny constant. For a discussion about Δ see Remark 4.
- $\alpha = 10^{-4}$ is a parameter used for the Andoni-Indyk LSH round (see Algorithm 3)
- $\delta_z = 10^{-6}$ is a parameter used in discretizing continuous densities in Definition 41.
- $\delta_x = 10^{-8}$ is used for defining a grid over $(0, \sqrt{2})$, such that for any x from this grid we prepare the data structure to recover points from $[x - \delta, x)$ (see Algorithm 9).

Algorithm 4 SPHERICAL-LSH: x is the target distance to recover, v is the node in recursion tree (corresponds to a subset of the dataset)

```
1: procedure SPHERICAL-LSH($v, x, \mu$)
2:     $\gamma \leftarrow \frac{1}{\log\log\log n}$
3:     $T \leftarrow \sqrt{\log n}$
4:     $U \leftarrow$ PSEUDORANDOMIFY($v, \gamma$)
5:     for $w \in U$ do
6:         Add $w$ as a child of $v$ in recursion tree $\mathcal{T}$
7:         $P \leftarrow w.P$                                        ▷ The dataset of $w$
8:         $R \leftarrow w.r$                                        ▷ Radius of the sphere of $w$
9:         if $R < R_{min}$ continue
10:        $\delta' \leftarrow \exp(-(\log\log n)^C)$
11:        $o \leftarrow w.o$                                        ▷ Center of sphere of $w$
12:        $W \leftarrow \left\{\lfloor\frac{\Delta-\delta}{\delta'}\rfloor \cdot \delta', \left(\lfloor\frac{\Delta-\delta}{\delta'}\rfloor + 1\right)\delta', \ldots\right\} \cap (0, R(\sqrt{2}+\gamma)]$
13:                ▷ The smallest element in $W$ is $\Theta(1)$ by the setting of parameters. See 5.2
14:        for $x'' \in W$ do                        ▷ Enumerate over potential target distances
15:            if $x'' > R(\sqrt{2}+\gamma)$ continue
16:            Choose $\eta$ such that $\frac{F(\eta)}{G(x''/R,\eta)} = \left(\frac{1}{\mu}\right)^{\frac{1}{T}}$
17:                ▷ Choose $\eta$ such that a query explores $\left(\frac{1}{\mu}\right)^{\frac{1}{T}}$ children in expectation
18:            for $i = 1, \ldots, \left\lceil\frac{100}{G(x''/R,\eta)}\right\rceil$ do
19:                Sample a Gaussian vector $g \sim N(0,1)^d$
20:                $P' \leftarrow \left\{\mathbf{p} \in P : \left\langle\frac{\mathbf{p}.new-o}{R}, g\right\rangle \geq \eta\right\}$
21:            ▷ $\mathbf{p}.new$ is the rounded $\mathbf{p}$ to the surface of the sphere (see line 10 of Algorithm 5)
22:                if $P' \neq \emptyset$ then
23:                    Add a node $v'$ as child of $w$ in $\mathcal{T}$
24:                    $v'.P \leftarrow P',\ v'.level \leftarrow v.level+1,\ v'.g \leftarrow g,\ v'.r \leftarrow R,\ v'.o \leftarrow o,\ v'.x \leftarrow x''$
25:                    $v'.\eta \leftarrow \eta$
26:                    if $v'.level \neq J$ then
27:                        ▷ Stop whenever the level becomes $J$ (see Section 5.2 for the value of $J$.)
28:                        SPHERICAL-LSH($v', x, \mu$)    ▷ Recurse unless budget has been exhausted
```

Algorithm 9 is the standard (similar to Section 4) outer algorithm and is presented in Appendix C. It simply calls PREPROCESS (Algorithm 3) presented in this section. The following lemma bounds the space complexity of the preprocessing algorithm:

Lemma 30. *Under Assumption 1, the expected space consumption of the datastructure generated by* PREPROCESS-KDE*($\widetilde{P}, \mu$) (Algorithm 9) is bounded by*

$$\min\left\{n\exp_\mu(0.173), \exp_\mu(1+c+o(1))\right\},$$

for small constant $c = 10^{-3}$.

Proof. First, we calculate the expected size of the data structure created by PREPROCESS($\widetilde{P}, x, \mu$) for any $x \in \{\delta_x, 2\delta_x, \ldots\} \cap (0, \sqrt{2})$ (see line 6 in Algorithm 9). Note that the expected size of the sampled dataset is

$$\mathbb{E}[|\widetilde{P}|] \leq \min\left\{\exp_\mu\left(1-\frac{x^2}{2}\right), n\right\}.$$

Since PseudoRandomify does not duplicate points, every point in the dataset is duplicated (due to their presence in different spherical caps) at most

$$\exp_\mu\left(\frac{1}{T}\right) \cdot |W| = \exp_\mu\left(\frac{1}{T}\right) \cdot \exp\left((\log\log n)^{O(1)}\right)$$

times in expectation each time we increase the level. So, in total every point is duplicated at most

$$\exp_\mu\left(\frac{J}{T}\right) \cdot |W|^J = \exp_\mu\left(\frac{J}{T}\right) \cdot |W|^J$$

in expectation. Indeed, in every level we enumerate over at most $|W| = \exp((\log\log n)^{O(1)})$ possibilities for x'', amounting to at most a factor of $|W|^J = \exp\left((\log\log n)^{O(1)} \cdot O(\sqrt{\log n})\right) = n^{o(1)}$ duplication due to the termination condition in line 27 of Algorithm 4. Finally, PreProcess itself hashes every point $100\exp_\mu(\alpha) \leq 100\exp_\mu(10^{-4})$ times (see line 6 and line 5 of Algorithm 3). Putting these bounds together yields that the space consumption of PreProcess$(\widetilde{P}, x, \mu)$ is at most

$$\min\left\{\exp_\mu\left(1-\frac{x^2}{2}\right), n\right\} \cdot \exp_\mu\left(10^{-4} + \min\left\{\alpha^*, \frac{x^2}{2}\left(1-\frac{x^2}{2}\right) + 10^{-4}\right\} + o(1)\right)$$

in expectation, where $c := 10^{-4}$. Now, note that we also repeat this procedure $\left(\frac{1}{\mu}\right)^{4\delta_x + o(1)}$ times (see Algorithm 9), which results in the following bound on the total space consumption

$$\max_{x\in(0,\sqrt{2})}\left(\min\left\{\exp_\mu\left(1-\frac{x^2}{2}\right), n\right\} \cdot \exp_\mu\left(10^{-4} + \min\left\{\alpha^*, \frac{x^2}{2}\left(1-\frac{x^2}{2}\right) + 10^{-4}\right\} + 4\delta_x + o(1)\right)\right)$$
$$\leq \min\left\{n\exp_\mu(0.173), \exp_\mu(1+c+o(1))\right\},$$

for $c = 10^{-3}$. □

Finally, we introduce the procedure PseudoRandomify (Algorithm 5 below) used in Spherical-LSH. This procedure is quite similar to the corresponding primitive in [ALRW17] and is guaranteed to output pseudo-random spheres with parameters τ and γ (See Definition 13).

Algorithm 5 PseudoRandomify

```
1:  procedure PseudoRandomify($v, \gamma$)
2:      $\delta \leftarrow \exp(-(\log\log n)^C)$
3:      $R_{min} \leftarrow$ sufficiently small constant larger than $\Delta$ and $\delta_x$ (see Section 5.2)
4:      $P \leftarrow v.P$                                   ▷ Dataset of node $v$
5:      $R \leftarrow v.r$                                   ▷ Radius of sphere of node $v$
6:      $o \leftarrow v.o$                                   ▷ Center of sphere of node $v$
7:      $\tau \leftarrow \frac{1}{10}$
8:      if $R < R_{min}$ return
9:      for $\mathbf{p} \in P$ do
10:         $\mathbf{p}.new \leftarrow o + \delta\lceil\frac{||\mathbf{p}-o||}{\delta}\rceil \cdot \frac{\mathbf{p}-o}{||\mathbf{p}-o||}$     ▷ $\mathbf{p}$ represents the initial coordinates of point $\mathbf{p}$
11:     $V \leftarrow \emptyset$
12:     for $i \leftarrow 1 \ldots \lceil\frac{R}{\delta}\rceil$ do         ▷ Process all resulting spheres
13:         $\widetilde{P} \leftarrow \{\mathbf{p} \in P\ : ||\mathbf{p}.new - o|| = \delta i\}$
14:         if $\widetilde{P} \neq \emptyset$ then
15:             $\hat{R} \leftarrow (\sqrt{2} - \gamma)R$
16:             $m \leftarrow |\widetilde{P}|$
17:             $m' \leftarrow 0$
18:             while $m' \leq \frac{m}{2}$ do
19:                 $m \leftarrow |\widetilde{P}|$
20:                 while $\exists \hat{o} \in \partial B(o, \delta i) : |B(\hat{o}, \hat{R}) \cap \widetilde{P}| \geq \frac{1}{2} \cdot \tau \cdot m$ do
21:                                  ▷ Using rounded $p.new$ coordinates (see line 10) in line above
22:                     $P' \leftarrow \widetilde{P} \cap B(\hat{o}, \hat{R})$
23:                     $B(o', R') \leftarrow \mathrm{SEB}(P')$          ▷ SEB=smallest enclosing ball
24:                     Create a node $tmp$
25:                     $tmp.P \leftarrow P',\ tmp.level \leftarrow v.level,\ tmp.g \leftarrow 0,\ tmp.r \leftarrow R',\ tmp.o \leftarrow o'$
26:                     $V \leftarrow V \cup$ PseudoRandomify($tmp, \gamma$)
27:                     $\widetilde{P} \leftarrow \widetilde{P} \setminus B(\hat{o}, \hat{R})$
28:                 $m' \leftarrow |\widetilde{P}|$
29:             Create a node $w$
30:             $w.P \leftarrow \widetilde{P},\ w.level \leftarrow v.level,\ w.g \leftarrow 0,\ w.r \leftarrow \delta i, w.o \leftarrow o$
31:             $V \leftarrow V \cup \{w\}$
32:     return $V$
```

5.3 Query procedure

We now present our query procedure (Algorithm 6 below). The procedure simply traverses the recursion tree $\mathcal{T}$ from the root, exploring leaves that the query is mapped to according to line 29. Since every node u of the tree $\mathcal{T}$ corresponds to a pseudorandom dataset $u.P$ residing (essentially) on a sphere of radius $u.r$ centered at $u.o$, the query is projected onto the sphere, after which one recursively explores the children of u in $\mathcal{T}$ whose Gaussian vectors (see line 29) are sufficiently correlated with the projected query. One notable feature in comparison to the corresponding procedure in [ALRW17] is the follows. Note that the procedure of [ALRW17] recurses on a sphere even if the intersection of a sphere of radius x around the query (i.e. the range in which we would like to report points) barely touches the sphere that the dataset resides on. Our data structure, however, uses an increased search range $x + \Delta$ (see Figure 7), which results in somewhat higher runtime, but allows one to only recurse when the extended search range has nontrivial overlap with the sphere in question – see lower bound on x' in line 12 of Algorithm 4. This additive Δ technique, can also be used to simplify the technical proofs of [ALRW17], by not allowing their algorithm to recurse on tiny spheres at distance roughly x (i.e., when the distance x barely touches the sphere). The reason is that all the points on these small spheres has distance at most $x + 2R_{min}$ from the query, and we have small number of such points in expectation, by sub-sampling and density constraints.

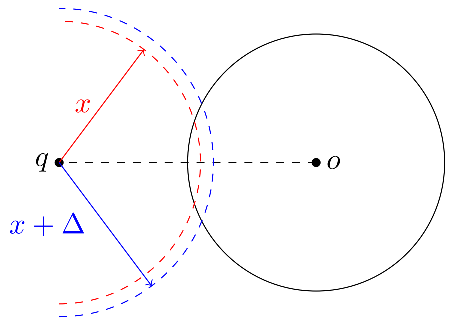


Figure 7: An (exaggerated) illustration of x and $x + \Delta$.

Algorithm 6 Query

```
1: procedure QUERY($\mathbf{q}, \mathcal{T}, x$)
2:     $P_x \leftarrow \emptyset$
3:     $\delta \leftarrow \exp(-(\log\log n)^C)$
4:     $v \leftarrow$ root of $\mathcal{T}$
5:     if $v.level = 0$ then
6:         $K_1 = 100\left(\frac{1}{\mu}\right)^{\alpha}$        ▷ $\alpha$ is the constant from line 5 in Algorithm 3
7:         for $j = 1, 2, \ldots, \lceil K_1 \rceil$ do
8:             Locate $\mathbf{q}$ in $h_j$ and $u \leftarrow$ the corresponding node in $\mathcal{T}$
9:             for each $w$ child of $u$ do
10:                $\mathcal{T}_w \leftarrow$ sub-tree of $w$ and its descendants.
11:                $P_x \leftarrow P_x \cup$ QUERY($\mathbf{q}, \mathcal{T}_w, x$)
           return $P_x$
12:    else if $\mathcal{T}$ is just one node, without any children then
13:        return $v.P$
14:    else
15:        $o \leftarrow v.o$
16:        $R \leftarrow v.r$
17:        $R_2 \leftarrow ||\mathbf{q} - o||$
18:        $x' \leftarrow$ PROJECT($x + \Delta, R_2, R$)
19:        $x'' \leftarrow$ smallest element in the grid $W$ (line 12 of Algorithm 4) which is not less than $x'$
20:        if $x + \delta < |R - R_2|$ then return
21:                                    ▷ Then no point from distance $x$ can be on this sphere
22:        if $\nexists u$ child of $v$, such that $u.x = x''$ then
23:            return $v.P$
24:        for each $u$ child of $v$ do
25:            $\Delta \leftarrow 10^{-20}$
26:            if $u.x = x''$ then
27:                $g \leftarrow u.g$
28:                $\eta \leftarrow u.\eta$
29:                if $\langle g, \frac{\mathbf{q}-o}{||\mathbf{q}-o||} \rangle \geq \eta$ then
30:                    for each $w$ child of $u$ do
31:                        $\mathcal{T}_w \leftarrow$ sub-tree of $w$ and its descendants.
32:                        $P_x \leftarrow P_x \cup$ QUERY($\mathbf{q}, \mathcal{T}_w, x$)
33:        return $P_x$
```

Since the correctness analysis of this procedure is standard and similar to [ALRW17], we present it in Appendix C (Lemma 60), for completeness. Basically, we prove the query procedure outputs any given point within distance x with high constant probability.

6 Query time analysis

The main result of this section is the following lemma which bounds the expected query time of the algorithm.

Lemma 31. *The expected query time is bounded by* $O\left(\left(\frac{1}{\mu}\right)^{0.173+o(1)}\right)$.

Throughout this section we consider the setting where one is given a query $\mathbf{q} \in \mathbb{R}^d$ and a parameter $\mu \in (0, 1]$ with the promise that

$$\mu^* \leq \mu, \tag{24}$$

where

$$\mu^* = K(P, \mathbf{q})$$

is the true kernel density value. We assume that $\mu^* = n^{-\Theta(1)}$, since this is the interesting regime for this problem. For $\mu^* = n^{-\omega(1)}$ under the Orthogonal Vectors Conjecture (e.g. [Rub18]), the problem cannot be solved faster than $n^{1-o(1)}$ using space $n^{2-o(1)}$ [CS19], and for larger values $\mu^* = n^{-o(1)}$ random sampling solves the problem in $n^{o(1)}/\epsilon^2$ time and space.

Densities of balls around query. Upper bounds on the number of points at various distances from the query point in dataset (i.e., densities of balls around the query) play a central part in our analysis. The core of our query time bound amounts to tracking the evolution of such densities in the recursion tree $\mathcal{T}$. In order to analyze the evolution of these upper bounds we let, for a query $\mathbf{q} \in \mathbb{R}^d$ (which we consider fixed throughout this section) and any $x \in (0, \sqrt{2})$ let

$$D_x(\mathbf{q}) := \{||\mathbf{p} - \mathbf{q}|| : \ \mathbf{p} \in P, ||\mathbf{p} - \mathbf{q}|| \geq x + 1.5\Delta\}, \tag{25}$$

denote the set of possible distances from the query to the points in the dataset which are further that $x + 1.5\Delta$ from the query. When there is no ambiguity we drop q and x and we simply call it D. For any $y \in D$ we let

$$P_y(\mathbf{q}) := \{\mathbf{p} \in P : ||\mathbf{p} - \mathbf{q}|| \leq y\} \tag{26}$$

be the set of points at distance y from $\mathbf{q}$. Since for every $y > 0$

$$\begin{aligned}\mu^* = K(P, \mathbf{q}) &= \frac{1}{n}\sum_{\mathbf{p} \in P} \mu^{||\mathbf{p}-\mathbf{q}||_2^2/2} \\ &\geq \frac{\mu^{y^2/2}}{n}|P_y(\mathbf{q})|\end{aligned}$$

we get

$$|P_y(\mathbf{q})| \leq n\mu^* \cdot \left(\frac{1}{\mu}\right)^{\frac{y^2}{2}} \leq n \cdot \left(\frac{1}{\mu}\right)^{\frac{y^2}{2}-1},$$

since $\mu^* \leq 4\mu$ by assumption.

Densities in the subsampled dataset. Fix $x \in (0, \sqrt{2})$, and recall that $\widetilde{P}$ contains every point in P independently with probability $\frac{1}{n} \cdot \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}}$. Note that for every y the expected number of points at distance at most y from query $\mathbf{q}$ that are included in $\widetilde{P}$ is upper bounded by

$$\min\left\{n \cdot \left(\frac{1}{\mu}\right)^{\frac{y^2}{2}-1}, n\right\} \cdot \frac{1}{n} \cdot \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}} \leq \min\left\{\exp_\mu\left(\frac{y^2 - x^2}{2}\right), \exp_\mu\left(\frac{1 - x^2}{2}\right)\right\}, \tag{27}$$

and Figure 8 illustrates this.

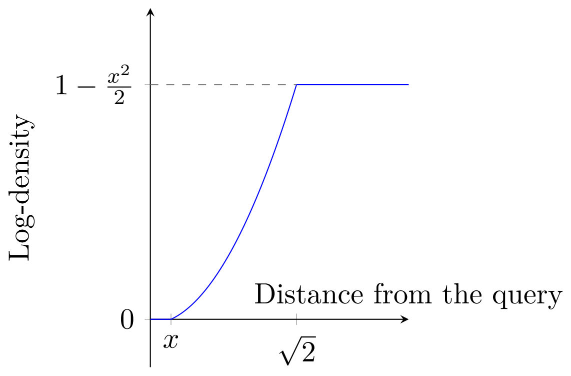


Figure 8: Upper-bound on log-densities after sub-sampling.

Our main goal is to track the progress of the query $\mathbf{q}$ and any $\mathbf{p}$, for which we have $||\mathbf{p}-\mathbf{q}||_2 \leq x$, that was included in the set $\widetilde{P}$, and exploit the upper bounds (27) on the number of points at various distances from $\mathbf{q}$ in $\mathbf{q}$'s 'hash bucket' to show that the process quickly converges to a constant size data set at a leaf of $\mathcal{T}$. It is not hard to see (Lemma 46 below) that the number of nodes in $\mathcal{T}$ that the query explores is low. The main challenge is to show that the expected size of a leaf data set in $\mathcal{T}$ is small, since for that one needs to prove strong upper bounds on the number of points at various distances from the query in dataset that the query traverses on its path to a leaf in $\mathcal{T}$. We exploit two effects:

(Removal of points due to truncation) The PSEUDORANDOMIFY procedure, which is crucial to ensuring that spheres at nodes on $\mathcal{T}$ are pseudorandom, essentially acts as a trunction primitive on the density curve. See conditions **(2)** in Definition 36 below.

(Removal of points due to LSH) As the query explores the children of an LSH node $v \in \mathcal{T}$ the probability that a given point $\mathbf{p} \in v.P$ belongs to the same spherical cap as $\mathbf{q}$ depends on the distance between $\mathbf{p}$ and $\mathbf{q}$. This implies bounds on the evolution of the density of points at various distances y from $\mathbf{q}$ in the datasets that $\mathbf{q}$ explores on its path towards a leaf in $\mathcal{T}$. See conditions **(3)** in Definition 36 below.

The bulk of our analysis is devoted to understanding the worst case sequence of geometric configurations, i.e. spheres, that the query encounters on its path towards a leaf in $\mathcal{T}$.

6.1 Path geometries

We start by defining the path geometries in the recursion tree. Assume an invocation of PREPROCESS algorithm (Algorithm 3) and let $\mathcal{T}$ be the sub-tree that the query explores. Let

$$\mathcal{P} := (w_0, v_0, w_1, v_1, \ldots, w_J, v_J)$$

be any path from root to a LSH leaf at level J.

For any $j \in [J]$, suppose that given $x'' := v_j.x$ and $r := v_i.r$, we are interested in the distance from the query to the center of the sphere $(v_j.o)$. For simplicity of notation let $\widetilde{\ell} = ||\mathbf{q}-v_j.o||$. Recall that x'' is the rounded value for $x' = \text{PROJECT}(x+\Delta, \widetilde{\ell}, r)$ (see lines 18 and 19 of Algorithm 6). However, this equation is a degree two polynomial in $\widetilde{\ell}$, so it has at most two solutions. For intuition, Figure 9 shows these two solutions with an example. The solutions to the equation correspond to the points that the dashed circle intersects with the dashed line, i.e., position of $\mathbf{q}$. Now, recall that x'' is the rounded x' (see line 19 of Algorithm 6). So, x' can change in a small interval. This corresponds to moving the center of the dashed circle over the red arc. This changes the position of intersections, however, they still belong to a relatively small interval (shown in blue in Figure 9), we denote this intervals by *left interval* and *right interval.* Now, given query $\mathbf{q}$, we check weather it corresponds to the left interval or the right interval, and based on that we set b_j to be 1 or 2, respectively. We also let ℓ be the distance of the leftmost point in the interval of the query, from the center of the sphere. And we call ℓ the distance induced by (x'', r) and $\mathbf{q}$. In appendix D we formally argue this procedure.

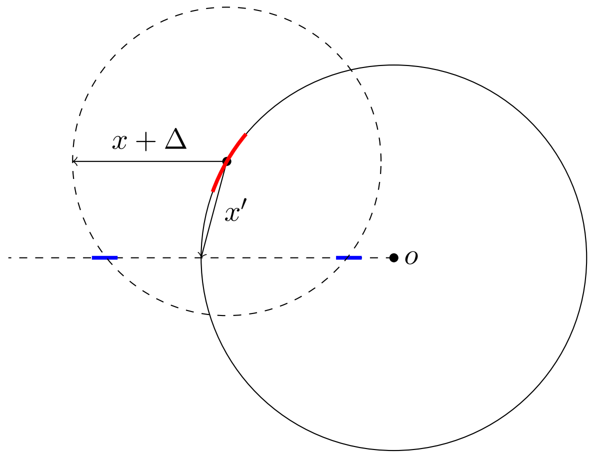


Figure 9: Geometric illustration of equation $x' = \textsc{Project}(x + \Delta, \widetilde{\ell}, r)$ when we have access to an approximation of x' (red arc).

Definition 32 (Path geometry and induced distances)**.** For any query $\mathbf{q}$ and tree $\mathcal{T}$ (as described above) for any root to leaf path

$$\mathcal{P} = (w_0, v_0, w_1, v_1, \dots, w_J, v_J),$$

we call

$$G(\mathcal{P}) := ((x''_1, r_1, b_1), \dots, (x''_J, r_J, b_J))$$

the geometry of path $\mathcal{P}$ where for all $i \in [J]$,

1. $x''_i := v_i.x$,
2. $r_i := v_i.r$,
3. b_i is as described above (formally defined in Appendix D).

Additionally, we call $L(\mathcal{P}) := (\ell_1, \dots, \ell_J)$ the induced distances of path $\mathcal{P}$, where for all $i \in [J]$, ℓ_i is induced by (x''_i, r_i) as explained above and formally defined in Appendix D.

Definition 33 (Sphere geometries)**.** For any query $\mathbf{q}$ and tree $\mathcal{T}$ (as described above) for any root to leaf path

$$\mathcal{P} = (w_0, v_0, w_1, v_1, \dots, w_J, v_J),$$

if the geometry of this path is defined as

$$G(\mathcal{P}) := ((x''_1, r_1, b_1), \dots, (x''_J, r_J, b_J))$$

then for any $j \in [J]$ we say that w_j and v_j has geometry (x''_j, r_j, b_j).

Recall from Definition 13 that the PSEUDORANOMIFY procedure (Algorithm 5) ensures that most of the points on any pseudorandom sphere w are nearly orthogonal to $\mathbf{q} - w.o$. We want to know, how the fact that a sphere is pseudorandom translates to densities. For the first step, we need to understand if a point on the sphere is almost orthogonal to the projection of the query on the sphere, then what the range of possible distances of these points from the query is. We define the $c := \sqrt{\ell^2 + r^2}$ which simplifies the notation. As Figure 10 suggests, we expect the orthogonal points to be at distance $\approx c$. The following claim formally argues how pseudorandomness of a sphere translates to a condition on the densities.

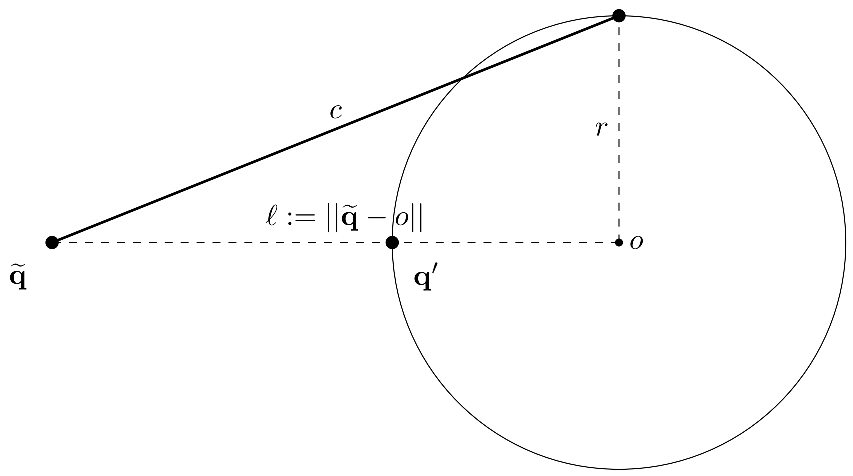


Figure 10: Illustration of the definition of c, the distance from the query to a 'typical' point on the sphere.

Claim 34 (Truncation claim). *Given query $\mathbf{q}$, let w be a pseudo random sphere with geometry (x'', r, b) which induces distance ℓ. Let $w.P$ be the set of points on this sphere, i.e., for any $\mathbf{p} \in w.P$, $\mathbf{p}.new$ is on the sphere. For all y let B_y be the number of points at distance y from $\mathbf{q}$ in $w.P$. Then, the following conditions hold.*

$$\sum_{y \le c - r\psi} B_y \le \frac{\tau}{1 - 2\tau} \cdot \sum_{y \in (c - r\psi, c + r\psi)} B_y,$$

and

$$\sum_{y \ge c + r\psi} B_y \le \frac{\tau}{1 - 2\tau} \cdot \sum_{y \in (c - r\psi, c + r\psi)} B_y,$$

where $\psi = \gamma^{1/3} + \delta'^{1/4} + \delta^{1/4}$ (the same as in Claim 63) and $c := \sqrt{\ell^2 + r^2}$.

Proof. The proof is just a simple application of Claim 63 to this sphere. □

Suppose that one has two points on the sphere at some distance from each other, we can use Lemma 8 and Lemma 9, to find collision probabilities under a spherical cap of size η. However, in general the query is not on the sphere, so we need to translate distance from $\mathbf{q}$ to any point $\mathbf{p}$ to distance from $\mathbf{q}'$ (projection of $\mathbf{q}$ on the sphere) to $\mathbf{p}$, using a function called PROJECT (formally defined in Definition 11 and its formula is given in Lemma 12). Also, there are some rounding steps, such as rounding the points to the sphere and rounding of the distance from the query to the center of the sphere (rounding of $\widetilde{\ell}$ to ℓ). Considering all these issues, the following claim illustrates the effect of spherical LSH on the points based on their distance from the query.

Claim 35 (Spherical LSH claim)**.** *Suppose that there is a sphere with geometry* (x'', r, b) *and induced distance* ℓ *(see Section 6.1 and Definition 32) for some* $x'' \in W$*,* $r \in \left[\left\lceil \frac{R_{max}}{\delta} \right\rceil\right]$ *and* $b \in \{1, 2\}$*. Let* o *be the center of the sphere. Also, let* $\mathbf{p}$ *be a point such that* $y = ||\mathbf{p} - \mathbf{q}||$ *and* $\mathbf{p}.new$ *is on the sphere (see line 10 of Algorithm 5). Now, suppose that one generates a Gaussian vector* g *as in Algorithm 4. Then, we have*

$$\Pr_{g\sim N(0,1)^d}\left[\langle g, \frac{\mathbf{p}.new - o}{||\mathbf{p}.new - o||}\rangle \geq \eta | \langle g, \frac{\mathbf{q} - o}{||\mathbf{q} - o||}\rangle \geq \eta\right] \leq \exp_\mu\left(-\frac{4(r/x')^2 - 1}{4(r/y')^2 - 1} \cdot \frac{1}{T}\right).$$

where

- η *is such that* $\frac{F(\eta)}{G(x''/r,\eta)} = \left(\frac{1}{\mu}\right)^{\frac{1}{T}}$ *(see line 16 of Algorithm 4).*
- $x' :=$ PROJECT$(x + \Delta, \ell, r)$.
- $y' :=$ PROJECT$(y - \Delta/2, \ell, r)$.

Proof. Let o be the center of the sphere. Let $\widetilde{\ell} := ||\mathbf{q} - o||$. Recall by the discussion in Section 6.1 and Definition 32 that any sphere geometry (x'', r, b) induces a distance ℓ. Now, suppose that we move the query in the direction of the vector from o to $\mathbf{q}$, such that for the new point $\widetilde{\mathbf{q}}$, we get $||\widetilde{\mathbf{q}} - o|| = \ell$. Now, one should note that the geometry of the sphere with respect to $\mathbf{q}$ and $\widetilde{\mathbf{q}}$ is the same. Also, the projections of $\mathbf{q}$ and $\widetilde{\mathbf{q}}$ on the sphere are identical. Also, for point $\mathbf{p}$ at distance y from $\mathbf{q}$, by the triangle inequality for $(\mathbf{q}, \widetilde{\mathbf{q}}, \mathbf{p})$, since $\widetilde{\ell} \in [\ell - \delta'^{1/3}, \ell]$ we get

$$||\mathbf{p} - \widetilde{\mathbf{q}}|| \in [y - \delta'^{1/3}, y + \delta'^{1/3}]. \tag{28}$$

Now, if we let point $\mathbf{q}'$ be the projection of $\widetilde{\mathbf{q}}$ on the sphere, and let $\mathbf{p}.new$ be the rounded $\mathbf{p}$ on the sphere, then $||\mathbf{p}.new - \widetilde{\mathbf{q}}|| \in [y - \delta - \delta'^{1/3}, y + \delta + \delta'^{1/3}]$, which implies

$$y'' := ||\mathbf{q}' - \mathbf{p}.new|| \in [\text{PROJECT}(y - \delta - \delta'^{1/3}, \ell, r), \text{PROJECT}(y + \delta + \delta'^{1/3}, \ell, r)].$$

Note that with this definition of y'' one has

$$\Pr_{g\sim N(0,1)^d}\left[\langle g, \frac{\mathbf{p}.new - o}{||\mathbf{p}.new - o||}\rangle \geq \eta | \langle g, \frac{\mathbf{q} - o}{||\mathbf{q} - o||}\rangle \geq \eta\right] = \frac{G(y''/r, \eta)}{F(\eta)}. \tag{29}$$

Now, by invoking Claim 64, **(b)**

$$\Pr_{g\sim N(0,1)^d}\left[\langle g, \frac{\mathbf{p}.new - o}{||\mathbf{p}.new - o||}\rangle \geq \eta | \langle g, \frac{\mathbf{q} - o}{||\mathbf{q} - o||}\rangle \geq \eta\right] \leq \exp_\mu\left(-\frac{4(r_j/x')^2 - 1}{4(r_j/y')^2 - 1} \cdot \frac{1}{T}\right) \tag{30}$$

Now, we verify the preconditions of Claim 64, **(b)**. Condition **(p1)** of Claim 64 is satisfied by setting of δ as $\delta + \delta'^{1/3}$.[13] Condition **(p2)** is satisfied by line 10 in Algorithm 4. Condition **(p3)** is satisfied by setting of Δ in line 25 of Algorithm 6. Finally, condition **(p4)** is satisfied due to line 15 in Algorithm 4 that ensures that a nontrivial data structure is only prepared for $x' \leq R(\sqrt{2} + \gamma)$, and no recursion is performed otherwise.

Conditioned on event $\mathcal{E}_{diam}$ (which ensures constant upper-bound on the radii of spheres, see the discussion in Section 5.1), $r = O(1)$. Thus, we can invoke part **(b)** of Claim 64 applies and gives (30). □

[13] To be more clear, we set the δ of claim 64 as $\delta + \delta'^{1/3}$ where δ and δ' are the parameters of the algorithm.

In the following definition we summarize the effect of sub-sampling the dataset, the truncation rounds and the spherical LSH rounds on densities along the path.

Definition 36 (Valid execution path)**.** Let $R := (r_j)_{j=1}^J$ and $L := (\ell_j)_{j=1}^J$ for some positive values r_j's and ℓ_j's such that for all $j \in [J]$, $x + \delta \geq |\ell_j - r_j|$. Also let D be as defined in (25). Then, for

$$\begin{aligned} A &:= (a_{y,j}), \quad y \in D, j \in [J] \cup \{0\} && \text{(Intermediate densities)} \\ B &:= (b_{y,j}), \quad y \in D, j \in [J+1] \cup \{0\} && \text{(Truncated intermediate densities)} \end{aligned}$$

(L, R, A, B) is called a *valid execution path*, if the conditions below are satisfied. We define $\psi := \gamma^{1/3} + \delta'^{1/4} + \delta^{1/4}$ and $c_j := \sqrt{r_j^2 + \ell_j^2}$ for convenience.

(1) Initial densities condition. The $a_{y,0}$ and $b_{y,0}$ variables are upper-bounded by the initial expected densities in the sampled dataset: for all $y \in D$

$$\sum_{y' \in [0,y] \cap D} a_{y',0} \leq \min\left\{\exp_\mu\left(\frac{y^2 - x^2}{2}\right), \exp_\mu\left(\frac{1 - x^2}{2}\right)\right\}$$

and

$$\sum_{y' \in [0,y] \cap D} b_{y',0} \leq \min\left\{\exp_\mu\left(\frac{y^2 - x^2}{2}\right), \exp_\mu\left(\frac{1 - x^2}{2}\right)\right\}$$

(2) Truncation conditions (effect of PseudoRandomify). For any $j \in [J]$, for all $y \in D \setminus [\ell_j - r_j, \ell_j + r_j]$ one has $b_{y,j} = 0$ (density is zero outside of the range corresponding to the j-th sphere on the path; condition **(2a)**), for all $y \in D \cap [\ell_j - r_j, \ell_j + r_j]$ one has $b_{y,j} \leq a_{y,j-1}$ (removing points arbitrarily **(2b)**) and

$$\sum_{y \in [0, c_j - \psi r_j] \cap D} b_{y,j} \leq \frac{\tau}{1 - 2\tau} \cdot \sum_{y \in (c_j - \psi r_j, c_j + \psi r_j) \cap D} b_{y,j} \qquad \text{(condition (2c))}$$

(3) LSH conditions. For every $j \in [J]$ and all $y \in [\ell_j - r_j, \ell_j + r_j] \cap D$

$$a_{y,j} \leq b_{y,j} \cdot \exp_\mu\left(-\frac{4\left(\frac{r_j}{x'}\right)^2 - 1}{4\left(\frac{r_j}{y'}\right)^2 - 1} \cdot \frac{1}{T}\right)$$

where $x' := \textsc{Project}(x + \Delta, \ell_j, r_j)$ and $y' := \textsc{Project}(y - \Delta/2, \ell_j, r_j)$. See Remark 4 below for a discussion about Δ factors.

(4) Terminal density condition. For any y such that $a_{y,J}$ is defined, $b_{y,J+1} \leq a_{y,J}$.

Remark 5. Throughout the paper we need good bounds on the probability that a random spherical cap encompasses a data point $\mathbf{p}$, given that the spherical cap captures the projection of the query. The expression in condition **(3)** of Definition 36 is a convenient upper bound for this quantity when the distance from $\mathbf{p}$ to $\mathbf{q}$ is equal to y. Exact expressions for such collision probabilities are unstable with respect to perturbations of the point $\mathbf{p}$ when $\mathbf{p}$ is antipodal to $\mathbf{q}$ on the sphere, and because of this it is more convenient to work with upper bounds. Specifically, we upper bound this probability by imagining that the point is slightly closer (by $\Delta/2$) than the actual distance y, for a small positive constant Δ that affects our query time bounds. The advantage is that such probabilities are more stable under small perturbations of the data point $\mathbf{p}$ – see the proof of Claim 64 for more details. One notes that the expression in condition **(3)** also depends on x. This is because we select spherical cap sizes based on x – see line 16 of Algorithm 4.

We introduce the notion of the length of an execution path (L, R, A, B).

Definition 37. We define the length of an execution path (L, R, A, B) by $\text{Length}((L, R, A, B)) := |R| = J$.

A special class of execution paths that we refer to as zero-distance monotone paths will be central to our analysis:

Definition 38. (Zero-distance and monotone path) Let (L, R, A, B) be an execution path defined in Definition 36. If for $R = (r_j)_{j=1}^J$, r_j's are non-increasing in j, and $L = R$, then we say that (L, R, A, B) is a *zero-distance and monotone* execution path. When $L = R$, we usually drop L, and simply write (R, A, B).

The following crucial lemma allows our LP based analysis of the query time:

Lemma 39. *(Reduction to zero-distance monotone execution paths) For every valid execution path (L, R, A, B) (see Definition 36), there exists a* zero-distance and monotone *valid execution path (R', A', B') (see Definition 38) such that $b'_{y,J+1} = b_{y,J+1}$ for all $y \in D$[14] and $|R'| = |R|$ (i.e., the length of the paths are equal).*

The proof of this lemma is given in Section 7.

6.1.1 Linear programming formulation

As we prove in Lemma 39, for any execution path there exists a zero-distance monotone path (see Definition 38) with the same length and the same final densities. This means that if we prove that for any zero-distance monotone path, the final densities are small, then this generalizes to all possible execution paths. So, from now on we only consider zero-distance monotone paths.

As mentioned before, we analyze the evolution of density of points at various distances. Instead of analyzing continuous densities, we define a new notion, called *discretized log-densities* (see Definition 41), for which we round densities to the discretized distances in a natural way, and for simplicity of calculations we take the log of these densities. These two steps allow us to analyze the evolution of densities over the course of time. More specifically, we define an LP (see (33)) such that any zero-distance monotone execution path with large enough final densities, imposes a feasible solution to the LP, with cost (almost) equal to the length of the execution path divided by T. Thus, if the length of the execution path is large, final densities cannot be too large (see Section 8 and Claim 55 for the formal statement), which means that we managed to reduce the densities to a small amount.

In section 8 we formally describe the procedure for constructing a feasible solution based on discretized densities.

We start by defining a convenient discretization of the distances on a valid execution path:

Definition 40 (x-centered grid Z_x)**.** For every $x \in (0, R_{max})$ define the grid $Z_x = \{z_I, z_{I-1}, \ldots, z_0\}$ by letting $z_I = x$, letting $z_{I-i} := (1 + \delta_z)^i \cdot z_I$ for all $i \in [I]$ and choosing the smallest integer I such that $z_0 \geq R_{max}\sqrt{2}$.

[14]We need the final condition to argue that we have the same number of points remaining at the end.

Definition 41 (Discretized log-densities $f_{z_i,j}$)**.** For any zero-distance monotone valid execution path (R, A, B) (as per Definition 36) with radii bounded by R_{max} and $J = |R|$, for all $j \in [J]$ let k_j be the index of the largest grid element which is not bigger than $r_j \cdot (\sqrt{2} + \psi)$, i.e.,

$$r_j \cdot (\sqrt{2} + \psi) \in [z_{k_j}, z_{k_j - 1}) \tag{31}$$

and for every integer $i \in \{k_j, \dots, I\}$ define

$$f_{z_i,j} := \log_{1/\mu} \left(\sum_{y \in D \cap [z_{i+1}, z_{i-1})} b_{y,j} \right) \tag{32}$$

Note that the variables $b_{y,j}$ on the right hand side of (32) are the $b_{y,j}$ variables of the execution path (R, A, B).

Letting $Z := Z_x$ to simplify notation, we will consider I linear programs defined below in (33), enumerating over all $j^* \in [I]$, where we let $x' = x + \Delta$:

$$\begin{array}{lll}
\mathrm{LP}(x, j^*): & \max_{\alpha \geq 0} \sum_{j=1}^{j^*-1} \alpha_j & (33) \\
& \forall y \in Z : g_{y,1} \leq \min \left\{ \frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2} \right\} & \text{Density constraints} \\
& \text{for all } j < j^*, y \in Z, y < z_j : & \\
& \quad g_{y,j} \leq g_{z_j,j} & \text{Truncation} \\
& \quad g_{y,j+1} \leq g_{y,j} - \frac{2 (z_j/x)^2 - 1}{2 (z_j/y)^2 - 1} \cdot \alpha_j & \text{Spherical LSH} \\
& \quad g_{z_{j^*},j^*} \geq 0 & \text{Non-empty range constraint}
\end{array}$$

Intuitively, LP (33) captures the evolution of the density of points at different distances from the query throughout the hashing process. Our main technical claim connecting the LP (33) and execution paths in the query process is Claim 55 in Section 8.

6.2 Upper-bounding the expected number of points examined by the query

In this section we bound the expected number of points that the query examines in the query procedure. Let $\mathcal{T}$ be the tree that the query traverses. Note that the query only examines the points that it sees in the leaves that it visits. One should note that some leaves (which are LSH nodes for this case) in the tree have level J (see line 27 of Algorithm 4). However, they are other leaves in tree $\mathcal{T}$, due to two cases:

1. Path termination due to $x'' > R(\sqrt{2} + \gamma)$. This case happens when query $\mathbf{q}$ is such that it needs to recover points at distance x'' on the sphere, but this distance corresponds to points beyond orthogonal. Note that in the preprocessing phase we did not prepare any child with this x'' (see line 15 in Algorithm 4), so the query will stop at this node and scan the points (see line 22 of Algorithm 6). Roughly speaking, since we only expect $O(1)$ number of points at distance x, and since the number of points on the sphere is dominated by the number of points in the orthogonal band, then we expect to see small number of points on this sphere. We formally prove this in Claim 42.

2. Path termination due to small sphere radius. This simple case corresponds to the cases when PSEUDORANDOMIFY does not process a ball further due to line 8 of Algorithm 5 or SPHERICALLSH does not partition the dataset further due to line 9 of Algorithm 4. Note that in that case the entire ball is necessarily at distance at most $x + 2R_{min}$, and hence the total number of points in the ball is small. We formally argue and prove this in Claim 42.

Claim 42. *For any tree $\mathcal{T}$ that the query $\mathbf{q}$ explores, the expected total number of points in the leaves with level less than J is bounded by*

$$\left(\frac{1}{\mu}\right)^{\alpha+\alpha^*+c},$$

for $c = 10^{-4}$.

Proof. We investigate the two cases mentioned above separately:

Path termination due to $x'' > R(\sqrt{2}+\gamma)$. First, suppose that the exploration process terminates at node $u \in \mathcal{T}$ because of line 15 in Algorithm 4 . In that case one has by invoking Claim 14 for two diametral points on the sphere, since the current dataset $u.P$ is pseudorandom as per Definition 13 and $\tau = 1/10$,

$$\left|\left\{\mathbf{p} \in u.P : ||\mathbf{p} - \mathbf{q}'|| \in \left(R(\sqrt{2}-\gamma), R(\sqrt{2}+\gamma)\right)\right\}\right| = \Omega\left(|u.P|\right).$$

Note that the expected number of points at distance at most $R(\sqrt{2}+\gamma)$ from the query is upper-bounded by the expected number of points at distance at most $x + \Delta + \delta'$, since $x'' > R(\sqrt{2}+\gamma)$ and by rounding of x' to x'' (see line 19 in Algorithm 6). So, after sub-sampling the data set and using the density constraints, we have at most

$$\begin{aligned}
\frac{1}{n}\cdot\left(\frac{1}{\mu}\right)^{1-x^2/2}\cdot 4n\cdot\mu^{1-\frac{(x+\Delta+\delta')^2}{2}} &= 4\exp_\mu\left(\frac{(x+\Delta+\delta')^2}{2} - \frac{x^2}{2}\right) \\
&\le 4\exp_\mu\left(\frac{(x+2\Delta)^2}{2} - \frac{x^2}{2}\right) \\
&\le 4\exp_\mu\left(\frac{1}{2}\cdot(4\Delta x + 4\Delta^2)\right) \\
&\le \exp_\mu\left(5\Delta\right) && \text{Since } x \le \sqrt{2} \text{ and } \Delta = 10^{-20}
\end{aligned}$$

points.

Path termination due to small sphere radius. As we discussed above for this case, the entire ball is necessarily at distance at most $x + 2R_{min}$, since this sphere passed the condition in line 20 of Algorithm 6, and hence on expectation the total number of points in this ball is bounded by

$$\frac{1}{n}\cdot\left(\frac{1}{\mu}\right)^{1-x^2/2}\cdot n\cdot\mu^{1-(x+2R_{min})^2/2} \le \exp_\mu\left(4R_{min}\right)$$

where the last line is by our choice of parameters, and since $x \le \sqrt{2}$.

Also, by Lemma 46 we know that the query explores at most $\left(\frac{1}{\mu}\right)^{\alpha^*+\alpha+o(1)}$ leaves. Now, by setting of parameters, the claim holds. □

Lemma 43. *Under Assumption 1, there exists an event $\mathcal{E}$ that depends on the choice of the hash function in* PREPROCESS *only and occurs with probability at least $1-(1/\mu)^{-4}$ such that conditioned on $\mathcal{E}$, the following holds. The query examines at most*

$$\left(\frac{1}{\mu}\right)^{0.173} \tag{34}$$

number of points in expectation.

Proof. First, we just calculate the expected size of the data set examined by the query in invocations of QUERY (Algorithm 6), and then we bound the expected total number of points of QUERY-KDE (Algorithm 10). Note that the goal is to prove an upper-bound on the expected number of points that the query examines.

Consider an invocation PREPROCESS and let $\mathcal{T}$ be the sub-tree of the recursion tree that the query explores. Now, we define processes on this tree that output a subset of leaves of this tree. Suppose that

$$\mathcal{H} = W \times \left[\left[\frac{R_{\max}}{\delta}\right]\right] \times \{1,2\}$$

And let J be the maximum number of times that we applied spherical LSH. Let $\mathbf{q}$ be the query. Let $M := |\mathcal{H}^J|$ and enumerate elements in $\mathcal{H}^J$. For any leaf in $\mathcal{T}$ if one looks at the path to the root from this leaf, this corresponds to one element in $\mathcal{H}^J$ (See the discussion in Section 6.1 and Definition 32). For i'th element of $\mathcal{H}^J$, $h_i = (h_i(j))_{j=1}^J$, the procedure $\mathcal{P}_i(\mathcal{T})$ outputs set E_i, which is the set of output(s) of SAMPLE$(\mathcal{T}, h_i, 0)$.[15] Note that Algorithm 7 outputs a set of leaves in the tree.

Algorithm 7

```
1: procedure SAMPLE($\mathcal{T}, h_i, k$)
2:     $v \leftarrow$ a uniformly random child of the root of $\mathcal{T}$ which is consistent with $h_i(k)$.
3:     if $k = J$ then
4:         Return $v$
5:     for all $w$ in the set of the childern of $v$ do
6:         if $w$ is consistant with $h_i(k)$ then
7:             $\mathcal{T}' \leftarrow$ the sub-tree of tree where the root is $w$.
8:             SAMPLE($\mathcal{T}', h, k+1$).
```

Also, for any pseudo-random node on the tree that the query visits, since $\mu = n^{-\Omega(1)}$ by assumption, using a simple Chernoff bound argument, we have that it explores at most

$$m := O(1)\left(\frac{1}{\mu}\right)^{\frac{1}{T}}$$

children of this node, with high probability.

Let V be the set of leaves in $\mathcal{T}$, with level J. Partition V into $V_1, \ldots, V_M$, such that for all $i \in [M]$, the leaves in V_i admit the geometry defined by h_i.

Claim 44. *For any $u \in U_i$ we have the following*

$$\Pr[u \in E_i|\mathcal{T}] \geq \left(\frac{1}{m}\right)^J \left(\frac{1}{100\left(\frac{1}{\mu}\right)^\alpha}\right).$$

[15] Also, for the purpose of consistency define $h_i(0) = (0,0,0)$ and let $h_i \leftarrow (h_i(j))_{j=0}^J$ and assume that every Andoni-Indyk LSH bucket is consistent with $h_i(0)$.

Proof. There is exactly one path from root to u. So, $u \in E_i$ if in all choices in line 2 of Algorithm 7, the algorithm chooses the correct child. This happens with probability at least $\left(\frac{1}{m}\right)^J \left(\frac{1}{100\left(\frac{1}{\mu}\right)^\alpha}\right)$. To be more clear, with probability $\left(\frac{1}{100\left(\frac{1}{\mu}\right)^\alpha}\right)$ the correct child of the root is chosen, and the other term correspond to the success probability in J steps. □

Now, we have the following:

$$\begin{aligned}
\sum_{i\in[M]} \mathbb{E}\left[\sum_{v\in E_i} |v.P| \;|\mathcal{T}\right] &= \sum_{i\in[M]} \mathbb{E}\left[\sum_{u\in V_i} \mathbb{I}\{u \in E_i\}|u.P| \;|\mathcal{T}\right] \\
&= \sum_{i\in[M]} \sum_{u\in V_i} \Pr[u \in E_i|\mathcal{T}] \cdot |u.P| \\
&\geq \left(\frac{1}{m}\right)^J \sum_{i\in[M]} \sum_{u\in V_i} |u.P| \\
&= \left(\frac{1}{m}\right)^J \sum_{u\in V} |u.P| \qquad (35)
\end{aligned}$$

where expectations are over the random choices of line 2 of Algorithm 7.

Let V' be the leaves with level $\neq J$. Note that $\sum_{u\in V} |u.P| + \sum_{u\in V'} |u.P|$ is equal to the number of points that the query examines in the leaves of $\mathcal{T}$. Note that Claim 42 proves that

$$\mathbb{E}_{\mathcal{T}}\left[\sum_{u\in V'} |u.P|\right] \leq \left(\frac{1}{\mu}\right)^{\alpha+\alpha^*+0.0001} \qquad (36)$$

Now, we need to take expectation over the tree $\mathcal{T}$. From now on, the goal is to prove an upper-bound on

$$\mathbb{E}_{\mathcal{T}}\left[\mathbb{E}\left[\sum_{v\in E_i} |v.P| \;|\mathcal{T}\right]\right]$$

where the outer expectation is over the randomness of trees, and the inner expectation is over the randomness of choices in line 2 of Algorithm 7.

For any $\mathcal{T}$, define $W^{(0,\mathcal{T})}$, as the root of T. For all $j \in [J] \cup \{0\}$ let $V^{(j,\mathcal{T})}$ be the nodes in the tree selected by line 2 of Algorithm 7, when $k = j$. Also, for all $j \in [J]$ let $W^{(j,\mathcal{T})}$ be the children of $V^{(j-1,\mathcal{T})}$ which are consistent with $h_i(j)$, .i.e., nodes satisfying the condition in line 6 of Algorithm 7 when $k = j$. We drop superscripts for the tree, when it is clear from the context.

For all $j \in [J] \cup \{0\}$, $A_{y,j}$ denote the number of points at distance y for all $y \geq x + 1.5\Delta$ from the query in $\cup_{u\in V^{(j)}} u.P$. And similarly, for all $j \in [J] \cup \{0\}$ define $B_{y,j}$ as the number of points at distance y from the query in $\cup_{u\in W^{(j)}} u.P$.

Also, let $L = (\ell_j)_{j=1}^J$ be the distances induced by the geometry h_i. Now, define $x'_j :=$ Project$(x+\Delta, \ell_j, r_j)$ and $y'_j =$ Project$(y - \Delta/2, \ell_j, r_j)$. Now, Claim 35 implies that for all $j \in [J]$

$$\mathbb{E}\left[A_{y,j}|B_{y,j}, \mathcal{T}_{<j}\right] \leq p_{y,j} \cdot B_{y,j}, \qquad (37)$$

where the expectation is over the randomness of the tree and the random choice of line 2 of Algorithm 7, and

$$p_{y,j} := \exp_\mu\left(-\frac{4(r_j/x'_j)^2 - 1}{4(r_j/y'_j)^2 - 1} \cdot \frac{1}{T}\right). \qquad (38)$$

On the other hand, since $B_{y,j}$ variables correspond to pseudo-random spheres, using Claim 34 they should satisfy the following:

$$\sum_{y \le c_j - \psi R_j} B_{y,j} \le \frac{\tau}{1-2\tau} \cdot \sum_{y \in (c_j - \psi R_j, c_j + \psi R_j)} B_{y,j}, \tag{39}$$

and

$$\sum_{y \ge c_j + \psi R_j} B_{y,j} \le \frac{\tau}{1-2\tau} \cdot \sum_{y \in (c_j - \psi R_j, c_j + \psi R_j)} B_{y,j}. \tag{40}$$

Also, since $\cup_{u \in V^{(j)}} u.P \subseteq \cup_{u \in W^{(j)}} u.P$, then $B_{y,j} \le A_{y,j-1}$. At this point, define Now, for all $j \in [J]$ define

$$\widetilde{B}_{y,j} := \mathbb{E}\left[B_{y,j}\right]$$

and

$$\widetilde{A}_{y,j} := \widetilde{B}_{y,j} \cdot p_{y,j} \tag{41}$$

and define

$$\widetilde{B}_{y,J+1} := \widetilde{A}_{y,J}. \tag{42}$$

Therefore, if $A := (\widetilde{A}_{y,j})_{j=1}^{J}$ and $B := (\widetilde{B}_{y,j})_{j=1}^{J+1}$ and L is the ordered set of distances induced by the path geometry h_i (see Section 6.1 and Definition 32) and R is the set of radii of the spheres, then we can argue that (L, R, A, B) is a valid execution path by Definition 36. Checking the conditions of Definition 36:

- Initial conditions: They are satisfied by the expectation of sub-sampling (see (27)), i.e.,

$$\sum_{y' \in [0,y] \cup D} \widetilde{A}_{y',0} \le \min\left\{\exp_\mu\left(\frac{y^2 - x^2}{2}\right), \exp_\mu\left(\frac{1-x^2}{2}\right)\right\}$$

  and

$$\sum_{y' \in [0,y] \cup D} \widetilde{B}_{y',0} \le \min\left\{\exp_\mu\left(\frac{y^2 - x^2}{2}\right), \exp_\mu\left(\frac{1-x^2}{2}\right)\right\}.$$

- Truncation conditions: **(2a)** is satisfied since if a point is on the sphere, its distance to the query can be in interval $[x + 1.5\Delta, \ell_j + r_j]$ which is a sub-interval of $[\ell_j - r_j, \ell_j + r_j]$, by the definition of induced distances and setting of parameters. **(2b)** holds, since the number of points in each distance is non-increasing from root to leaf. **(2c)** is satisfied by (39).

- LSH conditions: They are satisfied by (41) and the definition of $p_{y,j}$ in (38).

- Terminal density condition: It holds by (42).

we conclude that (L, R, A, B) is a valid execution path.

Now by Lemma 39 there exists a zero distance monotone execution path (R', A', B') such that $A' = (a'_{y,j})$, $B' = (b'_{y,j})$ and $b'_{y,J+1} = \widetilde{B}_{y,J+1}$. Let $f_{y,j}$'s be defined based on $b'_{y,j}$'s using Definition 41. More specifically, for every integer $i \in \{k_j, \ldots, I\}$ (see Definition 41 for the definition of k_j) define

$$f_{z_i,j} := \log_{1/\mu}\left(\sum_{y \in D \cap [z_{i+1}, z_{i-1})} b'_{y,j}\right) \tag{43}$$

Now, by Claim 55 and our setting of J (see Section 5.2), which ensures that $J > \frac{T}{1-10^{-4}}\mathrm{OPT(LP)}$, for all $y \le z_{j^*-1}$ we have $f_{y,J+1} < 7\delta_z$ for $j^* = k_J + 1$. Now, we need to prove that this implies that $\sum_y \widetilde{A}_{y,J}$ is small:

Claim 45. *If for all $y \le z_{j^*-1}$ we have $f_{y,J+1} < 7\delta_z$ for $j^* = k_J + 1$, then we have the following bound*

$$\sum_y \widetilde{A}_{y,J} \le \exp_\mu\left(7\delta_z + o(1)\right).$$

The proof is deferred to Appendix E.

We just proved that for any fixed $i \in [M]$, $\sum_y \widetilde{A}_{y,J}$ (which bounds the expected number of points at distance $\ge x + 1.5\Delta$ (see (25)) that the query examines in buckets with geometry h_i) is bounded by $\exp_\mu(7\delta_z + o(1))$. Moreover, recall that in this process we only considered points at distance $\ge x + 1.5\Delta$. We should also add the contribution of points at distance $< x + 1.5\Delta$. For this, just recall that after sub-sampling (even without considering any LSH effect on these points) in expectation we have

$$4\left(\frac{1}{\mu}\right)^{\frac{(x+1.5\Delta)^2 - x^2}{2}} \le \left(\frac{1}{\mu}\right)^{10\Delta}. \tag{44}$$

Now, in order to argue that the expected number of points examined by the query is bounded, we need to multiply by M, which results in the following bound

$$M \cdot \Big(\exp_\mu\left(7\delta_z + o(1)\right) + \exp_\mu\left(10\Delta\right)\Big) \tag{45}$$

which by the setting of parameters, combining with (35) and summing with (36), and considering the we call QUERY (Algorithm 6) at most $\left(\frac{1}{\mu}\right)^{4\delta_x + o(1)}$, gives the following bound on the expected number of points scanned by the query

$$\left(\frac{1}{\mu}\right)^{0.173}.$$

□

6.3 Proof of Lemma 31

Before we present the proof of Lemma 31, we need to show another auxiliary claim that helps us establish an upper bound on the expected number of leaves that a query explores, which helps upper bound the work done to reach a leaf (recall that Lemma 43 shows that the expected size of the dataset corresponding to a leaves of $\mathcal{T}$ that the query scans is bounded, so combining these two bounds will give us the final result).

Lemma 46. *For every* $\mathbf{q} \in \mathbb{R}^d$*, every* $x > 0$*, every* $\mu \in (0,1)$ *under Assumption 1, if* $\mathcal{T}$ *is the tree generated by* $\textsc{PreProcess}(P, x, \mu)$*, then the expected number of leaves explored by a query* q *in a call to* $\textsc{Query}(\mathbf{q}, x, \mathcal{T})$ *(Algorithm 6) is bounded by* $\exp_\mu(\alpha^* + \alpha + o(1))$*.*

The proof is given in Appendix E. We will also need the following technical claim, which we also prove in Appendix E.

Claim 47. *For every* $R \geq R_{min}$*, for every* $x' \in (\Delta, R(\sqrt{2} + \gamma))$ *and sufficiently large* η *we have that* $\frac{1}{G(x'/R,\eta)} = \left(\frac{F(\eta)}{G(x'/R,\eta)}\right)^{O(1/\Delta^2)}$.

Proof of Lemma 31: By Lemma 46 the expected number of leaves that the query explores is bounded by

$$\left(\frac{1}{\mu}\right)^{\alpha^* + \alpha + o(1)} \tag{46}$$

The expected size of the dataset that the query scans is bounded by $\left(\frac{1}{\mu}\right)^{0.173}$ with high probability by Lemma 43. Now by an application of Markov's inequality to (46) we have that the query explores at most $\left(\frac{1}{\mu}\right)^{0.173+o(1)}$ leaves with high probability, and hence the total work is bounded by $(\frac{1}{\mu})^{0.173} \cdot n^{o(1)}$, as required. Finally, we bound the work done in line 18 of Algorithm 4. Indeed, recall that $x' < R(\sqrt{2} + \gamma)$ by line 15 of Algorithm 4, and at the same time by Claim 47 we have

$$\frac{1}{G(x'/R,\eta)} = \left(\frac{F(\eta)}{G(x'/R,\eta)}\right)^{O(1/\Delta^2)}.$$

Equipped with this observation, we can now finish the proof. We get using the choice of T in line 16 of Algorithm 4

$$\frac{100}{G(x'/R,\eta)} = 100 \cdot \left(\frac{F(\eta)}{G(x'/R,\eta)}\right)^{O(1/\Delta^2)} = 100 \cdot (1/\mu)^{O(1/(\Delta^2 \cdot T))} = n^{o(1)}$$

by choice of $\Delta = \Omega(1)$ and $T = \sqrt{\log n}$ in line 3 of Algorithm 4. This completes the proof. $\square$

7 Reduction to zero-distance monotone execution paths

In this section, we prove Lemma 39, which proves that for any valid execution path, there exists a zero-distance valid execution path such that the final densities are identical and both have the same length. First, we state the following claims, and then assuming these claims, we prove Lemma 39. Then, we present the proof of these claims.

Claim 48 (Reduction to zero distance paths)**.** *For any* L*,* R*,* A *and* B *such that* (L, R, A, B) *is a valid execution path (see Definition 36), there exists* R' *and* A' *such that* (R', R', A', B) *is a valid execution path for some* A'*.*

Claim 49 (Local improvement towards monotonicity)**.** *For every valid zero-distance execution path* (R, A, B)*, if for some* $i \in [J-1]$ *one has* $r_i \leq r_{i+1}$*, then for* $R' := (r_1, \ldots, r_{i-1}, r_{i+1}, r_{i+1}, \ldots, r_J)$*, there exist* A', B' *such that the path* (R', A', B') *is a valid execution path and* $b'_{y,J+1} = b_{y,J+1}$ *for all* $y \in D$ *(see (25) for the definition of* D*).*

Now, assuming the correctness of Claim 48 and Claim 49 we present the proof of Lemma 39.

Proof of Lemma 39: First, using Claim 48, we find a zero-distance valid execution path (L'', R'', A'', B). Now, we repeat the procedure described in Claim 49 on (L'', R'', A'', B), until it becomes a zero-distance monotone execution path, (R', A', B'), which satisfies the conditions of the lemma. $\square$

Now we present the proof of Claim 48 and Claim 49.

Proof of Claim 48: Let $(\ell_j)_{j=1}^J = L$ and $(r_j)_{j=1}^J = R$. Then $\forall j \in [J]$ we define: [16]

$$r'_j := \sqrt{\frac{\ell_j^2 + r_j^2}{2}} \tag{47}$$

$$\ell'_j := \sqrt{\frac{\ell_j^2 + r_j^2}{2}} \tag{48}$$

and we let $R' := (\ell'_j)_{j=1}^J = (r'_j)_{j=1}^J$. The same as Definition 36 for all $j \in [J]$, we define

$$c'_j := \sqrt{(\ell'_j)^2 + (r'_j)^2} = \sqrt{2} \cdot r'_j$$

which translates to $c'_j = c_j$. First, we need to show that

$$[0, \ell_j + r_j] \subseteq [0, \ell'_j + r'_j].$$

Note that

$$(\ell'_j + r'_j)^2 - (\ell_j + r_j)^2 = 2c_j^2 - c_j^2 - 2\ell_j r_j \geq 0$$

where the last inequality is due to $c_j = \sqrt{r_j^2 + \ell_j^2}$. One can see that since we can set $a'_{y,0} = a_{y,0}$ for all $y \in D$, it suffices to show that for all $j \in [J]$, $x \in [|\ell_j - r_j| - \delta, \ell_j + r_j]$ (see line 20 of Algorithm 6 and Definition 36) and $y \in [\ell_j - r_j, \ell_j + r_j]$ such that $y - \Delta/2 \geq x + \Delta$:

$$\frac{4\left(\frac{r_j}{\textsc{Proj}(x+\Delta, \ell_j, r_j)}\right)^2 - 1}{4\left(\frac{r_j}{\textsc{Proj}(y-\Delta/2, \ell_j, r_j)}\right)^2 - 1} \geq \frac{4(r'_j/(x+\Delta))^2 - 1}{4(r'_j/(y-\Delta/2))^2 - 1} \tag{49}$$

We drop the indices j for ease of notation, and let $\alpha := x + \Delta$ and $\beta := y - \Delta/2$. Note that using the formula for $\textsc{Project}$ (see Lemma 12) have

$$\begin{aligned}
&\frac{4\left(\frac{r}{\textsc{Project}(\alpha, \ell, r)}\right)^2 - 1}{4\left(\frac{r}{\textsc{Project}(\beta, \ell, r)}\right)^2 - 1} \cdot \frac{4(r'/\beta)^2 - 1}{4(r'/\alpha)^2 - 1} \\
&= \frac{4\left(\frac{r^2}{\frac{r}{\ell}(\alpha^2 - (\ell - r)^2)}\right) - 1}{4\left(\frac{r^2}{\frac{r}{\ell}(\beta^2 - (\ell - r)^2)}\right) - 1} \cdot \frac{\frac{4r'^2 - \beta^2}{\beta^2}}{\frac{4r'^2 - \alpha^2}{\alpha^2}} \\
&= \frac{(\ell + r)^2 - \alpha^2}{\alpha^2 - (\ell - r)^2} \cdot \frac{\beta^2 - (\ell - r)^2}{(\ell + r)^2 - \beta^2} \cdot \frac{\alpha^2}{\beta^2} \cdot \frac{4r'^2 - \beta^2}{4r'^2 - \alpha^2}
\end{aligned}$$

[16]We define ℓ'_j's for the convenience of the reader, otherwise it is clear that $\ell'_j = r'_j$.

where in the second transition above we used the fact that

$$4\frac{r^2}{\frac{r}{\ell}\left(\alpha^2-(\ell-r)^2\right)}-1=\frac{4r^2\ell^2-\left(\alpha^2-(\ell-r)^2\right)}{\alpha^2-(\ell-r)^2}=\frac{(\ell+r)^2-\alpha^2}{\alpha^2-(\ell-r)^2}$$

and

$$4\frac{r^2}{\frac{r}{\ell}\left(\beta^2-(\ell-r)^2\right)}-1=\frac{4r\ell-\left(\beta^2-(\ell-r)^2\right)}{\beta^2-(\ell-r)^2}=\frac{(\ell+r)^2-\beta^2}{\beta^2-(\ell-r)^2}.$$

Now, by re-ordering the factors, and the fact that $4r'^2=2(\ell^2+r^2)$ by (47)

$$\begin{aligned}&\frac{(\ell+r)^2-\alpha^2}{\alpha^2-(\ell-r)^2}\cdot\frac{\beta^2-(\ell-r)^2}{(\ell+r)^2-\beta^2}\cdot\frac{\alpha^2}{\beta^2}\cdot\frac{4r'^2-\beta^2}{4r'^2-\alpha^2}\\&=\left(\frac{2(\ell^2+r^2)-\beta^2}{(r+\ell)^2-\beta^2}\cdot\frac{(r+\ell)^2-\alpha^2}{2(\ell^2+r^2)-\alpha^2}\right)\cdot\left(\frac{\alpha^2}{\alpha^2-(r-\ell)^2}\cdot\frac{\beta^2-(r-\ell)^2}{\beta^2}\right)\end{aligned}$$

We bound the two terms above separately. For the first term we have

$$\frac{2(\ell^2+r^2)-\beta^2}{(r+\ell)^2-\beta^2}\cdot\frac{(r+\ell)^2-\alpha^2}{2(\ell^2+r^2)-\alpha^2}=\frac{2(\ell^2+r^2)-\beta^2}{(2(r^2+\ell^2)-\beta^2)-(\ell-r)^2}\cdot\frac{(2(r^2+\ell^2)-\alpha^2)-(\ell-r)^2}{2(\ell^2+r^2)-\alpha^2}\geq 1$$

where the inequality follow since for any $0<d<a\leq b$ one has $\frac{a}{a-d}\frac{b-d}{b}\geq 1$. Set $a=2(r^2+\ell^2)-\beta^2$, $b=2(r^2+\ell^2)-\alpha^2$ and $d=(\ell-r)^2$. Note that, $a\leq b$ since $x+\Delta\leq y-\Delta/2$, and $d<a$ since $y-\Delta/2<\ell_j+r_j$.

Now, we bound the second term

$$\left(\frac{\alpha^2}{\alpha^2-(r-\ell)^2}\cdot\frac{\beta^2-(r-\ell)^2}{\beta^2}\right)\geq 1$$

again by the same argument as above, by setting $d=(r-\ell)^2$, $a=\alpha^2$ and $b=\beta^2$. Again, $a\leq b$ since $x+\Delta\leq y-\Delta/2$, and $d<a$ since $x+\Delta>|r-\ell|$ (since $\Delta>\delta$ by the setting of parameters).

Now, combining these two facts (49) holds. □

Remark 6. One should note that in some cases, the radius of a sphere may decrease when converting it to a zero distance sphere and it means that the size of the band corresponding to orthogonal bands may decrease and this may cause the sphere not being pseudo-random anymore. However, one should note that in our algorithm the radius of the sphere is always $\Theta(1)$, meaning that the radius may change by a constant multiplicative factor, so one can re-scale the size of the orthogonal band in the definition (Definition 36) to cover the previously covered distances.

Proof of Claim 49: First note that for $r'\geq r$, we have

$$\frac{4\left(\frac{r}{x+\Delta}\right)^2-1}{4\left(\frac{r}{y-\Delta/2}\right)^2-1}\geq\frac{4\left(\frac{r'}{x+\Delta}\right)^2-1}{4\left(\frac{r'}{y-\Delta/2}\right)^2-1},\tag{50}$$

since $f(c)=\frac{4\left(\frac{r}{x+\Delta}\right)^2-1}{4\left(\frac{r}{y-\Delta/2}\right)^2-1}$ is a decreasing function in r, assuming $y-\Delta/2>x+\Delta$. For the rest of the proof, let $x':=x+\Delta$ and $y':=y-\Delta/2$.

Defining A' and B'. We now construct the sequence of intermediate densities A' that satisfies the conditions in Definition 36 by modifying the original sequence A on position j (the position where non-monotonicity occurs in the original sequence). Let $a'_{y,i} := a_{y,i}$ and $b'_{y,i} := b_{y,i}$ for all $y \in D$ and $i \in ([J] \cup \{0\}) \setminus \{j\}$. Also, let $b'_{y,J+1} := b_{y,J+1}$ for all $y \in D$. Now, let

$$\forall y \in D: \; a'_{y,j} := b_{y,j+1} \tag{51}$$

and also set

$$b'_{y,j} := a'_{y,j} \cdot \exp_\mu\left(\frac{4(r_{j+1}/x')^2 - 1}{4(r_{j+1}/y')^2 - 1} \cdot \frac{1}{T}\right) = b_{y,j+1} \cdot \exp_\mu\left(\frac{4(r_{j+1}/x')^2 - 1}{4(r_{j+1}/y')^2 - 1} \cdot \frac{1}{T}\right) \tag{52}$$

since $r'_j = r_{j+1}$.

We now prove that our choice of A' and B' above satisfies the conditions of Definition 36, i.e. yields a valid execution path. Initial density condition (condition **(1)**) and the terminal density condition (condition **(4)**) are satisfied since they were satisfied by the original execution path (R, A, B), and we did not modify the path on the first and last coordinates. The LSH condition (condition **(3)**) is also satisfied by (52) and tha fact that the original execution path satisfied it. We now verify condition **(2)**. Condition **(2a)** follows since $r_{j+1} > r_j$.

Verifying condition (2b). One has, using the assumption that (L, R, A, B) is a valid execution path,

$$\begin{aligned}
b'_{y,j} &= b_{y,j+1} \cdot \exp_\mu\left(\frac{4(r_{j+1}/x')^2 - 1}{4(r_{j+1}/y')^2 - 1} \cdot \frac{1}{T}\right) && \text{(by (52))}\\
&\le a_{y,j} \cdot \exp_\mu\left(\frac{4(r_{j+1}/x')^2 - 1}{4(r_{j+1}/y')^2 - 1} \cdot \frac{1}{T}\right) && \text{(property \textbf{(2b)} for } (R, A, B))\\
&\le b_{y,j} \exp_\mu\left(-\frac{4(r_j/x')^2 - 1}{4(r_j/y')^2 - 1} \cdot \frac{1}{T}\right) \cdot \exp_\mu\left(\frac{4(r_{j+1}/x')^2 - 1}{4(r_{j+1}/y')^2 - 1} \cdot \frac{1}{T}\right) && \text{(property \textbf{(3)} for } (R, A, B))\\
&\le b_{y,j} && \text{(by (50) together with } r_j < r_{j+1})\\
&\le a_{y,j-1} && \text{(property \textbf{(2b)} for } (R, A, B))
\end{aligned}$$

Verifying condition (2c). We need to prove

$$\sum_{y \in [0, r_{j+1}(\sqrt{2}-\psi)] \cap D} b'_{y,j} \le \frac{\tau}{1-2\tau} \cdot \sum_{y \in (r_{j+1}(\sqrt{2}-\psi), r_{j+1}(\sqrt{2}+\psi)) \cap D} b'_{y,j} \tag{53}$$

Note that by property **(2c)** we have

$$\sum_{y \in [0, r_{j+1}(\sqrt{2}-\psi)] \cap D} b_{y,j+1} \le \frac{\tau}{1-2\tau} \cdot \sum_{y \in (r_{j+1}(\sqrt{2}-\psi), r_{j+1}(\sqrt{2}+\psi)) \cap D} b_{y,j+1} \tag{54}$$

Also, recall that by (52) we have

$$b'_{y,j} = b_{y,j+1} \cdot \exp_\mu\left(\frac{4(r_{j+1}/x')^2 - 1}{4(r_{j+1}/y')^2 - 1} \cdot \frac{1}{T}\right)$$

Now, combing the fact that $\exp_\mu\left(\frac{4(r_{j+1}/x')^2-1}{4(r_{j+1}/y')^2-1} \cdot \frac{1}{T}\right)$ is increasing in y' with (54), proves (53).

We have thus shown that (R', A', B') is a valid execution path. Note that $b'_{y,J+1} = b_{y,J+1}$ for all $y \in D$ by definition of b', as required. $\square$

8 Feasible LP solutions based on valid execution paths

First, we state the main result of this section informally below. We refer the reader to Claim 55 for the formal version of this claim.

Claim 50. *(Informal) If the length of a valid execution path is large enough, then the terminal densities must be small.*

We prove this claim, by arguing that if the terminal densities are not small then there exists a feasible solution to the LP. However, the feasible solution that we construct, has a cost larger than the optimal solution of the LP, which is a contradiction. This implies that we cannot have large terminal densities.

We use Definition 40, Definition 41 and the corresponding notations in the rest of this section. At this point, one should recall that the definition of valid execution paths (Definition 36) is over the continuous densities. Now, we need to present a similar notion for *discretized log-densities.*

Claim 51 (Discretized execution path)**.** *If the $f_{z_i,j}$ variables are defined as per* (32) *(based on a zero distance monotone execution path (R, A, B), with $J = |R|$) then*

(1) Initial densities: *For any integer i: $f_{z_i,1} \le \min\left\{\frac{\cdot z_i^2 - x^2}{2} + 3\delta_z, 1 - \frac{x^2}{2}\right\}$.*

(2) Truncation: *for any $j \in [J]$ and $i \in \{k_j + 1, \dots, I\}$ one has $f_{z_i,j} \le f_{z_{k_j},j} + \log_{1/\mu} \frac{2-2\tau}{1-2\tau}$.*

(3) Locality Sensitive Hashing: *for any $j \in [J]$ and any integer $i \in \{k_j, \dots, I\}$ one has $f_{z_i,j+1} \le f_{z_i,j} - \frac{2(z_{k_j}/x)^2-1}{2(z_{k_j}/z_i)^2-1} \cdot \frac{1}{T} \cdot (1 - 10^{-4})$.*

Proof. For the purposes of the proof it is convenient to introduce an auxiliary definition. For every $j \in [J]$ and every integer $i \in \{k_j, \dots, I\}$ define

$$\widetilde{a}_{z_i,j} := \sum_{y \in D_x \cap [z_{i+1}, z_{i-1})} a_{y,j}. \tag{55}$$

and

$$\widetilde{b}_{z_i,j} := \sum_{y \in D_x \cap [z_{i+1}, z_{i-1})} b_{y,j}. \tag{56}$$

Note that with these definitions in place (32) is equivalent to

$$f_{z_i,j} := \log_{1/\mu} \widetilde{b}_{z_i,j}. \tag{57}$$

We also let $D := D_x$, omitting the dependence on x, to simplify notation. We now prove the properties one by one.

(1) Initial densities condition: First, note that by the initial densities condition for the execution path (R, A, B) together with the truncation conditions (Definition 36) one has

$$\sum_{y' \in [0,y] \cap D} b_{y',1} \le \min\left\{\exp_\mu\left(\frac{y^2 - x^2}{2}\right), \exp_\mu\left(\frac{1 - x^2}{2}\right)\right\}.$$

Combining this with (56), we get

$$\begin{aligned}\widetilde{b}_{z_i,1} &= \sum_{y\in D\cap(z_{i+1},z_{i-1})} b_{y,1} \le \sum_{y\in[0,z_{i-1}]\cap D} b_{y,1}\\ &\le \min\left\{\exp_\mu\left(\frac{z_{i-1}^2-x^2}{2}\right), \exp_\mu\left(\frac{1-x^2}{2}\right)\right\}\\ &= \min\left\{\exp_\mu\left(\frac{(1+\delta_z)^2\cdot z_i^2-x^2}{2}\right), \exp_\mu\left(1-\frac{x^2}{2}\right)\right\}\\ &\le \min\left\{\exp_\mu\left(\frac{\cdot z_i^2-x^2}{2}+3\delta_z\right), \exp_\mu\left(1-\frac{x^2}{2}\right)\right\},\end{aligned}$$

where we used the definition of the grid Z, the fact that $\mu = o(1)$ and that for $z_i \ge \sqrt{2}$ the second term is the minimum term.

(2) Truncation conditions (effect of PseudoRandomify): We have, using (56),

$$\begin{aligned}\sum_{i=k_j+1}^{I} \widetilde{b}_{z_i,j} &\le 2\sum_{y\in D\cap(0,z_{k_j+1})} b_{y,j} + \sum_{y\in D\cap(z_{k_j+1},z_{k_j})} b_{y,j}\\ &\le 2\sum_{y\in D\cap(0,z_{k_j})} b_{y,j}\\ &\le 2\sum_{y\in D\cap(0,r_j(\sqrt{2}+\psi))} b_{y,j}.\end{aligned} \tag{58}$$

The last transition uses the fact that by definition of k_j (see (31)) we have $r_j\cdot(\sqrt{2}+\psi) \in [z_{k_j}, z_{k_j-1})$, and in particular, $r_j\cdot(\sqrt{2}+\psi) \ge z_{k_j}$.

We now note that since $10\psi \le \delta_z = 10^{-6}$ by assumption of the claim and $z_{k_j+1} = (1+\delta_z)^{-1}z_{k_j}$, we further have

$$(r_j(\sqrt{2}-\psi), r_j(\sqrt{2}+\psi)) \subset [z_{k_j+1}, z_{k_j-1})$$

which implies

$$\sum_{y\in D\cap(r_j(\sqrt{2}-\psi),r_j(\sqrt{2}+\psi))} b_{y,j} \le \widetilde{b}_{z_{k_j},j}. \tag{59}$$

At the same time, since (R, A, B) was a valid execution path, then by property **(2c)** in Definition 36, we have

$$\begin{aligned}\sum_{y\in D\cap(0,r_j(\sqrt{2}+\psi))} b_{y,j} &\le \left(1+\frac{\tau}{1-2\tau}\right)\sum_{y\in D\cap(r_j(\sqrt{2}-\psi),r_j(\sqrt{2}+\psi))} b_{y,j}\\ &= \frac{1-\tau}{1-2\tau}\sum_{y\in D\cap(r_j(\sqrt{2}-\psi),r_j(\sqrt{2}+\psi))} b_{y,j}.\end{aligned}$$

Substituting the bound above into (58) and using (59) yields

$$\sum_{i=k_j+1}^{I} \widetilde{b}_{z_i,j} \le \frac{2-2\tau}{1-2\tau}\cdot\widetilde{b}_{z_{k_j},j},$$

establishing the claim.

(3) LSH conditions: For all $j \in [J]$ let $c_j := \sqrt{2} r_j$. One can think of c_j as the distance from a query on the surface of the the j-th sphere in the execution path to a 'typical' point on the sphere. Note that (31) defines a rounding of c_j's points on the grid D. Specifically, c_j is rounded to z_{k_j}'s.

Claim 52. *Let $x' := x + \Delta$ and let $y \in (z_{i+1}, z_{i-1}]$, if $y' = y - \Delta/2$, and $i \in \{k_j, \dots, I\}$ then we have the following claim.*

$$-\frac{4\left(\frac{r_j}{x'}\right)^2 - 1}{4\left(\frac{r_j}{y'}\right)^2 - 1} \le -\frac{2\left(\frac{z_{k_j}}{x}\right)^2 - 1}{2\left(\frac{z_{k_j}}{z_i}\right)^2 - 1}(1 - 10^{-4})$$

We prove this claim in Appendix F.

By property **(3)** in Definition 36, one has

$$a_{y,j} \le b_{y,j} \cdot \exp_\mu\left(-\frac{2(c_j/x')^2 - 1}{2(c_j/y')^2 - 1} \cdot \frac{1}{T}\right). \tag{60}$$

Thus, for all $i \in \{k_j, \dots, I\}$ we have

$$\begin{aligned}
\widetilde{b}_{z_i,j+1} &= \sum_{y \in D \cap (z_{i+1}, z_{i-1})} b_{y,j+1} && \text{By (56)}\\
&\le \sum_{y \in D \cap (z_{i+1}, z_{i-1})} a_{y,j} && \text{Property } \textbf{(2b)} \text{ for } (R, A, B)\\
&\le \sum_{y \in D \cap (z_{i+1}, z_{i-1})} b_{y,j} \cdot \exp_\mu\left(-\frac{2(c_j/x')^2 - 1}{2(c_j/y')^2 - 1} \cdot \frac{1}{T}\right) && \text{By (60)}\\
&\le \widetilde{b}_{z_i,j} \cdot \exp_\mu\left(-\frac{2(z_{k_j}/x)^2 - 1}{2(z_{k_j}/z_i)^2 - 1} \cdot \frac{1}{T} \cdot (1 - 10^{-4})\right) && \text{By (56) and Claim 52}
\end{aligned}$$

This completes the proof of **(3)**. □

8.1 Construction of a feasible solution

In this section, we construct a feasible solution to the LP, i.e., $g_{y,j}$'s and α_j's, based on the execution path that we are considering. Later, we show the relation between the cost of this solution and the length of the execution path.

First, letting $J = |R|$, recall that $R = (r_j)_{j=1}^J$. Then, for all $s \in [J]$ define $c_s := \sqrt{2} \cdot r_s$ and let $c_0 = +\infty$ for convenience. Let $\widetilde{T}$ be such that

$$\frac{1}{T} \cdot (1 - 10^{-4}) = \frac{1}{\widetilde{T}}. \tag{61}$$

Let $x' = x + \Delta$. We classify steps $s = 1, \dots, J$ into three types:

- We say that a step s is **stationary** if $c_s = c_{s-1}$ (this corresponds to the algorithm performing multiple rounds of hashing on the same sphere).
- Otherwise we call step s **minor**, if $\frac{c_s}{c_{s-1}} \ge 1 - \frac{1}{\sqrt{T}}$,
- and call step s **major**, otherwise.

Let $R = R_{stat} \cup R_m \cup R_M$ denote the partition of R into stationary, minor and major steps. Let $j_1, \ldots, j_{|R_m \cup R_M|}$ be such that $z_{j_1} > z_{j_2} > \ldots > z_{j_{|R_m \cup R_M|}}$ are exactly the c_s values corresponding to non-stationary steps, in decreasing order.

Note that by Lemma 28 and parameter settings in the algorithm, $c_1 \le R_{max}\sqrt{2} = O(1)$. Since the grid D (see (25)) contains only elements at least as large as $x + 1.5\Delta$, and if we let x to be lower bounded by an absolute constant we have $|R_M| = O(\sqrt{T})$. The reason is that by the definition above, for any major step s, we have

$$\frac{c_s}{c_{s-1}} < 1 - \frac{1}{\sqrt{T}}.$$

We define the feasible solution $g_{y,j}$'s and α_j's to $\mathrm{LP}(x, j^*)$ as defined in (33) without the **non-empty range constraint**. We construct feasible $g_{y,j}$ and α_j by induction on $j = 1, \ldots, j^*$. It will be important that the constructed solutions for $g_{y,j}$'s are non-decreasing in y for $y \in [0, z_j]$ for every $j = 1, \ldots, j^*$.

For the rest of the section, whenever we are working with discrete functions and it is clear from the context, we drop the condition $y \in Z_x$.

On the other hand, it is more convenient to work with the following formulation of the LP constraints, since we construct the solution in an inductive way.

$$\boxed{\begin{aligned} &\forall y : g_{y,1} \le \min\left\{\frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\} \\ &\text{for all } j < j^*, y < z_j : \\ &g_{y,j+1} \le \min\left\{ g_{y,j} - \frac{2\left(z_j/x\right)^2 - 1}{2\left(z_j/y\right)^2 - 1} \cdot \alpha_j,\ g_{z_{j+1},j} - \frac{2\left(z_j/x\right)^2 - 1}{2(z_j/z_{j+1})^2 - 1} \cdot \alpha_j \right\} \end{aligned}}$$

Base: For all $j \in [j_1]$ such that $y \le z_j$, we set

$$g_{y,j} := \min\left\{\frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\}. \tag{62}$$

We let $\alpha_j := 0$ for all $j \in [j_1 - 1]$ so that **spherical LSH** constraints of the LP in (33) are satisfied for $j \in [j_1 - 1]$. That is, we don't have any progress using spherical LSH, since $\alpha_j = 0$ for all $j \in [j_1 - 1]$. The **truncation** constraints of the LP in (33) are satisfied since the rhs of (62) is non-decreasing in y.

Inductive step $j_i \to j_i + 1, \ldots, j_{i+1}$: We let $a := j_i$ and $b := j_{i+1}$ to simplify notation. Let s be the first step on sphere z_a, i.e., $c_s = z_a$ and $c_{s-1} \ne z_a$. Also, let N be the number of steps that we stay on sphere z_a, i.e.,

$$c_s = c_{s+1} = \ldots = c_{s+N-1} = z_a \text{ and } c_{s+N} \ne z_a.$$

Note that steps $s+1, \ldots, s+N-1$ are stationary as per our definitions.

It is convenient to define a sequence of auxiliary variables in order to handle the sequence of $N - 1$ stationary steps (note that $N - 1$ could be zero).

Upper bounds $h_y^{(q)}, q = 0, \ldots, N$, on density after (possible) stationary steps:

For all $y \le z_a$: (Starting density)

$$h_y^{(0)} := g_{y,a} \tag{63}$$

For all $q \in [N-1], y \le z_a$: (Stationary steps)

$$h_y^{(q)} := \min\left\{ h_y^{(q-1)} - \frac{2(z_a/x)^2 - 1}{2(z_a/y)^2 - 1}\frac{1}{\widetilde{T}},\ h_{z_a}^{(q-1)} - \frac{2(z_a/x)^2 - 1}{\widetilde{T}} \right\} \tag{64}$$

For $y \le z_a$: (Final density)

$$h_y^{(N)} := h_y^{(N-1)} - \frac{2(z_a/x)^2 - 1}{2(z_a/y)^2 - 1}\frac{1}{\widetilde{T}}. \tag{65}$$

Equipped with the definitions of h above, we now define g to satisfy the inductive step. First let $\alpha_a := \frac{N}{\widetilde{T}}$ and let $\alpha_{a+s} := 0$ for $s = 1, \ldots, N-1$. Then define for all $y \le z_b$:

$$g_{y,a+1} := \min\left\{ h_y^{(N)},\ h_{z_{a+1}}^{(N)} \right\}$$

and for $j = a+2, \ldots, b$ and all $y \le z_j$ let

$$g_{y,j} := \min\{g_{y,j-1}, g_{z_j,j-1}\}.$$

We note that this implies for all $y \le z_b$

$$g_{y,b} := \min\left\{ \min_{j \in \{a+1,\ldots,b\}} \left\{ h_{z_j}^{(N)} \right\}, h_y^{(N)} \right\}. \tag{66}$$

This finished the inductive step.

Note that in the last step, i.e., when $z_a = z_{j^*-1}$, since we do not have truncation condition for g_{y,j^*}'s, we define

$$\forall y \le z_{j^*-1} :\ g_{y,j^*} = h_y^{(N)}. \tag{67}$$

8.2 Monotonicity claims

Claim 53 (Unique maximum after LSH). *For every integer $t \ge 1$, $x \in (0, \sqrt{2})$ and any sequence $c_1 \ge c_2 \ge \ldots \ge c_t \ge x$, such that*

$$f(y) = \frac{y^2 - x^2}{2} - \sum_{s=1}^{t} \frac{2(c_s/x)^2 - 1}{2(c_s/y)^2 - 1} \cdot \frac{1}{T}$$

satisfies $f(\sqrt{2}c_t) > 0$, the following conditions hold. There exists $y^* \in (x, \sqrt{2}c_t]$ such that the function satisfies $f(y^*) = 0$ is monotone increasing on the interval $[y^*, \eta]$, where η is where the (unique) maximum of f on $(y^*, \sqrt{2}c_t]$ happens.

Proof. We prove that $\frac{\partial^2 f(y)}{\partial y^2}$ is a monotone decreasing function. One should note that

$$\frac{\partial^2 f(y)}{\partial y^2} = 1 - \sum_{s=1}^{t} \frac{4c_s^2(2c_s^2 + 3y^2)}{(2c_s^2 - y^2)^3} \cdot \frac{2(c_s/x)^2 - 1}{T}$$

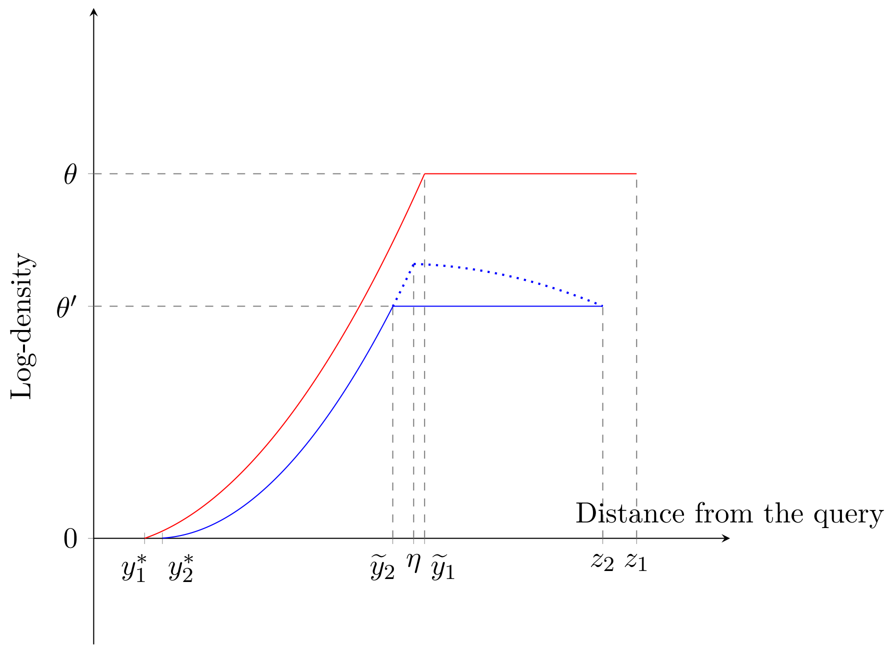


Figure 11: An illustration of proof of Claim 54. The red and blue curves represent functions G_1 and G_2. The dotted part of the blue curve represents $G_1 - \hat{q}$ function for interval $[\widetilde{y}_2, z_2]$, which gets truncated by θ'.

Now, one can see that $\frac{\partial^2 f(y)}{\partial y^2}$ is a monotone decreasing function in y. We then note that $f(x) \leq 0$, and the function $f(y)$ has exactly one maximum on $(y^*, \sqrt{2}c_s)$. □

We will need

Claim 54 (Monotonicity)**.** *For every $i \in [|R|]$ we have*

(a) *If $g_{z_{j_i}, j_i} > 0$ then there exists a $y^* \in (x, \sqrt{2})$ such that $g_{y^*, j_i} \geq 0$, $g_{y, j_i} \leq 0$ for any $y \in Z_x$ such that $y \leq y^*$, and g_{y, j_i} is non-decreasing in y for $y \in [y^*, z_{j_i}]$;*

(b) *If $h_{z_{j_i}}^{(N-1)} > 0$ then there exists a $y^* \in (x, \sqrt{2})$ such that $h_{y^*}^{(N-1)} \geq 0$, $h_y^{(N-1)} \leq 0$ for any $y \in Z_x$ such that $y \leq y^*$ and $h_y^{(N-1)}$ is non-decreasing in y for $y \in [y^*, z_{j_i}]$.*

Proof. Let
$$q(y) := \sum_{i=1}^{t} \frac{2(c_s/x)^2 - 1}{2(c_s/y)^2 - 1} \frac{1}{T},$$
where $c_1 \geq c_2 \geq \ldots \geq c_t \geq z_1 \geq x$ for some $z_1 \geq x$. And let y_1^* be such that $\frac{(y_1^*)^2 - x^2}{2} - q(y_1^*) = 0$ and let $\widetilde{y}_1$ be the smallest value such that $\widetilde{y}_1 \geq y_1^*$ and $\frac{\widetilde{y}_1^2 - x^2}{2} - q(\widetilde{y}_1) = \theta$ for some $\theta \geq 0$. Now define $G_1(y)$ on $[y_1^*, z_1]$, for some $z_1 \geq \widetilde{y}_1$ as follows
$$G_1(y) := \begin{cases} \frac{y^2 - x^2}{2} - q(y) & y \in [y_1^*, \widetilde{y}_1) \\ \theta & y \in [\widetilde{y}_1, z_1] \end{cases} \tag{68}$$

See the red curve in Figure 11.

Also, let $\hat{q}(y) := \frac{2(z_1/x)^2-1}{2(z_1/y)^2-1}\frac{1}{T}$. Let $y_2^* \geq y_1^*$ such that $G_1(y_2^*) - \hat{q}(y_2^*) = 0$. Now, we define $G_2(y)$ for $y \in [y_2^*, z_2]$ as follows:

$$G_2(y) := \min\{G_1(y) - \hat{q}(y), \theta'\}$$

where $\theta' := G_1(z_2) - \hat{q}(z_2)$ and $\theta' \geq 0$ for some $z_2 \leq z_1$. By the definition of y_2^*, function $G_2(y)$ for $y \in [y_2^*, \widetilde{y}_1]$ is in the form of the function in Claim 53 and thus, it has a unique maximum at some $\eta \in [y_2^*, \widetilde{y}_1]$. Also, recall that $G_1(y) = \theta$ for $y \in [\widetilde{y}_1, z_2]$. Also, one should note that since $\hat{q}(y)$ is a monotone increasing function for $y \in (0, \sqrt{2}z_1)$ and hence for $y \in [\widetilde{y}_1, z_2]$, then $\theta' \leq G_2(\widetilde{y}_1)$ and therefore $\theta' \leq G_2(\eta)$. This guarantees that there exist a $\widetilde{y}_2 \in [y_2^*, \eta]$ such that $G_2(\widetilde{y}_2) = \theta'$. The reason is that $G_2(y)$ is a continuous increasing function for $y \in [\widetilde{y}_2, \eta]$. So, we have

$$G_2(y) := \begin{cases} \frac{y^2-x^2}{2} - q'(y) & y \in [y_2^*, \widetilde{y}_2) \\ \theta' & y \in [\widetilde{y}_2, z_2] \end{cases} \tag{69}$$

where, $q'(y) := q(y) - \hat{q}(y)$. See the blue curve in Figure 11. Now, one can see that by a simple inductive argument starting with the initial densities

$$\min\left\{\frac{y^2-x^2}{2}, 1 - \frac{x^2}{2}\right\}$$

which is in the form of (68), the statement of the claim holds. □

8.3 Bounding terminal densities using feasible LP solutions

Claim 55 (Feasible LP solution from an execution path). *If integer J is such that $J > \frac{T}{1-10^{-4}}\mathrm{OPT}(\mathrm{LP})$ then, for all $y \leq z_{j^*-1}$, $f_{y,J+1} < 7\delta_z$ for $j^* = k_J + 1$ (see Definition 41 for the definition of k_J).*

Proof. We prove the claim by contradiction. Suppose that there exists $z \leq z_{j^*-1}$ such that $f_{z,J+1} \geq 7\delta_z$. We define the feasible solution $g_{y,j}$'s and α_j's to LP(x, j^*) as defined in (33). However, the cost of this solution will be more than the optimal cost of the LP, which gives us the contradiction. First, we construct the solution without considering the **non-empty range constraint**. Then, we show that applying $f_{z,J+1} \geq 7\delta_z$ the **non-empty range constraint** is satisfied too.

Let $g_{y,j}$ and α_j be defined as by induction on $j = 1, \ldots, j^*$ as above. We prove by induction on j that if there exists a s such that $c_s = z_j$ and $c_{s-1} \neq z_j$ then for all $y \leq z_j$,

$$f_{y,s} \leq g_{y,j} + X_s, \tag{70}$$

where we define

$$X_s := \sum_{r \in R_m \text{ s.t. } r \leq s} O\left(\frac{1}{\sqrt{T}}\right) \cdot \frac{1}{\widetilde{T}} + \sum_{r \in R_M \text{ s.t. } r \leq s} \frac{O(1)}{\widetilde{T}} + s \cdot \delta + 3\delta_z. \tag{71}$$

for ease of notation, and let $\delta = \log_{1/\mu} \frac{2-2\tau}{1-2\tau} = \Theta\left(\frac{1}{\log 1/\mu}\right)$. One should note that the δ used in this proof is not related to the δ used in the algorithm. Also, let $c_0 = \infty$ and $c_{J+1} = z_{j^*}$, for easing the corner case analysis.

Base: For all $j \in \{1, 2, \ldots, j_1\} = [j_1]$ and all $y \leq z_j$, in (62) we did set

$$g_{y,j} := \min\left\{\frac{y^2-x^2}{2}, 1 - \frac{x^2}{2}\right\}. \tag{72}$$

Also, recall that we let $\alpha_j := 0$ for all $j \in [j_1 - 1]$ (see **base case** in Section 8.1). Now, note that we have $f_{y,1} \le g_{y,1} + 3\delta_z$ for all y by Claim 51, **(1)** combined with the assumption that $\delta_z \le 1$. So, the base holds.

Inductive step $j_i \to j_i + 1, \ldots, j_{i+1}$: We let $a := j_i$ and $b := j_{i+1}$ to simplify notation. Let s be the first step on sphere z_a: $c_s = z_a$ and $c_{s-1} \neq z_a$. Also, let N be the number of steps that we stay on sphere z_a, i.e.,

$$c_s = c_{s+1} = \ldots = c_{s+N-1} = z_a \text{ and } c_{s+N} \neq z_a.$$

Note that steps $s+1, \ldots, s+N-1$ are stationary as per our definitions. We let $t := s+N$ for convenience. By the inductive hypothesis for any $y \le z_a$ we have

$$f_{y,s} \le g_{y,a} + X_s \tag{73}$$

We prove that for any $y \le z_b$

$$f_{y,t} \le g_{y,b} + X_t. \tag{74}$$

Let $h_y^{(q)}, q = 0, \ldots, N$ and $g_{y,j}, j = a, \ldots, b$, be defined as above. We now upper bound $f_{y,s+q}$ in terms of $f_{y,s+(q-1)}$. We have for all $q \in [N-1]$ and $y \le z_b$:

$$\begin{aligned} f_{y,s+q} &\le \min\left\{ f_{y,s+(q-1)} - \frac{2(z_a/x)^2 - 1}{2(z_a/y)^2 - 1}\frac{1}{\widetilde{\widetilde{T}}},\ f_{z_a,s+(q-1)} - \frac{2(z_a/x)^2-1}{\widetilde{\widetilde{T}}} + \delta \right\} \\ &\le \min\left\{ f_{y,s+(q-1)} - \frac{2(z_a/x)^2 - 1}{2(z_a/y)^2 - 1}\frac{1}{\widetilde{\widetilde{T}}},\ f_{z_a,s+(q-1)} - \frac{2(z_a/x)^2-1}{\widetilde{\widetilde{T}}} \right\} + \delta. \end{aligned} \tag{75}$$

where the first transition is by Claim 51. Similarly we have (again by Claim 51)

$$\begin{aligned} f_{y,t} &\le \min\left\{ f_{y,s+(N-1)} - \frac{2(z_a/x)^2 - 1}{2(z_a/y)^2 - 1}\frac{1}{\widetilde{\widetilde{T}}},\ f_{z_{j_b},s+(N-1)} - \frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1}\frac{1}{\widetilde{\widetilde{T}}} + \delta \right\} \\ &\le \min\left\{ f_{y,s+(N-1)} - \frac{2(z_a/x)^2 - 1}{2(z_a/y)^2 - 1}\frac{1}{\widetilde{\widetilde{T}}},\ f_{z_{j_b},s+(N-1)} - \frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1}\frac{1}{\widetilde{\widetilde{T}}} \right\} + \delta \end{aligned} \tag{76}$$

We now note that the recurrence relations (64) and (65) defining $h_y^{(q)}$ are only different from the above by an additive δ term, and the initial condition (63) for $h_y^{(0)}$ is only different from the inductive hypothesis (73) by an additive X_s term. Combining these observations, we get

$$f_{y,t} \le \min\left\{ h_y^{(N)}, h_{z_b}^{(N)} \right\} + X_s + \delta \cdot (t-s). \tag{77}$$

Now, one can see that we have the following upper bound for X_s for any s using the definition of X_s

$$\begin{aligned} X_s &= \sum_{r \in R_m \text{ s.t. } r \le s} O\left(\frac{1}{\sqrt{T}}\right) \cdot \frac{1}{\widetilde{\widetilde{T}}} + \sum_{r \in R_M \text{ s.t. } r \le s} \frac{O(1)}{\widetilde{T}} + s \cdot \delta + 3\delta_z \\ &\le O\left(\frac{1}{\sqrt{T}}\right) + 3\delta_z \end{aligned} \tag{78}$$

since we have at most $O(\sqrt{T})$ major steps, at most $O(T)$ minor steps, and $\delta = \Theta\left(\frac{1}{T^2}\right)$. This implies

$$f_{y,t} \le \min\left\{ h_y^{(N)}, h_{z_b}^{(N)} \right\} + 3\delta_z + O\left(\frac{1}{\sqrt{T}}\right) \le h_y^{(N)} + 4\delta_z. \tag{79}$$

Combining this with the assumption that there exists a $z \le z_{j^*-1}$ such that $f_{z,J+1} \ge 7\delta_z$ we have

$$h_y^{(N)} \ge 0 \text{ for all } y \ge z_{j^*-1}, \tag{80}$$

which we prove below and will be useful whenever we want to invoke Claim 54.

Claim 56. *$\forall y \ge z_{j^*-1}$, we have $h_y^{(N)} \ge 0$.*

Proof. $\forall y \ge z_{j^*}: \; h_y^{(N)} \ge 0$. Assume that there exists a $z \le z_{j^*-1}$ such that $f_{z,J+1} \ge 7\delta_z$. Now, by the fact that $f_{y,j}$'s are monotone in j, and by (79) we have

$$7\delta_z \le f_{z,J+1} \le f_{z,t} \le h_z^{(N)} + 4\delta_z,$$

which implies

$$3\delta_z \le h_z^{(N)}.$$

On the other hand, by (64) we get

$$h_z^{(N-1)} \le h_{z_a}^{(N-1)}.$$

which implies $h_{z_a}^{(N-1)} \ge 3\delta_z \ge 0$. Thus, by Claim 54, $h_y^{(N-1)}$ is non-decreasing in $[z, z_a]$. So, for any $y \in [z, z_a]$,

$$3\delta_z \le h_y^{(N-1)} \tag{81}$$

Also, by (65) we have

$$h_y^{(N)} = h_y^{(N-1)} - \frac{2(z_a/x)^2 - 1}{2(z_a/y)^2 - 1}\frac{1}{\widetilde{T}} = h_y^{(N-1)} - O\left(\frac{1}{T}\right) \tag{82}$$

Combing (81) and (82), we prove that for $y \ge z_{j^*-1}$:

$$h_y^{(N)} \ge 2\delta_z \ge 0.$$

□

The following characterization of $X_t - X_s$ will be useful:

$$X_t - X_s = \delta \cdot (t-s) + \begin{cases} O\left(\frac{1}{\sqrt{T}}\right) \cdot \frac{1}{\widetilde{T}} & \text{if } t \in R_m \\ \frac{O(1)}{\widetilde{T}} & \text{if } t \in R_M. \end{cases} \tag{83}$$

The above follows by (71) since all steps between s and t are stationary.

We now the upper bound the minimum on the rhs in (77). Recall from (66) that

$$g_{y,b} := \min\left\{\min_{j \in \{a+1,\dots,b\}} \left\{h_{z_j}^{(N)}\right\}, h_y^{(N)}\right\}.$$

Let y'' be such that $g_{y,b} = h_{y''}^{(N)}$. We consider two cases, depending on whether $y'' = y$.

Case 1: $y = y''$ (the simple case). In that case we have

$$\begin{aligned}
f_{y,t} &\le \min\left\{h_y^{(N)}, h_b^{(N)}\right\} + X_s + \delta \cdot (t-s) && \text{By (77)}\\
&= h_y^{(N)} + X_s + \delta \cdot (t-s) && \text{Since } y'' = y\\
&= g_{y,b} + X_s + \delta \cdot (t-s) && \text{Combining } y'' = y \text{ and (66)}\\
&\le g_{y,b} + X_t, && \text{By (83)}
\end{aligned}$$

as required.

Case 2: $y \neq y''$ (the main case).

$$\begin{aligned}
f_{y,t} &\le \min\left\{h_y^{(N)}, h_{z_b}^{(N)}\right\} + X_s + \delta \cdot (t-s)\\
&\le h_{z_b}^{(N)} + X_s + \delta \cdot (t-s)\\
&= g_{y,b} + X_s + \delta \cdot (t-s) + (h_{z_b}^{(N)} - g_{y,b}).
\end{aligned} \tag{84}$$

In what follows we show that

$$h_{z_b}^{(N)} - g_{y,b} = \begin{cases} O\left(\frac{1}{\sqrt{T}}\right) \cdot \frac{1}{\widetilde{T}} & \text{if } t \in R_m \\ \frac{O(1)}{\widetilde{T}} & \text{if } t \in R_M, \end{cases} \tag{85}$$

which gives the result once substituted in (84), as per (83).

We now consider two case, depending on whether t is a *minor* or a *major* step. For both steps we use the fact that $y'' \neq y$ implies $y'' \ge z_b$ (this follows by definition of y'' together with (66)).

Minor steps ($t \in R_m$). In this case we have

$$\begin{aligned}
&h_{z_b}^{(N)} - g_{y,b}\\
=&h_{z_b}^{(N)} - h_{y''}^{(N)} && \text{By definition of } y''\\
=&h_{z_b}^{(N-1)} - \frac{2(z_a/x)^2 - 1}{2(z_a/z_b)^2 - 1} \cdot \frac{1}{\widetilde{T}} - h_{y''}^{(N-1)} + \frac{2(z_a/x)^2 - 1}{2(z_a/y'')^2 - 1} \cdot \frac{1}{\widetilde{T}} && \text{By (65)}\\
\le&\frac{2(z_a/x)^2 - 1}{2(z_a/y'')^2 - 1} \cdot \frac{1}{\widetilde{T}} - \frac{2(z_a/x)^2 - 1}{2(z_a/z_b)^2 - 1} \cdot \frac{1}{\widetilde{T}}.
\end{aligned}$$

The last transition used Claim 54, **(b)**, and the fact that $y'' \ge z_b$: we only need to verify the preconditions of Claim 54, which follows by (80) together with the fact that $h_y^{(N-1)} \ge h_y^{(N)}$ for all y.

We now bound the rhs of the equation above by

$$\begin{aligned}
&\frac{2(z_a/x)^2-1}{2(z_a/y'')^2-1}\cdot\frac{1}{\widetilde{T}}-\frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1}\cdot\frac{1}{\widetilde{T}} \\
\leq&\frac{2(z_a/x)^2-1}{\widetilde{T}}\left(1-\frac{1}{2(z_a/z_b)^2-1}\right) && \text{Since } y''\leq z_a \\
=&\frac{2(z_a/x)^2-1}{\widetilde{T}}\left(1-\frac{1}{2(1-1/\sqrt{T})^{-2}-1}\right) && \text{Since this is a \textit{minor} step} \\
=&\frac{2(z_a/x)^2-1}{\widetilde{T}}\cdot O(1/\sqrt{T}) \\
\leq&\frac{2(z_a/\Delta)^2-1}{\widetilde{T}}\cdot O(1/\sqrt{T}) \\
=&\frac{1}{\widetilde{T}}\cdot O\left(\frac{1}{\sqrt{T}}\right), && \text{Since } z_a\leq R_{max}=O(1) \text{ and } \Delta=\Omega(1)
\end{aligned}$$

Major steps ($t\in R_M$). Now, we consider the case when the step is *major*.

$$\begin{aligned}
&h^{(N)}_{z_b}-g_{z_b,b} \\
=&h^{(N)}_{z_b}-h^{(N)}_{y''} && \text{By definition of } y'' \text{ and (66)} \\
=&h^{(N-1)}_{z_b}-\frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1}\cdot\frac{1}{\widetilde{T}}-h^{(N-1)}_{y''}+\frac{2(z_a/x)^2-1}{2(z_a/y'')^2-1}\cdot\frac{1}{\widetilde{T}} && \text{By (65)} \\
\leq&\frac{2(z_a/x)^2-1}{2(z_a/y'')^2-1}\cdot\frac{1}{\widetilde{T}}-\frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1}\cdot\frac{1}{\widetilde{T}}.
\end{aligned}$$

The last transition used Claim 54, **(b)**, and the fact that $y''\geq z_b$: we only need to verify the preconditions of Claim 54, which follows by (80) together with the fact that $h^{(N-1)}_y\geq h^{(N)}_y$ for all y. We now upper bound the rhs of the equation above:

$$\begin{aligned}
&\frac{2(z_a/x)^2-1}{2(z_a/y'')^2-1}\cdot\frac{1}{\widetilde{T}}-\frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1}\cdot\frac{1}{\widetilde{T}} \\
\leq&\frac{2(z_a/x)^2-1}{\widetilde{T}}\left(1-\frac{1}{2(z_a/z_b)^2-1}\right) \\
\leq&\frac{2(z_a/x)^2-1}{\widetilde{T}} \\
\leq&\frac{2(z_a/\Delta)^2-1}{\widetilde{T}} \\
=&\frac{O(1)}{\widetilde{T}} && \text{Since } z_a\leq R_{max}=O(1) \text{ and } \Delta=\Omega(1)
\end{aligned}$$

This completes the inductive claim and establishes (70) for all $j=1,\ldots,j^*$.

The only thing we need to verify is that the solution that we presented, satisfies the **non-empty range constraint**. For the sake of this proof, let us define $g_{z_{j^*-1},j^*}$ as follows:

$$g_{z_{j^*-1},j^*}:=g_{z_{j^*-1},j^*-1}-\frac{2\left(z_{j^*-1}/x\right)^2-1}{2\left(z_{j^*-1}/z_{j^*-1}\right)^2-1}\cdot\alpha_{j^*-1}=g_{z_j,j}-\frac{2\left(z_{j^*-1}/x\right)^2-1}{1}\cdot\alpha_{j^*-1} \tag{86}$$

If s is such that $c_s = z_{j^*-1}$ and $c_{s-1} \neq z_{j^*-1}$, then by the discussion above

$$f_{y,s} \leq g_{y,j^*-1} + X_s. \tag{87}$$

and more specifically, when $y = z$ by the assumption we have

$$7\delta_z \leq f_{z,J+1} \leq g_{z,j^*} + O\left(\frac{1}{\sqrt{T}}\right) + 3\delta_z \qquad \text{By (78)}$$

which implies

$$g_{z,j^*} \geq 3\delta_z. \tag{88}$$

Now, we prove that $g_{z_{j^*},j^*} \geq 0$. The same as the discussion above, if we took N steps on sphere z_{j^*-1} then by (67) we have

$$g_{z,j^*} = h_z^{(N)},$$

where h's are the auxiliary variables defined for sphere z_{j^*-1}. Now, we also have

$$\begin{aligned} g_{z,j^*} = h_z^{(N)} &\leq h_z^{(N-1)} && \text{By (65)} \\ &\leq h_{z_{j^*-1}}^{(N-1)} && \text{By (64)} \end{aligned}$$

On the other hand, we have

$$h_{z_{j^*-1}}^{(N)} = h_{z_{j^*-1}}^{(N-1)} - \frac{2(z_{j^*-1}/x)^2 - 1}{1}\frac{1}{\tilde{T}} = h_{z_{j^*-1}}^{(N-1)} - O\left(\frac{1}{T}\right) \geq h_{z_{j^*-1}}^{(N-1)} - \delta_z$$

Combining these facts we get

$$g_{z_{j^*-1},j^*} = h_{z_{j^*-1}}^{(N)} \geq h_{z_{j^*-1}}^{(N-1)} - \delta_z \geq g_{z,j^*} - \delta_z \geq 2\delta_z$$

where the last inequality is due to (88). Also, by the construction of the solution and the fact that the function $\frac{2(z/x)^2-1}{2(z/y)^2-1}$ is increasing in y, we have

$$\begin{aligned} g_{z_{j^*-1},j^*} - g_{z_{j^*},j^*} &\leq \min\left\{\frac{z_{j^*-1}^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\} - \min\left\{\frac{z_{j^*}^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\} \\ &\leq \frac{(\sqrt{2})^2 - \left((1-\delta_z)\sqrt{2}\right)^2}{2} \leq 2\delta_z \end{aligned}$$

which implies that $g_{z_{j^*},j^*} \geq 0$ (the non-empty range constraint in the LP (33)).

Now, recalling the values of α_j's, one can see the cost of this solution of LP is equal to $\frac{J(1-10^{-4})}{T}$, which is greater than the optimal solution for the LP, which is a contradiction. So, the claim holds. □

It is important to note that Claim 55 is not universally true for any shift-invariant kernel, even under natural monotonicity assumptions. An example is presented in Section 2 in Figure 4. We show, however, that our linear programming formulation is indeed a tight relaxation for a wide class of kernels that includes the Gaussian kernel, the exponential kernel as well as any log-convex kernel.

9 Upper bounding LP value

The main result of this section is a proof of Lemma 56 below:

Lemma 57. *For every $x \geq 0, y \geq x$ the value of the LP in (33) (restated below as (89)) is upper bounded by* 0.1718*. Furthermore, for every $x \in (0, \sqrt{2})$ and $y \geq \sqrt{2}$ the LP value is bounded by $3 - 2\sqrt{2} < 0.1718$, and for every $x \in (0, \sqrt{2})$ and every $y \in [x, \sqrt{2}]$ the LP value is bounded by $\frac{x^2}{2}\left(1 - \frac{x^2}{2}\right)$.*

The main result of this section is an upper bound on the value of the LP (89) below. We first derive a dual formulation, then exhibit a feasible dual solution and then verify numerically that the value of dual is bounded by 0.172 for all values of the input parameters x and y^*.

Fix $x \in [0, \sqrt{2}]$. Let $z_1 > z_2 > \ldots > z_I$, denote the distances on the grid, and we define $Z := \{z_1, z_2, \ldots, z_I\}$, we will consider I linear programs, enumerating over all $j^* \in [I]$ such that $z_{j^*} \geq x$.

$$\max_{\alpha \geq 0} \sum_{j=1}^{j^*-1} \alpha_j \tag{89}$$

$$\begin{aligned}
&\text{such that :} \\
&\forall y \in Z : g_{y,1} \leq \min\left\{\frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\} && (r_{y,0}) \quad \text{Density constraints} \\
&\forall j \in [j^*-1], \forall y \in Z \text{ s.t. } y < z_j : \; g_{y,j} \leq g_{z_j,j} && (q_{y,j}) \quad \text{Truncation} \\
&g_{y,j+1} \leq g_{y,j} - \frac{2\left(z_j/x\right)^2 - 1}{2\left(z_j/y\right)^2 - 1} \cdot \alpha_j && (r_{y,j}) \quad \text{Spherical LSH} \\
&g_{z_{j^*},j^*} \geq 0 && (\eta)
\end{aligned}$$

The dual of (89) is

$$\min \sum_{y \in Z} \left\{\frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\} r_{y,0} \tag{90}$$

$$\begin{aligned}
&\text{such that :} \\
&\forall j \in [j^*-1], y \in Z, y < z_j : r_{y,j-1} - r_{y,j} + q_{y,j} = 0 && (g_{y,j}) \quad \text{Mass transportation} \\
&\forall j \in [j^*-1] : r_{z_j,j-1} - \sum_{x \in Z, x < z_j} q_{x,j} = 0 && (g_{z_j,j}) \quad \text{Max tracking} \\
&\forall y \in Z, y < z_{j^*} : r_{y,j^*-1} = 0 && (g_{y,j^*}) \quad \text{Sink} \\
&-\eta + r_{z_{j^*},j^*-1} = 0 && (g_{z_{j^*},j^*}) \quad \text{Terminal flow} \\
&j \in [j^*-1] : \sum_{y \in Z : y < z_j} \frac{2\left(z_j/x\right)^2 - 1}{2\left(z_j/y\right)^2 - 1} r_{y,j} \geq 1 && (\alpha_j) \\
&r_{y,j}, q_{y,j} \geq 0 \\
&\eta \geq 0
\end{aligned}$$

We start by exhibiting a simple feasible solution for the dual that reproduces our result from Section 4.

Upper bound of $\frac{x^2}{2}\cdot(1-\frac{x^2}{2})$ for every x. Let $q_{y,j}=0$ for all y,j. Let

$$r_{z_{j^*},j}=\left(\frac{x}{y}\right)^2$$

for all $j=0,1,\ldots,j^*-1$ and let $r_{y,j}=0$ for $y\neq z_{j^*}$ and all y. We let $\eta=r_{z_{j^*},j^*-1}$. We first verify feasibility. We have for every $j=1,\ldots,j^*-1$

$$\begin{aligned}\sum_{y\in Z:y<z_j}\frac{2\left(z_j/x\right)^2-1}{2\left(z_j/y\right)^2-1}r_{y,j}&=\frac{2\left(z_{j^*}/x\right)^2-1}{2\left(z_{j^*}/y\right)^2-1}r_{z_{j^*},j^*-1}\\&\geq\frac{2\left(z_{j^*}/x\right)^2}{2\left(z_{j^*}/y\right)^2}r_{z_{j^*},j^*-1}\\&=\left(\frac{y}{x}\right)^2\cdot\left(\frac{x}{y}\right)^2\\&=1,\end{aligned}$$

where we used the fact that $x\leq y$. We thus have a feasible solution. The value of the solution is

$$\left(\frac{x}{y}\right)^2\cdot\min\left\{\frac{y^2-x^2}{2},1-\frac{x^2}{2}\right\}\leq\left(\frac{x}{y}\right)^2\cdot\min\left\{\frac{y^2-x^2}{2},1-\frac{x^2}{2}\right\}$$

When $y\geq\sqrt{2}$, we get, $\left(\frac{x}{y}\right)^2\cdot\left(1-\frac{x^2}{2}\right)$, which is maximized at $y=\sqrt{2}$ and gives $\left(\frac{x}{\sqrt{2}}\right)^2\cdot(1-\frac{x^2}{2})$. Similarly, when $y\leq\sqrt{2}$, we get

$$\left(\frac{x}{y}\right)^2\cdot\left(\frac{y^2-x^2}{2}\right)=\frac{x^2}{2}-\frac{x^4}{2y^2},$$

which is again maximized when $y=\sqrt{2}$. Thus, we get that the value of the LP in (89) is bounded by

$$\frac{x^2}{2}\cdot(1-\frac{x^2}{2}).$$

and we obtain the exponent of 0.25. This (almost) recovers the result of Section 4 . In what follows we obtain a stronger bound of 0.1718 on the value of the LP in (89), obtaining our main result on data-dependent KDE.

Upper bound of 0.1718 on LP value for all x. We exhibit a feasible solution for the dual in which for every $j<j^*$

$$\begin{aligned}q_{z_{j+1},j}&>0\\q_{y,j}&=0\text{ for all }y<z_{j+1}\end{aligned}\tag{91}$$

and $q_{y,j^*}=0$ for all $y<z_{j^*}$. We later show numerically that our dual solution is optimal for the Gaussian kernel.

Simplifying (90) under the assumptions from (91), we get

$$
\begin{aligned}
&\min \sum_{y\in Z} \left\{ \frac{y^2-x^2}{2}, 1-\frac{x^2}{2} \right\} r_{y,0} \\
&\text{such that :} \\
&\forall j\in[j^*-1], y\in Z, y<z_{j+1} : r_{y,j-1}-r_{y,j}=0 && (g_{y,j}) \\
&\forall j\in[j^*-1] : r_{z_{j+1},j-1}-r_{z_{j+1},j}+q_{z_{j+1},j}=0 && (g_{y,j}) \\
&\forall j\in[j^*-1] : r_{z_j,j-1}-q_{z_{j+1},j}=0 && (g_{z_j,j}) \\
&\forall y\in Z, y<z_{j^*} : r_{y,j^*-1}=0 && (g_{y,j^*}) \\
&-\eta+r_{z_{j^*},j^*-1}=0 && (g_{z_{j^*},j^*}) \\
&j\in[j^*-1] : \sum_{y\in Z:y<z_j} \frac{2\left(z_j/x\right)^2-1}{2\left(z_j/y\right)^2-1} r_{y,j}\ge 1 && (\alpha_j) \\
&r_{y,j}, q_{y,j}\ge 0 \\
&\eta\ge 0
\end{aligned}
$$

Eliminating the q variables from the above for simplicity, we get, making the inequality for the (α_j) constraints an equality (recall that we only need to exhibit a dual feasible solution),

$$
\begin{aligned}
&\min \sum_{y\in Z} \left\{ \frac{y^2-x^2}{2}, 1-\frac{x^2}{2} \right\} r_{y,0} && (92) \\
&\text{such that :} \\
&\forall j\in[j^*-1], y\in Z, y<z_{j+1} : r_{y,j-1}-r_{y,j}=0 && (g_{y,j}) \\
&\forall j\in[j^*-1] : r_{z_{j+1},j-1}=r_{z_{j+1},j}-r_{z_j,j-1} && (g_{y,j}) \\
&\forall y\in Z, y<z_{j^*} : r_{y,j^*-1}=0 && (g_{y,j^*}) \\
&r_{z_{j^*},j^*-1}=\eta && (g_{z_{j^*},j^*}) \\
&j\in[j^*-1] : \sum_{y\in Z:y<z_j} \frac{2\left(z_j/x\right)^2-1}{2\left(z_j/y\right)^2-1} r_{y,j}=1 && (\alpha_j) \\
&r_{y,j}\ge 0 \\
&\eta\ge 0
\end{aligned}
$$

Defining a dual feasible solution r. We now derive an expression for a feasible solution r. The construction is by **induction**: starting with $j=j^*-1$ as the **base** we define $r_{y,j}$ variables for $y\in Z, y\le z_j$ that satisfy dual feasibility. The **base** is provided by

$$
r_{z_{j^*},j^*-1}=\eta=\left(\frac{2\left(z_{j^*-1}/x\right)^2-1}{2\left(z_{j^*-1}/z_{j^*}\right)^2-1}\right)^{-1}. \tag{93}
$$

Note that this fully defines $r_{y,j}$ for $j=j^*-1$, since $r_{y,j^*-1}=0$ for $y<z_{j^*}$.

We now give the **inductive step:** $j\to j-1$. By the inductive hypothesis the variables $r_{y,j}$ that we defined satisfy the (α_j) constraints in the dual (92), which means:

$$
\sum_{i>j} \frac{2\left(z_j/x\right)^2-1}{2\left(z_j/z_i\right)^2-1} r_{z_i,j}=1. \tag{94}
$$

We will define $r_{y,j-1}$ so that

$$\sum_{i>j-1} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j-1} = 1 \tag{95}$$

and at the same time the $(g_{y,j})$ constraints relating $r_{y,j}$ to $r_{y,j-1}$ are satisfied.

By the first constraint in (92) we have $r_{z_i,j-1} = r_{z_i,j}$ for all $i > j+1$, since $y < z_{j+1}$ is equivalent to $i > j+1$. By the second constraint in (92) we have $r_{z_{j+1},j-1} = r_{z_{j+1},j} - r_{z_j,j-1}$. Putting these two constraints together, we now find $r_{z_j,j-1}$ and therefore $r_{z_{j+1},j-1}$. We rewrite the left hand side of (95) as

$$\begin{aligned}
\sum_{i>j-1} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j-1} &= \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_j\right)^2-1} r_{z_j,j-1} + \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_{j+1}\right)^2-1}\left(r_{z_{j+1},j} - r_{z_j,j-1}\right) \\
&\quad + \sum_{i>j+1} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j} \\
&= \left(\frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_j\right)^2-1} - \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_{j+1}\right)^2-1}\right) r_{z_j,j-1} \\
&\quad + \sum_{i>j} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j}
\end{aligned} \tag{96}$$

Combining this with (95), we thus get that

$$r_{z_j,j-1} = \left(\frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_j\right)^2-1} - \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_{j+1}\right)^2-1}\right)^{-1} \left(1 - \sum_{i>j} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j}\right). \tag{97}$$

We now show that $r_{z_j,j-1} \geq 0$. The first multiplier in the expression above is non-negative since $\frac{2(z_{j-1}/x)^2-1}{2(z_{j-1}/z)^2-1}$ is increasing in z and $z_j \geq z_{j+1}$. For the second multiplier we have

$$\begin{aligned}
1 - \sum_{i>j} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j} &= 1 - \sum_{i>j} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j} \\
&\geq 1 - \sum_{i>j} \frac{2\left(z_j/x\right)^2-1}{2\left(z_j/z_i\right)^2-1} r_{z_i,j} \\
&= 0.
\end{aligned} \tag{98}$$

Here the first transition used (94) (the inductive hypothesis), and the second transition used the fact that the function $\frac{2(z/x)^2-1}{2(z/y)^2-1}$ is non-increasing in z for $x \leq y$. To summarize, we let $r_{z_j,j-1}$ be defined by (97). Also, we let

$$r_{z_{j+1},j-1} = r_{z_{j+1},j} - r_{z_j,j-1} \tag{99}$$

and let $r_{z_i,j-1} = r_{z_i,j}$ for $i > j+1$. We verify numerically that $r_{z_{j+1},j-1} \geq 0$.

Integral equation representation of the dual solution. While we do not use the following in our analysis, it is interesting to note that the dual solution that we propose satisfies an integral equation in the limit as the grid size goes to 0. Let z_j denote a uniform grid with step size $\Delta \to 0$ on the interval $[0, C]$ for some constant $C \geq \sqrt{2}$. We now rewrite (97) as

$$\left(\frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_j\right)^2-1}-\frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_{j+1}\right)^2-1}\right) r_{z_j,j-1}=\left(1-\sum_{i>j}\frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j}\right). \tag{100}$$

Note by the Mean Value Theorem and the fact that the derivative $\left(\frac{2(z_{j-1}/x)^2-1}{2(z_{j-1}/z)^2-1}\right)'_z$ is Lipschitz within $[z_{j+1}, z_j]$ it follows that

$$\left(\frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_j\right)^2-1}-\frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_{j+1}\right)^2-1}\right)=\left.\left(\frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z\right)^2-1}\right)'_z\right|_{z=z_j}\cdot\Delta\cdot(1+O(\Delta)).$$

We thus have that $r_{z_j,j-1}/\Delta$ converges to the solution $g(y)$ to the following integral equation:

$$\left.\left(\frac{2\left(u/x\right)^2-1}{2\left(u/z\right)^2-1}\right)'_z\right|_{z=u}\cdot g(u)=1-\int_{y^*}^{u}\frac{2\left(u/x\right)^2-1}{2\left(u/r\right)^2-1}g(r)dr \tag{101}$$

The initial condition is a point mass at y^*.

Exact solution to the primal when $z_{j^*} \geq \sqrt{2}$. We note that if $z_{j^*} \geq \sqrt{2}$ an optimal solution to the LP (89) is easy to obtain. The reason is that we can simplify the constraints of LP for band z_{j^*} as follows: for all $j \in [j^* - 2]$

$$\begin{aligned} g_{z_{j+2},j+1} &\leq \min\left\{g_{z_{j+2},j}-\frac{2\left(z_j/x\right)^2-1}{2\left(z_j/z_{j+2}\right)^2-1}\cdot\alpha_j,\ g_{z_{j+1},j}-\frac{2\left(z_j/x\right)^2-1}{2(z_j/z_{j+1})^2-1}\cdot\alpha_j\right\} \\ &\leq g_{z_{j+1},j}-\frac{2\left(z_j/x\right)^2-1}{2(z_j/z_{j+1})^2-1}\cdot\alpha_j \end{aligned}$$

and

$$g_{z_{j^*},j^*}\leq g_{z_{j^*},j^*-1}-\frac{2\left(z_{j^*-1}/x\right)^2-1}{2\left(z_{j^*-1}/z_{j^*}\right)^2-1}\cdot\alpha_{j^*-1}.$$

Now, combining these inequalities with the fact that $g_{z_2,1} \leq 1 - \frac{x^2}{2}$, one has

$$\begin{aligned} g_{z_{j^*},j^*} &\leq 1-x^2/2-\sum_{j=1}^{j^*-1}\frac{2\left(z_j/x\right)^2-1}{2\left(z_j/z_{j+1}\right)^2-1}\alpha_j \\ &\leq 1-\frac{x^2}{2}-\frac{1}{1+5\delta_z}\sum_{j=1}^{j^*-1}\left(2\left(z_j/x\right)^2-1\right)\cdot\alpha_j \\ &\leq 1-\frac{x^2}{2}-\frac{1}{1+5\delta_z}\sum_{j=1}^{j^*-1}\left(2\left(z_{j^*}/x\right)^2-1\right)\cdot\alpha_j \\ &\leq 1-\frac{x^2}{2}-\frac{1}{1+5\delta_z}\left(2\left(\sqrt{2}/x\right)^2-1\right)\cdot\sum_{j=1}^{j^*-1}\alpha_j, \end{aligned} \tag{102}$$

where we used the fact that $2\left(z_j/z_{j+1}\right)^2 - 1 = 2(1+\delta_z)^2 - 1 \le 1 + 5\delta_z$ and the function $2\left(z/x\right)^2 - 1$ is increasing in z.

Letting $\gamma := \sum_{j=1}^{j^*-1} \alpha_j$ denote the LP objective we need to maximize γ subject to $1 - x^2/2 - \frac{1}{1+5\delta_z}\left(2\left(\sqrt{2}/x\right)^2 - 1\right) \cdot \gamma \ge 0$ (the nonempty range LP constraint), where $y^* = z_{j^*}$. The solution is

$$\gamma = \left(1 + 5\delta_z\right)\left(1 - x^2/2\right)\left(2\left(\sqrt{2}/x\right)^2 - 1\right)^{-1}.$$

Finally, one has

$$\max_{x \in [0,\sqrt{2}]}\left(\frac{1 - x^2/2}{2(2/x^2) - 1}\right) = 3 - 2\sqrt{2} \approx 0.171573,$$

which is achieved at $x = \sqrt{4 - 2\sqrt{2}}$. It remains to note that this is achievable by letting $\alpha_{j^*-1} = \gamma$ and letting $\alpha_j = 0$ for $j < j^* - 1$, when $z_{j^*} = \sqrt{2}$.

Numerical verification for $x \in [0, \sqrt{2}], y \in [0, \sqrt{2}]$. Implementing this in Matlab and optimizing over x and j^* (with a uniform grid on $[0, \sqrt{2}]$ consisting of $J = 400$ points) yields the exponent of ≈ 0.1716, achieved at $x \approx 1.0842$ and $z_{j^*} \approx \sqrt{2}$. The Matlab code is given below. Then 0.1718 is an upper-bound on the optimal cost of LP. Moreover, for the analysis if we set $\alpha^* = 0.172$ (as in Section 5.2) then $\alpha^*(1 - 10^{-4})$ strictly upper bounds OPT(LP) (this simplifies the notation in other sections).

```
J=400;
vmax=0;
xIdxMax=0;
yIdxMax=0;

for xIdx=5:5:J-5,
    for yIdx=xIdx-5:-5:1,
        z=sqrt(2)*(J-(1:J))/J;
        density=zeros(J);
        for j=1:J,
          %% density for exp(-x^2/2)
          density(j)=min((z(j)^2-z(xIdx)^2)/2, 1-z(xIdx)^2/2);
        end;
        r=zeros(J);
        r(yIdx-1)=((2*(z(yIdx-1)/z(xIdx))^2-1)/(2*(z(yIdx-1)/z(yIdx))^2-1))^(-1);
        for j=yIdx-2:-1:1,
          coeff=zeros(J);
          for i=j:yIdx,
              coeff(i)=(2*(z(j)/z(xIdx))^2-1)/(2*(z(j)/z(i))^2-1);
          end;
          val=0;
          for i=j+1:yIdx-1,
              val=val+coeff(i+1)*r(i);
          end;

          r(j)=(coeff(j+1)-coeff(j+2))^(-1)*(1-val);
          r(j+1)=r(j+1)-r(j);
        end;
        val=0;
        for i=1:J,
          val=val+density(i)*r(i);
        end;
        if vmax<val
          vmax=val;
          xIdxMax=xIdx;
          yIdxMax=yIdx;
        end;
    end;
end;

vmax
xIdxMax
yIdxMax
%%%%%%%%%%%%

Matlab output:

vmax = 0.1716

xIdxMax = 95

yIdxMax = 5
%%%%%%%%%%%%
```

For other densities replace the density assignment above accordingly. For example, for the $\exp(-||x||_2)$ (exponential kernel, scaled by $\sqrt{2}$ for convenience) set

```
          %% density for exp(-|x|/sqrt{2})
          density(j)=min((z(j)-z(xIdx))/sqrt(2), 1-z(xIdx)/sqrt(2));
```

and for the $\exp\left(-\sqrt{||x||_2}\right)$ kernel (scaled to $\sqrt{2}$ for convenience) set

```
          %% density for exp(-(x/\sqrt{2})^{1/2})
          density(j)=min(sqrt(z(j)/sqrt(2))-sqrt(z(xIdx)/sqrt(2)), 1-sqrt(z(xIdx)/sqrt(2)));
```

respectively.

References

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

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

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

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

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

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

[AR15a] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 793–801. ACM, 2015.

[AR15b] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 793–801, New York, NY, USA, 2015. ACM.

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

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

[BIW19a] 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.

[BIW19b] Arturs Backurs, Piotr Indyk, and Tal Wagner. Space and time efficient kernel density estimation in high dimensions. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 15773–15782, 2019.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A Omitted proofs from Section 3

Proof of Lemma 12: Let $x' := ||\mathbf{q}' - \mathbf{p}||$. If we consider the plane containing q, p, o, then q' also belongs to this plane, since this plane contains $\mathbf{q}, o$.[17] Then, without loss of generality we can assume that we are working on $\mathbb{R}^2$, where $o = (0,0), \mathbf{q} = (R_1, 0), \mathbf{q}' = (R_2, 0)$. Let $\mathbf{p} = (\alpha, \beta)$ such that

$$\begin{aligned}
x^2 &= (\alpha - R_1)^2 + \beta^2 = \alpha^2 + \beta^2 - 2\alpha R_1 + R_1^2 \\
R_2^2 &= \alpha^2 + \beta^2 \\
x'^2 &= (\alpha - R_2)^2 + \beta^2 = \alpha^2 + \beta^2 - 2\alpha R_2 + R_2^2
\end{aligned}$$

Therefore, one has

$$x^2 = R_2^2 - 2\alpha R_1 + R_1^2.$$

Thus,

$$\begin{aligned}
x'^2 &= R_2^2 - 2\alpha R_2 + R_2^2 \\
&= 2R_2(R_2 - \alpha) \\
&= 2R_2\left(R_2 - \frac{R_2^2 + R_1^2 - x^2}{2R_1}\right) \\
&= \frac{R_2}{R_1}\left(x^2 - (R_1 - R_2)^2\right),
\end{aligned}$$

which proves the claim. One should note that the claim holds for both $R_1 \geq R_2$ and $R_1 < R_2$. $\square$

[17]The cases when $\mathbf{q}' = \mathbf{p}$ or $||\mathbf{p} - \mathbf{q}|| = R_1 + R_2$ are the cases when the plane is not unique, but the reader should note that these cases correspond to $x' = 0, x' = 2R_2$ cases, which are trivial.

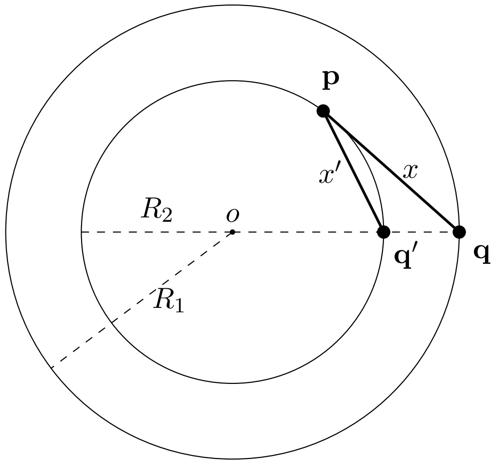


Figure 12: Illustration of $x' = \textsc{Project}(x, R_1, R_2)$

Proof of Claim 14: Now let $\mathbf{q}'$ be the projection of the query on the sphere and let $\mathbf{q}''$ be the antipodal point of $\mathbf{q}'$ on this sphere (see Figure 13). By Definition 13, we have

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}'|| \le r(\sqrt{2} - \gamma)\right\}\right| \le \tau \cdot |P|$$

and

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}''|| \le r(\sqrt{2} - \gamma)\right\}\right| \le \tau \cdot |P|. \tag{103}$$

On the other hand, in Figure 13 let a, c be points at distances $r(\sqrt{2} - \gamma)$ and $r\sqrt{2}$ respectively from $\mathbf{q}'$ and $\mathbf{d}$ be a point at distance $r(\sqrt{2} - \gamma)$ from $\mathbf{q}''$. Then by Pythagoras theorem, we have

$$||\mathbf{q}' - \mathbf{d}||^2 + |\mathbf{q}'' - \mathbf{d}||^2 = ||\mathbf{q}' - \mathbf{q}''||^2,$$

which implies

$$||\mathbf{q}' - \mathbf{d}|| = r\sqrt{4 - \left(\sqrt{2} - \gamma\right)^2} = r\sqrt{2 - \gamma^2 + 2\sqrt{2}\gamma},$$

since $||\mathbf{q}'' - \mathbf{d}|| = ||\mathbf{q}' - \mathbf{a}||$. Therefore, we have the following

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}''|| \le r\left(\sqrt{2} - \gamma\right)\right\}\right| = \left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}'|| \ge r\left(\sqrt{2 - \gamma^2 + 2\sqrt{2}\gamma}\right)\right\}\right| \le \tau \cdot |P|.$$

So one has

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}'|| \in \left(r\left(\sqrt{2} - \gamma\right), r \cdot \sqrt{2 - \gamma^2 + 2\sqrt{2}\gamma}\right)\right\}\right| \ge (1 - 2\tau) \cdot |P|.$$

On the other hand, we have

$$\left(r\left(\sqrt{2} - \gamma\right), r \cdot \sqrt{2 - \gamma^2 + 2\sqrt{2}\gamma}\right) \subseteq \left(r\left(\sqrt{2} - \gamma\right), r\left(\sqrt{2} + \gamma\right)\right),$$

and hence

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}'|| \in \left(r\left(\sqrt{2} - \gamma\right), r \cdot \left(\sqrt{2} + \gamma\right)\right)\right\}\right| \ge (1 - 2\tau) \cdot |P|.$$

which proves the second part of the claim. Now, using (103) we have

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}'|| \le r(\sqrt{2} - \gamma)\right\}\right| \le \frac{\tau}{1 - 2\tau} \cdot \left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}'|| \in \left(r\left(\sqrt{2} - \gamma\right), r \cdot \left(\sqrt{2} + \gamma\right)\right)\right\}\right|.$$

which proves the first part of the claim. $\square$

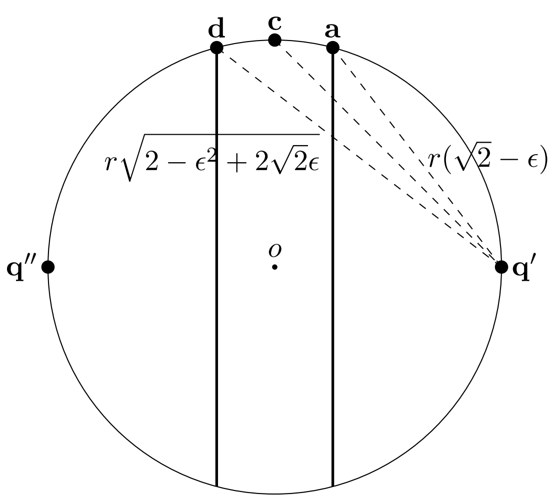


Figure 13: ϵ-neighborhood of orthogonal points in a sphere of radius r

B Pseudo-random data sets via Ball carving

In this section we provide a simple self-contained proof of the claim that one can efficiently (near-linear time) detect and remove a dense ball on the sphere when it exists that avoids invoking VC-dimension arguments as in [**?**].

Lemma 58. *There is a randomized procedure* $\textsc{Certify}_{\epsilon,\tau,\delta}(P)$ *that given a set* $P \subset \mathcal{S}^{d-1}$ *and parameters* $\epsilon, \tau, \delta \in (0, \frac{1}{3})$*, runs in time* $O(\frac{d}{\epsilon^2\tau}\log(2n/\delta)\cdot n)$ *and with probability* $1-\delta$

1. *either returns a point* $p^* \in P$ *such that* $|B(p^*, \sqrt{2(1-\epsilon^2)}) \cap P| \geq \Omega(\epsilon^2)\cdot\tau|P|$
2. *or certifies that the set* P *is* (ϵ', τ)-pseudo-random *with* $\epsilon' = \sqrt{2}(1-\sqrt{1-2\epsilon}) = \Theta(\epsilon)$.

Algorithm 8 $\textsc{Certify}_{\epsilon,\tau,\delta}(P)$

```
1: Input: parameters $\epsilon, \tau, \delta \in (0, \frac{1}{3})$ and $P \subset \mathcal{S}^{d-1}$.
2: $\zeta \leftarrow \frac{1}{4}$, $m \leftarrow \lceil \frac{48}{\epsilon^2\tau}\log(2n/\delta)\rceil$
3: $Q_m \leftarrow \{m$ uniform random points with replacement from $P\}$          ▷ Sub-sampling
4: $p^* \leftarrow \arg\max_{p\in P}\{|B(p, \sqrt{2(1-\epsilon^2)}) \cap Q_m|\}$
5: if $|B(p^*, \sqrt{2(1-\epsilon^2)}) \cap Q_m| \geq (1-\zeta)(3\epsilon^2)\tau\cdot m$ then
6:     return $p^* \in P$          ▷ Center for a "dense ball" is found.
7: else
8:     return $\perp$.          ▷ Set $P$ is $(\Theta(\epsilon), \tau)$-pseudo random
```

The partitioning procedure is based on the following lemma adapted from [**?**] showing that in any set that contains a dense ball on the unit-sphere, one can find a point in the dataset that captures a large fraction of the points in the dense ball.

Lemma 59 (Certificate)**.** *Let* $S' \subset \mathcal{S}^{d-1}$ *and* $x^* \in \mathcal{S}^{d-1}$ *such that for* $\epsilon \in (0, \frac{1}{3})$ *and all* $x \in S'$, $\|x^* - x\| \leq \sqrt{2(1-2\epsilon)}$. *There exists a point* $x_0 \in S'$ *such that*

$$\left|\left\{x \in S' : \|x - x_0\| \leq \sqrt{2(1-\epsilon^2)}\right\}\right| \geq (3\epsilon^2)\cdot|S|'. \tag{104}$$

The contrapositive is that if no balls of a certain radius and density exist with points of the dataset as centers, then the dataset is *pseudo-random* with appropriate constants. We can use this lemma to show that either the data set is pseudo-random or we can always find a dense ball and decrease the size of the remaining data set by a non-trivial factor. The issue that is left to discuss is efficiency of the process.

By repeatedly applying the lemma and by stopping only when $|P| \leq \frac{1}{\tau}$ we can decompose any set on the sphere in at most $T = O(\frac{\log |P|}{\epsilon^2 \tau})$ "dense balls" and a pseudo-random remainder. Let $\chi \in (0,1)$ be a bound on the failure probability and set $\delta = \chi/T$, then this can be done in time

$$O\left(\frac{d}{\epsilon^4 \tau^2} \log\left(2n \log(n)/\epsilon^2 \tau \chi\right) \log(n) \cdot n\right). \tag{105}$$

For any point $p \in P$ let $B_p := B(p, \sqrt{2(1-\epsilon^2)}) \cap P$ and $\mathcal{B} := \{B_p : p \in P \wedge |B_p| \geq 3\epsilon^2 \tau\}$. If Q_m is a random sample of m points from P with replacement, then by Chernoff bounds $\forall B \in \mathcal{B}$ we get $\Pr\left[\left||B \cap Q_m| - \frac{|B|}{|P|}m\right| \geq \zeta \frac{|B|}{|P|}m\right] \leq 2e^{-\frac{\zeta^2}{3}\frac{|B|}{|P|}m}$. Setting $\zeta = \frac{1}{4}$ and $m \geq \frac{48}{\epsilon^2 \tau}\log(2n/\delta)$ and taking union bound over at most $|\mathcal{B}| \leq |P|$ events, we get that with probability at least $1-\chi$ for all $B \in \mathcal{B}$ we have $\frac{3}{4}\frac{|B|}{|P|} \leq \frac{|B \cap Q_m|}{m} \leq \frac{5}{4}\frac{|B|}{|P|}$. Conditional on the above event we have that:

- If there exists $p^* \in P$ such that $|B_{p^*} \cap Q_m| \geq \frac{3}{4}(3\epsilon^2)\tau m$, then $|B_{p^*}| \geq \frac{9}{16}(3\epsilon^2)\tau \cdot |P|$.
- If for all $p \in P$, $|B_p \cap Q_m| < (1-\zeta)(3\epsilon^2)\tau m$ then $|B_p| < 3\epsilon^2 \tau$ for all $p \in P$ and by Lemma 58 the set P is (ϵ', τ)-pseudo random, with $\epsilon' = \sqrt{2}(1 - \sqrt{1-2\epsilon}) \Rightarrow \epsilon = \frac{\epsilon'}{\sqrt{2}}(1 - \frac{\epsilon'}{2\sqrt{2}})$ as $\sqrt{2(1-2\epsilon)} = \sqrt{2(1-\sqrt{2}\epsilon'(1-\frac{\epsilon'}{2\sqrt{2}}))} = \sqrt{(\sqrt{2}-\epsilon')^2} = \sqrt{2}-\epsilon'$

This shows correctness of the Procedure $\textsc{Certify}_{\epsilon,\tau,\delta}$ (Algorithm 8). The overall cost of this procedure is $O(dmn) = O(\frac{d\log(2n/\chi)}{\epsilon^2\tau}n)$ dominated by the cost of finding the ball of radius $\sqrt{2(1-\epsilon^2)}$ centered at one of the points in P with the most number of points in Q_m. □

To prove Lemma 58 we are going to use the following simple lemma.

Proposition 60 ([?]). *For any set $S \subset \mathcal{S}^{d-1}$ such that there exists $c \in \mathcal{S}^{d-1}$, $\|c - x\| \leq r_\epsilon = \sqrt{2(1-2\epsilon)}$ for all $x \in S$,*

$$\frac{1}{|S|^2}\sum_{x,y\in S}\langle x, y\rangle \geq \left(1 - \frac{r_\epsilon^2}{2}\right)^2 = 4\epsilon^2 \tag{106}$$

Proof. Given $x, c \in \mathcal{S}^{d-1}$, $\|x - c\| \leq r_\epsilon \Rightarrow \langle x, c\rangle \geq 1 - \frac{r_\epsilon^2}{2}$. Thus,

$$\sum_{i,j\in S}\langle x_i, x_j\rangle = \|\sum_{x\in S} x\|^2\|c\|^2 \geq \left|\sum_{x\in S}\langle x, c\rangle\right|^2 \geq (1 - \frac{r_\epsilon^2}{2})^2|S|^2 \tag{107}$$

The proof is concluded by substituting $r_\epsilon = \sqrt{2(1-2\epsilon)}$. □

Proof of Lemma 58. We proceed with a proof by contradiction. Assuming that the statement is not true, then

$$\forall y \in S, \ \left|\left\{x \in S : \|x - y\| \leq \sqrt{2(1-\epsilon^2)}\right\}\right| < (3\epsilon^2)\cdot|S| \tag{108}$$

Moreover, $\|x - y\| > \sqrt{2(1-\epsilon^2)} \Rightarrow \langle x, y\rangle < \epsilon^2$. Therefore, we get:

$$\frac{1}{|S|^2}\sum_{x,y\in S}\langle x, y\rangle < \left(1 - 3\epsilon^2\right)\epsilon^2 + 3\epsilon^2 \cdot 1 = (4 - 3\epsilon^2)\epsilon^2 < 4\epsilon^2 \tag{109}$$

Using Proposition 1 and the hypothesis we arrive at a contradiction. □

C Correctness proof of the data-dependent algorithm

In this section we present the outer algorithms for our approach. The procedure is quite routine and similar to Section 4. First, in Algorithm 9 we present the outer procedure of preprocessing phase. In Algorithm 9 for any $x \in \{\delta_x, 2\delta_x, 3\delta_x, \ldots \delta_x \lfloor \frac{\sqrt{2}}{\delta_x} \rfloor\}$, we sample the data set with probability $\min \left\{ \frac{1}{n} \left(\frac{1}{\mu}\right)^{1-x^2/2}, 1 \right\}$, and then using Algorithm 3, we prepare a data structure that after receiving the query, one can recover any point that is present in the sample and has distance $[x - \delta_x, x)$ from the query using Algorithm 10, with probability 0.8 (see Lemma 60 below).

Lemma 61. *Under Assumption 1, if $\mathcal{T} = \textsc{PreProcess}(P, x, \mu)$, then for every point $\mathbf{p} \in P$ such that $\mathbf{p} \in v_0.P$, where v_0 is the root of tree $\mathcal{T}$ and $||\mathbf{q} - \mathbf{p}|| \leq x$, one has $p \in \textsc{Query}(\mathbf{q}, \mathcal{T}, x)$ with probability at least* 0.8*.*

Proof. By Corollary 27, if $\mathcal{H}$ is a (α, x, μ)-AI hash family then for any point $\mathbf{p}$ such that $||\mathbf{p}-\mathbf{q}|| \leq x$

$$\Pr_{h \sim \mathcal{H}}[h(\mathbf{q}) = h(\mathbf{p})] \geq \mu^\alpha$$

Now, noting the number of repetitions of the Andoni-Indyk LSH round, i.e., setting of $K_1 = 100 \left(\frac{1}{\mu}\right)^\alpha$ (see line 6 of Algorithm 3), with probability at least 0.9 we know that there exists a hash bucket that both query and point $\mathbf{p}$ are hashed. Now, we prove by induction on depth of the tree, that if $\mathbf{p}$ belongs to the dataset of root of any tree $\mathcal{T}'$ then $\textsc{Query}(\mathbf{q}, \mathcal{T}', x)$ recovers it with probability at least 0.9.

Base: If the depth of $\mathcal{T}'$ is 1, then $\mathbf{p} \in P_x$ by line 13 of Algorithm 6.
Inductive step: Suppose that $\mathbf{p} \in v.P$ such that v is the root of $\mathcal{T}'$. One should note that v is a pseudo-random sphere. Also, suppose that $\mathbf{q}$ is at distance R_2 from the center of this sphere. Then let $x' := \textsc{Project}(x + \Delta, R_2, R)$ and let x'' be the smallest element in the grid W which is not less than x'. Let $N := \left\lceil \frac{100}{G(x''/R, \eta)} \right\rceil$. Then, by Algorithm 4 we know that v has N children $u_1, u_2, \ldots, u_N$ such that $u_j.x = x''$ for all $j \in [N]$. If $\mathbf{p} \in u_j.P$ for some j then $\mathbf{p}$ will appear in exactly one of the children of u_j, we call this node $u_j(\mathbf{p})$, and if $\mathbf{p} \notin u_j.P$ then $u_j(\mathbf{p}) = \perp$. Let $\mathbf{q}'$ be the projection of $\mathbf{q}$ on the sphere. Now, note that for any $j \in [N]$, if w is a child of u_j then

$$\begin{aligned}
&\Pr\left[\textsc{Query}(\mathbf{q}, \mathcal{T}_{u_j(\mathbf{p})}, x) \text{ will be called and } \mathbf{p} \in u_j(\mathbf{p}).P \text{ and } \mathbf{p} \in \textsc{Query}(\mathbf{q}, \mathcal{T}_{u_j(\mathbf{p})}, x)\right] \\
&= \Pr\left[\left\langle u_j.g, \frac{\mathbf{q} - o}{||\mathbf{q} - o||}\right\rangle \geq \eta \text{ and } \mathbf{p} \in u_j.P \text{ and } \mathbf{p} \in \textsc{Query}(\mathbf{q}, \mathcal{T}_{u_j(\mathbf{p})}, x)\right] \\
&= \Pr\left[\mathbf{p} \in \textsc{Query}(\mathbf{q}, \mathcal{T}_{u_j(\mathbf{p})}, x) \mid \left\langle u_j.g, \frac{\mathbf{q} - o}{||\mathbf{q} - o||}\right\rangle \geq \eta \text{ and } \mathbf{p} \in u_j.P\right] \\
&\quad \cdot \Pr\left[\left\langle u_j.g, \frac{\mathbf{q} - o}{||\mathbf{q} - o||}\right\rangle \geq \eta \text{ and } \mathbf{p} \in u_j.P\right] \\
&\geq 0.9 \cdot \Pr\left[\left\langle u_j.g, \frac{\mathbf{q} - o}{||\mathbf{q} - o||}\right\rangle \geq \eta \text{ and } \mathbf{p} \in u_j.P\right] \\
&\geq 0.9 \cdot G(||\mathbf{q}' - \mathbf{p}.new||/R, \eta) \\
&\geq 0.9 \cdot G\left(\textsc{Project}(x + \delta, R_2, R)/R, \eta\right) \\
&\geq 0.9 \cdot G(x'/R, \eta).
\end{aligned}$$

The first inequality holds by induction. The second inequality holds by Definition 10. The third inequality holds since $||\mathbf{q}' - \mathbf{p}.new|| \leq \textsc{Project}(x + \delta, R_2, R)$. Also, the last inequality holds since

$\Delta \geq \delta$. Then, we have

$$\begin{aligned}&\Pr\left[\textsc{Query}(\mathbf{q}, \mathcal{T}_{u_j(\mathbf{p})}, x) \text{ will not be called or } \mathbf{p} \notin u_j(\mathbf{p}).P \text{ or } \mathbf{p} \notin \textsc{Query}(\mathbf{q}, \mathcal{T}_{u_j(\mathbf{p})}, x)\right] \\ &\leq 1 - 0.9 \cdot G(x'/R, \eta).\end{aligned}$$

Consequently

$$\begin{aligned}\Pr\left[\mathbf{p} \notin \textsc{Query}(\mathbf{q}, \mathcal{T}', x)\right] &\leq (1 - 0.9 \cdot G(x'/R, \eta))^N \\ &\leq 0.1,\end{aligned}$$

where the second inequality uses the following fact that since $x'' \geq x'$, we have

$$N = \left\lceil \frac{100}{G(x''/R, \eta)} \right\rceil \geq \left\lceil \frac{100}{G(x'/R, \eta)} \right\rceil \geq \frac{100}{G(x'/R, \eta)}.$$

So the inductive step goes through, and the statement of the lemma holds. Now, by taking the union bound over the failure probability of the Andoni-Indyk round (which succeeds with high probability) and the failure probability of the data dependent part, we succeed by probability at least $1 - 0.1 - 0.1 = 0.8$. □

For points beyond $\delta_x \lfloor \frac{\sqrt{2}}{\delta_x} \rfloor$ we just sample the data set with rate $\frac{1}{n}$ and just store the sampled set (see line 10 in Algorithm 9). In the query procedure we just scan the sub-sampled data set for recovering these points (see line 10 of Algorithm 10). We repeat this procedure $O(\log n)$ times to boost the success probability to high probability. After recovering the sampled points from the various bands using corresponding data structures, Algorithm 10 applies the standard procedure of importance sampling by calculating Z_μ.

Algorithm 9 PREPROCESS-KDE: P is the data-set

```
1:  procedure PREPROCESS-KDE($P, \mu$)
2:      $\delta_x \leftarrow 10^{-8}$                                        ▷ Step size for grid over $x$
3:      $K_1 \leftarrow \lceil \frac{C \log n}{\epsilon^2} \cdot \mu^{-4\delta_x} \rceil$          ▷ where $C$ is some large enough constant
4:      for $k = 1, 2, \ldots, K_1$ do
5:          for $j = 1, \ldots, \lfloor \sqrt{2}/\delta_x \rfloor$ do     ▷ Uniform grid with step size $\delta_x$ over $[0, \sqrt{2}]$
6:              $x \leftarrow j \cdot \delta_x$
7:              $\widetilde{P}_{k,x} \leftarrow$ sample each point in $P$ with probability $\min\left\{\frac{1}{n}\left(\frac{1}{\mu}\right)^{1-x^2/2}, 1\right\}$
8:              for $i = 1, \ldots, 10 \log n$ do
9:                  $\mathcal{T}_{x,k,i} \leftarrow$ PREPROCESS($\widetilde{P}_{k,x}, x, \mu$)
10:         $\widetilde{P}_k \leftarrow$ sample each point in $P$ with probability $\frac{1}{n}$
11:         Store $\widetilde{P}_k$              ▷ This set will be used to recover points beyond $\delta_x \lfloor \sqrt{2}/\delta_x \rfloor$.
```

Algorithm 10 QUERY-KDE: $\mathbf{q}$ is the query point

```
 1: procedure QUERY-KDE($\mathbf{q}, \mu$)
 2:     $\delta_x \leftarrow 10^{-8}$
 3:     $C_x \leftarrow \lfloor \frac{\sqrt{2}}{\delta_x} \rfloor$
 4:     $K_1 \leftarrow \lceil \frac{C \log n}{\epsilon^2} \cdot \mu^{-4\delta_x} \rceil$          ▷ where $C$ is some large enough constant
 5:     $Z_\mu \leftarrow 0$
 6:     for $k = 1, 2, \ldots, K_1$ do
 7:         $Z_{\mu,k}, \leftarrow 0$
 8:         for $j = 1, \ldots, C_x$ do
 9:             $x \leftarrow j \cdot \delta_x$
10:             $S_x \leftarrow \emptyset$
11:             for $i = 1, \ldots, 10 \log n$ do
12:                 $\mathcal{T}_{x,k,i} \leftarrow$ the data structure prepared by line 9 of Algorithm 9
13:                 $P_{x,k,i} \leftarrow$ QUERY($\mathbf{q}, \mathcal{T}_{x,k,i}, x$)
14:                 for $\mathbf{p} \in P_{x,k,i}$ do
15:                     if $||\mathbf{q} - \mathbf{p}|| \in [x - \delta_x, x)$ then
16:                         $S_x \leftarrow S_x \cup \{\mathbf{p}\}$
17:             for $\mathbf{p} \in S_x$ do
18:                 $\hat{x} \leftarrow ||\mathbf{q} - \mathbf{p}||$
19:                 $Z_{\mu,k} \leftarrow Z_{\mu,k} + \left(\mu^{\frac{\hat{x}^2}{2}}\right)\left(\min\left\{\frac{1}{n}\exp_\mu\left(1 - \frac{x^2}{2}\right), 1\right\}\right)^{-1}$
20:         for $p \in \widetilde{P}_k$ do          ▷ Importance sampling for points beyond $\delta_x C_x$.
21:             if $||\mathbf{q} - \mathbf{p}|| \geq \delta_x C_x$ then
22:                 $\hat{x} \leftarrow ||\mathbf{q} - \mathbf{p}||$
23:                 $Z_{\mu,k} \leftarrow Z_{\mu,k} + n\left(\mu^{\hat{x}^2/2}\right)$
24:         $Z_\mu \leftarrow Z_\mu + \frac{Z_{\mu,k}}{K_1}$
25:     return $Z_\mu$
```

Below, we present the proof of correctness for the outer algorithm, which is very similar to the proof in Section 4.

Claim 62 (Unbiasedness of the estimator)**.** *The estimator $Z_{\mu,k}$ for any $\mu \geq \mu^*$ and any $k \in [K_1]$(see line 6 of Algorithm 10) satisfies the following:*

$$(1 - n^{-9})n\mu^* \leq \mathbb{E}[Z_{\mu,k}] \leq n\mu^*.$$

Proof. First note that if a point $\mathbf{p} \in \widetilde{P}_{k,x}$ for some k and x in line 7 of Algorithm 9 is such that $||\mathbf{q} - \mathbf{p}|| \in [x - \delta_x, x)$, then since we are preparing $10 \log n$ data structures, alongside with Lemma 60 with probability at least $1 - n^{-10}$, $\mathbf{p} \in S_x$ (see line 16 of Algorithm 10). Taking union bound over all the points, with probability $1 - n^{-9}$, any point in distance $[x - \delta_x, x)$ is being sampled with probability $\min\left\{\frac{1}{n}\left(\frac{1}{\mu}\right)^{1-x^2/2}, 1\right\}$ for any $x \in \{\delta_x, 2\delta_x, \ldots\} \cap (0, \sqrt{2})$. We call this event $\mathcal{E}$. Now, since $p_i \cdot (1 - n^{-9}) \leq \Pr[\chi_i = 1] \leq p_i$, we have

$$\mathbb{E}[Z_{\mu,k}] = \mathbb{E}\left[\sum_{i=1}^{n} \chi_i \frac{w_i}{p_i}\right] \geq (1 - n^{-9}) \sum_{i=1}^{n} w_i = (1 - n^{-10})n\mu^*.$$

and

$$\mathbb{E}[Z_{\mu,k}] \leq n\mu^*$$

where p_i is the probability of sampling i'th point, and χ_i is the indicator for the event that i'th point is recovered. □

We proved that our estimator is unbiased[18] for **any choice** of $\mu \geq \mu^*$. Therefore if $\mu \geq 4\mu^*$, by Markov's inequality the estimator outputs a value larger than μ at most with probability $1/4$. We perform $O(\log n)$ independent estimates, and conclude that μ is higher than μ^* if the median of the estimated values is below μ. This estimate is correct with high probability, which suffices to ensure that we find a value of μ that satisfies $\mu/4 < \mu^* \leq \mu$ with high probability by starting with $\mu = n^{-\Theta(1)}$ (since our analysis assumes $\mu^* = n^{-\Theta(1)}$) and repeatedly halving our estimate (the number of times that we need to halve the estimate is $O(\log n)$ assuming that μ is lower bounded by a polynomial in n, an assumption that we make).

Claim 63. *For μ such that $\mu/4 \leq \mu^* \leq \mu$,* QUERY-KDE$(\mathbf{q}, \mu)$ *(Algorithm 10) returns a $(1 \pm \epsilon)$-approximation to μ^*.*

Proof. Also, one should note that $Z_{\mu,k} < n^2 \left(\frac{1}{\mu}\right)$ which implies

$$\mathbb{E}\left[Z_{\mu,k}|\mathcal{E}\right] \cdot \Pr[\mathcal{E}] + n^2 \left(\frac{1}{\mu}\right)(1 - \Pr[\mathcal{E}]) \geq \mathbb{E}[Z_{\mu,k}]$$

So,

$$\mathbb{E}[Z_{\mu,k}]|\mathcal{E}] \geq \left((1 - n^{-10})n\mu^* - \frac{1}{n^2}\frac{1}{\mu}\right) = n\mu^* - o(1/n^5)$$

Also, since $Z_{\mu,k}$ is a non-negative random variable, we have

$$\mathbb{E}\left[Z_{\mu,k}|\mathcal{E}\right] \leq \frac{\mathbb{E}\left[Z_{\mu,k}\right]}{\Pr[\mathcal{E}]} \leq \frac{n\mu^*}{\Pr[\mathcal{E}]} = n\mu^* + o(1/n^5)$$

Also,

$$\begin{aligned}
\mathbb{E}[Z_{\mu,k}^2] &= \mathbb{E}\left[\left(\sum_{i\in[n]} \chi_i \frac{w_i}{p_i}\right)^2\right] \\
&= \sum_{i\neq j} \mathbb{E}\left[\chi_i\chi_j \frac{w_i w_j}{p_i p_j}\right] + \sum_{i\in[n]} \mathbb{E}\left[\chi_i \frac{w_i^2}{p_i^2}\right] \\
&\leq \sum_{i\neq j} w_i w_j + \sum_{i\in[n]} \frac{w_i^2}{p_i}\mathbb{I}[p_i = 1] + \sum_{i\in[n]} \frac{w_i^2}{p_i}\mathbb{I}[p_i \neq 1] \\
&\leq \left(\sum_i w_i\right)^2 + \sum_i w_i^2 + \max_i \left\{\frac{w_i}{p_i}\mathbb{I}[p_i \neq 1]\right\} \sum_{i\in[n]} w_i \\
&\leq 2n^2\mu^{*2} + n^2\left(\frac{1}{\mu}\right)^{-1+4\delta_x} \cdot \mu^* \leq n^2\mu^{2-4\delta_x}
\end{aligned}$$

[18]Up to some small inverse polynomial error.

and

$$\mathbb{E}[Z_{\mu,k}^2|\mathcal{E}] \leq \frac{\mathbb{E}[Z_{\mu,k}^2]}{\Pr[\mathcal{E}]} \leq n^2\mu^{2-4\delta_x} + o(1/n^5)$$

So in order to get a $(1 \pm \epsilon)$-factor approximation to $n\mu$, with high probability, it suffices to repeat the whole process $K_1 = \frac{C \log n}{\epsilon^2} \cdot \mu^{-4\delta_x}$ times (see Algorithm 9 and Algorithm 10), where C is a universal constant. □

D Omitted discussion from Section 6.1

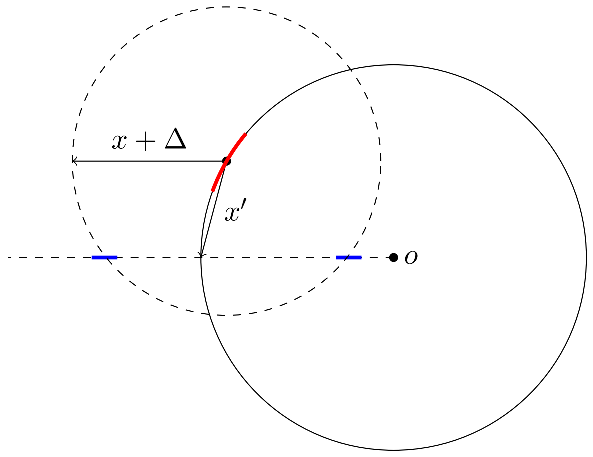


Figure 14: Geometric illustration of equation $x' = \textsc{Project}(x + \Delta, \widetilde{\ell}, r)$ when we have access to an approximation of x' (red arc).

Given query $\mathbf{q}$, and a LSH node v with $(v.x, v.r) = (x'', r)$, we define $b \in \{1, 2\}$ which we use in the definition of the path geometry (Definition 32). Let $\widetilde{\ell} := ||\mathbf{q} - o||$, where o is the center of the sphere. Note that, if we solve $x' = \textsc{Project}(x + \Delta, \ell, r)$ for ℓ we get the following roots for this equation.

$$\ell_1 = \frac{\sqrt{4r^2((x+\Delta)^2 - x'^2) + x'^4} + 2r^2 - x'^2}{2r} \tag{110}$$

and

$$\ell_2 = \frac{-\sqrt{4r^2((x+\Delta)^2 - x'^2) + x'^4} + 2r^2 - x'^2}{2r} \tag{111}$$

Stability of ℓ_1 and ℓ_2 for small changes of x': Let $\widetilde{x}'$ be such that $\widetilde{x}' \in [x', x' + \delta']$. Since $\delta' = o(1)$, $r = \Theta(1)$ and $x = \Theta(1)$, if we solve equation $\widetilde{x}' = \textsc{Project}(x + \Delta, \ell, r)$ for ℓ then we get roots $\widetilde{\ell}_1$ and $\widetilde{\ell}_2$ such that $\widetilde{\ell}_1 \in (\ell_1 - \delta'^{1/3}, \ell_1 + \delta^{1/3})$ and $\widetilde{\ell}_2 = (\ell_2 - \delta'^{1/3}, \ell_2 + \delta'^{1/3})$ for large enough n, since $\delta' = \exp(-(\log \log n)^C)$ (see line 10 of Algorithm 4).

Suppose that we solve $x' = \textsc{Project}(x + \Delta, \ell, r)$ for ℓ for all values of $x' \in [x'' - \delta', x'']$, and let ℓ_1^* be the largest quantity that we get by (110) and let ℓ_2^* be the largest quantity that we get by (111). More formally,

$$\ell_1^* := \max_{x' \in [x''-\delta', x'']} \frac{\sqrt{4r^2((x+\Delta)^2 - x'^2) + x'^4} + 2r^2 - x'^2}{2r}$$

and

$$\ell_2^* := \max_{x' \in [x''-\delta', x'']} \frac{-\sqrt{4r^2((x+\Delta)^2 - x'^2) + x'^4} + 2r^2 - x'^2}{2r}.$$

Now, if $\widetilde{\ell} \in [\ell_1^* - \delta'^{1/3}, \ell_1^*]$ then we let $b = 1$ and $\ell := \ell_1^*$, and otherwise we let $b = 2$ and $\ell := \ell_2^*$. Note that when $b = 2$ it is guaranteed that $\widetilde{\ell} \in [\ell_2^* - \delta'^{1/3}, \ell_2^*]$. One should note that since we define geometry for root to leaf paths, then it is guaranteed that $x' = \textsc{Project}(x + \Delta, \ell, r)$ has at least a real valued solution for ℓ, because otherwise such a root to leaf path is not possible in the tree that the query explores. Also, note that the maximizations above are over the real values, and we ignore the imaginary solutions.

E Omitted claims and proofs from Section 6

Claim 64. *Given query $\mathbf{q}$ and a pseudo random sphere with geometry (x'', r, b) that induces distance ℓ let $\mathbf{q}'$ be the projection of $\mathbf{q}$ on the sphere. In that case, if a point $\mathbf{p}.new$ on the sphere is such that $||\mathbf{q}' - \mathbf{p}.new|| \in (r(\sqrt{2} - \gamma), r(\sqrt{2} + \gamma))$, then*

$$||\mathbf{p} - \mathbf{q}|| \in (c - r\psi, c + r\psi)$$

where $\psi := \gamma^{1/3} + \delta'^{1/4} + \delta^{1/4}$, $c := \sqrt{\ell^2 + r^2}$.

Proof. Since $\mathbf{q}$ and the geometry of the sphere induce distance ℓ, then $||\mathbf{q} - o|| \in [\ell - \delta'^{1/3}, \ell]$. Now, suppose that we move $\mathbf{q}$ in the direction of the vector from o to $\mathbf{q}$ and reach a point $\widetilde{\mathbf{q}}$ such that $||\widetilde{\mathbf{q}} - o|| = \ell$. Then, by assumption

$$\textsc{Project}(||\widetilde{\mathbf{q}} - \mathbf{p}.new||, \ell, r) \in (r(\sqrt{2} - \gamma), r(\sqrt{2} + \gamma)).$$

Let $\widetilde{y} := ||\widetilde{\mathbf{q}} - \mathbf{p}.new||$. Then,

$$\frac{r}{\ell}\left(\widetilde{y}^2 - (\ell - r)^2\right) \in \left(r^2(\sqrt{2} - \gamma)^2, r^2(\sqrt{2} + \gamma)^2\right)$$

which using the definition $c := \sqrt{r^2 + \ell^2}$ (see Figure 10) translates to

$$\begin{aligned}\widetilde{y}^2 &\in \left(r\ell(2 - 2\sqrt{2}\gamma + \gamma^2) + (\ell - r)^2, r\ell(2 + 2\sqrt{2}\gamma + \gamma^2) + (\ell - r)^2\right) \\ &= \left(c^2 - 2\sqrt{2}r\ell\gamma + r\ell\gamma^2, c^2 + 2\sqrt{2}r\ell\gamma + r\ell\gamma^2\right)\end{aligned}$$

which also translates to

$$\widetilde{y} \in \left(\sqrt{c^2 - 2\sqrt{2}r\ell\gamma + r\ell\gamma^2}, \sqrt{c^2 + 2\sqrt{2}r\ell\gamma + r\ell\gamma^2}\right)$$

Now, noting that $\ell = O(1)$ and $r = \Theta(1)$, for large enough n we get that

$$\begin{aligned}\sqrt{c^2 - 2\sqrt{2}r\ell\gamma + r\ell\gamma^2} &\geq c - \sqrt{2\sqrt{2}r\ell\gamma - r\ell\gamma^2} \\ &\geq c - r\gamma^{1/3}.\end{aligned}$$

And Similarly,

$$\begin{aligned}\sqrt{c^2 + 2\sqrt{2}r\ell\gamma + r\ell\gamma^2} &\leq c - \sqrt{2\sqrt{2}r\ell\gamma + \ell\gamma^2} \\ &\leq c + r\gamma^{1/3}.\end{aligned}$$

So, overall

$$\widetilde{y} \in \left(c - r\gamma^{1/3}, c + r\gamma^{1/3}\right)$$

Noting that $||\mathbf{q} - \widetilde{\mathbf{q}}|| \leq \delta'^{1/3}$ and $||\mathbf{p} - \mathbf{p}.new|| \leq \delta$, using the triangle inequality, for $y := ||\mathbf{q} - \mathbf{p}||$ we get

$$y \in \left(c - r\gamma^{1/3} - \delta'^{1/3} - \delta, c + r\gamma^{1/3} + \delta'^{1/3} + \delta\right)$$

Again noting that $r = \Theta(1)$ and setting $\psi = \gamma^{1/3} + \delta'^{1/4} + \delta^{1/4}$

$$y \in (c - r\psi, c + r\psi).$$

Note that in this proof we did not optimize the inequalities and we were generous in bounding variables for the sake of brevity. □

Claim 65. *Let y be such that $y \geq x + \Delta$ for some $x \in (\delta_x, \sqrt{2})$, and y'' is such that*

$$y'' \in [\textsc{Project}(y - \delta, R_2, R), \textsc{Project}(y + \delta, R_2, R)]$$

for some R_2 and R. Let $x' = \textsc{Project}(x + \Delta, R_2, R)$, and let x'' be the smallest element in W_x which is not larger than x'. Additionally, assume that we have the following properties:

(p1) $\frac{\delta}{\Delta} = o(1)$

(p2) $\frac{\delta'}{\Delta} = o(1)$

(p3) $\Delta = \Theta(1)$

(p4) $x' \leq \frac{8}{5} \cdot R$

If η is such that $\frac{F(\eta)}{G(x''/R,\eta)} = \exp_\mu\left(\frac{1}{T}\right)$, then, we have **(a)**

$$\frac{G(y''/R, \eta)}{F(\eta)} \leq \exp_\mu\left(-(1 - o(1))\frac{4(R/x')^2 - 1}{4(R/y')^2 - 1} \cdot \frac{1}{T}\right)$$

and Furthermore, **(b)** *when $R = O(1)$, then*

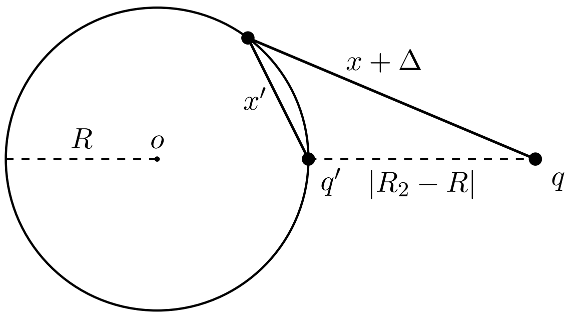
$$\frac{G(y''/R, \eta)}{F(\eta)} \leq \exp_\mu\left(-\frac{4(R/x')^2 - 1}{4(R/y')^2 - 1} \cdot \frac{1}{T}\right).$$

Proof. By assumption we have

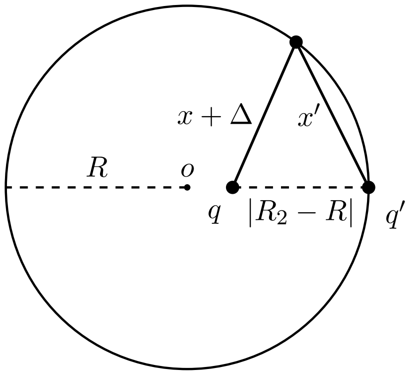
$$\frac{F(\eta)}{G(x''/R, \eta)} = \exp_\mu\left(\frac{1}{T}\right). \tag{112}$$

On the other hand, if we set $s = x''/R$

$$\begin{aligned}\frac{F(\eta)}{G(x''/R, \eta)} &= \frac{e^{-(1+o(1))\frac{\eta^2}{2}}}{e^{-(1+o(1))\frac{\eta^2}{2}\cdot\frac{4}{4-s^2}}} \\ &= e^{(1+o(1))\frac{\eta^2}{2}\cdot\frac{s^2}{4-s^2}} \\ &= \exp\left((1 + o(1))\frac{\eta^2}{2} \cdot \frac{x''^2}{4R^2 - x''^2}\right).\end{aligned} \tag{113}$$



(a) When the query is outside the sphere



(b) When the query is inside the sphere

Figure 15: Triangle inequality instances for (114)

By triangle inequality in Euclidean space (see Figure 15) we have,

$$\begin{aligned} x' &\geq (x+\Delta) - |R_2 - R| && \\ &\geq (x+\Delta) - (x+\delta) && \text{By line 20 of Algorithm 6} \\ &= \Delta - \delta = (1-o(1))\Delta && \text{By property } \textbf{(p1)}. \end{aligned} \tag{114}$$

Also note that by assumption

$$x'' \in \left[x', x' + \delta'\right],$$

then, since $\frac{\delta'}{\Delta} = o(1)$ by property **(p2)**, we have

$$x'' = (1 \pm o(1)) \cdot x'.$$

And by property **(p4)**, we have

$$\frac{x''^2}{4R^2 - x''^2} = (1 \pm o(1)) \cdot \frac{x'^2}{4R^2 - x'^2}$$

Therefore

$$\frac{F(\eta)}{G(x'/R, \eta)} = \exp\left((1+o(1))\frac{\eta^2}{2} \cdot \frac{x'^2}{4R^2 - x'^2}\right) \tag{115}$$

$$= \exp_\mu\left((1 \pm o(1))\frac{1}{T}\right). \tag{116}$$

On the other hand, if we set $s' = y'/R$, similarly

$$\begin{aligned}\frac{F(\eta)}{G(y'/R,\eta)} &= \frac{e^{-(1+o(1))\frac{\eta^2}{2}}}{e^{-(1+o(1))\frac{\eta^2}{2}\cdot\frac{4}{4-s'^2}}}\\ &= e^{(1+o(1))\frac{\eta^2}{2}\cdot\frac{s'^2}{4-s'^2}}\\ &= \exp\left((1+o(1))\frac{\eta^2}{2}\cdot\frac{y'^2}{4R^2-y'^2}\right). \qquad (117)\end{aligned}$$

Thus, by (115), (116) and (117)

$$\begin{aligned}\frac{G(y'/R,\eta)}{F(\eta)} &= \exp_\mu\left(-(1\pm o(1))\frac{1}{T}\frac{4-s^2}{s^2}\cdot\frac{s'^2}{4-s'^2}\right)\\ &= \exp_\mu\left(-(1\pm o(1))\frac{1}{T}\frac{4(R/x')^2-1}{4(R/y')^2-1}\right). \qquad (118)\end{aligned}$$

Note that

$$y'' \in [\textsc{Project}(y-\delta,R_2,R),\textsc{Project}(y+\delta,R_2,R)]$$

Then since $\frac{\delta'}{\Delta} = o(1)$ by property **(p2)**, we have

$$y'' \geq \textsc{Project}(y-\Delta/2,R_2,R) = y',$$

Now since $G(s,\eta)$ is monotone decreasing in s, we have

$$\frac{G(y'/R,\eta)}{F(\eta)} \geq \frac{G(y''/R,\eta)}{F(\eta)}. \qquad (119)$$

Now, by (118) and (119), we have the statement of the first part of the claim.

For the case when $R = O(1)$, and consequently $R_2 = O(1)$ (by the assumption fact that $x \leq \sqrt{2}$), by property **(p3)**:

$$\begin{aligned}y''^2 - y'^2 &\geq (\textsc{Project}\,(y-\delta,R_2,R))^2 - (\textsc{Project}\,(y-\Delta/2,R_2,R))^2\\ &= \frac{R}{R_2}\left((y-\delta)^2-(R_2-R)^2\right) - \frac{R}{R_2}\left((y-\Delta/2)^2-(R_2-R)^2\right)\\ &= \Omega(1) \qquad (120)\end{aligned}$$

Then,

$$\begin{aligned}(1-o(1))\cdot\frac{1}{4(R/y'')^2-1} &\geq (1-o(1))\cdot\frac{y'^2}{4R^2-y''^2} && \text{Since } y''\geq y'\\ &\geq \frac{y'^2}{4R^2-y'^2} && \text{By (120)}\\ &\geq \frac{1}{4(R/y')^2-1}\end{aligned}$$

which implies,

$$\exp_\mu\left(-\frac{1}{T}\frac{4(R/x')^2-1}{4(R/y')^2-1}\right) \geq \exp_\mu\left(-(1\pm o(1))\frac{1}{T}\frac{4(R/x')^2-1}{4(R/y'')^2-1}\right) = \frac{G(y''/R,\eta)}{F(\eta)},$$

which proves the second part of the claim. $\square$

Claim 66. *Let V denote the output of* PSEUDORANDOMIFY(v, γ) *on a node v of a recursion tree $\mathcal{T}$ associated with a dataset P of diameter bounded by D. Then for every positive integer j, where R_{min} is the parameter from line 3 of Algorithm 5, the number of sets with diameter at least $(1-\gamma^2/2)^j D$ contained in V is upper bounded by Λ^j for $\Lambda = O(D \log |P|)/\delta$.*

Proof. Note that an input dataset is first partitioned into at most $\lceil R/\delta \rceil = O(D/\delta)$ spherical shells. For each spherical shell one repeatedly removes dense clusters (containing at least a $1/10$ fraction of the current dataset), repeating this process $O(\log |P|)$ times, since at most 10 clusters are removed before the dataset size decreases by a constant factor. Every such ball has radius smaller than the original dataset by a $(1-\gamma^2/2)$ factor [ALRW17]. This gives the claimed bound. $\square$

We now give

Proof of Lemma 46: The proof is by induction on (a, b), where a is the number ℓ of LSH nodes on the path from $v \in \mathcal{T}$ to the closest leaf, b is the number of pseudorandomification nodes on such a path and $r = v.R$ is the radius of the dataset. We prove that the expected number of nodes in the subtree of such a node v in $\mathcal{T}$ is upper bounded by

$$(L \cdot \Lambda)^a \cdot (100/\mu^{1/T})^b \cdot \Lambda^j.$$

Here $\Lambda = (O(D \log |P|)/\delta)$ is the parameter from Claim 65, $j = \log_{\frac{1}{1-\gamma^2/2}}(R_{max}/r)$ is an upper bound on the number of times the radius of the sphere could have shrunk through calls to PSEUDORANDOMIFY from the largest possible (bounded R_{max}) to its current value r, and $L = \log_{\frac{1}{1-\gamma^2/2}}(R_{max}/R_{min})$ is the maximum number of times a point can be part of a dataset that PSEUDORANDOMIFY is called on (since the radius reduces by a factor of $1-\gamma^2/2$ in every such call).

The **base** is provided by the case of v being a leaf. We now give the **inductive step**. First suppose that $u \in \mathcal{T}$ is a pseudorandomification node. Let x' denote the value of rounded projected distance computed in line 19 of Algorithm 6. Then Algorithm 4 generates $\frac{100}{G(x'/R,\eta)}$ Gaussians, and the expected number of Gaussians for which the condition in line 29 is satisfied (i.e. the number of children of u that the query q explores) is exactly $\frac{100F(\eta)}{G(x'/R,\eta)}$ by definition of $F(\eta)$ (see Lemma 8 in Section 3). We also have $\frac{F(\eta)}{G(x'/R,\eta)} = (1/\mu)^{1/T}$ by setting of parameters in line 16 of Algorithm 6. Putting this together with the inductive hypothesis and noting that LSH nodes do not change the radius of the sphere, we get that the expected number of nodes of $\mathcal{T}$ that the query explores is bounded by

$$100\,(1/\mu)^{1/T} \cdot (L \cdot \Lambda)^a \cdot (100/\mu^{1/T})^{b-1} \cdot \Lambda^j = (L \cdot \Lambda)^a \cdot (100/\mu^{1/T})^b \cdot \Lambda^j,$$

as required.

Now suppose that $u \in \mathcal{T}$ is a pseudorandomification node. Then by Claim 65 for every i the number of datasets with diameter at least $(1-\gamma^2/2)^i r$ generated by PSEUDORANDOMIFY is bounded by Λ^i. For every $i = 0, \ldots, L$ the number of nodes with radius in $((1-\gamma^2/2)^{i-1} r, (1-\gamma^2/2)^i r]$ that are generated is bounded by Λ^{i-1}. For such nodes we have by the inductive hypothesis that the expected number of nodes of $\mathcal{T}$ explored in their subtree is upper bounded by

$$(L \cdot \Lambda)^{a-1} \left(100/\mu^{1/T}\right)^b \cdot \Lambda^{j-i+1}.$$

Summing over all i between 1 and $\log_{\frac{1}{1-\gamma^2/2}}(r/R_{min})$, we get that the total number of nodes that the query is expected to explore in the subtree of u is bounded by

$$\begin{aligned}\sum_{i=1}^{\log_{\frac{1}{1-\gamma^2/2}}(r/R_{min})} (L\cdot\Lambda)^{a-1}\left(100/\mu^{1/T}\right)^b\cdot\Lambda^{j-i+1}\cdot\Lambda^i &\le L\cdot(L\cdot\Lambda)^{a-1}\left(100/\mu^{1/T}\right)^b\cdot\Lambda^{j+1}\\ &\le (L\cdot\Lambda)\cdot(L\cdot\Lambda)^{a-1}\left(100/\mu^{1/T}\right)^b\cdot\Lambda^{j+1}\\ &\le (L\cdot\Lambda)^a\cdot\left(100/\mu^{1/T}\right)^b\cdot\Lambda^j\end{aligned}$$

proving the inductive step.

Substituting $\alpha^*\cdot T$ as the upper bound on the number of levels in $\mathcal{T}$ as per Algorithm 4, we thus get that the number of nodes explored by the query is bounded by

$$(L\cdot\Lambda)^T\cdot(100/\mu^{1/T})^{\alpha^*T}\cdot\Lambda^L\le(100L\cdot\Lambda)^T\cdot\Lambda^L\cdot(1/\mu)^{\alpha^*}=n^{o(1)}\cdot(1/\mu)^{\alpha^*}$$

in expectation. In the last transition we used the fact that

$$(100L\cdot\Lambda)^T\cdot\Lambda^L=(100\cdot\log_{\frac{1}{1-\gamma^2/2}}(R_{max}/R_{min})\cdot(O(R_{max}\log|P|)/\delta))^{\sqrt{\log n}}\cdot((O(D\log|P|)/\delta)^{\sqrt{\log n}}=n^{o(1)}$$

by our setting of parameters since $\gamma=1/\log\log\log n$, $R_{max}=O(1)$, $R_{min}=\Omega(1)$ and $\delta=\exp(-(\log\log n)^{O(1)})$ as per Algorithm 4 and Algorithm 5. And also since we use $100\left(\frac{1}{\mu}\right)^\alpha$ Andoni-Indyk hash functions (see Algorithm 3), we get

$$n^{o(1)}\cdot\left(\frac{1}{\mu}\right)^{\alpha^*+\alpha}$$

in total. □

Proof of Claim 47:

By Lemma 8 and Lemma 9 and Definition 10 one has

$$F(\eta)=e^{-(1+o(1))\cdot\frac{\eta^2}{2}}$$

and

$$G(x'/R,\eta)=e^{-(1+o(1))\cdot\frac{2\eta^2(1-\alpha(x'/R))}{2\beta^2(x'/R)}}=e^{-(1+o(1))\cdot\frac{2\eta^2}{2(1+\alpha(x'/R))}},$$

where $\alpha(x'/R):=1-\frac{(x'/R)^2}{2}$. Using the assumption that $x'>\Delta$ we get that

$$G(x'/R,\eta)\le e^{-(1+o(1))\cdot\frac{\eta^2}{2-((\Delta/R)^2/2)}}.$$

And in particular using the fact that $R\ge\Delta$

$$\frac{F(\eta)}{G(x'/R,\eta)}\ge e^{-(1+o(1))\cdot\frac{\eta^2}{2}+(1+o(1))\cdot\frac{\eta^2}{2-((\Delta/R)^2/2)}}=(G(x'/R,\eta))^{\Omega(\Delta^2)},$$

or, equivalently, $\frac{1}{G(x'/R,\eta)}=\left(\frac{F(\eta)}{G(x'/R,\eta)}\right)^{O(1/\Delta^2)}$. □

Proof of Claim 45: For $j^*=k_J+1$, by definition of $f_{z_j,J+1}$ for $i\in\{j^*-1,\dots,I\}$ and the fact that

$$b'_{y,J+1}=\widetilde{B}_{y,J+1}=\widetilde{A}_{y,J},$$

we have

$$f_{z_i,J+1} = \log_{1/\mu}\left(\sum_{y\in D\cap[z_{i+1},z_{i-1})} \widetilde{A}_{y,J}\right) \tag{121}$$

On the other hand, (40) and the fact that $\widetilde{B}_{y,j} = \mathbb{E}[B_{y,j}]$ we have

$$\sum_{y\geq c_J+\psi R_J} \widetilde{B}_{y,J} \leq \frac{\tau}{1-2\tau} \cdot \sum_{y\in(c_J-\psi R_J, c_J+\psi R_J)} \widetilde{B}_{y,J}.$$

Also recall (41), where we have

$$\widetilde{A}_{y,J} = \widetilde{B}_{y,J} \cdot p_{y,J}$$

where $p_{y,J}$ is a decreasing and non-negative function in y (for the valid range of y). This implies that

$$\sum_{y\geq c_J+\psi R_J} \widetilde{A}_{y,J} \leq \frac{\tau}{1-2\tau} \cdot \sum_{y\in(c_J-\psi R_J, c_J+\psi R_J)} \widetilde{A}_{y,J}.$$

Now, we have

$$\begin{aligned}
\sum_y \widetilde{A}_{y,J} &= \sum_{y<c_J+\psi R_J} \widetilde{A}_{y,J} + \sum_{y\geq c_J+\psi R_J} \widetilde{A}_{y,J} \\
&\leq \sum_{y<c_J+\psi R_J} \widetilde{A}_{y,J} + \frac{\tau}{1-2\tau} \cdot \sum_{y\in(c_J-\psi R_J, c_J+\psi R_J)} \widetilde{A}_{y,J} \\
&\leq \sum_{y\leq z_{j^*-1}} \exp_\mu\left(f_{y,J+1}\right) + \exp_\mu\left(f_{z_{j^*-1},J+1}\right) \\
&\leq O(1)\cdot \exp_\mu\left(7\delta_z\right) = \exp_\mu\left(7\delta_z + o(1)\right)
\end{aligned}$$

where the second inequality is based on Definition 41, (121) and setting of parameters (the fact that $\psi = o(1)$, $\delta_z = \Theta(1)$ and $R_{\max} = O(1)$). The last inequality is by the assumption that $f_{y,J+1} < 7\delta_z$ for $y \leq z_{j^*-1}$. $\square$

F Proof of Claim 52

Proof of Claim 52: We want to prove that

$$\left(\frac{4\left(\frac{r_j}{x'}\right)^2 - 1}{4\left(\frac{r_j}{y'}\right)^2 - 1}\right)\left(\frac{2\left(\frac{z_{k_j}}{z_i}\right)^2 - 1}{2\left(\frac{z_{k_j}}{x}\right)^2 - 1}\right) \geq (1-10^{-4}).$$

By defining $z := z_{k_j}$, $s := z_i$ and $r := r_j$ for the sake of brevity, the left hand side becomes

$$\left(\frac{x^2}{x'^2}\right)\cdot\left(\frac{y'^2}{s^2}\right)\cdot\left(\frac{4r^2 - x'^2}{2z^2 - x^2}\right)\left(\frac{2z^2 - s^2}{4r^2 - y'^2}\right).$$

We upper-bound each term one by one.

First term: Since $x' := x + \Delta$ then

$$\frac{x}{x'} = \frac{x}{x+\Delta} = 1 - \frac{\Delta}{x+\Delta} \geq 1 - \frac{\Delta}{\delta_x + \Delta} \geq 1 - \frac{\Delta}{\delta_x} = 1 - 10^{-12}$$

where we used the fact that $x \geq \delta_x$, and the last transition is by the setting of parameters. So,

$$\frac{x^2}{x'^2} \geq 1 - 10^{-11}$$

Second term: Since $y' := y - \Delta/2$ and $y \in \big(s(1+\delta_z)^{-1}, s(1+\delta_z)\big)$ then

$$\frac{y'}{s} \geq \frac{y - \Delta/2}{y(1+\delta_z)} \geq \frac{1-\delta_z}{1+\delta_z} \geq 1 - 3\delta_z$$

where we used the fact that $y \geq \delta_x$ (since $y \geq x$) and also considered that by the parameter setting $\delta_z = 10^{-6}$, $\delta_x = 10^{-8}$ and $\Delta = 10^{-20}$. Consequently, we have

$$\frac{y'^2}{s^2} \geq 1 + 9\delta_z^2 - 6\delta_z \geq 1 - 10^{-5}.$$

Third term: Note that by (31) we have

$$r(\sqrt{2} + \psi) \in [z, z(1+\delta_z))$$

which combining with the fact that $\psi = o(1)$ implies

$$r \in \left[\frac{z(1-o(1))}{\sqrt{2}}, \frac{z(1+\delta_z)}{\sqrt{2}}\right). \tag{122}$$

Note that we used the fact that $z \geq x$ so $z = \Omega(1)$ (actually we have $z = \Theta(1)$). On the other hand, by the bound for the **first term** we have

$$x'^2 \leq x^2 \left(1 + 10^{-10}\right)$$

Now, we use these tools to bound the third term[19]:

$$\begin{aligned}
\frac{4r^2 - x'^2}{2z^2 - x^2} &\geq \frac{2z^2(1-o(1)) - x^2(1+10^{-10})}{2z^2 - x^2} \\
&\geq 1 - \frac{2o(1)z^2 + 10^{-10}x^2}{2z^2 - x^2} \\
&\geq 1 - \frac{2 \times 10^{-10}x^2}{2z^2 - x^2} \\
&\geq 1 - 10^{-9}
\end{aligned}$$

[19]Note that for the sake of brevity we are being generous in bounding terms and the inequalities are not tight

Fourth term: For the fourth term, actually its easier to upper-bound the inverse of it. First, note that

$$y' = y - \Delta/2 \geq y(1-\delta_z) \geq s(1-\delta_z)(1+\delta_z)^{-1} \geq s(1-10^{-5}).$$

The first inequality is due to $y \geq x \geq \delta_x = 10^{-8}$ and $\delta_z = 10^{-6}$. This also implies that

$$y'^2 \geq s^2(1 - 3 \times 10^{-5}).$$

On the other hand, by (122) we have

$$2r^2 \leq z^2(1+\delta_z)^2 \leq z^2(1+10^{-5}).$$

Combining these facts we have

$$\begin{aligned}\frac{4r^2 - y'^2}{2z^2 - s^2} &\leq \frac{2z^2(1+10^{-5}) - s^2(1-3\times 10^{-5})}{2z^2 - s^2} \\ &\leq 1 + 10^{-5} + 4 \times 10^{-5}\frac{s^2}{2z^2 - s^2} \\ &\leq 1 + 5 \times 10^{-5}\end{aligned}$$

where the last transition is due to the fact that $s \leq z$ (or $z_i \leq z_{k_j}$ equivalently). Therefore, we have a lower-bound of $1 - 5 \times 10^{-5}$ for the fourth term.

Combining the bounds: Now, we have:

$$\begin{aligned}\left(\frac{x^2}{x'^2}\right) \cdot \left(\frac{y'^2}{s^2}\right) \cdot \left(\frac{4r^2 - x'^2}{2z^2 - x^2}\right)\left(\frac{2z^2 - s^2}{4r^2 - y'^2}\right) &\geq (1-10^{-11})(1-10^{-5})(1-10^{-9})(1-5\times 10^{-5}) \\ &\geq 1 - 10^{-4}\end{aligned}$$

which proves the claim. □

General Kernels

Lemma 67 (Uniqueness of Maximum). *Let $f : [a,b] \to \mathbb{R}$ be a three times differentiable function in (a,b) such that:*

- *$f(a) < 0$*
- *$\exists y' \in (a,b]$ such that $f(y') > 0$*
- *for all $y \in (a,b)$ it holds $\frac{d^3}{dy^3}f(y) \leq 0$.*

Then

1. *$\exists y^* \in (a,y')$ such that $f(y^*) = 0$.*
2. *$\exists \eta \in (y^*,b]$ such that η is the unique maximum of f in $[a,b]$ and the function is monotone increasing in $[a,\eta]$.*

Proof. We prove the statements in order:

1. Using the first two assumptions and continuity of f (since it is differentiable) we get by the Intermediate Value Theorem that $\exists y^* \in (a, y')$ such that $f(y^*) = 0$.

2. Since the function is defined on a closed interval it attains a maximum. We show that there exists only one maximum. Assume that there exist two local maxima $\eta_1 < \eta_2 \in (a, b]$. Then, there must be a local minimum $\eta_0 \in (\eta_1, \eta_2)$ for which $f''(\eta_0) > 0$. However, this is impossible since $f''(\eta_1) < 0$ and the function f'' is non-increasing. Hence, there is exactly one local maximum η in $(a, b]$ and the function is increasing in $[a, \eta]$ (and decreasing in $(\eta, b]$ if $\eta \neq b$).

□

Corollary 68. *Let $\phi : \mathbb{R}_+ \to \mathbb{R}$ be any function such that $\phi'''(y) \le 0$. For all $x > 0$, $T \ge 1$ and $c_1 \ge c_2 \ge \ldots \ge c_t > \frac{x}{\sqrt{2}}$ such that $\exists y' \in (x, \sqrt{2}c_t]$ with $f(y') > 0$ define:*

$$f(y) := [\phi(y) - \phi(x)] - \sum_{s=1}^{t} \frac{2(c_s/x)^2 - 1}{2(c_s/y)^2 - 1} \cdot \frac{1}{T}.$$

Then, the conclusion of Lemma 66 holds. In particular, it holds for all $\phi(y) \propto (y)^p$ with $p \le 2$.

Proof. Follows by observing that the second derivative of the summation term is decreasing and that $\phi'''(y) \propto -(2-p)p \cdot (p-1)\frac{1}{y^{2-p}} \le 0$ for all $p \le 2$ and $y > 0$. □

Claim 69 (Monotonicity)**.** *For every $i \in [|R|]$ and $\phi : \mathbb{R}_+ \to \mathbb{R}$ as in Corollary 67 we have*

(a) *there exists a $y^* \in (x, \sqrt{2})$ such that $g_{y^*, j_i} \ge 0$, $g_{y, j_i} \le 0$ for any $y \in Z_x$ such that $y \le y^*$, and g_{y, j_i} is non-decreasing in y for $y \in [y^*, z_{j_i}]$;*

(b) *there exists a $y^* \in (x, \sqrt{2})$ such that $h_{y^*}^{(N-1)} \ge 0$, $h_y^{(N-1)} \le 0$ for any $y \in Z_x$ such that $y \le y^*$ and $h_y^{(N-1)}$ is non-decreasing in y for $y \in [y^*, z_{j_i}]$.*

Proof. Let

$$q(y) := \sum_{i=1}^{t} \frac{2(c_s/x)^2 - 1}{2(c_s/y)^2 - 1} \frac{1}{T},$$

where $c_1 \ge c_2 \ge \ldots \ge c_t \ge z_1 \ge x$ for some $z_1 \ge x$. And let y_1^* be such that $\phi(y_1^*) - \phi(x) - q(y_1^*) = 0$ and let $\widetilde{y}_1$ be the smallest value such that $\widetilde{y}_1 \ge y_1^*$ and $\phi(\widetilde{y}_1) - \phi(x) - q(\widetilde{y}_1) = \theta$ for some $\theta \ge 0$. Now define $G_1(y)$ on $[y_1^*, z_1]$, for some $z_1 \ge \widetilde{y}_1$ as follows

$$G_1(y) := \begin{cases} \phi(y) - \phi(x) - q(y) & y \in [y_1^*, \widetilde{y}_1) \\ \theta & y \in [\widetilde{y}_1, z_1] \end{cases} \tag{123}$$

See the red curve in Figure 11.

Also, let $\hat{q}(y) := \frac{2(z_1/x)^2 - 1}{2(z_1/y)^2 - 1}\frac{1}{T}$. Let $y_2^* \ge y_1^*$ such that $G_1(y_2^*) - \hat{q}(y_2^*) = 0$. Now, we define $G_2(y)$ for $y \in [y_2^*, z_2]$ as follows:

$$G_2(y) := \min\{G_1(y) - \hat{q}(y), \theta'\}$$

where $\theta' := G_1(z_2) - \hat{q}(z_2)$ and $\theta' \ge 0$ for some $z_2 \le z_1$. By the definition of y_2^*, function $G_2(y)$ for $y \in [y_2^*, \widetilde{y}_1]$ is in the form of the function in Claim 53 and thus, it has a unique maximum at some

$\eta \in [y_2^*, \widetilde{y}_1]$. Also, recall that $G_1(y) = \theta$ for $y \in [\widetilde{y}_1, z_2]$. Also, one should note that since $\hat{q}(y)$ is a monotone increasing function for $y \in (0, \sqrt{2}z_1)$ and hence for $y \in [\widetilde{y}_1, z_2]$, then $\theta' \leq G_2(\widetilde{y}_1)$ and therefore $\theta' \leq G_2(\eta)$. This guarantees that there exist a $\widetilde{y}_2 \in [y_2^*, \eta]$ such that $G_2(\widetilde{y}_2) = \theta'$. The reason is that $G_2(y)$ is a continuous increasing function for $y \in [\widetilde{y}_2, \eta]$. So, we have

$$G_2(y) := \begin{cases} \phi(y) - \phi(x) - q'(y) & y \in [y_2^*, \widetilde{y}_2) \\ \theta' & y \in [\widetilde{y}_2, z_2] \end{cases} \tag{124}$$

where, $q'(y) := q(y) - \hat{q}(y)$. See the blue curve in Figure 11.

□