Optimal Data-Dependent Hashing for Approximate Near Neighbors

Alexandr Andoni*† Ilya Razenshteyn*‡

January 7, 2015

Abstract

We show an optimal data-dependent hashing scheme for the approximate near neighbor problem. For an n-point data set in a d-dimensional space our data structure achieves query time $O(dn^{\rho+o(1)})$ and space $O(n^{1+\rho+o(1)}+dn)$, where $\rho=\frac{1}{2c^2-1}$ for the Euclidean space and approximation $c>1$. For the Hamming space, we obtain an exponent of $\rho=\frac{1}{2c-1}$.

Our result completes the direction set forth in [AINR14] who gave a proof-of-concept that data-dependent hashing can outperform classic Locality Sensitive Hashing (LSH). In contrast to [AINR14], the new bound is not only optimal, but in fact improves over the best (optimal) LSH data structures [IM98, AI06] for all approximation factors $c>1$.

From the technical perspective, we proceed by decomposing an *arbitrary* dataset into several subsets that are, in a certain sense, *pseudo-random*.

*Work was done in part while both authors were at Microsoft Research Silicon Valley.

†Simons Institute for the Theory of Computing, UC Berkeley, andoni@mit.edu

‡CSAIL MIT, ilyaraz@mit.edu

Contents

1 Introduction

In the near neighbor search problem, we are given a set P of n points in a d-dimensional space, and the goal is to build a data structure that, given a query point q, reports any point within a given distance r to the query. The problem is of major importance in several areas, such as databases, data mining, information retrieval, computer vision, computational geometry, signal processing, etc.

Efficient near(est) neighbor algorithms are known for the case when the dimension d is "low" (e.g., see [Cla88, Mei93]). However, the current solutions suffer from "the curse of dimensionality" phenomenon: either space or query time are exponential in the dimension d. To escape this curse, researchers proposed *approximation* algorithms for the problem. In the (c, r)-approximate near neighbor problem (ANN), the data structure may return any data point whose distance from the query is at most cr, for an approximation factor $c > 1$ (provided that there exists a data point within distance r). Many approximation algorithms for the problem are known; e.g., see surveys [AI08, And09].

To address the ANN problem, Indyk and Motwani proposed the *Locality Sensitive Hashing* scheme (LSH), which has since proved to be influential in theory and practice [IM98, HPIM12]. In particular, LSH yields the best ANN data structures for the regime of sub-quadratic space and constant approximation factor, which turns out to be the most important regime from the practical perspective. The main idea is to hash the points such that the probability of collision is much higher for points which are close to each other (at distance $\leq r$) than for those which are far apart (at distance $\geq cr$). Given such hash functions, one can retrieve near neighbors by hashing the query point and retrieving elements stored in buckets containing that point. If the probability of collision is at least p_1 for the close points and at most p_2 for the far points, the algorithm solves the (c,r)-ANN using $n^{1+\rho}$ extra space and dn^{ρ} query time[1], where $\rho = \log(1/p_1)/\log(1/p_2)$ [HPIM12]. The value of the exponent ρ thus determines the "quality" of the LSH families used.

A natural question emerged: what is the best possible exponent ρ? The original LSH paper [IM98] showed $\rho \leq 1/c$ for both Hamming and Euclidean spaces. Focusing on the Euclidean space, subsequent research showed that one can obtain a better exponent: $\rho \leq 1/c^2$ [DIIM04, AI06][2]. Complementing these results, lower bounds on ρ showed that this bound is tight [MNP07, OWZ11], thus settling the question: the best exponent is $\rho = 1/c^2$ for the Euclidean space.

Data-dependent hashing. Surprisingly, while the best possible LSH exponent ρ has been settled, it turns out there exist *more efficient* ANN data structures, which step outside the LSH framework. In particular, [AINR14] obtain the exponent of $\rho = \frac{7/8}{c^2} + \frac{O(1)}{c^3}$ by considering *data-dependent hashing*, i.e., a randomized hash family that itself depends on the actual points in the dataset. We stress that this approach gives improvement for *worst-case* datasets, which is somewhat unexpected. To put this into a perspective: if one were to assume that the dataset has some *special structure*, it would be more natural to expect speed-ups with data-dependent hashing: such hashing may adapt to the special structure, perhaps implicitly, as was done in, say, [DF08, VKD09, AAKK14]. However, in our setting there is no assumed structure to adapt to, and hence it is unclear why data-dependent hashing shall help. (To compare with classic, non-geometric hashing, the most similar situation where data-dependent hashing helps in the worst-case seems to be the perfect hashing [FKS84].) Note that for the case of Hamming space, [AINR14] has been the first and only improvement over [IM98] since the introduction of LSH (see Section 1.3).

Thus the core question resurfaced: what is the best possible exponent for *data-dependent* hashing schemes? To formulate the question correctly, we also need to require that the hash family is "nice": otherwise, the trivially best solution is the Voronoi diagram of n points at hand, which is obviously useless (computing the hash function is as hard as the original problem!). A natural way to preclude such non-viable solutions is to require that each hash function can be "efficiently described", i.e., it has a description of $n^{o(1)}$ bits (e.g., this property is satisfied by all the LSH functions we know of).

1.1 Main result

We present an optimal data-dependent hash family that achieves the following exponent[3] for ANN under the Euclidean distance:

$$\rho = \frac{1}{2c^2 - 1}. \tag{1}$$

Specifically, we obtain the following main theorem.

[1] For exposition purposes, we are suppressing the time to compute hash functions, which we assume to require $n^{o(1)}$ time and $n^{o(1)}$ space. We also assume distances can be computed in $O(d)$ time, and that $1/p_1 = n^{o(1)}$.

[2] Ignoring terms vanishing with n; the exact dependence is $\rho = 1/c^2 + 1/\log^{\Omega(1)} n$.

[3] Again, we ignore the additive term that vanishes with n.

Theorem 1.1. *For fixed approximation $c > 1$ and threshold $r > 0$, one can solve the (c,r)-ANN problem in d-dimensional Euclidean space on n points with $O(d \cdot n^{\rho+o_c(1)})$ query time, $O(n^{1+\rho+o_c(1)} + dn)$ space, and $O(d \cdot n^{1+\rho+o_c(1)})$ preprocessing time, where $\rho = \frac{1}{2c^2-1}$.*

Our bound (1) is optimal since it matches the LSH lower bound of $\rho \geq \frac{1}{2c^2-1}$ from [MNP07, Dub10], which turns out to be applicable to the data-dependent hashing as well. In particular, [MNP07] proves a lower bound of roughly $\rho \geq \frac{1}{2c^2}$ for large c. [Dub10] shows a lower bound matching the bound (1) in the context of *closest pair problem*, where the goal is to find the closest pair of points from a given dataset. The closest pair problem is essentially the "offline version" of ANN, as it can be solved by performing n queries on the given dataset. Hence lower bounds there apply to our ANN setting as well. We note that [Dub10] obtains an optimal hashing algorithm for the closest pair problem on *random instances* in the Hamming space, running in $n^{1+\rho+o(1)}$ time where $\rho = \frac{1}{2c-1}$.

The lower bounds of [MNP07, Dub10] apply to the data-dependent hashing in the following sense. These lower bounds are for the setting of a random dataset instance, where the "far" points are two random, uncorrelated points. Hence, even if the LSH designer has access to the distribution from which the dataset is sampled, the designer learns nothing about the query, and thus the lower bounds from [MNP07, Dub10] still apply. It is concievable that one can improve over the bound (1) if the LSH designer knows the particular n-point dataset instance. Nonetheless, we remove this possibility as well in a separate note [AR15]. In the same note, we also show that the tight lower bound of $\rho \geq \frac{1}{2c^2-1}$ may be proven directly by strenghening the argument from [MNP07], simplifying the argument from [Dub10].[4]

An important aspect of our algorithm is that it effectively reduces ANN on a generic dataset to ANN on an (essentially) random dataset. The latter is the most natural "hard distribution" for the ANN problem. Besides the aforementioned lower bounds, it is also a source of *cell-probe* lower bounds for ANN [PTW08, PTW10]. Hence, looking forward, to further improve the efficiency of ANN, one would have first to improve the random dataset case, which seems to require fundamentally different techniques, if possible at all.

The importance of this reduction is highlighted by the progress on the closest pair problem by Valiant [Val12]. In particular, Valiant gives an algorithm with $n^{1.62} \cdot \mathrm{poly}(\frac{d}{c-1})$ runtime for the aforementioned *random instances*.[5] Obtaining similar runtime for the *worst case* (e.g., via a similar reduction) would *refute the Strong Exponential-Time Hypothesis* (SETH).

We also point out that, besides achieving the optimal bound, the new algorithm has two further advantages over the one from [AINR14]. First, our bound (1) is better than the optimal LSH bound $1/c^2$ *for every* $c > 1$ (the bound from [AINR14] is better only for sufficiently large c). Second, the preprocessing time of our algorithm is near-linear in the amount of space used, improving over the quadratic preprocessing time of [AINR14].

1.2 Techniques

The general approach is via data-dependent LSH families, which can be equivalently seen as data-dependent random *space partitions*. Such space partitions are usually constructed iteratively: first we partition the space very coarsely, then we refine the partition iteratively a number of times. In standard LSH, each iteration of partitioning is random i.i.d., and the overall data structure consists of n^ρ such iterative space partitions, constructed independently (see [HPIM12] for details).

[4] We note that [Dub10] claims the tight lower bound of $\rho \geq \frac{1}{2c^2-1}$ in the journal version, although it relies on an inequality deferred to an unpublished manuscript. In our note [AR15], we prove an alternative, complete proof of this lower bound in the LSH setting.

[5] Note that this improves over [Dub10]: Valiant exploits fast matrix multiplication algorithms to step outside Dubiner's hashing framework altogether.

For the latter discussion, it is useful to keep in mind what is the *random dataset instances* for ANN. Consider a sphere of radius $cr/\sqrt{2}$ in $\mathbb{R}^d$ for $d = 1000 \log n$. The data set is obtained by sampling n points on the sphere uniformly at random. To generate a query, one chooses a data point uniformly at random, and plants a query at distance at most within r from it uniformly at random. It is not hard to see that with very high probability, the query will be at least $cr - o(1)$ apart from all the data points except one. Thus, any data structure for $(c - o(1), r)$-ANN must be able to recover the data point the query was planted to.

Let us first contrast our approach to the previous algorithm of [AINR14]. That result improved over LSH by identifying a "nice configuration" of a dataset, for which one can design a hash family with better $\rho < 1/c^2$. It turns out that the right notion of niceness is the ability to enclose dataset into a ball of small radius, of order $O(cr)$ (the aforementioned random instance corresponds to "tightest possible" radius of $cr/\sqrt{2}$). Moreover, the smaller the enclosing ball is, the better the exponent ρ one can obtain. The iterative partitioning from [AINR14] consists of two rounds. During the first round, one partitions the space so that the dataset in each part would be "low-diameter" with high probability. This step uses classic, data-independent LSH, and hence effectively has quality $\rho = 1/c^2$. During the second round, one would apply "low-diameter" LSH with quality $\rho < 1/c^2$. The final exponent ρ is a weighted average of qualities of the two rounds. While one can generalize their approach to any number of rounds, the best achievable exponent ρ is around $0.73/c^2 + O(1/c^3)$ [Raz14].

In fact, [AINR14] cannot obtain the optimal ρ as in (1) *in principle*. The fundamental issue is that, before one completes the reduction to a "nice configuration", one must incur some "waste". In particular, the first round uses (non-optimal) data-independent hashing, and hence the "average" ρ cannot meet the best-achievable ρ. (Moreover, even the second round of the algorithm does not achieve the optimal ρ.)

Thus, the real challenge remained: how to perform *each* iteration of the partitioning with *optimal* ρ? E.g., we must be able to perform such partitioning even during the very first iteration, on a dataset without any structure whatsoever.

Our new algorithm resolves precisely this challenge. For simplicity, let us assume that all the data points lie on a sphere of radius R. (It is helpful to think of R as being, say, $1000cr$ — e.g., the low-diameter family from [AINR14] gives almost no advantage over the data-independent $\rho = 1/c^2$ for such R). We start by decomposing the data set into a small number of *dense clusters* (by this we mean a spherical cap that is slightly smaller than a hemisphere and that covers $n^{1-o(1)}$ points) and a *pseudo-random* remainder that has no dense parts. For dense clusters we recurse enclosing them in balls of radius slightly smaller than R, and for the pseudorandom part we just apply one iteration of the "low-diameter" LSH from [AINR14] and then recurse on each part. This partitioning subroutine makes progress in two ways. For dense clusters we slightly reduce the radius of the instance. Thus, after a bounded number of such reductions we will arrive to an instance that can be easily handled using the low-diameter family. For the pseudorandom remainder, we can argue that the low-diameter family works unreasonably well: intuitively, it follows from the fact that almost all pairs of data points are very far apart (roughly speaking, at distance almost $\sqrt{2}R$). We call the remainder pseudo-random precisely because of the latter property: random points are essentially $(\sqrt{2}R)$-separated.

We note that one can see the partition from above as a (kind of) decision tree, albeit with a few particularities. Each node of the decision tree has a number of children that partition the dataset points (reaching this node), corresponding to the dense clusters as well as to all the (non-empty) parts of the aforementioned "low-diameter" LSH partition. In fact, it will be necessary for some points to be replicated in a few children (hence the decision tree size may be $\gg n$), though we will show that the degree of replication is bounded (a factor of $n^{o(1)}$ overall). The query procedure will

search the decision tree by following a path from the root to the leaves. Again, it may be necessary to follow a few children at a decision node at a time; but the replication is bounded as well. The final data structure is composed of n^ρ such decision trees, and the query algorithm queries each of them.

A new aspect of our analysis is that, among other things, we will need to understand how the low-diameter family introduced in [AINR14] works on *triples of points*: without this analysis, one can only get a bound of

$$\rho = \frac{1}{2c^2 - 2},$$

which is much worse than (1) for small c. To the best of our knowledge, this is the first time when one needs to go beyond the "pairwise" analysis in the study of LSH.

1.3 Further implications and connections

Our algorithm also directly applies to the Hamming metric, for which it achieves[6] an exponent of

$$\rho = \frac{1}{2c - 1}.$$

This is a nearly quadratic improvement over the original LSH paper [IM98], which obtained exponent $\rho = 1/c$, previously shown to be optimal for the classic LSH in [OWZ11]. The result of [AINR14] was the first to bypass the 1998 bound via data-dependent hashing, achieving $\rho = \frac{7/8}{c} + \frac{O(1)}{c^{3/2}}$, an improvement for large enough c. Our new bound improves over [IM98] for all values of c, and is also optimal (the above discussion for ℓ_2 applies here as well).

From a broader perspective, we would like to point out the related developments in practice. Many or most of the practical applications of LSH involve designing *data-aware* hash functions [Spr91, McN01, VKD09, WTF08, SH09, YSRL11] (see also a recent survey [WSSJ14]). The challenge of understanding and exploiting the relative strengths of data-oblivious versus data-aware methods has been recognized as a major open question in the area (e.g., see [Cou13], page 77). This paper can be seen as part of the efforts addressing the challenge.

We conclude with mentioning a natural question for future research: can we design *dynamic* data-dependent hashing algorithms for ANN? In contrast to the case of data-independent hashing, it appears harder to maintain a data-dependent hashing structure under insertions/deletions of points.

2 Preliminaries

In the text we denote the ℓ_2 norm by $\|\cdot\|$. When we use $O(\cdot)$, $o(\cdot)$, $\Omega(\cdot)$ or $\omega(\cdot)$ we explicitly write all the parameters that the corresponding constant factors depend on as subscripts (the variable is always n or derived functions of n). Our main tool will be random partitions of a metric space. For a partition $\mathcal{R}$ and a point p we denote $\mathcal{R}(p)$ the part of $\mathcal{R}$, which p belongs to. If $\mathcal{R}$ is a partition of a *subset* of the space, then we denote $\bigcup\mathcal{R}$ the union of all the pieces of $\mathcal{R}$. By $N(a,\sigma^2)$ we denote a standard Gaussian with mean a and variance σ^2. We denote the closed Euclidean ball with a center u and a radius $r \geq 0$ by $B(u,r)$. By $\partial B(u,r)$ we denote the corresponding *sphere*. We denote $S^{d-1} \subset \mathbb{R}^d$ the unit Euclidean sphere in $\mathbb{R}^d$ with the center being the origin. A spherical cap can be considered as a ball on a sphere with metric inherited from the ambient Euclidean distance. We define a *radius* of a spherical cap to be the radius of the corresponding ball. For instance, the radius of a hemisphere of a unit sphere is equal to $\sqrt{2}$.

[6] This follows from a standard embedding of the ℓ_1 norm into ℓ_2-squared [LLR95].

Definition 2.1. The (c,r)-*Approximate Near Neighbor problem (ANN)* with failure probability f is to construct a data structure over a set of points $P \subseteq \mathbb{R}^d$ supporting the following query: given any fixed query point $q \in \mathbb{R}^d$, if there exists $p \in P$ with $\|p-q\| \leq r$, then report some $p' \in P$ such that $\|p'-q\| \leq cr$, with probability at least $1-f$.

Note that we allow preprocessing to be randomized as well, and we measure the probability of success over the random coins tossed during *both* preprocessing and query phases.

Definition 2.2 ([HPIM12])**.** We call a random partition $\mathcal{R}$ of $\mathbb{R}^d$ (r_1, r_2, p_1, p_2)-*sensitive*, if for every $x, y \in X$ we have $\Pr_{\mathcal{R}}[\mathcal{R}(x) = \mathcal{R}(y)] \geq p_1$ if $\|x-y\| \leq r_1$, and $\Pr_{\mathcal{R}}[\mathcal{R}(x) = \mathcal{R}(y)] \leq p_2$ if $\|x-y\| \geq r_2$.

Remark: For $\mathcal{R}$ to be useful we require that $r_1 < r_2$ and $p_1 > p_2$.

Now we are ready to state a very general way to solve ANN, if we have a good (r, cr, p_1, p_2)-sensitive partition [IM98, HPIM12]. The following theorem gives a data structure with near-linear space and small query time, but with probability of success being only inversely polynomial in the number of points.

Theorem 2.3 ([IM98, HPIM12])**.** *Suppose that there is a (r, cr, p_1, p_2)-sensitive partition $\mathcal{R}$ of $\mathbb{R}^d$, where $(p_1, p_2) \in (0,1)$ and let $\rho = \ln(1/p_1)/\ln(1/p_2)$. Assume that $p_1, p_2 \geq 1/n^{o_c(1)}$, one can sample a partition from $\mathcal{R}$ in time $n^{o_c(1)}$, store it in space $n^{o_c(1)}$ and perform point location in time $n^{o_c(1)}$. Then there exists a data structure for (c,r)-ANN over a set $P \subseteq \mathbb{R}^d$ with $|P| = n$ with preprocessing time $O(dn^{1+o_c(1)})$, probability of success at least $n^{-\rho-o_c(1)}$, space consumption (in addition to the data points) $O(n^{1+o_c(1)})$ and* expected *query time $O(dn^{o_c(1)})$.*

Remark 2.4. *To obtain the full data structure we increase the probability of success from $n^{-\rho-o_c(1)}$ to 0.99 by building $O(n^{\rho+o_c(1)})$ independent data structures from Theorem 2.3. Overall, we obtain $O(dn^{\rho+o_c(1)})$ query time, $O(dn + n^{1+\rho+o_c(1)})$ space, and $O(dn^{1+\rho+o_c(1)})$ preprocessing time.*

In our algorithm, we will also be using the following (data-independent) LSH scheme.

Theorem 2.5 ([DIIM04])**.** *There exists a random partition $\mathcal{R}$ of $\mathbb{R}^d$ such that for every $u, v \in \mathbb{R}^d$ with $\|u-v\| = \tau$ one has $\ln \frac{1}{\Pr_{\mathcal{R}}[\mathcal{R}(u)=\mathcal{R}(v)]} = (1 + O_\tau(1/d)) \cdot \tau\sqrt{d}$. Moreover, $\mathcal{R}$ can be sampled in time $d^{O(1)}$, stored in space $d^{O(1)}$ and a point can be located in $\mathcal{R}$ in time $d^{O(1)}$.*

3 Spherical LSH

In this section, we describe a partitioning scheme of the unit sphere S^{d-1}, termed *Spherical LSH*. We will use Spherical LSH in our data structure described in the next section. While the Spherical LSH was introduced in [AINR14], we need to show a new important property of it. We then illustrate how Spherical LSH achieves optimal ρ for the ANN problem in the "base case" of *random* instances. As mentioned in the Introduction, the main thrust of the new data structure will be to reduce a worst-case dataset to this "base case".

The main idea of the Spherical LSH is to "carve" spherical caps of radius $\sqrt{2} - o(1)$ (*almost hemispheres*). The partitioning proceeds as follows:

$\mathcal{R} \leftarrow \emptyset$
while $\bigcup \mathcal{R} \neq S^{d-1}$ **do**
  sample $g \sim N(0,1)^d$
  $U \leftarrow \left\{u \in S^{d-1} : \langle u, g \rangle \geq d^{1/4}\right\} \setminus \bigcup \mathcal{R}$

if $U \neq \emptyset$ **then**
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{U\}$

Here $\mathcal{R}$ denotes the resulting partition of S^{d-1}, $\bigcup\mathcal{R}$ denotes the union of all elements of $\mathcal{R}$ (the currently partitioned subset of S^{d-1}) and $N(0,1)^d$ is a standard d-dimensional Gaussian.

This partitioning scheme is not efficient: we can not quickly compute $\bigcup\mathcal{R}$, and the partitioning process can potentially be infinite. We will address these issues in Section A.3.

Now let us state the properties of Spherical LSH (the efficiency aspects are for a slightly modified variant of it, according to Section A.3). The proof is deferred to Appendix A.

Theorem 3.1. *There exists a positive constant $\delta > 0$ such that for sufficiently large d, Spherical LSH $\mathcal{R}$ on S^{d-1} has the following properties. Suppose that $\varepsilon = \varepsilon(d) > 0$ tends to 0 as d tends to infinity; also let $\tau \in [d^{-\delta}; 2 - d^{-\delta}]$. Then, for every $u, v, w \in S^{d-1}$, we have*

$$\|u-v\| \geq \tau \Longrightarrow \ln(1/\Pr_{\mathcal{R}}[\mathcal{R}(u) = \mathcal{R}(v)]) \geq (1 - d^{-\Omega(1)}) \cdot (\tau^2/(4-\tau^2)) \cdot \sqrt{d}/2, \tag{2}$$

$$\|u-v\| \leq \tau \Longrightarrow \ln(1/\Pr_{\mathcal{R}}[\mathcal{R}(u) = \mathcal{R}(v)]) \leq (1 + d^{-\Omega(1)}) \cdot (\tau^2/(4-\tau^2)) \cdot \sqrt{d}/2, \tag{3}$$

$$\begin{aligned} &\text{if } \|u-w\|, \|v-w\| \in \sqrt{2} \pm \varepsilon, \|u-v\| \leq 1.99 \\ &\qquad \Longrightarrow \ln(1/\Pr_{\mathcal{R}}[\mathcal{R}(u) = \mathcal{R}(w) \mid \mathcal{R}(u) = \mathcal{R}(v)]) \geq (1 - \varepsilon^{\Omega(1)} - d^{-\Omega(1)}) \cdot \sqrt{d}/2. \end{aligned} \tag{4}$$

Moreover, we can sample a partition in time $\exp(O(\sqrt{d}))$, store it in space $\exp(O(\sqrt{d}))$ and locate a point from S^{d-1} in it in time $\exp(O(\sqrt{d}))$.

Discussion of the three-point collision property (4). While properties (2) and (3) were derived in [AINR14] (under an additional assumption $\tau \leq \sqrt{2}$), the property (4) is the new contribution of Theorem 3.1.

Let us elaborate why proving (4) is challenging. First, one can easily prove that

$$\begin{aligned} \ln(1/\Pr_{\mathcal{R}}[\mathcal{R}(u) = \mathcal{R}(w) \mid \mathcal{R}(u) = \mathcal{R}(v)]) &\geq \ln(\Pr_{\mathcal{R}}[\mathcal{R}(u) = \mathcal{R}(v)] / \Pr_{\mathcal{R}}[\mathcal{R}(u) = \mathcal{R}(w)]) \\ &= \ln(1/\Pr_{\mathcal{R}}[\mathcal{R}(u) = \mathcal{R}(w)]) - \ln(1/\Pr_{\mathcal{R}}[\mathcal{R}(u) = \mathcal{R}(v)]) \\ &\approx \sqrt{d}/2 - \ln(1/\Pr_{\mathcal{R}}[\mathcal{R}(u) = \mathcal{R}(v)]), \end{aligned} \tag{5}$$

where the last step follows from (2), (3) and the fact that $\|u-w\| \approx \sqrt{2}$. However this is worse than $\sqrt{d}/2$ claimed in (4) provided that u and v are not too close. It turns out that (5) is tight, if we do not assume anything about $\|v-w\|$ (for instance, when u, v and w lie on a great circle). So, we have to "open the black box" and derive (4) from the first principles. A high-level idea of the analysis is to observe that, when $\|v-w\| \approx \sqrt{2}$, certain directions of interest become almost orthogonal, so the corresponding Gaussians are almost independent, which gives almost independence of the events "$\mathcal{R}(u) = \mathcal{R}(w)$" and "$\mathcal{R}(u) = \mathcal{R}(v)$", which, in turn, implies

$$\ln(1/\Pr_{\mathcal{R}}[\mathcal{R}(u) = \mathcal{R}(w) \mid \mathcal{R}(u) = \mathcal{R}(v)]) \approx \ln(1/\Pr_{\mathcal{R}}[\mathcal{R}(u) = \mathcal{R}(w)]) \approx \sqrt{d}/2$$

as required. Again, see the full argument in Appendix A.

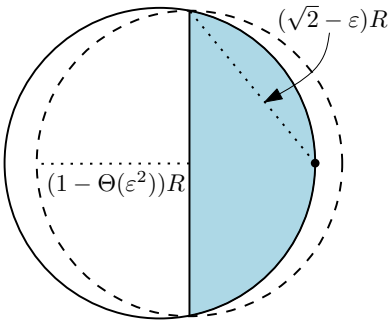


Figure 1: Covering a spherical cap of radius $(\sqrt{2}-\varepsilon)R$

3.1 Implications for ANN

It is illuminating to see what Spherical LSH implies for a random instance (as defined in the Introduction). Since all the points lie on a sphere of radius $cr/\sqrt{2}$, we can plug in Theorem 3.1 into Theorem 2.3, and thus obtain the exponent

$$\rho \leq \frac{1}{2c^2-1} + o_c(1).$$

Note that we achieve the desired bound (1) for *random instances* by using the Spherical LSH directly.

4 The data structure

In this section we describe the new data structure. First, we show how to achieve $n^{-\rho}$ success probability, $n^{o_c(1)}$ query time, and $n^{1+o_c(1)}$ space and preprocessing time, where $\rho = \frac{1}{2c^2-1} + o_c(1)$. Finally, to obtain the final result, one then builds $O(n^\rho)$ copies of the above data structure to amplify the probability of success to 0.99 (as explained in Remark 2.4). We analyze the data structure in Section 5.

4.1 Overview

We start with a high-level overview. Consider a dataset P_0 of n points. We can assume that $r = 1$ by rescaling. We may also assume that the dataset lies in the Euclidean space of dimension $d = \Theta(\log n \cdot \log\log n)$: one can always reduce the dimension to d by applying Johnson-Lindenstrauss lemma [JL84, DG03] while incurring distortion at most $1 + 1/(\log\log n)^{\Omega(1)}$ with high probability.

For simplicity, suppose that the entire dataset P_0 and a query lie on a sphere $\partial B(0, R)$ of radius $R = O_c(1)$. If $R \leq c/\sqrt{2}$, we are done: this case corresponds to the "nicest configuration" of points and we can apply Theorem 2.3 equipped with Theorem 3.1 (see the discussion in Section 3.1).

Now suppose that $R > c/\sqrt{2}$. We split P_0 into a number of disjoint components: l *dense* components, termed $C_1, C_2, \ldots, C_l$, and one *pseudorandom* component, termed $\widetilde{P}$. The properties of these components are as follows. For each dense component C_i we require that $|C_i| \geq \tau n$ and that C_i can be covered by a spherical cap of radius $(\sqrt{2}-\varepsilon)R$. Here $\tau, \varepsilon > 0$ are small positive quantities to be chosen later. The pseudorandom component $\widetilde{P}$ is such that it contains no more dense components inside.

We proceed separately for each C_i and $\widetilde{P}$ as follows. For every dense component C_i, we enclose it in a ball E_i of radius $(1-\Theta(\varepsilon^2))R$ (see Figure 1). For simplicity, let us first ignore the issue that C_i

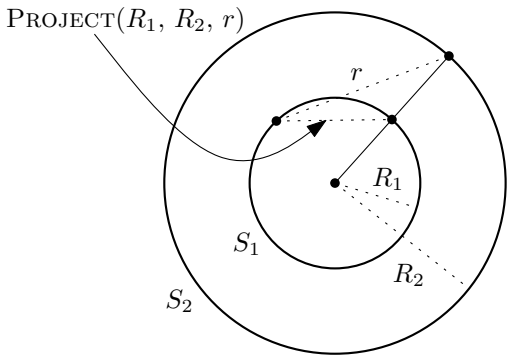


Figure 2: The definition of Project

does not necessarily lie on the boundary ∂E_i. Then, we can just recurse for the resulting spherical instance with radius $(1-\Theta(\varepsilon^2))R$. We treat the pseudorandom part $\widetilde{P}$ completely differently. We sample a partition (hash function) $\mathcal{R}$ of $\partial B(0,R)$ using Theorem 3.1. Then we partition $\widetilde{P}$ using $\mathcal{R}$ and recurse on each non-empty part. Note that after we have partitioned $\widetilde{P}$, there may appear new dense clusters in some parts (since it may become easier to satisfy the minimum size constraint).

During the query procedure, we will recursively query *each* C_i. Furthermore, for the pseudorandom component $\widetilde{P}$, we locate the part of $\mathcal{R}$ that captures the query point, and recursively query this part. Overall, there are $(l+1)$ recursive calls.

To analyze our algorithm, we show that we make progress in two ways. First, for dense clusters we reduce the radius of a sphere by a factor of $(1-\Theta(\varepsilon^2))$. Hence, in $O_c(1/\varepsilon^2)$ iterations we must arrive to the case of $R \le c/\sqrt{2}$, which is easy (as argued above). Second, for the pseudorandom component $\widetilde{P}$, we argue that most of the points lie at distance $(\sqrt{2}-\varepsilon)R$ from each other. In particular, the ratio of R to a typical inter-point distance is $\approx 1/\sqrt{2}$, like in a random case for which Spherical LSH from Theorem 3.1 is efficient, as discussed in Section 3.1. (This is exactly the reason why we call $\widetilde{P}$ pseudorandom.) Despite the simplicity of this intuition, the actual analysis is quite involved: in particular, this is the place, where we use the three-point property (4) of the Spherical LSH.

It remains to address the issue deferred in the above high-level description: namely, that a dense component C_i does not generally lie on ∂E_i, but rather can occupy the interior of E_i. We deal with it by partitioning E_i into very thin annuli of carefully chosen width δ. We then treat each annulus as a sphere. This discretization of a ball adds to the complexity of the analysis, although it does not seem to be fundamental from the conceptual point of view.

Finally, we also show how to obtain *fast* preprocessing, which turns out to be a non-trivial task, as we discuss in Section 6. The main bottleneck is in finding dense components, for which we show a near-linear time algorithm. Roughly, the idea is to restrict ourselves to dense components with centers in data points: this gives preprocessing time $n^{2+o(1)}$; we improve it further, to $n^{1+o(1)}$, by sampling the dataset and searching for dense components in the sample only (intuitively, this works because we require the dense components to contain many points).

4.2 Algorithm

We are now ready to describe the data structure formally. It depends on the (small positive) parameters τ, ε and δ, which we will need to choose carefully later on. The pseudocode appears as Figure 3.

Preprocessing. Our preprocessing algorithm consists of the following functions:

- ProcessSphere(P, r_1, r_2, o, R) builds the data structure for a pointset P that lies on a sphere $\partial B(o,R)$, assuming we need to solve ANN with distance thresholds r_1 and r_2. Moreover, we are guaranteed that queries will lie on $\partial B(o,R)$, too.
- ProcessBall(P, r_1, r_2, o, R) builds the data structure for a dataset P that lies inside the ball $B(o,R)$, assuming we need to solve ANN with distance thresholds r_1 and r_2. Unlike ProcessSphere, here queries can be arbitrary.
- Process(P) builds the data structure for a dataset P to solve the general $(1,c)$-ANN;
- Project(R_1, R_2, r) is an auxiliary function computing the following projection. Suppose we have two spheres S_1 and S_2 with a common center and radii R_1 and R_2. Suppose there are points $p_1 \in S_1$ and $p_2 \in S_2$ with $\|p_1 - p_2\| = r$. Project(R_1, R_2, r) returns the distance between p_1 and the point $\widetilde{p_2}$ that lies on S_1 and is the closest to p_2 (see Figure 2).

We now elaborate on algorithms in each of the above functions.

ProcessSphere. Function ProcessSphere follows the exposition from Section 4.1. First, we consider three base cases. If $r_2 \geq 2R$, then the goal can be achieved trivially, since any point from P works as an answer for any valid query. If $r_1/r_2 \leq 1/(2c^2-1)$, then Theorem 2.3 coupled with the hash family from Theorem 2.5 does the job. Similarly, if $r_2 \geq \sqrt{2}R$, then we can use the family from Theorem 3.1 (see the discussion in Section 3.1). Otherwise, we find non-trivially smaller balls (of radius $(\sqrt{2}-\varepsilon)R$) with centers on $\partial B(o,R)$ that contain many data points (at least $\tau|P|$). These balls can be enclosed into balls (with unconstrained center) of radius $\widetilde{R} \leq (1-\Omega(\varepsilon^2))R$ (proven in Claim 5.1). For these balls we invoke ProcessBall. Then, for the remaining points we sample a partition of $\partial B(o,R)$ using Theorem 3.1, and recurse on each part. We note that in order to apply Theorem 2.5 and Theorem 3.1 we need certain conditions on r_1, r_2 and R to hold (we define and verify them in Claim 5.2).

ProcessBall. First, we consider the following simple base case. If $r_1 + 2R \leq r_2$, then any point from $B(o,R)$ could serve as a valid answer to any query.

In general, we reduce to the spherical case via a discretization of the ball $B(o,R)$. First, we round all the distances to o up to a multiple of δ, which can change distance between any pair of points by at most 2δ (by the triangle inequality). Then, for every possible distance δi from o to a data point and every possible distance δj from o to a query (for admissible integers i,j), we build a separate data structure via ProcessSphere (we also need to check that $|\delta(i-j)| \leq r_1 + 2\delta$ to ensure that the corresponding pair (i,j) does not yield a trivial instance). We compute the new distance thresholds $\widetilde{r}_1$ and $\widetilde{r}_2$ for this data structure as follows. After rounding, the new thresholds for the ball instance should be $r_1 + 2\delta$ and $r_2 - 2\delta$, since distances can change by at most 2δ. To compute the final thresholds (after projecting the query to the sphere of radius δi), we just invoke Project (see the definition above).

Process. Process reduces the general case to the ball case. We proceed similarly to ProcessSphere, with a three modifications. First, instead of the family from Theorem 3.1, we use the family from Theorem 2.5 which is designed for partitioning the whole $\mathbb{R}^d$ rather than just a sphere. Second, we seek to find clusters of radius $2c^2$. Third, we do not need to find the smallest enclosing ball for $P \cap B(x, 2c^2)$: instead, $B(x, 2c^2)$ itself is enough.

function PROCESS(P)
  $m \leftarrow |P|$
  while $\exists x \in \mathbb{R}^d : |B(x, 2c^2) \cap P| \geq \tau m$ **do**
    PROCESSBALL($P \cap B(x, 2c^2)$, 1, c, x, $2c^2$)
    $P \leftarrow P \setminus B(x, 2c^2)$
  sample $\mathcal{R}$ according to Theorem 2.5
  for $U \in \mathcal{R}$ **do**
    if $P \cap U \neq \emptyset$ **then**
      PROCESS($P \cap U$)

function PROCESSBALL(P, r_1, r_2, o, R)
  if $r_1 + 2R \leq r_2$ **then**
    store any point from P
    return
  $P \leftarrow \{o + \delta \lceil \frac{\|p-o\|}{\delta} \rceil \cdot \frac{p-o}{\|p-o\|} \mid p \in P\}$
  for $i \leftarrow 0 \dots \lceil \frac{R}{\delta} \rceil$ **do**
    $\widetilde{P} \leftarrow \{p \in P : \|p - o\| = \delta i\}$
    if $\widetilde{P} \neq \emptyset$ **then**
      for $j \leftarrow 0 \dots \lceil \frac{R + r_1 + 2\delta}{\delta} \rceil$ **do**
        if $\delta |i - j| \leq r_1 + 2\delta$ **then**
          $\widetilde{r_1} \leftarrow$ PROJECT(δi, δj, $r_1 + 2\delta$)
          $\widetilde{r_2} \leftarrow$ PROJECT(δi, δj, $r_2 - 2\delta$)
          PROCESSSPHERE($\widetilde{P}$, $\widetilde{r_1}$, $\widetilde{r_2}$, o, δi)

function PROJECT(R_1, R_2, r)
  return $\sqrt{R_1(r^2 - (R_1 - R_2)^2)/R_2}$

function PROCESSSPHERE(P, r_1, r_2, o, R)
  if $r_2 \geq 2R$ **then**
    store any point from P
    return
  if $\frac{r_1}{r_2} \leq \frac{1}{2c^2 - 1}$ **then**
    apply Theorem 2.3 with Theorem 2.5 to P
    return
  if $r_2 \geq \sqrt{2}R$ **then**
    apply Theorem 2.3 with Theorem 3.1 to P
    return
  $m \leftarrow |P|$
  $\widehat{R} \leftarrow (\sqrt{2} - \varepsilon)R$
  while $\exists x \in \partial B(o, R) : |B(x, \widehat{R}) \cap P| \geq \tau m$ **do**
    $B(\widetilde{o}, \widetilde{R}) \leftarrow$ the SEB for $P \cap B(x, \widehat{R})$
    PROCESSBALL($P \cap B(x, \widehat{R})$, r_1, r_2, $\widetilde{o}$, $\widetilde{R}$)
    $P \leftarrow P \setminus B(x, \widehat{R})$
  sample $\mathcal{R}$ according to Theorem 3.1
  for $U \in \mathcal{R}$ **do**
    if $P \cap U \neq \emptyset$ **then**
      PROCESSSPHERE($P \cap U$, r_1, r_2, o, R)

Figure 3: Pseudocode of the data structure (SEB stands for *smallest enclosing ball*)

Project. This is implemented by a formula (see Figure 2).

Overall, the preprocessing creates a decision tree, where the nodes correspond to procedures PROCESSSPHERE, PROCESSBALL, PROCESS. We refer to the tree nodes correspondingly, using the labels in the below description of the query algorithm.

Observe that currently the preprocessing is expensive: a priori it is not even clear how to make it *polynomial* in n as we need to search over all possible ball centers o. We address this challenge in Section 6.

Query procedure. Consider a query point $q \in \mathbb{R}^d$. We run the query on the decision tree, starting with the root, and applying the following algorithms depending on the label of the nodes:

- In PROCESS we first recursively query the ball data structures. Second, we locate q in $\mathcal{R}$, and query the data structure we built for $P \cap \mathcal{R}(q)$.
- In PROCESSBALL, we first consider the base case, where we just return the stored point if it is close enough. In general, we check if $\|q - o\| \leq R + r_1$. If not, we can return. Otherwise, we round q so that the distance from o to q is a multiple of δ. Next, we enumerate the distances from o to the potential near neighbor we are looking for, and query the corresponding PROCESSSPHERE children after projecting q on the sphere with a tentative near neighbor (using, naturally, PROJECT).
- In PROCESSSPHERE, we proceed exactly the same way as PROCESS modulo the base cases, which we handle according to Theorem 2.3.

5 Analysis of the data structure

In this section we analyze the above data structure.

5.1 Overview of the analysis

The most interesting part of the analysis is lower bounding the probability of success: we need to show that it is at least $n^{-\rho-o_c(1)}$, where $\rho = \frac{1}{2c^2-1}$. The challenge is that we need to analyze a (somewhat adaptive) random process. In particular, we cannot just use *probability of collision of far points* as is usually done in the analysis of (data-independent) LSH families. Instead, we use its *empirical estimate*: namely, the actual count of points remaining in the part containing the query q (in fact, its expectation). While this allows to make some good-quality progress, this only lasts for a few iterations of partitioning, until we run into the fact that the set is only *pseudo*random and the deviations from the "ideal structure" begin showing up more prominently (which is the reason that, after partitioning we again need again check for dense balls). Furthermore, while computing this empirical estimate, we need to condition on the fact that the near neighbor is colliding with the query.

A bit more formally, the proof proceeds in two steps. First, we show that whenever we apply a partition $\mathcal{R}$ to a pseudorandom remainder P, the partitioning quality is very good: it achieves the exponent of $\ln(1/p_1)/\ln(1/p_2) \leq \frac{1}{2c^2-1} + o_c(1)$ (see Claim 5.5). Here $p_1 = \Pr_{\mathcal{R}}[\mathcal{R}(p) = \mathcal{R}(q)]$ is the probability for the query $q \in \mathbb{R}^d$ and its near neighbor $p \in P$ to collide under $\mathcal{R}$, and

$$p_2 = \mathrm{E}_{\mathcal{R}}\left[\frac{|\mathcal{R}(p) \cap P|}{|P|} \;\middle|\; \mathcal{R}(p) = \mathcal{R}(q)\right]$$

is the (conditioned) empirical estimate of the *fraction of P that collides with p*. Note that, when computing p_2, we condition on the fact that the query and its near neighbor collide (i.e., $\mathcal{R}(p) = \mathcal{R}(q)$). It is exactly this conditioning that requires the three-point property (4) of the Spherical LSH. Furthermore, we use the fact that all the "dense" balls have been carved out, in order to argue that, on average, many points are far away and so p_2 is essentially governed by the collision probability of the Spherical LSH for distances around $\sqrt{2}R$. In the second step, we proceed by lower bounding the probability of success via a careful inductive proof analyzing the corresponding random process (Claim 5.6). Along the way, we use the above estimate crucially. See Section 5.4 for details.

The rest of the analysis proves that the data structure occupies $n^{1+o_c(1)}$ space and has $n^{o_c(1)}$ query time (in expectation). While a bit tedious, this is relatively straightforward. See Section 5.5. Finally, we highlight that obtaining the near-linear preprocessing algorithm requires further ideas and in particular utilizes the van der Corput lemma. See Section 6.

5.2 Setting parameters

Recall that the dimension is $d = \Theta(\log n \cdot \log\log n)$. We set $\varepsilon, \delta, \tau$ as follows:

- $\varepsilon = \frac{1}{\log\log\log n}$;
- $\delta = \exp\bigl(-(\log\log\log n)^C\bigr)$;
- $\tau = \exp\bigl(-\log^{2/3} n\bigr)$,

where C is a sufficiently large positive constant (concrete value of C is only important for the proof of Claim 5.2).

5.3 Invariants and auxiliary properties

We now state and prove several invariants and properties of the data structure that are needed for the subsequent analysis.

Claim 5.1. *In* ProcessSphere *we have* $\widetilde{R} \le (1-\Omega(\varepsilon^2))R$ *(see Figure 1).*

Proof. It is enough to show that for $x \in \partial B(o,R)$ there is a ball of radius $(1-\Omega(\varepsilon^2))R$ that covers $\partial B(o,R) \cap B(x,(\sqrt{2}-\varepsilon)R)$. Without loss of generality we can assume that $o=0$ and $x=(R,0,\dots,0)$. Then, $\partial B(0,R)\cap B(x,(\sqrt{2}-\varepsilon)R) = \{u \in \partial B(0,R)\colon u_1 \ge \eta R\}$, where $\eta = \Theta(\varepsilon)$. At the same time, we can cover $\{u \in \partial B(0,R)\colon u_1 \ge \eta R\}$ with the ball centered in $(\eta R,0,\dots,0)$ and radius $R\sqrt{1-\eta^2} = (1-\Omega(\eta^2))R = (1-\Omega(\varepsilon^2))R$. □

Lemma 5.2. *The following invariants hold.*

- *At every moment of time we have* $\frac{r_2}{r_1} \ge c - o_c(1)$, $r_2 \le c + o_c(1)$ *and* $R \le O_c(1)$*;*
- *After checking for the base case in* ProcessBall *and the first base case in* ProcessSphere *we have* $r_2/R \ge \exp(-O_c(\log\log\log n)^{O(1)})$*;*
- *At every moment of the preprocessing or the query procedures, the number of calls to* ProcessBall *in the recursion stack is* $O_c(\varepsilon^{-O(1)})$*;*
- *The expected length of any contiguous run of calls to* Process *or* ProcessSphere *in the recursion stack is* $O(\log n)$ *(again, during the preprocessing or the query).*

The rest of the Section is devoted to proving Lemma 5.2.

Consider the recursion stack at any moment of the preprocessing or the query algorithms. It has several calls to ProcessBall interleaved with sequences of calls to Process and ProcessSphere. Our current goal is to bound the number of calls to ProcessBall that can appear in the recursion stack at any given moment (we want to bound it by $O_c(1/\varepsilon^6)$). First, we prove that this bound holds under the (unrealistic) assumption that in ProcessBall the rounding of distances has no effect (that is, all the distances of points to o are already multiples of δ). Second, we prove the bound in full generality by showing that this rounding introduces only a tiny multiplicative error to r_1, r_2 and R.

Claim 5.3. *Suppose that the rounding of distances in* ProcessBall *has no effect (i.e., distances from o to all points are multiples of δ). Then the number of the calls to* ProcessBall *in the recursion stack at any given moment of time is* $O_c(1/\varepsilon^6)$.

Proof. Let us keep track of two quantities: $\eta = r_2^2/r_1^2$ and $\xi = r_2^2/R^2$. It is immediate to see that the initial value of η is c^2, it is non-decreasing (it can only change, when we apply Project, which can only increase the ratio between r_2 and r_1), and it is at most $(2c^2-1)^2$ (otherwise, the base case in ProcessSphere is triggered). Similarly, ξ is initially equal to $1/(4c^2)$ and it can be at most 2 (otherwise, the base in ProcessSphere is triggered). Unlike η, the value of ξ *can* decrease.

Suppose that in ProcessBall we call ProcessSphere for some $R_1 = \delta i$ and $R_2 = \delta j$ with $|R_1 - R_2| = \Delta R$. Suppose that $\widetilde{\eta}$ is the new value of η after this call. We have

$$\frac{\widetilde{\eta}}{\eta} = \frac{\widetilde{r_2}^2/\widetilde{r_1}^2}{r_2^2/r_1^2} = \frac{\textsc{Project}(R_1,\, R_2,\, r_2)^2/\textsc{Project}(R_1,\, R_2,\, r_1)^2}{r_2^2/r_1^2} = \frac{(r_2^2-\Delta R^2)/(r_1^2-\Delta R^2)}{r_2^2/r_1^2}$$
$$= \frac{1-\frac{\Delta R^2}{r_2^2}}{1-\frac{\Delta R^2}{r_1^2}} = \frac{1-\frac{\Delta R^2}{r_2^2}}{1-\eta\cdot\frac{\Delta R^2}{r_2^2}} = 1+\Omega_c\Big(\frac{\Delta R^2}{r_2^2}\Big), \quad (6)$$

where the third step follows from the formula for PROJECT and the last step follows from the fact that $\eta \geq c^2 > 1$.

Let us call an invocation of PROCESSSPHERE within PROCESSBALL λ-*shrinking* for some $\lambda > 0$, if $\Delta R^2 / r_2^2 \geq \lambda$. From (6), the fact that $\eta \in [c^2; (2c^2-1)^2]$ and that η is non-decreasing, we conclude that there can be at most $O_c(1/\lambda)$ λ-shrinking invocations of PROCESSSPHERE in the recursive stack at any given moment of time.

Now let us see how ξ evolves. Suppose that within some PROCESSBALL we call PROCESSSPHERE with some R_1 and R_2. Then, in PROCESSSPHERE we call PROCESSSPHERE recursively several times without any change in r_1, r_2 and R, and finally we call PROCESSBALL again. Denote $\widetilde{\xi} = \frac{\widetilde{r_2}^2}{\widetilde{R}^2}$ the new value of ξ after this call of PROCESSBALL. We have

$$\frac{\widetilde{\xi}}{\xi} = \frac{\widetilde{r_2}^2/\widetilde{R}^2}{r_2^2/R^2} \geq (1+\Omega(\varepsilon^2)) \frac{\widetilde{r_2}^2/R_1^2}{r_2^2/R^2} = (1+\Omega(\varepsilon^2)) \frac{(r_2^2 - \Delta R^2)/(R_1 R_2)}{r_2^2/R^2}$$
$$= (1+\Omega(\varepsilon^2)) \Big(1 - \frac{\Delta R^2}{r_2^2}\Big) \frac{R^2}{R_1 R_2} \geq (1+\Omega(\varepsilon^2)) \Big(1 - \frac{\Delta R^2}{r_2^2}\Big) \frac{1}{1+\frac{\Delta R}{R}}, \quad (7)$$

where the second step follows from Claim 5.1, the third step follows from the formula for PROJECT, the fifth step follows from the fact that $R_1 \leq R$ and $R_2 \leq R_1 + \Delta R \leq R + \Delta R$.

Denote $\lambda^* = \varepsilon^4 / C$ for a sufficiently large constant $C > 0$. If the call to PROCESSSPHERE within PROCESSBALL is not λ^*-shrinking (that is, $\Delta R / r_2 \leq \varepsilon^2 / \sqrt{C}$), then since

$$\frac{\Delta R}{R} = \frac{\Delta R}{r_2} \cdot \sqrt{\xi} \leq \frac{\sqrt{2} \cdot \Delta R}{r_2} \leq \sqrt{\frac{2}{C}} \cdot \varepsilon^2,$$

where we use $\xi \leq 2$, from (7) we have that $\widetilde{\xi}/\xi = 1 + \Omega(\varepsilon^2)$ (provided that C is sufficiently large). On the other hand, if the call is λ^*-shrinking, then since

$$\frac{\Delta R^2}{r_2^2} \leq \frac{r_1^2}{r_2^2} \leq \frac{1}{c^2},$$

and

$$\frac{\Delta R}{R} \leq \frac{r_1}{R} = \frac{r_1}{r_2} \cdot \frac{r_2}{R} \leq \frac{\sqrt{2}}{c},$$

we have from (7)

$$\frac{\widetilde{\xi}}{\xi} \geq \Big(1 - \frac{1}{c^2}\Big) \frac{1}{1+\sqrt{2}/c} = \Omega_c(1),$$

since $c > 1$. That being said, non-λ^*-shrinking calls increase ξ non-trivially (by at least $(1+\Omega(\varepsilon^2))$), while λ^*-shrinking calls decrease ξ by not too much, and by the above discussion, there are at most $O_c(1/\lambda^*) = O_c(1/\varepsilon^4)$ of them.

More formally, suppose we have A calls to PROCESSSPHERE that are λ^*-shrinking and B calls that are not λ^*-shrinking. We have that $A = O_c(1/\lambda^*) = O_c(1/\varepsilon^4)$. On the other hand, since every λ^*-shrinking call multiplies ξ by at least $\Omega_c(1)$, the initial value of ξ is $1/(4c^2)$, and the final value of ξ is at most 2, we have

$$(1+\Omega(\varepsilon^2))^B \leq \exp(O_c(A)),$$

thus, $B = O_c(1/\varepsilon^6)$. Overall, we have at most $A + B \leq O_c(1/\varepsilon^6)$ invocations of PROCESSBALL in the recursion stack at any moment of time. □

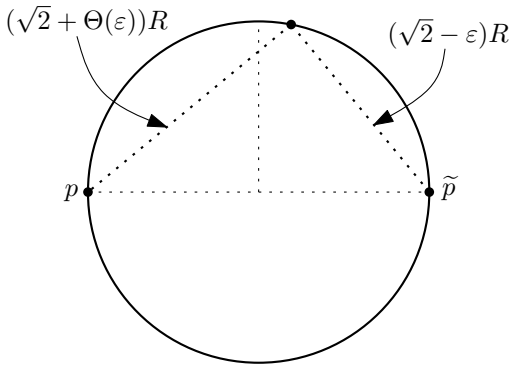


Figure 4: For the proof of Claim 5.5: distances to points in U

Claim 5.4. *Suppose that the rounding of distances in* ProcessBall *has no effect (i.e., distances from o to all points are multiples of δ). At any moment of time, in* ProcessBall *and* ProcessSphere *outside of the base cases one has $r_1, r_2, R \geq \exp(-O_c(1/\varepsilon^{10}))$.*

Proof. By the proof of Claim 5.3, r_2/r_1 is $\Theta_c(1)$ and $r_2/R \in [\exp(-O_c(1/\varepsilon^4)); \sqrt{2}]$.

After calling ProcessSphere within ProcessBall (and vice versa) the new value of R is at least $\Omega(r_1)$, since otherwise the first base case in ProcessSphere (or the base case of ProcessBall) will be triggered.

So, overall we have $O_c(1/\varepsilon^6)$ calls to ProcessBall in the recursion stack each of which can decrease r_1, r_2, R by a factor of at most $\exp(O_c(1/\varepsilon^4))$. Hence the claim. □

Since $\varepsilon = 1/\log\log\log n$, we can choose C in the definition of δ (see Section 5.2) so that, by Claim 5.4, rounding distance to multiples of δ gives only a small *multiplicative* change to r_1, r_2, R that accumulates to $1 + o_c(1)$ over $O_c(1/\varepsilon^6)$ iterations.

This way, we obtain all the items in Claim 5.2, except the last one (by staring at the above proofs and taking the previous paragraph into account).

Let us show the last item for Process (for ProcessSphere the proof is the same verbatim). Let us look at any data point $p \in P_0$. Suppose that p ends up in a pseudorandom remainder. Then, there are only τn points in the $2c^2$-neighborhood of p. When we sample $\mathcal{R}$, the number of points outside this neighborhood multiplies by a constant stricly smaller than one (in expectation). Thus, overall, the number of points in P multiplies by a constant smaller than one on average every call to Process. It means that in $O(\log n)$ with high probability either p will be captured by a dense cluster or it will be the only remaining point.

5.4 Probability of success

We now analyze the probability of success for a query $q \in \mathbb{R}^d$ for which there exists $p \in P_0$ with $\|q - p\| \leq 1$. We perform this analysis in two steps. First, we upper bound $\mathrm{E}_{\mathcal{R}}[|P \cap \mathcal{R}(q)| \mid \mathcal{R}(p) = \mathcal{R}(q)]$ in Process and ProcessSphere provided that $p \in P$ after removing dense clusters. This upper bound formalizes the intuition that after removing dense clusters the remaining pseudorandom instance becomes easy to partition. While upper bounding this expectation for ProcessSphere, we crucially rely on the estimate (4) for triples of points from Theorem 3.1 (see also the remark after the proof of Claim 5.5). Second, we use this estimate to lower bound the overall probability of success by analyzing the corresponding random process.

Claim 5.5. *Suppose that in* Process *or* ProcessSphere*, we have $p \in P$ after removing all dense balls. Let $q \in \mathbb{R}^d$ be such that $\|p-q\| \le 1$ for* Process *or $\|p-q\| \le r_1$ and $q \in \partial B(o,R)$ for* ProcessSphere*. Then, after sampling $\mathcal{R}$, we have*

$$\frac{\ln(1/p_1)}{\ln(1/p_2)} \le \frac{1}{2c^2-1} + o_c(1), \tag{8}$$

where $p_1 = \Pr_{\mathcal{R}}[\mathcal{R}(p) = \mathcal{R}(q)]$, and $p_2 = \mathrm{E}_{\mathcal{R}}\left[\frac{|P \cap \mathcal{R}(q)|}{m} \mid \mathcal{R}(p) = \mathcal{R}(q)\right]$, where m is the size of P at the beginning of Process *or* ProcessSphere*.*

Proof. Let us first analyze Process, for which the analysis is somewhat simpler. In this case, by Theorem 2.5, we have

$$\ln(1/p_1) = \ln \frac{1}{\Pr_{\mathcal{R}}[\mathcal{R}(p) = \mathcal{R}(q)]} = (1+o(1))\sqrt{d}. \tag{9}$$

On the other hand,

$$\begin{aligned} p_2 = \frac{\mathrm{E}_{\mathcal{R}}[|P \cap \mathcal{R}(q)| \mid \mathcal{R}(p) = \mathcal{R}(q)]}{m} &\le \tau + \frac{\mathrm{E}_{\mathcal{R}}\left[|(P \cap \mathcal{R}(q)) \setminus B(q, 2c^2)| \mid \mathcal{R}(p) = \mathcal{R}(q)\right]}{m} \\ &\le \tau + \sup_{p' \in P \setminus B(q,2c^2)} \Pr_{\mathcal{R}}[\mathcal{R}(p') = \mathcal{R}(q) \mid \mathcal{R}(p) = \mathcal{R}(q)] \\ &\le \tau + \frac{\sup_{p' \in P \setminus B(q,2c^2)} \Pr_{\mathcal{R}}[\mathcal{R}(p') = \mathcal{R}(q)]}{\Pr_{\mathcal{R}}[\mathcal{R}(p) = \mathcal{R}(q)]} \\ &\le \tau + e^{-(2c^2-1)\sqrt{d}\cdot(1+o_c(1))} \\ &\le e^{-(2c^2-1)\sqrt{d}\cdot(1+o_c(1))}, \end{aligned} \tag{10}$$

where the first step follows from the fact that after removing the dense clusters we have $|\mathcal{R} \cap B(q, 2c^2)| < \tau m$, the second step follows from the linearity of expectation, the fourth step follows from Theorem 2.5 and the last step uses that $d = \Theta(\log n \log\log n)$ and $\tau = \exp(-\log^{2/3} n)$. Now, combining (9) and (10), we get

$$\frac{\ln(1/p_1)}{\ln(1/p_2)} \le \frac{1}{2c^2-1} + o_c(1)$$

as desired.

Now let us analyze ProcessSphere. By Claim 5.2 and the fact that $r_2/r_1 \le O_c(1)$ (otherwise, we would have triggered the second base case in ProcessSphere), we have $r_1/R \ge \exp(-O_c(\log\log\log n)^{O(1)})$, and $d = \Theta(\log n \log\log n)$, so we are in position to apply Theorem 3.1. We have

$$\begin{aligned} \ln(1/p_1) &\le \frac{(r_1/R)^2}{4-(r_1/R)^2} \cdot \frac{\sqrt{d}}{2} \cdot (1+o(1)) \\ &\le \frac{1}{2c^2-1} \cdot \frac{\sqrt{d}}{2} \cdot (1+o_c(1)), \end{aligned} \tag{11}$$

where the second step follows from the estimate

$$\frac{r_1}{R} = \frac{r_1}{r_2} \cdot \frac{r_2}{R} \le \frac{\sqrt{2}}{c} \cdot (1+o_c(1)),$$

where the second step is due to the assumptions $r_2/r_1 \geq c - o_c(1)$ (which is true by Claim 5.2) and $r_2 \leq \sqrt{2}R$ (if the latter was not the case, we would have triggered the third base case in ProcessSphere). Let $\widetilde{p}$ and $\widetilde{q}$ be reflections of p and q respectively with respect to o. Define

$$\begin{aligned} U = B(q, (\sqrt{2}-\varepsilon)R) \cup B(\widetilde{q}, (\sqrt{2}-\varepsilon)R) \\ \cup B(p, (\sqrt{2}-\varepsilon)R) \cup B(\widetilde{p}, (\sqrt{2}-\varepsilon)R). \end{aligned}$$

Then,

$$\begin{aligned} p_2 = \frac{\mathrm{E}_{\mathcal{R}}\left[|P \cap \mathcal{R}(q)| \mid \mathcal{R}(p) = \mathcal{R}(q)\right]}{m} &\leq 4\tau + \frac{\mathrm{E}_{\mathcal{R}}\left[|(P \cap \mathcal{R}(q)) \setminus U| \mid \mathcal{R}(p) = \mathcal{R}(q)\right]}{m} \\ &\leq 4\tau + \sup_{p' \in P \setminus U} \Pr_{\mathcal{R}}\left[\mathcal{R}(p') = \mathcal{R}(q) \mid \mathcal{R}(p) = \mathcal{R}(q)\right] \\ &\leq 4\tau + e^{-\frac{\sqrt{d}}{2} \cdot (1+o(1))} \\ &\leq e^{-\frac{\sqrt{d}}{2} \cdot (1+o(1))}, \end{aligned} \tag{12}$$

where the first step follows from the definition of U and that we have all the dense clusters removed, the second step follows from the linearity of expectation, the third step follows from Theorem 3.1 (namely, (4)) and the fact that all the points from $P \setminus U$ are within $(\sqrt{2} \pm \Theta(\varepsilon))R$ from *both* p and q (see Figure 4), and the fourth step uses the values for d and τ. Overall, combining (11) and (12), we get

$$\frac{\ln(1/p_1)}{\ln(1/p_2)} \leq \frac{1}{2c^2-1} + o_c(1).$$

□

Remark: If instead of (4) we used (5), then the best we could hope for would be

$$\frac{\ln(1/p_1)}{\ln(1/p_2)} \leq \frac{1}{2c^2-2} + o_c(1),$$

which is much worse than (8), if c is close to 1.

Claim 5.6. *Suppose that $p \in P_0$ and $q \in \mathbb{R}^d$ with $\|p-q\| \leq 1$. Then, the probability of success of the data structure is at least*

$$n^{-\frac{1}{2c^2-1} - o_c(1)}.$$

Proof. Let us prove by induction that any query of a data structure built by Process, ProcessBall and ProcessSphere with $p \in P$ has probability of success at least $|P|^{-\rho} \cdot n^{-\alpha}$, where $\rho \leq 1/(2c^2-1) + o_c(1)$ and $\alpha = o_c(1)$. First, we prove this for Process assuming that the same bound holds for ProcessBall. Then, we argue that for ProcessBall and ProcessSphere essentially the same argument works as well.

Let us denote $f(m)$ a lower bound on the probability of success for Process when $|P| \leq m$ and denote p_1 and p_2 the quantities from Claim 5.5 introduced for Process.

Let us lower bound $f(m)$ by induction. If p belongs to one of the dense clusters, then $f(m) \geq$

$m^{-\rho} \cdot n^{-\alpha}$ by the assumption for ProcessBall. If p does not belong to any dense cluster, we have

$$\begin{aligned}
f(m) &\geq \mathrm{E}_{\mathcal{R}}\left[f(|P \cap \mathcal{R}(q)|) \cdot \mathbf{1}_{\mathcal{R}(p)=\mathcal{R}(q)}\right] \\
&= \Pr_{\mathcal{R}}\left[\mathcal{R}(p) = \mathcal{R}(q)\right] \cdot \mathrm{E}_{\mathcal{R}}\left[f(|P \cap \mathcal{R}(q)|) \mid \mathcal{R}(p) = \mathcal{R}(q)\right] \\
&\geq p_1 \cdot \inf_{\substack{\operatorname{supp} X \subseteq [m]: \\ \mathrm{E}[X] \leq p_2 \cdot m}} \mathrm{E}\left[f(X)\right] \\
&\geq p_1 \cdot \inf_{\substack{\operatorname{supp} X \subseteq [m]: \\ \mathrm{E}[X] \leq p_2 \cdot m}} \Pr\left[X < m\right] \cdot \mathrm{E}\left[f(X) \mid X < m\right] \\
&\geq p_1(1-p_2) \cdot \inf_{\substack{\operatorname{supp} Y \subseteq [m-1]: \\ \mathrm{E}[Y] \leq p_2 \cdot m}} \mathrm{E}\left[f(Y)\right] \\
&\geq p_1(1-p_2)n^{-\alpha} \cdot \inf_{\substack{\operatorname{supp} Y \subseteq [m-1]: \\ \mathrm{E}[Y] \leq p_2 \cdot m}} \mathrm{E}\left[Y^{-\rho}\right] \\
&\geq p_1(1-p_2)n^{-\alpha} \cdot \inf_{\substack{\operatorname{supp} Y \subseteq [m-1]: \\ \mathrm{E}[Y] \leq p_2 \cdot m}} \mathrm{E}\left[Y\right]^{-\rho} \\
&\geq p_1(1-p_2)p_2^{-\rho}m^{-\rho}n^{-\alpha},
\end{aligned}$$

where the third step is due to the definition of p_1 and p_2, the fifth step is due to Markov's inequality, the sixth step is by the induction hypothesis and the seventh step is due to Jensen's inequality. For the induction to go through we must have

$$p_1(1-p_2)p_2^{-\rho} \geq 1, \tag{13}$$

so by Claim 5.5 and the fact that $p_2 \leq 1 - \Omega(1)$ (which follows from Theorem 2.5), we can set $\rho \leq \frac{1}{2c^2-1} + o_c(1)$.

For ProcessBall and ProcessSphere we can perform a similar induction with two modifications. First, we need to verify the bound for the base cases in ProcessSphere (in particular, this will determine the value of α). Second, since ProcessBall and ProcessSphere depend on each other, we need to carry on the "outer" induction on the number of calls to ProcessBall in the recursion stack. The latter issue is easy to address, since by Claim 5.2 the maximum number of calls to ProcessBall in the recursion stack is bounded, and from the above estimate it is immediate to see that there is no "error" that would accumulate. Thus, it is only left to look at the base cases of ProcessSphere. The first base case is trivial. For the remaining cases we use Theorem 2.3. For the second base case we use the Theorem 2.5: we are in position to apply it, since by Claim 5.2 $r_2 \leq c + o_c(1)$. As for the third case, since by Claim 5.2 and the fact that for the third case $r_2/r_1 = O_c(1)$ one has that r_1 is not too small compared to R, we are in position to apply Theorem 3.1. Since by Claim 5.2 $r_2/r_1 \geq c - o_c(1)$ and by the assumption $r_2 \geq \sqrt{2}R$, applying Theorem 3.1, we get the required bound on the probability of success. □

5.5 Space and query time

In this section we show that the expected space the data structure occupies is $n^{1+o_c(1)}$ and the expected query time is $n^{o_c(1)}$.

Claim 5.7. *The overall expected space the data structure occupies is $n^{1+o_c(1)}$.*

Proof. Every particular point $p \in P_0$ can participate in several branches of recursion during the preprocessing: the reason for this is ProcessBall, where every data point participates in many

sphere data structures. But observe that by Claim 5.2 every point can end up in at most

$$O_c\Big(\frac{1}{\delta}\Big)^{O_c(\varepsilon^{-O(1)})} = n^{o_c(1)} \tag{14}$$

branches, since there are at most $O_c(\varepsilon^{-O(1)})$ calls to ProcessBall in every branch, and each such call introduces branching factor of at most $O((R+r_1+2\delta)/\delta) = O_c(1/\delta)$.

Next, for every point $p \in P_0$ and for every branch it participates in, one can see that by Claim 5.2 the total expected number of calls in the stack is at most

$$O_c\Big(\frac{\log n}{\varepsilon^{O(1)}}\Big) = n^{o_c(1)}, \tag{15}$$

since the number of ProcessBall's is at most $O_c(\varepsilon^{-O(1)})$ they separate the runs of Process and ProcessSphere, each of which is of length $O(\log n)$ in expectation.

Since every partition $\mathcal{R}$ sampled in Process or ProcessSphere takes $n^{o_c(1)}$ space by Theorem 2.5 and Theorem 3.1, we get, combining (14) and (15), that the space consumption of partitions and hash tables per point is $n^{o_c(1)}$. Also, the base cases in ProcessSphere are cheap too: indeed, from Theorem 2.3 (coupled with Theorem 3.1 and Theorem 2.5) we see that the space consumption for the base cases is $n^{o_c(1)}$ per point per branch.

Thus, the overall bound $n^{1+o_c(1)}$ follows. □

Claim 5.8. *For every query $q \in \mathbb{R}^d$, the expected query time is at most $n^{o_c(1)}$.*

Proof. Consider a recursion tree for a particular query $q \in \mathbb{R}^d$.

First, observe that each sequence of calls to Process or ProcessSphere can spawn at most $O(\tau^{-1} \cdot \log n)$ calls to ProcessBall, since every such call multiplies the number of remaining points by a factor of at most $(1-\tau)$. At the same time by Claim 5.2, each such sequence is of size $O(\log n)$ in expectation.

Since by Claim 5.2 in the recursion stack there can be at most $O_c(\varepsilon^{-O(1)})$ calls to ProcessBall, which induces the branching factor of at most $O_c(1/\delta)$, overall, the expected number of nodes in the tree is at most

$$\Big(\frac{\log n}{\delta\tau}\Big)^{O_c(\varepsilon^{-O(1)})} = n^{o_c(1)}.$$

In every node we need to do one point location in a partition, which by Theorem 2.5 and Theorem 3.1 can be done in time $n^{o_c(1)}$, and then for the base cases we have the expected query time $n^{o_c(1)}$ (by Theorem 2.3). Thus, the overall expected query time is $n^{o_c(1)}$. □

6 Fast preprocessing

A priori, it is not clear how to implement the preprocessing in polynomial time, let alone near-linear. We will first show how to get preprocessing time to $n^{2+o_c(1)}$ and then reduce it to $n^{1+o_c(1)}$.

6.1 Near-quadratic time

To get preprocessing time $n^{2+o_c(1)}$ we need to observe that during the clustering step in Process and ProcessSphere we can look only for balls with centers being points from P.

For Process it is easy to see that we can find balls of radius $4c^2$ with centers being points from P. Then, the proofs of Claim 5.5 and, as a result, Claim 5.6 go through (we use that, as a result of such a preprocessing, there are no dense clusters of radius $2c^2$ with *arbitrary* centers). Also, we

need to make sure that Claim 5.2 is still true, despite we start with clusters of radius $4c^2$ instead of $2c^2$, but that is straightforward.

For ProcessSphere the situation is slightly more delicate, since we can not afford to lose a factor of two in the radius here. We build upon the following Lemma.

Proposition 6.1 (van der Corput Lemma)**.** *For any $v^*, v_1, v_2, \ldots, v_n \in S^{d-1}$ one has*

$$\sum_{i,j} \langle v_i, v_j \rangle \geq \Big| \sum_i \langle v^*, v_i \rangle \Big|^2.$$

Proof. We have

$$\Big| \sum_i \langle v^*, v_i \rangle \Big|^2 = \Big| \langle v^*, \sum_i v_i \rangle \Big|^2 \leq \|v^*\|^2 \cdot \| \sum_i v_i \|^2 = \| \sum_i v_i \|^2 = \sum_{i,j} \langle v_i, v_j \rangle,$$

where the second step is an application of the Cauchy-Schwartz inequality. □

The following Claim is the main estimate we use to analyze the variant of ProcessSphere, where we are looking only for clusters with centers in data points. Informally, we prove that if a non-trivially small spherical cap covers n points, then there is a non-trivially small cap *centered in one of the points* that covers a substantial fraction of points.

Claim 6.2. *Fix $\varepsilon > 0$. Suppose that $U \subset S^{d-1}$ with $|U| = n$. Suppose that there exists $u^* \in S^{d-1}$ such that $\|u^* - u\| \leq \sqrt{2} - \varepsilon$ for every $u \in U$. Then, there exists $u_0 \in U$ such that*

$$\Big| \Big\{ u \in U \colon \|u - u_0\| \leq \sqrt{2} - \Omega(\varepsilon^2) \Big\} \Big| \geq \Omega(\varepsilon^2 n).$$

Proof. First, observe that $\|u^* - u\| \leq \sqrt{2} - \varepsilon$ iff $\langle u^*, u \rangle \geq \Omega(\varepsilon)$. By van der Corput Lemma (Proposition 6.1),

$$\sum_{u,v \in U} \langle u, v \rangle \geq \left| \sum_{u \in U} \langle u^*, u \rangle \right|^2 \geq \Omega(\varepsilon^2 n^2).$$

Thus, there exists $u_0 \in U$ such that

$$\sum_{u \in U} \langle u_0, u \rangle \geq \Omega(\varepsilon^2 n).$$

This implies

$$\Big| \Big\{ u \in U \mid \langle u_0, u \rangle \geq \Omega(\varepsilon^2) \Big\} \Big| \geq \Omega(\varepsilon^2 n),$$

which is equivalent to

$$\Big| \Big\{ u \in U \mid \|u - u_0\| \leq \sqrt{2} - \Omega(\varepsilon^2) \Big\} \Big| \geq \Omega(\varepsilon^2 n).$$

□

It means that if in ProcessSphere we search for clusters of radius $\sqrt{2} - \Omega(\varepsilon^2)$ centered in data points that cover at least $\Theta(\varepsilon^2 \tau m)$ points, then after we remove all of them, we are sure that there are no clusters of radius $\sqrt{2} - \varepsilon$ with *arbitrary* centers that cover at least τm points, so Claims 5.5 and 5.6 go through. The proof of Claim 5.2 needs to be adjusted accordingly, but we claim that by setting C in the definition of δ large enough, the proof of Claim 5.2 still goes through.

It is easy to see that by reducing the time of each clustering step to near-quadratic, we reduce the total preprocessing time to $n^{2+o_c(1)}$. This follows from the proof of Claim 5.7 (intuitively, each point participates in $n^{o_c(1)}$ instances of the clustering subroutine) and from the fact that it takes time $n^{o_c(1)}$ to sample a partition $\mathcal{R}$ in Process and ProcessSphere.

6.2 Near-linear time

To get $n^{1+o_c(1)}$ preprocessing time, we just sample the dataset and compute dense balls in the sample. Indeed, since we care about the clusters with at least $\varepsilon^2\tau n = n^{1-o_c(1)}$ data points, we can sample $n^{o_c(1)}$ points from the dataset and find dense clusters for the sample. Then, using the fact that the VC-dimension for balls in $\mathbb{R}^d$ is $O(d)$, we can argue that this sample is accurate enough with probability at least $1 - n^{-10}$. Then, taking the union bound over all clusterings, we are done.

Acknowledgments

We thank Piotr Indyk and Sepideh Mahabadi for insightful discussions about the problem and for reading early drafts of this write-up. In particular, discussions with them led us to the fast preprocessing algorithm.

References

[AAKK14] Amirali Abdullah, Alexandr Andoni, Ravindran Kannan, and Robert Krauthgamer. Spectral approaches to nearest neighbor search. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS '2014)*, 2014.

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

[AI08] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

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

[And09] Alexandr Andoni. *Nearest Neighbor Search: the Old, the New, and the Impossible*. PhD thesis, Massachusetts Institute of Technology, 2009.

[AR15] Alexandr Andoni and Ilya Razenshteyn. Data-dependent hashing lower bounds. 2015.

[Cla88] Kenneth L. Clarkson. A randomized algorithm for closest-point queries. *SIAM Journal on Computing*, 17(4):830–847, 1988.

[Cou13] National Research Council. *Frontiers in Massive Data Analysis*. The National Academies Press, Washington, DC, 2013.

[DF08] Sanjoy Dasgupta and Yoav Freund. Random projection trees and low dimensional manifolds. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 537–546. ACM, 2008.

[DG03] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures and Algorithms*, 22(1):60–65, 2003.

[DIIM04] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th ACM Symposium on Computational Geometry (SoCG '2004)*, pages 253–262, 2004.

[Dub10] Moshe Dubiner. Bucketing coding and information theory for the statistical highdimensional nearest-neighbor problem. *IEEE Transactions on Information Theory*, 56(8):4166–4179, 2010.

[FKS84] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with O(1) worst case access time. *Journal of the ACM*, 31(3):538–544, 1984.

[HPIM12] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, 2012.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the 30th ACM Symposium on the Theory of Computing (STOC '1998)*, pages 604–613, 1998.

[JL84] William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Connecticut, 1982)*, volume 26 of *Contemporary Mathematics*, pages 189–206. 1984.

[KMS98] David R. Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM*, 45(2):246–265, 1998.

[LLR95] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[McN01] James McNames. A fast nearest-neighbor algorithm based on a principal axis search tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):964–976, 2001.

[Mei93] Stefan Meiser. Point location in arrangements of hyperplanes. *Information and Computation*, 106(2):286–303, 1993.

[MNP07] Rajeev Motwani, Assaf Naor, and Rina Panigrahy. Lower bounds on locality sensitive hashing. *SIAM Journal on Discrete Mathematics*, 21(4):930–935, 2007.

[OWZ11] Ryan O'Donnell, Yi Wu, and Yuan Zhou. Optimal lower bounds for locality sensitive hashing (except when q is tiny). In *Proceedings of Innovations in Computer Science (ICS '2011)*, pages 275–283, 2011.

[PTW08] Rina Panigrahy, Kunal Talwar, and Udi Wieder. A geometric approach to lower bounds for approximate near-neighbor search and partial match. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2008)*, pages 414–423, 2008.

[PTW10] Rina Panigrahy, Kunal Talwar, and Udi Wieder. Lower bounds on near neighbor search via metric expansion. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS '2010)*, pages 805–814, 2010.

[Raz14] Ilya Razenshteyn. Beyond Locality-Sensitive Hashing. Master's thesis, MIT, 2014.

[SH09] Ruslan Salakhutdinov and Geoffrey E. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[Spr91] Robert F. Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica*, 6(1-6):579–589, 1991.

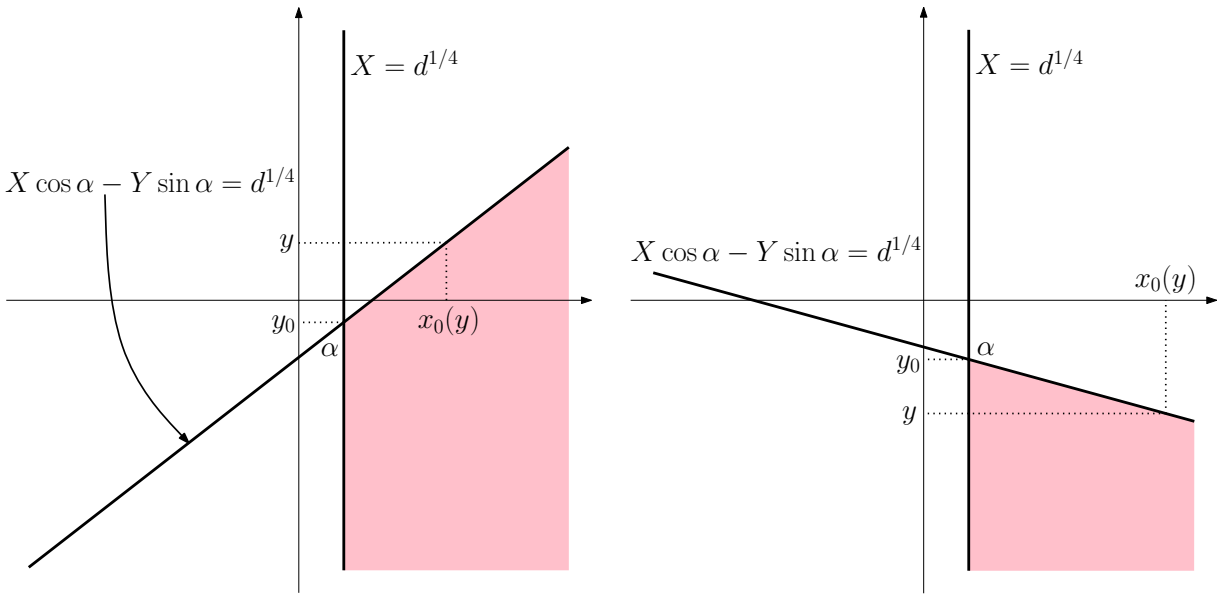
[Val12] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '2012)*, pages 11–20, 2012.

[VKD09] Nakul Verma, Samory Kpotufe, and Sanjoy Dasgupta. Which spatial partition trees are adaptive to intrinsic dimension? In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI '2009)*, pages 565–574, 2009.

[WSSJ14] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *CoRR*, abs/1408.2927, 2014.

[WTF08] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Proceedings of 22nd Annual Conference on Neural Information Processing Systems (NIPS '2008)*, pages 1753–1760, 2008.

[YSRL11] Jay Yagnik, Dennis Strelow, David A. Ross, and Ruei-Sung Lin. The power of comparative reasoning. In *Proceedings of 13th IEEE International Conference on Computer Vision (ICCV '2011)*, pages 2431–2438, 2011.



(a) The case $\alpha < \pi/2$

(b) The case $\alpha > \pi/2$

Figure 5: The region corresponding to $X \geq d^{1/4}$ and $X\cos\alpha - Y\sin\alpha \geq d^{1/4}$

A Analysis of Spherical LSH

In this section we prove Theorem 3.1. We will use the following basic estimate from [KMS98] repeatedly.

Lemma A.1 ([KMS98])**.** *For every $t > 0$*

$$\frac{1}{\sqrt{2\pi}} \cdot \left(\frac{1}{t} - \frac{1}{t^3}\right) \cdot e^{-t^2/2} \leq \Pr_{X\sim N(0,1)}[X \geq t] \leq \frac{1}{\sqrt{2\pi}} \cdot \frac{1}{t} \cdot e^{-t^2/2}.$$

A.1 Collision probability for a pair

It is immediate to see that for every $u, v \in S^{d-1}$ with angle α between them we have

$$\begin{aligned}\Pr_{\mathcal{R}}[\mathcal{R}(u) = \mathcal{R}(v)] &= \frac{\Pr_{g\sim N(0,1)^d}\left[\langle u, g\rangle \geq d^{1/4} \text{ and } \langle v, g\rangle \geq d^{1/4}\right]}{\Pr_{g\sim N(0,1)^d}\left[\langle u, g\rangle \geq d^{1/4} \text{ or } \langle v, g\rangle \geq d^{1/4}\right]} \\ &\in [0.5; 1] \cdot \frac{\Pr_{X,Y\sim N(0,1)}\left[X \geq d^{1/4} \text{ and } X\cos\alpha - Y\sin\alpha \geq d^{1/4}\right]}{\Pr_{X\sim N(0,1)}\left[X \geq d^{1/4}\right]},\end{aligned}$$

where the second step follows from the 2-stability and spherical symmetry of Gaussians. By Lemma A.1,

$$\Pr_{X\sim N(0,1)}\left[X \geq d^{1/4}\right] = (1 + O(d^{-1/2})) \cdot \frac{e^{-\sqrt{d}/2}}{(2\pi)^{1/2} d^{1/4}},$$

so we have

$$\Pr_{\mathcal{R}}[\mathcal{R}(u) = \mathcal{R}(v)] = \Theta(d^{1/4}) \cdot \frac{\Pr_{X,Y\sim N(0,1)}\left[X \geq d^{1/4} \text{ and } X\cos\alpha - Y\sin\alpha \geq d^{1/4}\right]}{e^{-\sqrt{d}/2}}. \tag{16}$$

Thus, estimating $\Pr_{\mathcal{R}}[\mathcal{R}(u)=\mathcal{R}(v)]$ boils down to computing the Gaussian measure of a simple two-dimensional set (see Figure 5). For $d\in\mathbb{N}$, $0\le\alpha\le\pi$ denote

$$S(d,\alpha):=\Pr_{X,Y\sim N(0,1)}\left[X\ge d^{1/4}\text{ and }X\cos\alpha-Y\sin\alpha\ge d^{1/4}\right].$$

Claim A.2. *For every d the function $S(d,\alpha)$ is non-increasing in α.*

Proof. One can check that for every $d\in\mathbb{N}$ and $0\le\alpha'\le\alpha''\le\pi$ we have

$$\left\{(x,y)\colon x\ge d^{1/4}\text{ and }x\cos\alpha'-y\sin\alpha'\ge d^{1/4}\right\}\supseteq\left\{(x,y)\colon x\ge d^{1/4}\text{ and }x\cos\alpha''-y\sin\alpha''\ge d^{1/4}\right\},$$

hence the claim. □

Our next goal will be estimating $S(d,\alpha)$ for $\alpha\in[d^{-\Omega(1)};\pi-d^{-\Omega(1)}]$ within a factor polynomial in d. We would like to claim that $S(d,\alpha)$ is close to $\widetilde{S}(d,\alpha):=\Pr_{X,Y\sim N(0,1)}\left[X\ge d^{1/4}\text{ and }Y\le y_0\right]$, where $y_0=-d^{1/4}\tan\frac{\alpha}{2}$ is the y-coordinate of the intersection of two lines (see Figure 5). But first let us compute $\widetilde{S}(d,\alpha)$: this is easy due to the independence of X and Y.

Claim A.3. *If $\alpha=\Omega(d^{-1/5})$, then*

$$\widetilde{S}(d,\alpha)\in\frac{1\pm d^{-\Omega(1)}}{2\pi d^{1/2}\tan\frac{\alpha}{2}}\exp\left(-\left(1+\tan^2\frac{\alpha}{2}\right)\frac{\sqrt{d}}{2}\right).$$

Proof. First, observe that since $\alpha=\Omega(d^{-1/5})$, we have $y_0=-d^{1/4}\tan\frac{\alpha}{2}=-d^{\Omega(1)}$. Next, we have

$$\begin{aligned}\widetilde{S}(d,\alpha)&=\Pr_{X,Y\sim N(0,1)}\left[X\ge d^{1/4}\text{ and }Y\le y_0\right]\\&=\Pr_{X\sim N(0,1)}\left[X\ge d^{1/4}\right]\Pr_{Y\sim N(0,1)}\left[Y\le y_0\right]\\&\in(1\pm d^{-\Omega(1)})\cdot\frac{e^{-\sqrt{d}/2}}{\sqrt{2\pi}d^{1/4}}\cdot\frac{e^{-|y_0|^2/2}}{\sqrt{2\pi}|y_0|}\\&\in\frac{1\pm d^{-\Omega(1)}}{2\pi d^{1/2}\tan\frac{\alpha}{2}}\exp\left(-\left(1+\tan^2\frac{\alpha}{2}\right)\frac{\sqrt{d}}{2}\right),\end{aligned}$$

where the third step is due to $y_0=-d^{\Omega(1)}$ and Lemma A.1 and the fourth step is due to $y_0=-d^{1/4}\tan\frac{\alpha}{2}$. □

Now the goal is to prove that if α is not too close to 0 or π, then $S(d,\alpha)$ is close to $\widetilde{S}(d,\alpha)$.

Claim A.4. *If $\alpha=\Omega(d^{-1/5})$ and $\alpha=\pi-\Omega(d^{-\delta})$ for a sufficiently small $\delta>0$, then*

$$d^{-O(1)}\le\frac{S(d,\alpha)}{\widetilde{S}(d,\alpha)}\le d^{O(1)}.$$

Proof. First, for $\alpha=\pi/2$ the claim is trivial, since $S(d,\pi/2)=\widetilde{S}(d,\pi/2)$. Next, we consider the cases $\alpha<\pi/2$ and $\alpha>\pi/2$ separately (see Figure 5).

The case $\alpha < \pi/2$: in this case we have

$$S(d,\alpha) - \widetilde{S}(d,\alpha) = \frac{1}{2\pi} \int_{y_0}^{\infty} \int_{x_0(y)}^{\infty} e^{-\frac{x^2+y^2}{2}}\, dx\, dy,$$

where $x_0(y) = \frac{d^{1/4} + y\sin\alpha}{\cos\alpha}$ (see Figure 5). For every $y \geq y_0$ we have $x_0(y) \geq d^{1/4}$, so by Lemma A.1

$$\int_{x_0(y)}^{\infty} e^{-x^2/2} dx \in \frac{(1 \pm d^{-\Omega(1)}) e^{-x_0(y)^2/2}}{x_0(y)}.$$

Thus,

$$S(d,\alpha) - \widetilde{S}(d,\alpha) = \frac{1 \pm d^{-\Omega(1)}}{2\pi} \int_{y_0}^{\infty} \frac{e^{-\frac{y^2 + x_0(y)^2}{2}}}{x_0(y)}\, dy.$$

By direct computation,

$$y^2 + x_0(y)^2 = y^2 + \left(\frac{d^{1/4} + y\sin\alpha}{\cos\alpha}\right)^2 = \left(\frac{d^{1/4}\sin\alpha + y}{\cos\alpha}\right)^2 + d^{1/2}.$$

Let us perform the following change of variables:

$$u = \frac{d^{1/4}\sin\alpha + y}{\cos\alpha}.$$

We have

$$\begin{aligned} S(d,\alpha) - \widetilde{S}(d,\alpha) &= \frac{(1 \pm d^{-\Omega(1)}) e^{-\sqrt{d}/2}}{2\pi} \int_{d^{1/4}\tan\frac{\alpha}{2}}^{\infty} \frac{e^{-u^2/2}}{d^{1/4} + u\tan\alpha}\, du \\ &\leq \frac{(1 \pm d^{-\Omega(1)}) e^{-\sqrt{d}/2}}{2\pi d^{1/4}} \int_{d^{1/4}\tan\frac{\alpha}{2}}^{\infty} e^{-u^2/2}\, du \\ &= (1 \pm d^{-\Omega(1)}) \widetilde{S}(d,\alpha), \end{aligned}$$

where the last step is due to Lemma A.1 and Claim A.3. Overall, we have

$$\widetilde{S}(d,\alpha) \leq S(d,\alpha) \leq (2 \pm d^{-\Omega(1)}) \widetilde{S}(d,\alpha).$$

The case $\alpha > \pi/2$: this case is similar. We have

$$\begin{aligned} \widetilde{S}(d,\alpha) - S(d,\alpha) &= \frac{1}{2\pi} \int_{-\infty}^{y_0} \int_{x_0(y)}^{\infty} e^{-\frac{x^2+y^2}{2}}\, dx\, dy \\ &= \frac{1 \pm d^{-\Omega(1)}}{2\pi} \int_{-\infty}^{y_0} \frac{e^{-\frac{y^2 + x_0(y)^2}{2}}}{x_0(y)}\, dy. \end{aligned}$$

After the change

$$u = \frac{d^{1/4}\sin\alpha + y}{\cos\alpha}$$

we get (note that $\cos\alpha < 0$)

$$\begin{aligned}\widetilde{S}(d,\alpha) - S(d,\alpha) &= \frac{(1\pm d^{-\Omega(1)})e^{-\sqrt{d}/2}}{2\pi}\int_{d^{1/4}\tan\frac{\alpha}{2}}^{\infty}\frac{e^{-u^2/2}}{u|\tan\alpha| - d^{1/4}}\,du \\ &\le \frac{(1\pm d^{-\Omega(1)})e^{-\sqrt{d}/2}}{2\pi d^{1/4}(|\tan\alpha|\tan\frac{\alpha}{2}-1)}\int_{d^{1/4}\tan\frac{\alpha}{2}}^{\infty}e^{-u^2/2}\,du \\ &= \frac{1\pm d^{-\Omega(1)}}{|\tan\alpha|\tan\frac{\alpha}{2}-1}\widetilde{S}(d,\alpha).\end{aligned}$$

Since $\alpha = \pi - \Omega(d^{-\delta})$, we have

$$\frac{1}{|\tan\alpha|\tan\frac{\alpha}{2}-1} = \frac{1}{1+\Omega(d^{-\delta})} = 1-\Omega(d^{-\delta}).$$

We can choose δ such that

$$\widetilde{S}(d,\alpha) - S(d,\alpha) \le (1-d^{-O(1)})\widetilde{S}(d,\alpha),$$

so

$$d^{-O(1)}\cdot\widetilde{S}(d,\alpha) \le S(d,\alpha) \le \widetilde{S}(d,\alpha).$$

□

Combining (16), Claim A.2, Claim A.3, Claim A.4 and the formula

$$\tan^2\frac{\alpha}{2} = \frac{\|u-v\|^2}{4-\|u-v\|^2},$$

we get the first two bullet points from the statement of Theorem 3.1.

A.2 Three-way collision probabilities

In this section we prove (4). We start with a simple case $\varepsilon = 0$.

A.2.1 The case $\varepsilon = 0$

Suppose we have $u,v,w \in S^{d-1}$ with $\|u-v\| = \tau$ with $\tau \le 1.99$, $\|u-w\|, \|v-w\| = \sqrt{2}$. Our goal is to upper bound

$$\Pr_{\mathcal{R}}\left[\mathcal{R}(u) = \mathcal{R}(v) = \mathcal{R}(w)\right].$$

Similarly to the two-point case, we have

$$\begin{aligned}&\Pr_{\mathcal{R}}\left[\mathcal{R}(u) = \mathcal{R}(v) = \mathcal{R}(w)\right] \\ &= \Theta(1)\cdot\frac{\Pr_{X,Y,Z\sim N(0,1)}\left[X \ge d^{1/4} \text{ and } Y\sqrt{1-\frac{\tau^2}{4}} + Z\cdot\frac{\tau}{2} \ge d^{1/4} \text{ and } Y\sqrt{1-\frac{\tau^2}{4}} - Z\cdot\frac{\tau}{2} \ge d^{1/4}\right]}{\Pr_{X\sim N(0,1)}\left[X \ge d^{1/4}\right]} \\ &= \Theta(1)\cdot\Pr_{X\sim N(0,1)}\left[X \ge d^{1/4}\right]\cdot\Pr_{\mathcal{R}}\left[\mathcal{R}(u) = \mathcal{R}(v)\right].\end{aligned}$$

Thus,

$$\Pr_{\mathcal{R}}\left[\mathcal{R}(u) = \mathcal{R}(w) \mid \mathcal{R}(u) = \mathcal{R}(v)\right] = \Theta(1)\cdot\Pr_{X\sim N(0,1)}\left[X \ge d^{1/4}\right],$$

which, combined with Lemma A.1, gives the desired claim.

A.2.2 The case of arbitrary ε

Suppose that $u, v, w \in S^{d-1}$ are such that $\|u-v\| = \tau$ with $\tau \le 1.99$, $\|u-w\|, \|v-w\| \in \sqrt{2} \pm \varepsilon$ with $\varepsilon = o(1)$. We are interested in upper bounding

$$\Pr_{\mathcal{R}}\left[\mathcal{R}(u) = \mathcal{R}(w) \mid \mathcal{R}(u) = \mathcal{R}(v)\right].$$

By the spherical symmetry of Gaussians, we can assume that $u, v, w \in S^2$. It is not hard to see that there exist $u', v' \in S^2$ such that

- $\|u-u'\| \in \pm O(\varepsilon)$, $\|v-v'\| \in \pm O(\varepsilon)$;
- $\|u'-w\| = \|v'-w\| = \sqrt{2}$;
- $\|u'-v'\| \in \tau \pm O(\varepsilon)$.

Then,

$$\begin{aligned}
&\Pr_{\mathcal{R}}\left[\mathcal{R}(u) = \mathcal{R}(v) = \mathcal{R}(w)\right] \\
&\qquad \in \Theta(1) \cdot \frac{\Pr_{g\sim N(0,1)^3}\left[\langle u, g\rangle \ge d^{1/4} \text{ and } \langle v, g\rangle \ge d^{1/4} \text{ and } \langle w, g\rangle \ge d^{1/4}\right]}{\Pr_{X\sim N(0,1)}\left[X \ge d^{1/4}\right]} \\
&\qquad \in \Theta(d^{1/4}) \cdot \frac{\Pr_{g\sim N(0,1)^3}\left[\langle u, g\rangle \ge d^{1/4} \text{ and } \langle v, g\rangle \ge d^{1/4} \text{ and } \langle w, g\rangle \ge d^{1/4}\right]}{e^{-\sqrt{d}/2}}. \qquad (17)
\end{aligned}$$

Let us study the numerator.

$$\begin{aligned}
&\Pr_{g\sim N(0,1)^3}\left[\langle u, g\rangle \ge d^{1/4} \text{ and } \langle v, g\rangle \ge d^{1/4} \text{ and } \langle w, g\rangle \ge d^{1/4}\right] \\
&\quad \le \Pr_{g\sim N(0,1)^3}\left[\langle u', g\rangle \ge (1-\sqrt{\varepsilon})d^{1/4} \text{ and } \langle v', g\rangle \ge (1-\sqrt{\varepsilon})d^{1/4} \text{ and } \langle w, g\rangle \ge d^{1/4}\right] \\
&\qquad + \Pr_{g\sim N(0,1)^d}\left[\langle u-u', g\rangle \ge \sqrt{\varepsilon}\cdot d^{1/4}\right] + \Pr_{g\sim N(0,1)^d}\left[\langle v-v', g\rangle \ge \sqrt{\varepsilon}\cdot d^{1/4}\right] \\
&\quad \le \Pr_{g\sim N(0,1)^3}\left[\langle u', g\rangle \ge (1-\sqrt{\varepsilon})d^{1/4} \text{ and } \langle v', g\rangle \ge (1-\sqrt{\varepsilon})d^{1/4} \text{ and } \langle w, g\rangle \ge d^{1/4}\right] \\
&\qquad + e^{-\Omega(\sqrt{d/\varepsilon})} \\
&= \Pr_{X,Y,Z\sim N(0,1)}\left[X \ge d^{1/4} \text{ and } \min\left\{Y\sqrt{1-\frac{\tau'^2}{4}} + Z\cdot\frac{\tau'}{2}, Y\sqrt{1-\frac{\tau'^2}{4}} - Z\cdot\frac{\tau'}{2}\right\} \ge (1-\sqrt{\varepsilon})d^{1/4}\right] \\
&\qquad + e^{-\Omega(\sqrt{d/\varepsilon})}, \qquad (18)
\end{aligned}$$

where $\tau' \in \tau \pm O(\varepsilon)$. After plugging (18) into (17), and using that $\tau \le 1.99$, we get the result.

A.3 Efficiency

As has been already observed, the above partitioning scheme is not efficient. One can fix this issue as follows. Instead of checking, whether $\bigcup \mathcal{R} \neq S^{d-1}$, we can just stop after T iterations. If we choose T such that the probability of the event "$\bigcup \mathcal{R} \neq S^{d-1}$" after T steps is less than $e^{-d^{100}}$, then we are done, since the bounds stated in Theorem 3.1 would remain true.

For a fixed point $u \in S^{d-1}$ we have

$$\Pr_{g\sim N(0,1)^d}\left[\langle u, g\rangle \ge d^{1/4}\right] = \Pr_{X\sim N(0,1)}\left[X \ge d^{1/4}\right] \ge e^{-O(\sqrt{d})},$$

where the first step is by 2-stability of Gaussians and the second step is by Lemma A.1.

Now it is not hard to see taking the union bound over a sufficiently fine ε-net that if we set $T = e^{O(\sqrt{d})}$, then we get what we want. This way, we conclude the time and space bounds in Theorem 3.1.