Optimal Las Vegas Approximate Near Neighbors in L_p

Optimal Las Vegas Approximate Near Neighbors in ℓ_p

Alexander Wei

April 3, 2020

Advisor: Jelani Nelson

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Arts in Computer Science and Mathematics
at Harvard College.

Abstract

In the *approximate near neighbor search* problem, one is given a database $\mathcal{D} \subset \mathbb{R}^d$ of n points and asked to construct a data structure which, given a query point $q \in \mathbb{R}^d$, returns a "near" neighbor of q in $\mathcal{D}$. As a fundamental algorithmic primitive for processing high-dimensional data, approximate near neighbor search has found a wide variety of applications, both within computer science and beyond, including to problems in machine learning, computer vision, data mining, and genomics.

We show that high-dimensional approximate near neighbor search can be solved without false negatives while matching the performance of the state-of-the-art algorithms (which can produce false negatives). Our results hold for approximate near neighbor search for Euclidean space as well as all ℓ_p spaces with $p \in [1, 2)$. We achieve this result in two ways: In the data-independent setting, in which algorithms do not adapt to the dataset, we show that the optimal ball-carving LSH of Andoni and Indyk (FOCS 2006) can be adapted into an algorithm without false negatives. In the data-dependent setting, in which algorithms can adapt to structure found in the dataset, we modify the data structure of Andoni et al. (SODA 2017) to produce no false negatives by constructing a new "Las Vegas" locality-sensitive filter family for the unit sphere.

Our data-independent construction improves on the recent Las Vegas data structure of Ahle (FOCS 2017) for ℓ_p when $1 < p \leq 2$. Our data-dependent construction does even better for ℓ_p for all $p \in [1, 2]$ and is the first Las Vegas approximate near neighbors data structure to make use of data-dependent approaches. We also answer open questions of Indyk (SODA 2000), Pagh (SODA 2016), and Ahle by showing that for approximate near neighbors, Las Vegas data structures can match state-of-the-art Monte Carlo data structures in performance for both the data-independent and data-dependent settings and across space-time tradeoffs.

Comments on Published Work

The last chapter of this thesis was published as a conference paper at the 2019 Symposium on Discrete Algorithms (SODA 2019) [Wei19].

Acknowledgements

First and foremost, I'd like to thank Jelani Nelson for being an incredible mentor and adviser throughout my college years. Three years ago, he was willing to entertain a freshman taking his algorithms class—who had no idea what research was—with a few open problem suggestions. Although I never made much progress on those problems, that experience gave me my first taste of research. Since then, Jelani has been unwaveringly supportive, always answering my questions, encouraging my research interests, and opening doors into a world that I have come to love. For that I am incredibly grateful.

I'd additionally like to thank all of the other faculty who I've been lucky to work with, who have advised and inspired me over the past few years: Hu Fu, Piotr Indyk, Scott Kominers, Kevin Leyton-Brown, James Mickens, and Madhu Sudan. They have done so much to broaden my view of computer science and also economics; I owe many of my current research interests to them. Furthermore, I'd like to thank Boaz Barak for reading this thesis and David Parkes for being an excellent academic adviser.

I'm incredibly grateful to the Computer Science and Mathematics departments at Harvard for their support of my research and for being my academic homes during college. My research has also been generously supported by the Herchel Smith Undergraduate Research Fellowship and Harvard Program for Research in Science and Engineering.

And finally, I'd like to thank all of my friends, who have been there with me through the good times and the bad, the Quad community for being a constant source of light in my life, my blockmates, who have made the past few years goofier than I ever could have imagined, Eric for being a baller, Meena for everything, and my family for always wanting the best for me. I would not have made it through without you.

Contents

List of Figures

1. Introduction

Nearest neighbor search, in its essence, is the following algorithmic task:

Given a set of points $\mathcal{D}$ and a query point q, find the point in $\mathcal{D}$ closest to q.

Typically, we think of the points as living in some metric space (e.g., $\mathbb{R}^d$ under the Euclidean norm), and our goal is to *preprocess* the set $\mathcal{D}$ to get a data structure that can answer nearest neighbor queries for points q in the metric space. Algorithms for this fundamental problem have found myriad applications, both within computer science and far beyond: Nearest neighbor search has been used for machine learning [CH67], data mining [BGMZ97], computer vision [SDI06], database retrieval [GIM99], drug discovery [DGJC06], genomics [BKC+15], and fraud detection[1], among others, emerging as a key algorithmic primitive time and time again.

In the first part of this thesis, we provide a gentle introduction to the rich algorithmic theory of (approximate) nearest neighbor search. Namely, we give an exposition of locality-sensitive hashing and the associated technical tools from probability, geometry, and combinatorics. We also build up the machinery and background necessary for the second part of this thesis.

In the second part of this thesis, which is based on our results in [Wei19], we describe a new algorithm for approximate near neighbor search in $\mathbb{R}^d$ under the ℓ_p norm for $1 \leq p \leq 2$. Our algorithm is as efficient as the state-of-the-art approach while also *having no false negatives*. Viewed through a derandomization lens, our result shows that Las Vegas algorithms are just as powerful as Monte Carlo algorithms for approximate near neighbor search.

1.1. A History of Nearest Neighbor Search

Although nearest neighbor search in low-dimensions has been well-understood for decades [Ben75, Cla88, Mei93], our interest lies in tackling nearest neighbor search for massive datasets of *high-dimensional* data. High-dimensional datasets arise through many natural data-generation processes: In natural language processing, modern contextualized word representations have thousands of dimensions [DCLT]; signal data such as images, audio, and video are sampled and stored in high-dimensional formats; DNA sequence data are an example of high-dimensional string data, where each base pair corresponds to a dimension.

As a concrete example, suppose we have a dataset of one billion 1000×1000 pixel images. Each image has a natural representation as a vector of dimension 10^6. Even if we wanted to perform a single image search over this dataset (e.g., with Euclidean distance as our similarity measure), a naïve linear scan would be infeasible, needing on the order of $10^9 \times 10^6 = 10^{15}$

[1] See `https://eng.uber.com/lsh/` for a practical example.

Figure 1.1.: An instance of nearest neighbor search in $\mathbb{R}^2$.

arithmetic operations. And these numbers are hardly extreme: Facebook, in its 2012 S-1 filing[2], reported receiving 300 million photo uploads *per day*. In the years since, high-dimensional data at such scale has only become more common across the computing world.

Applications like the above motivate the design of nearest neighbor search algorithms that significantly outperform a linear scan. Ideally, our algorithm would:

1. Only inspect a small, carefully chosen fraction of the dataset per query, and
2. Remain tractable when applied in high-dimensional spaces.

More precisely, we would like the computational cost to scale *sublinearly in the dataset size* $|\mathcal{D}|$ and *polynomially in the dimension* d. However, despite decades of research effort, an algorithm simultaneously achieving both properties has remained out of reach.[3] All known algorithms for exact nearest neighbor search are either linear in the dataset size or blow up exponentially with dimension. The latter can manifest as either exponentially growing space consumption or exponentially growing query time. This behavior is part of a general qualitive phenomenon in algorithms known as the *curse of dimensionality*.

To sidestep this curse of dimensionality, significant research effort over the past two decades has focused on an approximation version of the problem. In this setting, instead of returning the exact nearest neighbor of the query point, the objective is to return a point that is "almost" the nearest point. This version of the problem, known as *approximate near neighbor search*, is of both immense practical and theoretical relevance. For many real-world datasets, having an approximate near neighbor is often almost as good as the exact nearest neighbor [Raz17]; hence this relaxation to an approximation problem does not come at a significant practical cost. And, as it turns out, accepting a bit of slack in the output deftly avoids the curse of dimensionality. Since the seminal results of Indyk and Motwani [IM98] and Kushilevitz et al. [KOR00], efficient algorithms for high-dimensional approximate near neighbor search have been developed for a variety of metrics, including for $\mathbb{R}^d$ under the ℓ_1 and ℓ_2 norms.

Much of the theoretical work for approximate near neighbor search has taken place within the framework of *locality-sensitive hashing* (LSH), introduced by Indyk and Motwani [IM98]. At

[2] `https://www.sec.gov/Archives/edgar/data/1326801/000119312512235588/d287954ds1a.htm`.

[3] In fact, the existence of such an algorithm for Euclidean space would have strong implications for complexity theory, namely, refuting the strong exponential time hypothesis [AW15].

a high level, LSH algorithms group the points in $\mathcal{D}$ into "buckets" using a randomized hashing scheme during the preprocessing phase. Then, to answer a query, such algorithms only search through buckets located near the query point. By taking advantage of the geometric structure of the underlying space, LSH is able to eliminate all but a small fraction of the search space. This approach to algorithm design has had an outsized influence on nearest neighbor search in practice: Most of the applications cited in the first paragraph use LSH-based algorithms for nearest neighbor search.[4]

The current state-of-the-art for approximate near neighbor search in $\mathbb{R}^d$ under the Euclidean norm is *data-dependent LSH* [AR15, ALRW17]. Data-dependent hashing, unlike previous approaches, adapts the LSH scheme to the dataset at hand by discovering and exploiting pseudo-random structure within the dataset. Although data-dependent hashing has been ubiquitous in practice, this line of work was the first to provide theoretical understanding on why adaptive approaches tend to perform exceptionally well in practice. The data-dependent adaptation in [AR15, ALRW17] consists of reducing to "random looking" datasets by separating out dense clusters of points and then designing an efficient LSH scheme for datasets that look approximately random. Matching lower bounds for these results have also been obtained in restricted models of computation [AR16, ALRW17], suggesting that significantly different ideas will be needed if these results are to be improved further.

1.1.1. Randomness in Approximate Near Neighbor Search

A line of work within the approximate near neighbor search literature has been to understand the extent to which randomness is needed for approximate near neighbor search. Classical LSH algorithms rely heavily on randomness: The grouping of $\mathcal{D}$ into buckets is a randomized process, and when answering queries, a query could have an approximate near neighbor that does not get found. It is thus natural to ask whether this randomness can be eliminated or at least reduced. The second part of this thesis delves into this question in detail.

Indyk [Ind00] initiated the study of this question, describing a deterministic algorithm for approximate near neighbor search based on expander graphs. However, his algorithm works only when the approximation needed is rather coarse. Somewhat surprisingly, this algorithm is the only efficient deterministic algorithm known for approximate near neighbor search.

More success has been had the design of *Las Vegas* algorithms for approximate near neighbor search [Pag16, Ahl17], i.e., algorithms with the property an approximate near neighbor of a query point will always be found. Such algorithms are said to *have no false negatives*. This is in contrast with *Monte Carlo* algorithms (e.g., classical LSH), which may have false negatives with non-zero probability. Thus, Las Vegas algorithms can be understood as a sort of middle ground between Monte Carlo and deterministic algorithms: They are randomized, but always produce the correct result.

Pagh [Pag16] gave the first Las Vegas algorithm for approximate near neighbor search, with an LSH scheme for Hamming space that differs from the optimal by a constant factor in the exponent of the time and space bounds. Ahle [Ahl17] improves on this result with an algorithm matching optimal data-independent LSH for Hamming space [IM98] in performance. However,

[4] For recent surveys of the literature on LSH, see also [Raz17, AI17, AIR18].

these works leave open whether the same can be achieved for optimal data-independent LSH for Euclidean space [AI06] or whether Las Vegas algorithms can be competitive with state-of-the-art Monte Carlo data-dependent algorithms. Sankowski and Wygocki [SW17, Wyg17] make partial progress on Las Vegas algorithms for Euclidean approximate near neighbor search, but do not close the performance gap between Las Vegas and Monte Carlo algorithms.

1.2. Our Contribution

We close the gap between Las Vegas and Monte Carlo algorithms for approximate near neighbor search, answering questions of Ahle [Ahl17], Pagh [Pag16], and Indyk [Ind00].

First, we give a construction of a *data-independent* Las Vegas locality-sensitive data structure that is optimal for Euclidean space, matching the performance achieved in [AI06]. Stated as a theorem, we have:

Theorem 1.1 (Informal)**.** *There exists a Las Vegas data structure for Euclidean approximate near neighbor search in Euclidean space with performance matching that of optimal data-independent LSH [AI06].*

Second, we show that the data-dependent Monte Carlo algorithm of [ALRW17] can be made Las Vegas using similar techniques. Thus, not only do we match optimal data-dependent hashing [AR15] in performance, but we also provide a full range of space-time tradeoffs a la [ALRW17]. Stated as a theorem, we have:

Theorem 1.2 (Informal)**.** *There exists a Las Vegas data structure for Euclidean approximate near neighbor search in Euclidean with performance matching that of optimal data-dependent hashing [AR15]; moreover, this data structure can be adapted to achieve the space-time tradeoffs of Andoni et al. [ALRW17].*

Moreover, by a reduction of Nguyen [Ngu14], our results provide optimal locality-sensitive data structures for $\mathbb{R}^d$ under the ℓ_p norm for all $p \in [1, 2]$ in both the data-independent and the data-dependent settings, giving us the following corollaries:

Corollary 1.3 (Informal)**.** *There exists a Las Vegas data structure for Euclidean approximate near neighbor search in $\mathbb{R}^d$ under the ℓ_p norm for $p \in [1, 2]$ with performance matching that of optimal data-independent LSH [Ngu14].*

Corollary 1.4 (Informal)**.** *There exists a Las Vegas data structure for Euclidean approximate near neighbor search in $\mathbb{R}^d$ under the ℓ_p norm for $p \in [1, 2]$ with performance matching that of optimal data-dependent LSH [AR15]; moreover, this data structure can be adapted to achieve the space-time tradeoffs of Andoni et al. [ALRW17].*

Finally, in the process of proving our results, we construct a geometric adaptation of the *splitters* of Naor et al. [NSS95] using CountSketch [CCF04], which may be of independent interest.

Our results are summarized in a more quantitative form in Table 1.1, where a lower ρ value corresponds to performance that scales better with dataset size.

Technique		Metric: Hamming	Metric: ℓ_2
LSH[IM98, AI06]	(Monte Carlo)	$\rho = 1/2$	$\rho \approx 1/4$
Data-dependent LSH[AR15]	(Monte Carlo)	$\rho \approx 1/3$	$\rho \approx 1/7$
CoveringLSH [Pag16]		$\rho \approx 0.69$	$\rho \approx 0.69$
LSF + Splitters [Ahl17]		$\rho \approx 1/2$	$\rho \approx 1/2$
Our result (data-independent)		—	$\boldsymbol{\rho \approx 1/4}$
Our result (data-dependent)		$\boldsymbol{\rho \approx 1/3}$	$\boldsymbol{\rho \approx 1/7}$

Table 1.1.: Progress on Las Vegas (r, cr)-approximate near neighbor search for $c = 2$.

1.3. Organization

The remainder of this thesis is organized as follows:

- In Chapter 2, we provide an expository survey and introduction to the theory of locality-sensitive hashing. We describe optimal locality-sensitive hashing schemes for Hamming space and ℓ_1 in Section 2.2. We sketch the construction of an optimal locality-sensitive hashing scheme for Euclidean space in Section 2.3. In addition, we introduce related topics—dimensionality reduction (Section 2.3) and derandomization with bounded independence (Section 2.4)—that will be of use in Chapter 3.
- In Chapter 3, we develop and prove our main technical results. In Sections 3.3, 3.4, 3.5 and 3.6, we develop the result for data-independent setting in detail. In Section 3.5, we also develop the family of CountSketch splitters that adapt the splitters of Naor et al. [NSS95] to Euclidean space. In Section 3.7, we show how these same ideas apply to spherical locality-sensitive filtering in the context of data-dependent algorithms. Finally, a note showing that good "Las Vegas" filter families imply good Monte Carlo filter familes (as defined in [Chr17]) is included in Appendix A.

2. A Survey of Locality-Sensitive Hashing for Approximate Near Neighbor Search

In this chapter, we provide an introduction to the algorithmic theory of (approximate) nearest neighbor search, highlighting key historical results and introducing the necessary technical tools along the way. We assume no background other than a basic familiarity with algorithms and probability.

2.1. Introduction

In the nearest neighbor search problem, we are given a database $\mathcal{D}$ of n points in a metric space (X, D) and asked to construct a data structure to efficiently answer queries for the point in $\mathcal{D}$ that is closest to a query point $q \in X$. For many application settings, the space X is $\mathbb{R}^d$ and the metric D is an ℓ_p *norm* ($1 \leq p \leq \infty$), the latter defined as follows:

$$D(x, y) = \|x - y\|_p = \left(\sum_{i=1}^{d} (x_i - y_i)^p \right)^{1/p},$$

where we take the appropriate limit for $p = \infty$.

The nearest neighbor search problem in Euclidean space can be solved efficiently when the dimension of the space X is "low" (e.g., [Cla88, Mei93]). However, all known data structures with strongly sublinear query time for nearest neighbor search suffer from a "curse of dimensionality," in which either the space or time complexity of the data structure is exponential in the dimension d. Thus researchers (e.g., [IM98, GIM99, KOR00]) have also considered an approximate version of nearest neighbor search. Fixing parameters $r > 0$ (search radius) and $c > 1$ (approximation ratio), this problem is stated as follows:

Definition 2.1 ((r, cr)-approximate near neighbors)**.** Given a database $\mathcal{D}$ of n points in a metric space (X, D), construct a data structure that, for any query point $q \in X$, returns a point in $\mathcal{D}$ within distance cr of q, provided there exists a point in $\mathcal{D}$ within distance r of q.

If one has encountered approximation algorithms before, one might ask why we consider this problem, which has has an additional search radius parameter r, rather than the more natural approximation problem where we ask for any point in $\mathcal{D}$ that is no more than c times as far from x as the nearest neighbor of x. Har-Peled et al. [HIM12] establish a relationship between these two problems: The latter can be reduced to the former with a polylog(n) overhead in time and space and a $1 + o(1)$ loss in the approximation factor. Thus, it more or less suffices to solve the (r, cr)-approximate near neighbors problem, which is easier to reason about.

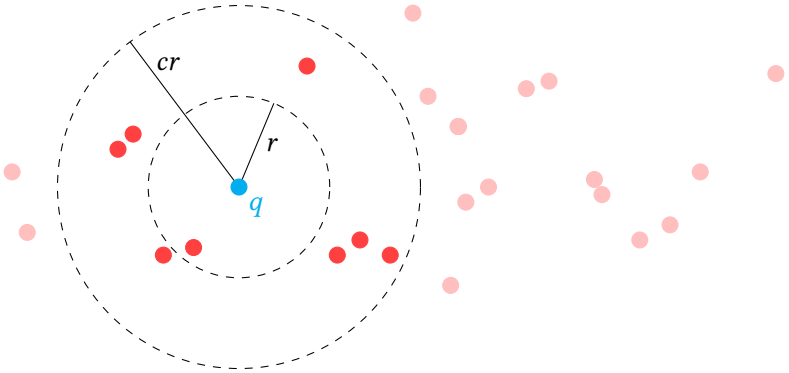

Figure 2.1.: An instance of $(2, r)$-approximate nearest neighbor search in $\mathbb{R}^2$.

And indeed, (r, cr)-approximate near neighbor search is the main problem that we will tackle in this thesis. More specifically, our goal will be to obtain algorithms for (r, cr)-approximate near neighbor search in $\mathbb{R}^d$ that are efficient in the following three performance metrics:

1. Query time: Answer queries in time *sublinear in* n and polynomial in d.
2. Space usage: Require no more than a polynomial amount of space in n and d.
3. Preprocessing time: Construct the data structure in time polynomial in n and d.

In the remainder of this section, we will briefly summarize the results surveyed in Chapter 1 in slightly more formal terms, now that we have defined approximate near neighbor search.

2.1.1. Bounds for Approximate Near Neighbor Search

In their seminal work, Indyk and Motwani [IM98] propose *locality-sensitive hashing* (LSH) as a solution for approximate near neighbors and provide an LSH data structure for Hamming space that overcomes the curse of dimensionality. In particular, their data structure obtains query time $O(dn^{\rho})$ and space usage $O(dn^{1+\rho})$ for an exponent of $\rho = 1/c$. Over the past two decades, the LSH framework and its generalization *locality-sensitive filters* (LSF) [BDGL16, Chr17, ALRW17] have emerged as leading approaches for finding approximate near neighbors, both in theory and in practice. These frameworks serve as the basis of efficient approximate near neighbors data structures for a variety of metric spaces (e.g., [IM98, BGMZ97, DIIM04, AI06, CP17]; see also [AI17, AIR18] for recent surveys). Corresponding lower bounds for the LSH framework have also been found [MNP07, OWZ14], with the construction of [IM98] known to be optimal for Hamming space via the result of [OWZ14].

Recently, a line of work [AINR14, AR15, ALRW17] has focused on *data-dependent* LSH, which are approximate near neighbors data structures where the "bucketing" of points depends on the database $\mathcal{D}$. This stands in contrast to earlier *data-independent* LSH and LSF (e.g., [IM98]), in which the families of hash functions and filters used are determined independently of $\mathcal{D}$. The data dependence allows such data structures to overcome the lower bound of [OWZ14] for

data-independent LSH, with [AR15] obtaining an improved exponent of $\rho = 1/(2c^2 - 1) + o(1)$ for Euclidean approximate near neighbors.

More recently, [ALRW17] obtain space-time tradeoffs with data-dependent hashing, extending the result of [AR15] to data structures with different exponents for space usage and query time. The state-of-the-art bounds for approximate near neighbor search in $\mathbb{R}^d$ under the ℓ_p norm with $p \in [1, 2]$ are given by exponents $\rho_u, \rho_q \geq 0$ such that

$$c^p\sqrt{\rho_q} + (c^p - 1)\sqrt{\rho_u} \geq \sqrt{2c^p - 1}. \tag{2.1}$$

That is, given ρ_u, ρ_q satisfying 2.1, [ALRW17] show that there is an algorithm with expected $O(dn^{\rho_q + o(1)})$ query time, $O(dn^{1+\rho_u+o(1)})$ space usage, and $O(dn^{1+\rho_u+o(1)})$ preprocessing time.

2.2. Locality-Sensitive Hashing

In this section, we introduce the locality-sensitive hashing framework [IM98]. In this framework, randomized partitions of the metric space X are used to reduce the search space for approximate near neighbor search. After setting up the necessary machinery (Theorem 2.1), we apply this framework to obtain efficient approximate near neighbors algorithms for the Hamming and ℓ_1 spaces.

2.2.1. The Locality-Sensitive Hashing Framework

The key definition in the LSH framework is that of a *locality-sensitive hash family*:

Definition 2.2 ((r, cr, p_1, p_2)-sensitive hash family)**.** For a metric space (X, D) and a set U, a distribution $\mathcal{H}$ over maps $h : X \to U$ is said to be (r, cr, p_1, p_2)-*sensitive* if, for all $x, y \in X$, the following holds:

- If $D(x, y) < r$, then $\mathbb{P}_{h\sim\mathcal{H}}(h(x) = h(y)) \geq p_1$;
- If $D(x, y) > cr$, then $\mathbb{P}_{h\sim\mathcal{H}}(h(x) = h(y)) \leq p_2$.

We also refer to such distributions $\mathcal{H}$ as *LSH families*.

To provide some intuition for the definition, note that the first condition says any two "near" points will be hashed to the same element of U (i.e., bucket) with probability at least p_1. Thus, if we sample sufficiently many hash functions, there will likely be one that maps a given pair x, y with $D(x, y) < r$ to the same bucket. Similarly, the second condition implies any two "far" points will be hashed to the same bucket with probability at most p_2. For this definition to be useful, we will need that $p_1 > p_2$, which we assume in the sequel.

Remarkably, all one needs is an (efficient) LSH family to obtain an efficient algorithm for approximate near neighbor search that fails with probability at most 0.9. (Note that we can always reduce the failure probability by running independent copies of the data structure.) This is formalized through the following theorem:

Theorem 2.1 ([IM98]). *Consider the (r, cr)-approximate near neighbors problem on (X, D). Let $\mathcal{H}$ be a (r, cr, p_1, p_2)-sensitive hash family on (X, D). Define*

$$\rho = \frac{\log(1/p_1)}{\log(1/p_2)}.$$

Then (r, cr)-approximate near neighbors can be solved with failure probability at most 0.9 in:

- *Query time that is the total cost of:*
  - *$O(n^\rho/p_1)$ distance computations;*
  - *$O(n^\rho/p_1 \cdot \lceil \log_{1/p_2}(n) \rceil)$ hash function evaluations;*
- *Space usage that is the total cost of:*
  - *Storing n data points;*
  - *$O(n^{1+\rho}/p_1)$ machine words;*
  - *Storing $O(n^\rho/p_1 \cdot \lceil \log_{1/p_2}(n) \rceil)$ hash functions;*
- *Preprocessing time that is the total cost of:*
  - *Sampling $O(n^\rho/p_1 \cdot \lceil \log_{1/p_2}(n) \rceil)$ hash functions from $\mathcal{H}$;*
  - *$O(n^{1+\rho}/p_1 \cdot \lceil \log_{1/p_2}(n) \rceil)$ hash function evaluations.*

Proof. Our basic procedure will be as follows: During preprocessing, sample a hash function $h \sim \mathcal{H}$ and hash each $x \in \mathcal{D}$ into its corresponding bucket $h(x) \in U$. To answer a query for q, first compute $h(q)$. Then, for each $x \in \mathcal{D}$ hashed to $h(q)$, check if $D(x, q) < cr$, returning once a near neighbor is encountered.

We will make this basic procedure more powerful in two ways: First, as alluded to before, we can replicate this procedure multiple times by sampling multiple hash functions h. This will make false negatives less likely, but also make us encounter more "false positives," i.e., points x such that $D(x, q) > cr$, during our search. To remedy this, our second trick is to "amplify" the hash family: We sample a tuple $g = (h_1, \ldots, h_k)$ from the product distribution $\mathcal{H}^k$ and define

$$g(x) = (h_1(x), \ldots, h_k(x)).$$

It is not hard to see that this identification makes $\mathcal{H}^k$ into a (r, cr, p_1^k, p_2^k)-sensitive hash family. This LSH family is "amplified" in the sense that the ratio p_1^k/p_2^k is now larger than p_1/p_2.

Now, let us analyze the basic procedure with the amplified LSH family $\mathcal{H}^k$ replicated L times. First, we bound the probability of a false negative. That is, suppose for a query q, there exists some $x \in \mathcal{D}$ such that $D(x, q) < r$. The probability that we do not find x in our search, i.e., the probability that x and q are never hashed to the same bucket, is at most

$$\left(1 - p_1^k\right)^L \leq e^{-Lp_1^k}.$$

To make this a constant so that our algorithm succeeds with at least constant probability, we will want Lp_1^k to be at least a constant. Next, we bound the number of false positives. For any

$x \in \mathcal{D}$ such that $D(x, q) > cr$, the expected number of hash collisions is at most Lp_2^k. Thus, by linearity of expectation, we expect at most $n \cdot Lp_2^k$ false positives total. By Markov's inequality, the probability that there are more than $3n \cdot Lp_2^k$ false positives is at most $1/3$. Therefore, we can guarantee that we encounter at most this many false positives by terminating the search early, incurring an additive $1/3$ cost in failure probability.

To bound the performance, note that for any query, we evaluate Lk different hash functions and compute at most $O(Lnp_2^k)$ distances. Next, for space usage, in addition to storing the dataset, we need to store a lookup table for each of the replicas that lets us look up dataset points by hash bucket, coming in at a cost of Ln machine words. Furthermore, we need to store each of the Lk hash functions that we sampled. Finally, for preprocessing, we sample Lk hash functions and generate lookup tables based on the hash functions evaluated on $\mathcal{D}$. For the latter, the cost is dominated by that of making Lnk hash function evaluations.

For parameters, we set $L = p_1^{-k}$, so we get an at most $1/e$ probability of false negatives, and $k = \lceil \log_{1/p_2}(n) \rceil$, so $n \cdot p_2^k < 1$. Plugging these values into the costs enumerated above, we get the desired performance bounds. In particular, $L \leq n^\rho / p_1$ because $k \leq \log_{1/p_2}(n) + 1$. And by a union bound, our failure probability is at most $1/e + 1/3 < 0.9$. □

Remark. We keep careful track of the impacts of p_1 and p_2 in our bookkeeping above, as they may factor into the asymptotic costs as a function of c, r, and d. In particular, the costs blow if either $p_1 \to 0$ and $p_2 \to 1$, both of which may happen in practice.

Given this cornerstone of a theorem, one approach to efficient approximate near neighbors algorithms is to construct good LSH families. This is the approach taken by [IM98, DIIM04, AI06], among others. In the sequel, we will see basic applications of this theorem to approximate near neighbor search in Hamming space and ℓ_1 space.

2.2.2. LSH for Hamming Space

Hamming space is the discrete metric space (X, D) where $X = \{0, 1\}^n$ and $D(x, y) = \|x - y\|$ is the number of indices i such that $x_i \neq y_i$. Hamming space is more or less the simplest metric space for which high-dimensional approximate near neighbor search is a relevant problem. This metric is useful for modeling datasets with binary features, a natural application setting.

From the theory angle, Hamming space is of importance because it is "easy" in two different ways:

1. It provides a simple setting to demonstrate the power of the LSH framework;
2. It reduces to approximate near neighbor search in $\mathbb{R}^d$ under the ℓ_p norm for $1 \leq p \leq 2$.

The latter property makes Hamming space ideal for approximate near neighbor search lower bounds [MNP07, OWZ14]: A lower bound for Hamming space implies lower bounds for all ℓ_p spaces, and as noted before, these lower bounds are tight. In our discussion of Hamming space, however, we will focus primarily on the upper bound.[1]

[1] Detailed discussion of lower bounds, although of major importance (and a fascinating application of Fourier analysis over Boolean functions), is out of the scope of this exposition.

Theorem 2.2. *(r, cr)-approximate near neighbor search for Hamming space can be solved with query time $\tilde{O}(dn^\rho)$, space usage $\tilde{O}(dn+n^{1+\rho})$, and preprocessing time $\tilde{O}(dn^{1+\rho})$ for $\rho = 1/c$. (Here, the $\tilde{O}$ hides* polylog(n) *factors.)*

Proof. Let $\mathcal{H}$ be the uniform distribution over the dictator functions $\{0,1\}^d \to \{0,1\}$. That is, $\mathcal{H}$ is uniform over the set of functions h such that $h(x) = x_i$ for some $i \in [d]$. Observe that each hash function takes constant time to sample and evaluate, but that distances take $O(d)$ time to compute.

To compute ρ, notice that $\mathbb{P}_{h\sim\mathcal{H}}(h(x) = h(y)) = 1 - \|x - y\|/d$. Thus, we have

$$\rho = \frac{\log(1/p_1)}{\log(1/p_2)} = \frac{\log(1 - r/d)}{\log(1 - cr/d)} \leq \frac{1}{c}$$

(where the inequality follows from $(1 - r/d)^c \geq 1 - cr/d$ and taking logs). The theorem now follows from applying Theorem 2.1. In particular, p_1 is bounded away from 0, as $p_1 > 1 - 1/c$ if $cr < d$, and p_2 satisfies $1/\log(1/p_2) = O(d/r)$. □

Although the result for Hamming space is almost a direct consequence of Theorem 2.1, it is instructive to unravel the construction a bit to get a feel for what the algorithm is doing. Our hash functions consist of sampling $\sim \frac{d}{r} \log n$ bits from each point into a "fingerprint" of sorts. For an approximation factor of $c = 2$, we use $\Theta(n^{1/2})$ such hash functions. And when $cr \approx d/2$, the setting corresponding approximately to random data, we hash into $\Theta(n)$ buckets.

Finally, we note that by the result of O'Donnell et al. [OWZ14], this LSH scheme is the best possible for Hamming space.

2.2.3. LSH for ℓ_1

In this section, we describe a simple LSH scheme for $\mathbb{R}^d$ under ℓ_1 norm that is similar in spirit to the reduction of Har-Peled et al. [HIM12] of ℓ_1 approximate near neighbor search to Hamming approximate near neighbor search.

We assume without loss of generality that $r = 1$. With this assumption, we reduce approximate near neighbors on $\mathbb{R}^d$ under the ℓ_1 norm to approximate near neighbors on the hypercube $[0, a]^d$ under the ℓ_1 norm. This reduction is a standard trick in the approximate near neighbor search literature and is useful for a variety of metrics (e.g., Section 3.6.2). We then give an LSH family for $[0, a]^d$ that is, in a sense, a generalization of the LSH family from Theorem 2.2.

Lemma 2.3 ([HIM12])**.** *Suppose we can solve $(1, c)$-approximate near neighbor search in $[0, a]^d$ under the ℓ_1 norm and failure probability f. Then we can solve $(1, c)$-approximate near neighbor search in $\mathbb{R}^d$ under the ℓ_1 norm with the same performance bounds and failure probability $f + 1/a$.*

Proof. We sample v uniformly from $[0, a]^d$ and partition $\mathbb{R}^d$ into hypercubes of side length a using the lattice $a \cdot \mathbb{Z}^d + v$. We then build a separate data structure for each hypercube that has a non-empty intersection with $\mathcal{D}$. To handle queries, we only query the data structure of the hypercube containing the query point (if it exists). With this approach, the performance bounds are clearly the same as that of the data structure for $[0, a]^d$.

It remains to bound the failure probability. The only way that we increase false negatives is by separating two "near" points x, q such that $\|x - q\|_1 \leq 1$ into different hypercubes. For this

to occur, there must exist some coordinate i such that a face of the hypercube separates x_i and q_i. Hence this event occurs with probability at most

$$\mathbb{P}(x, q \text{ lie in different hypercubes}) \leq \sum_{i=1}^{d} \frac{|x_i - q_i|}{a} \leq \frac{1}{a}$$

by a union bound over the coordinates. Union bounding with the original failure probability, we get a final failure probability of at most $f + 1/a$. □

Theorem 2.4. *(r, cr)-approximate near neighbor search for $\mathbb{R}^d$ under the ℓ_1 norm can be solved with query time $\tilde{O}(dn^\rho)$, space usage $\tilde{O}(dn + n^{1+\rho})$, and preprocessing time $\tilde{O}(dn^{1+\rho})$ for $\rho = 1/c$.*

Proof. Let $\mathcal{H}$ be the distribution over maps $[0, a]^d \to \{0, 1\}$ given by selecting a coordinate i and a threshold $t \in [0, a]$, each uniformly at random, with the map sending x to 0 if and only if $x_i < t$. Note that a hash function of this form takes constant time to sample and evaluate, but distances still take $O(d)$ to compute.

To evaluate $\mathcal{H}$ as an LSH family, note that for any $x, q \in [0, a]^d$,

$$\mathop{\mathbb{P}}_{h \sim \mathcal{H}}(h(x) = h(q)) = \frac{1}{d} \sum_{i=1}^{d} \left(1 - \frac{|x_i - q_i|}{a}\right) = 1 - \frac{\|x - q\|_1}{ad}.$$

It follows that the quality of this family is

$$\rho = \frac{\log(1/p_1)}{\log(1/p_2)} = \frac{\log(1 - 1/ad)}{\log(1 - c/ad)} \leq \frac{1}{c}.$$

Taking a to be a sufficiently large constant, we have $1/p_1 = O(1)$ and $1/\log(1/p_2) = O(d)$. Thus, approximate near neighbor search on $[0, a]^d$ can be solved with the desired time bounds, and by Lemma 2.3, the same is true of approximate near neighbor search on $\mathbb{R}^d$ under the ℓ_1 norm. □

Remark. In Har-Peled et al. [HIM12], a slightly different approach is taken: Instead of constructing a LSH family for $[0, a]^d$ directly, they discretize $[0, a]^d$ and embed the discretized version of the problem into Hamming space. The resulting LSH families, however, are morally the same.

And on the lower bound side, we note that the exponent $\rho = 1/c$ is again the best we can hope for from the LSH framework, as Hamming space $\{0, 1\}^d$ embeds isometrically into $\mathbb{R}^d$ under the ℓ_1 norm.

2.3. Dimensionality Reductions

When designing algorithms for high-dimensional data, the theory of dimensionality reductions provides a natural toolkit to apply. At the highest level, the goal of a *dimensionality reduction* is to map a high-dimensional space onto a space of lower dimension while preserving some property (e.g., distance or norm) in expectation. Such maps are inherently lossy; nonetheless, a well-tuned dimensionality reduction can capture enough of the geometry of the original space to make much more efficient approximation algorithms possible.

In this section, we introduce distance-preserving dimensionality reductions for Euclidean space. We start with the classical Johnson-Lindenstrauss lemma and then explore more computational aspects of Euclidean dimensionality reduction. Each of the dimensionality reduction schemes we discuss here will be applied in our construction of a Las Vegas data structure for approximate near neighbor search in $\mathbb{R}^d$ in Chapter 3.

2.3.1. The Johnson-Lindenstrauss Lemma

The Johnson-Lindenstrauss lemma [JL84] states that for any $\mathcal{D} \subseteq \mathbb{R}^d$ of size n, there exists a linear map $\mathbb{R}^d \to \mathbb{R}^t$ for $t = \Theta(\varepsilon^{-2} \log n)$ that preserves all pairwise Euclidean distances in $\mathcal{D}$ up to a multiplicative factor of $1 \pm \varepsilon$. That is, there exists a matrix $A \in \mathbb{R}^{t \times d}$ such that

$$(1 - \varepsilon) \|x - y\|_2 \leq \|Ax - Ay\|_2 \leq (1 + \varepsilon) \|x - y\|_2 \tag{2.2}$$

for all $x, y \in \mathcal{D}$. A linear map with this property is said to be of *distortion* $1 \pm \varepsilon$ for $\mathcal{D}$.

We prove this result using the probabilistic method. That is, we define a distribution over matrices $A \in \mathbb{R}^{t \times d}$ and show that 2.2 is satisfied with positive probability. A probabilistic proof of this form is also useful in a computational context: It gives us a way to "construct" such an A in practice.

Before we give the proof, we make a small reduction. Observe that $\|Ax - Ay\|_2 = \|A(x - y)\|_2$. Thus, it is equivalent to ask that A preserves the norms of all z such that $z = x - y$ for $x, y \in \mathcal{D}$. So if we show that the norm of z is preserved (up to $1 \pm \varepsilon$) with probability $1 - 1/2n^2$, then by a union bound, 2.2 holds with probability at least $1/2$. It therefore suffices to show the following, where we care about the case $\delta = 1/2n^2$ in particular:

Theorem 2.5 (Distributional Johnson-Lindenstrauss)**.** *For any $d > 0$ and ε, δ such that $0 < \varepsilon, \delta < 1/2$, there exists a probability distribution $\mathcal{A}$ over matrices in $\mathbb{R}^{t \times d}$ for $t = \Theta(\varepsilon^{-2} \log(1/\delta))$ such that for any $x \in \mathbb{R}^d$,*

$$\mathop{\mathbb{P}}_{A \sim \mathcal{A}} \left(\left| \frac{\|Ax\|_2^2}{\|x\|_2^2} - 1 \right| > \varepsilon \right) < \delta.$$

Proof. We take $\mathcal{A}$ to be a distribution where the entries of A are independent Gaussians. This stems from the observation that if g is a d-dimensional spherical Gaussian, then $\langle g, x \rangle$ is normally distributed with variance $\|x\|_2^2$. In other words, $\langle g, x \rangle^2$ is an unbiased estimator for $\|x\|_2^2$. We can stack independent copies of these estimators and take the mean (i.e., scale each Gaussian by $m^{-1/2}$) to get a more concentrated estimate. The question now becomes: How many samples do we need in order to obtain an estimate that is within a multiplicative factor of $1 \pm \varepsilon$ with probability $1 - \delta$?

At an intuitive level, we need $1/\varepsilon^2$ samples to get an estimator with standard deviation $1/\varepsilon$. To prove this formally, we assume without loss of generality that $\|x\|_2 = 1$. Let

$$Z = \frac{1}{t} \sum_{i=1}^{t} \langle g_i, x \rangle^2 ,$$

where g_i for $i = 1, \ldots, t$ are spherical Gaussians. Observe that Z is a χ-squared distribution with t degrees of freedom. Hence we may apply the following tail bound for χ-squared random variables (see [Wai19] for a proof):

Lemma 2.6. *Suppose Z is distributed as a χ-squared distribution with t degrees of freedom. Then*

$$\mathbb{P}\left(\left|\frac{Z}{t} - 1\right| \geq \varepsilon\right) \leq 2\exp\left(\frac{-t\varepsilon^2}{8}\right).$$

This lemma tells us that $O(\varepsilon^{-2}\log(1/\delta))$ samples suffice to get within ε of the true mean with probability at least $1-\delta$, yielding the desired bound. □

Although the sampling of the matrix A is simple to describe, it can be unwieldy when applied in a computational setting: Multiplication by this matrix can be slow, especially when ε is small. This motivates the discussion of faster algorithms for dimensionality reduction, which we will discuss next.

2.3.2. Fast Johnson-Lindenstrauss Transform

The *Fast Johnson-Lindenstrauss Transform* (FJLT) is a Euclidean dimensionality reduction scheme introduced by Ailon and Chazelle [AC09]. Instead of multiplying by a matrix of Gaussians, we first precondition our input with a randomized Walsh-Hadamard transform. After preconditioning, we sample from our preconditioned vector with a sparse sampling matrix. We have a speed gain over the Johnson-Lindenstrauss distribution defined above because multiplication by a Walsh-Hadamard matrix can be done in time $O(d\log d)$, and sampling can be done in time proportional to the output dimension. This speedup does not come for free, however: Our output dimension is worse than that of Theorem 2.5.

For $d = 2^k$, define the d-th (normalized) Walsh-Hadamard matrix as

$$H_d = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}^{\otimes k},$$

where $A^{\otimes k}$ denotes the k-fold Kronecker product of A with itself. Let D_d be a diagonal matrix all of whose diagonal entries are in $\{-1, 1\}$, sampled independently and uniformly at random. Finally, let S_d be a $t \times d$ (for t to be determined later) sampling matrix, where each of the t rows samples a single coordinate uniformly at random from $\mathbb{R}^d$ and scales it by $\sqrt{d/t}$. Our FJTL is then the product

$$A = S_d H_d D_d.$$

We can assume without loss of generality that d is indeed a power of 2, since we can always pad our vectors with 0 to reach a power of 2 with only a constant factor overhead.

Remark. We note that there are multiple ways to set the sparse sampling matrix S_d. Our choice of S_d is simple to analyze, but comes at the expense of getting the optimal output dimension.

With this setup, we show the following preconditioning lemma, which says that after multiplication by D_d and H_d, any vector has small ℓ_∞ norm with high probability:

Lemma 2.7. *With probability at least $1-\delta$,*

$$\|H_d D_d x\|_\infty \leq \sqrt{\frac{2\log(d/\delta)}{d}}.$$

To prove this lemma, we will need the Azuma-Hoeffding concentration inequality, which we state here:

Theorem 2.8 ([Hoe63, Azu67]). *Let $X_1, \ldots, X_n$ be independent random variables bounded by $|X_i| \leq x_i$. Then*

$$\mathbb{P}\left(\left|\sum_{i=1}^{n} X_i - \sum_{i=1}^{n} \mathbb{E}(X_i)\right| \geq \varepsilon\right) \leq 2\exp\left(-\frac{1}{2}\frac{\varepsilon^2}{\sum_{i=1}^{n} x_i^2}\right).$$

Proof of Lemma 2.7. For any vector $x \in \mathbb{R}^d$ of unit norm, we have that each entry of $H_d D_d x$ is distributed as

$$\frac{1}{\sqrt{d}} \sum_{i=1}^{d} Z_i x_i,$$

where the Z_i are independent Rademacher random variables. (A Rademacher random variable is a uniform random variable over $\{-1, 1\}$.) To bound this sum, we apply Azuma-Hoeffding:

$$\mathbb{P}\left(\left|\sum_{i=1}^{d} Z_i x_i\right| \geq \varepsilon\right) \leq 2\exp\left(-\frac{1}{2}\varepsilon^2\right).$$

Setting $\varepsilon = \sqrt{2\log(d/\delta)}$ and dividing by $\sqrt{d}$, we get that

$$\mathbb{P}\left(\left|\frac{1}{\sqrt{d}} \sum_{i=1}^{d} Z_i x_i\right| \geq \sqrt{\frac{2\log(d/\delta)}{d}}\right) \leq \frac{\delta}{d}.$$

By taking a union bound over all entries of $H_d D_d x$, the desired ℓ_∞ bound follows. □

Theorem 2.9 (Fast Johnson-Lindenstrauss Transform). *For any $d > 0$ and ε, δ such that $0 < \varepsilon, \delta < 1/2$, the FJLT distribution $\mathcal{A}$ over matrices in $\mathbb{R}^{t \times d}$ for $t = \Theta(\varepsilon^{-2}\log(1/\delta)\log(d/\delta))$ is such that for any $x \in \mathbb{R}^d$,*

$$\mathbb{P}_{A \sim \mathcal{A}}\left(\left|\frac{\|Ax\|_2^2}{\|x\|_2^2} - 1\right| > \varepsilon\right) < \delta.$$

Proof. By Lemma 2.7, after multiplication by $H_d D_d$, we obtain a vector with low ℓ_∞ norm with probability at least $1 - \delta$. Assuming that this is the case, we can now apply Azuma-Hoeffding again: Each coordinate of $y = H_d D_d x$ is bounded by $\sqrt{2\log(d/\delta)/d}$ in absolute value. Sampling t coordinates $Z_1, \ldots, Z_t$ from y, we get an estimator for $\|y\|_2^2 = 1$ with concentration

$$\mathbb{P}\left(\left|\frac{d}{t}\sum_{i=1}^{t} Z_i^2 - 1\right| > \varepsilon\right) \leq 2\exp\left(-\frac{1}{2}\frac{\varepsilon^2 t}{2\log(d/\delta)}\right).$$

For $t = \Theta(\varepsilon^{-2}\log(1/\delta)\log(d/\delta))$, we get a failure probability of δ. □

Putting everything together, the FJLT gives us an algorithm for dimensionality reduction to dimension $\Theta(\varepsilon^{-2}\log(1/\delta)\log(d/\delta))$ in time $O(d\log d + \varepsilon^{-2}\log(1/\delta)\log(d/\delta))$.

2.3.3. CountSketch

In our last section on dimensionality reductions, we discuss CountSketch [CCF04], a dimensionality reduction scheme that is extraordinarily simple, but achieves a worse dimensionality reduction guarantee than that of the Johnson-Lindenstrauss lemma or even the FJLT. CountSketch was introduced by Charikar et al. [CCF04], where it was originally used to find frequent items in streams of data. Since then, CountSketch has also been used as a component in more powerful dimensionality reductions that have excellent sparsity guarantees [KN14, CJN18]. For our purposes in Chapter 3, CountSketch will be useful because it can be easily derandomized and can be easily made to have orthogonal rows; we show that a slight variant of CountSketch has the same dimensionality reduction property.

The CountSketch distribution is defined as follows: We uniformly select from the set of $t \times d$ matrices (with t to be determined later) with exactly one non-zero entry in every column and only have non-zero entries from the set $\{-1, 1\}$. Another way to think about this distribution is that we hash each element of $[t]$ to an element of $[d]$ and sum with random signs over each hash bucket to get the output vector.

We obtain the below bound for CountSketch as a dimensionality scheme:

Theorem 2.10 (CountSketch)**.** *For any $d > 0$ and ε, δ such that $0 < \varepsilon, \delta < 1/2$, the CountSketch distribution $\mathcal{A}$ over matrices in $\mathbb{R}^{t \times d}$ for $t = \Theta(\varepsilon^{-2}\delta^{-1})$ is such that for any $x \in \mathbb{R}^d$,*

$$\mathop{\mathbb{P}}_{A \sim \mathcal{A}}\left(\left|\frac{\|Ax\|_2^2}{\|x\|_2^2} - 1\right| > \varepsilon\right) < \delta.$$

Our proof technique for this result comes from the class of *moment methods*. That is, in order to bound the tails of a non-negative random variable X, we bound a moment (i.e., an expectation of the form $\mathbb{E}(X^t)$) of X. Then, to get a tail bound, we apply Markov's inequality in the form

$$\mathbb{P}(X \geq \lambda) = \mathbb{P}(X^t \geq \lambda^t) \leq \frac{\mathbb{E}(X^t)}{\lambda^t}.$$

Moment bounding is a remarkably precise and versatile technique for controlling the tails of a random variable; it has been a cornerstone for the analysis of dimensionality reductions (e.g., [KN14, CJN18]). The moment bound we will be using is *Chebyshev's inequality*, which is simply the above for $t = 2$.

Proof of Theorem 2.10. Again, assume without loss of generality that $\|x\|_2 = 1$. We use the second moment method to analyze the random variable $Z = \|Ax\|_2^2 - 1$. Let $\eta_{r,i}$ be an indicator for whether $A_{r,i}$ is non-zero, and let σ_i be the sign of the non-zero entry of $A_{\cdot,i}$. Note that the $\eta_{r,i}$'s and σ_i's are mutually independent. Then

$$Z = \sum_{r=1}^{t} \sum_{i \neq j} \eta_{r,i}\eta_{r,j}\sigma_i\sigma_j x_i x_j.$$

By Chebyshev's inequality,

$$\begin{aligned}\mathbb{P}(|Z| > \varepsilon) &\leq \frac{1}{\varepsilon^2}\,\mathbb{E}(Z^2) \\ &= \frac{1}{\varepsilon^2}\,\mathbb{E}\left(\sum_{r=1}^{t}\left(\sum_{i\neq j}\eta_{r,i}\eta_{r,j}\sigma_i\sigma_j x_i x_j\right)^2\right) \\ &\qquad + \frac{1}{\varepsilon^2}\,\mathbb{E}\left(\sum_{r\neq s}\left(\left(\sum_{i\neq j}\eta_{r,i}\eta_{r,j}\sigma_i\sigma_j x_i x_j\right)\left(\sum_{i\neq j}\eta_{s,i}\eta_{s,j}\sigma_i\sigma_j x_i x_j\right)\right)\right).\end{aligned}$$

Due to the independence of the random variables and since each σ_i has expected value 0, the expected values of all terms in the second summation are zero. By similar logic, the expected value of the first term is equal to

$$2\sum_{r=1}^{t}\sum_{i\neq j}\mathbb{E}(\eta_{r,i}\eta_{r,j})x_i^2x_j^2 = 2t\sum_{i\neq j}\frac{1}{t^2}x_i^2x_j^2 \leq \frac{2}{t}.$$

(The only terms that don't cancel are the ones where all the σ_i's involved are squared.) Hence $\mathbb{P}(|Z| > \varepsilon) \leq 2/(\varepsilon^2 t)$. For $t = \Theta(\varepsilon^{-2}\delta^{-1})$, we get the desired bound. □

2.3.4. LSH for Euclidean Space via Dimensionality Reduction

In this section, we describe how the theory dimensionality reductions can be applied to obtain an efficient approximate near neighbors algorithm for Euclidean space. The LSH scheme we describe is due to Andoni and Indyk [AI06] and is optimal for Euclidean space. Instead of giving full technical details in this section, however, we will only give an idealized sketch of the main ideas, as the full details are rather messy. We do note that these ideas are developed in full in the context of locality-sensitive filters in Section 3.4.

The result that Andoni and Indyk show is the following:

Theorem 2.11 ([AI06]). *For sufficiently large n and d, there exists a (r, cr, p_1, p_2)-sensitive family of hash functions $\mathcal{H}$ on $\mathbb{R}^d$ with the following properties:*

1. *Sampling, storing, and evaluating a function $h \sim \mathcal{H}$ can be done in time $O(\mathrm{poly}(t^t))$, where $t = \Theta(\log^\alpha n)$ for some $\alpha < 1$.*
2. *The probabilities p_1 and p_2 are such that p_1, p_2 are bounded away from 1 and $1/p_1 = n^{o(1)}$. Furthermore,*
$$\rho = \frac{\log(1/p_1)}{\log(1/p_2)} = \frac{1}{c^2} + o(1).$$

As before, we can assume without loss of generality that $r = 1$. By the reduction of Lemma 2.3, it suffices to solve the problem for a hypercube $[0, a]^d$ under the ℓ_2 norm for a a sufficiently large constant. (Note that Lemma 2.3 applies here since the unit ℓ_2 ball in $\mathbb{R}^d$ is a subset of the unit ℓ_1 ball in $\mathbb{R}^d$.)

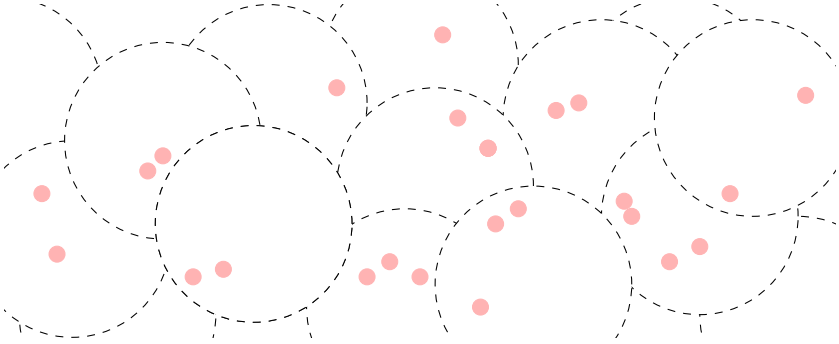

Figure 2.2.: The "ball-carving" LSH of Andoni and Indyk [AI06].

The first step in the construction of $\mathcal{H}$ is to sample a projection matrix $A \sim \mathcal{A}$, where $\mathcal{A}$ is the Johnson-Lindenstrauss distribution of Theorem 2.5 that maps $\mathbb{R}^d$ into $\mathbb{R}^t$, for some $t = \Theta(\log^\alpha n)$. Let us choose an error parameter ε such that $\varepsilon^{-2} \ll t^{1/2}$. Then, by Theorem 2.5, the probability that the distortion of projecting a single vector is greater than $1 \pm \varepsilon$ is at most $\exp(-\Theta(\varepsilon^2 t))$.

Now that we are mapped into $\mathbb{R}^t$, we select an almost optimal LSH family for $\mathbb{R}^t$ in a brute force way. In particular, since $t \ll \log n$, even $O(\text{poly}(t^t)) = O(n^{o(1)})$. With high probability, no point is mapped outside of $[0, 2a]^t$. Therefore, it suffices to focus on this hypercube. Using the probabilistic method, one can show that one can obtain an almost optimal LSH family by carving up $[0, 2a]^t$ with hyperballs selected uniformly at random (see Figure 2.2). We give a quick impression of this approach in the next paragraph.

Let w be a slowly growing function of n such that $\varepsilon^{-2} \ll w^2 \ll t$. We can construct something that is almost an LSH function as follows: Sample N points from the set $[-w, 2a + w]^t$ uniformly at random and consider the hyperballs of radius w centered at each of these N points. We will choose N big enough so that $[0, 2a]$ is barely covered by these hyperballs (e.g., such that each point expects to be in $O(1)$ hyperballs). Conditioning on a point $x \in [0, 2a]^t$ being contained in a random hyperball S, it turns out that the probability that a point y is also in S is approximately

$$\exp\left(-\frac{t}{2} \cdot \frac{\|x - y\|^2}{4w^2}\right)$$

up to $o(1/w^2)$ terms; for details, see Section 3.4. Playing fast and loose with the probabilities, if we were able to create a LSH function by hashing into these hyperballs, then we would expect $p_1 \approx \exp(-\frac{t}{2}\frac{1}{4w^2})$ and $p_2 \approx \exp(-\frac{t}{2}\frac{c^2}{4w^2})$. This would give us the desired ρ.

For this approach to really work though, we would also need to show that we aren't severely affected by the distortion from the dimensionality reduction. Based on our parameter settings, this is indeed true: Since $\varepsilon^{-2} \ll w^2$, we have $p_1 \approx p_1 - \exp(-\Theta(\varepsilon^2 t))$ and $p_2 \approx p_2 + \exp(-\Theta(\varepsilon^2 t))$, where the adjustments to p_1 and p_2 correspond to union bounds with event that the dimensionality reduction fails.

Although many details are left out, this proof sketch describes the high level approach of using dimensionality reductions for approximate near neighbor search: Reducing to a much lower dimension gives us access to "exponential" time algorithms, which are optimal but too

expensive in higher dimensions. However, doing so comes at the cost of incurring distortion, which must be carefully controlled.

2.4. Derandomization

The final technical toolkit that we will introduce is that of derandomization. The techniques for derandomizing algorithms encompass a vast expanse of theoretical computer science research. In this section, we restrict our attention to the particular approach of using random variables of bounded independence.

A model in which to reason about randomized algorithms is the following: Define a *randomized algorithm* to be an algorithm that behaves *deterministically* as a function of its input and an additional stream of uniformly random bits. Then, to quantify the amount of randomness used by an algorithm, we can look at the maximum number of random bits that the algorithm ever reads. At one extreme, a deterministic algorithm always reads zero bits.

The goal of *derandomizing* algorithms is then to reduce (and ideally bring to 0) the number of random bits needed by an algorithm while minimizing the impact on performance. We will see in this section how using random variables that are bounded in their independence can bring us towards this goal.

2.4.1. k-wise Independent Random Variables

One motivation for k-wise independent random variables comes from hashing: Imagine that one wants to build a hash table that maps elements in a universe $\mathcal{U}$ to a set of m buckets. If one wanted a fully random hash function $\mathcal{U} \to [m]$, then one would need $O(|\mathcal{U}| \log m)$ random bits to sample/store the hash function. However, for most practical applications of hashing, $\mathcal{U}$ is massive (e.g., all 64-bit integers), so this approach is clearly infeasible.

The class of k-wise independent random variables offers a compromise between space usage and full independence: Instead of requiring that n random variables all look independent at the same time, we only ask that they look independent within every subset of size k. Formally, the definition is as follows:

Definition 2.3 (k-wise independent random variables)**.** Given a finite set T and random variables $X_1, \ldots, X_n$ taking values in T, the random variables X_i are said to be *k-wise independent* if for all $(t_1, \ldots, t_n) \in T^n$, we have

$$\mathbb{P}\left(\bigwedge_{i=1}^{n} X_i = t_i\right) = \prod_{i=1}^{n} \mathbb{P}(X_i = t_i).$$

Now that we have defined k-wise independent variables, the natural next step is to construct them. One natural construction of k-wise independent variables comes from finite fields:

Lemma 2.12. *Let $\mathbb{F}_q$ be a finite field of size q. For $a_0, \ldots, a_{k-1}$ sampled uniformly at random from $\mathbb{F}_q$, the random variables X_u, $u \in \mathbb{F}_q$ given by*

$$X_u = \sum_{i=0}^{k-1} a_i u^i$$

are k-wise independent.

Proof. To show k-wise independence, it suffices to show that for all distinct $u_1, \ldots, u_k$ in $\mathbb{F}_q$ and $(x_1, \ldots, x_k) \in \mathbb{F}_q^k$, there exists a unique degree k polynomial $P \in \mathbb{F}_q[x]$ such that $P(u_i) = x_i$ for all i. Let

$$V = \begin{bmatrix} 1 & u_1 & \cdots & u_1^{k-1} \\ 1 & u_2 & \cdots & u_2^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & u_k & \cdots & u_k^{k-1} \end{bmatrix}$$

be the Vandermonde matrix on $u_1, \ldots, u_k$. Then a vector $a = [a_0, \ldots, a_{k-1}]$ could be the coefficients of P if and only if $Va = [x_1, \ldots, x_k]$. It is a standard exercise to show that

$$\det(V) = \prod_{i<j} (u_i - u_j).$$

Since the u_i are all distinct, $\det(V) \neq 0$. Therefore, V is invertible and P is unique. □

Note that in this construction, the tuple $(a_0, \ldots, a_{k-1})$ requires only $k \log q$ bits to store. Since there exist finite fields of size 2^m, this means we can have up to $n = 2^m$ random variables that are k-wise independent and take values in $\{0,1\}^m$ using just $O(km)$ bits. Returning to our hashing example, if we are happy with k-wise independence, then we only need $O(k \min(\log |\mathcal{U}|, \log m))$ bits to store a k-wise independent hash function.

2.4.2. Derandomizing with k-wise Independence

We now turn to designing, analyzing, and derandomizing algorithms with k-wise independent random variables. A key property of k-wise independent random variables we will need is the following:

Lemma 2.13. *Let $X_1, \ldots, X_n$ be a collection of k-wise independent random variables taking values in the reals. Then, for any $i_1, \ldots, i_k \in [n]$,*

$$\mathbb{E}\left(\prod_{j=1}^{k} X_{i_j}\right) = \prod_{j=1}^{k} \mathbb{E}(X_{i_j}).$$

Proof. This is immediate from the definition of k-wise independence and expanding the sums for the expected values. □

This lemma unlocks the technique of moment bounding for k-wise independent random variables. That is, we can obtain tail bounds for sums of k-wise independent random variables using up to the k-th moment. And conversely, when an analysis of an algorithm on independent random variables only looks at the first k moments, the same analysis goes through if one replaces the independent random variables by k-wise independent random variables with the same distribution. Note that this gives us a partial derandomization of the original algorithm.

We now apply exactly this technique to derandomize CountSketch (Theorem 2.10). Recall that in the proof of CountSketch, we had indicator random variables $\eta_{r,i}$ for whether $A_{r,i}$ was

non-zero and Rademacher random variables σ_i giving the sign of the non-zero entry in the i-th column of A. We can also characterize the CountSketch distribution in terms of these random variables: We independently sample one-hot column vectors $\eta_{\cdot,i}$ and random signs σ_i. For our derandomized CountSketch, we instead sample the columns $\eta_{\cdot,i}$ according a 2-wise independent distribution and the signs σ_i according to a 4-wise independent distribution, with the sets $\{\eta_{\cdot,i}\}$ and $\{\sigma_i\}$ sampled independently from each other.

Theorem 2.14 (Derandomized CountSketch)**.** *For any $d > 0$ and ε, δ such that $0 < \varepsilon, \delta < 1/2$, the derandomized CountSketch distribution $\mathcal{A}$ over matrices in $\mathbb{R}^{t \times d}$ for $t = \Theta(\varepsilon^{-2}\delta^{-1})$ is such that for any $x \in \mathbb{R}^d$,*

$$\mathop{\mathbb{P}}_{A \sim \mathcal{A}}\left(\left|\frac{\|Ax\|_2^2}{\|x\|_2^2} - 1\right| > \varepsilon\right) < \delta.$$

Proof. The same moment bounding argument as that of Theorem 2.10 works. In particular, we use the 4-wise independence of the σ_i to argue that terms involving $\sigma_i\sigma_j\sigma_{i'}\sigma_{j'}$ have expected value 0 if $\{i, j\} \neq \{i', j'\}$; we use the 2-wise independence of the $\eta_{\cdot,i}$ to expand $\mathbb{E}(\eta_{r,i}\eta_{r,j})$. □

This result shows that instead of using $O(d \log t)$ bits of randomness by sampling with full independence, we can use just $O(\log d)$ bits of randomness with k-wise independent random variables to implement CountSketch. Moreover, this tells us that we can *simulate* the results of CountSketch by iterating over all possible random seeds in polynomial time.

In Section 3.5, we apply a similar argument to derandomize the variant of CountSketch that we use during the "tensoring" step of our algorithm. The only difference is that instead of using k-wise independent random variables for $\eta_{\cdot,i}$, we use k-wise independent permutations, which we introduce next.

2.4.3. k-wise Independent Permutations

For our construction of CountSketch splitters in Section 3.5, we will need k-wise independent permutations. These are the permutation analogs of k-wise independent random variables, with the property that the distribution of any k entries is indistinguishable from that of a random permutation.

Definition 2.4 (k-wise independent permutations)**.** A family H of permutations $[n] \to [n]$ is *k-wise independent* if for all distinct $i_1, i_2, \ldots, i_k \in [n]$ and all distinct $j_1, j_2, \ldots, j_k \in [n]$,

$$\mathop{\mathbb{P}}_{h \sim H}((h(i_1), h(i_2), \ldots, h(i_k)) = (j_1, j_2, \ldots, j_k)) = \frac{1}{n(n-1)\cdots(n-k+1)}.$$

Although defined analogously to k-wise random variables, the dependence between entries for k-wise independent permutations makes them much harder to construct. In fact, no efficient families of k-wise independent permutations are known for $k \geq 4$ [AL13]. Fortunately for us, our use case needs only $k = 2$, for which explicit constructions are known:

Lemma 2.15. *Let $\mathbb{F}_q$ be a finite field of size q. The set of invertible, affine maps $\mathbb{F}_q \to \mathbb{F}_q : x \mapsto \alpha x + \beta$ forms a 2-wise independent family of permutations on $\mathbb{F}_q$.*

Proof. It suffices to show that for any distinct $x_1, x_2 \in \mathbb{F}_q$ and any distinct $y_1, y_2 \in \mathbb{F}_q$, there exists a unique invertible affine map $\mathbb{F}_q \to \mathbb{F}_q$. Indeed, given these values, the unique invertible affine map is given by $\alpha = (y_1 - y_2)/(x_1 - x_2)$ and $\beta = (x_1 y_2 - y_1 x_2)/(x_1 - x_2)$. □

Note that our proof also shows that this family consists $q(q-1)$ permutations. Thus each element can be represented with $O(\log q)$ bits.

3. Optimal Las Vegas Approximate Near Neighbors in ℓ_p

In this chapter, we develop and prove the main technical results of this thesis. We show that approximate near neighbor search in high dimensions can be solved in a Las Vegas fashion (i.e., without false negatives) for ℓ_p ($1 \leq p \leq 2$) while matching the performance of optimal locality-sensitive hashing. Specifically, we construct a data-independent Las Vegas data structure with query time $O(dn^\rho)$ and space usage $O(dn^{1+\rho})$ for (r, cr)-approximate near neighbors in $\mathbb{R}^d$ under the ℓ_p norm, where $\rho = 1/c^p + o(1)$. Furthermore, we give a Las Vegas locality-sensitive filter construction for the unit sphere that can be used with the data-dependent data structure of Andoni et al. (SODA 2017) to achieve optimal space-time tradeoffs in the data-dependent setting. For the symmetric case, this gives us a data-dependent Las Vegas data structure with query time $O(dn^\rho)$ and space usage $O(dn^{1+\rho})$ for (r, cr)-approximate near neighbors in $\mathbb{R}^d$ under the ℓ_p norm, where $\rho = 1/(2c^p - 1) + o(1)$.

3.1. Introduction

Almost all data structures that solve approximate near neighbors while overcoming the curse of dimensionality are Monte Carlo. That is, they fail with some probability to find a near neighbor of the query point when one exists. There are applications, however, where such a failure probability is undesirable, e.g., fraud detection or fingerprint lookups within a criminal database. Furthermore, tuning the failure probability precisely can be difficult in practical settings [GIM99]. These shortcomings of the Monte Carlo approach have motivated the study of Las Vegas data structures for approximate near neighbor search, in which the data structure must *always* return a near neighbor whenever one exists, but the runtime of the queries is permitted to be a random variable.[1] In the literature, this problem has also been referred to as that of constructing approximate near neighbors data structures that are "without false negatives" [Pag16, GPSS17], "with total recall" [PP16], "exact" [AGK06], or "explicit" [KKKC16].

The problem of constructing Las Vegas data structures for approximate near neighbors traces back to [Ind00] and [AGK06]. However, it was not until recently that Las Vegas data structures have come close to matching the bounds achieved by the best Monte Carlo LSH data structures: [Pag16] constructs a Hamming space data structure with exponent $\rho = O(1/c)$, coming within

[1] Note that a Las Vegas data structure can be converted into a Monte Carlo data structure with failure probability δ as follows: Run the data structure twice for the expected runtime (i.e., breaking if no near neighbor is returned), and repeat this $\log(1/\delta)$ times independently. However, no reduction in the other direction (from Monte Carlo to Las Vegas) is known to exist.

a constant factor of the optimal $\rho = 1/c$ of [HIM12], and [Ahl17] closes the gap, achieving $\rho = 1/c + o(1)$ with a new construction based on dimensionality reductions and locality-sensitive filters. Through standard reductions (see [HIM12]), the Hamming space data structure of [Ahl17] can be used to solve (r, cr)-approximate neighbors for ℓ_p with $\rho = 1/c + o(1)$ for all $p \in [1, 2]$. Although this result is optimal for Hamming space and ℓ_1 in the data-independent setting, it leaves open whether a Las Vegas data structure can match the performance of Monte Carlo data structures for $p \in (1, 2]$ and in particular, for the important case of Euclidean space (i.e., $p = 2$). Progress on this front was made by [SW17, Wyg17], which give new Las Vegas data structures for Euclidean (r, cr)-approximate near neighbors. However, these data structures do not match the performance of the best Monte Carlo approaches for this problem and in some instances require exponential space or an approximation factor that is $\omega(1)$.

In this chapter, we resolve the above open problem from [Ind00, Pag16, Ahl17] by constructing the first Las Vegas data structure for Euclidean (r, cr)-approximate near neighbors with exponent $\rho = 1/c^2 + o(1)$. This exponent matches that of the ball lattice LSH construction of [AI06], which was later shown to be optimal in the data-independent setting by [OWZ14]. By a reduction [Ngu14, Section 5], our data structure implies a data structure for approximate near neighbors in ℓ_p for $p \in (1, 2)$ with exponent $\rho = 1/c^p + o(1)$, which is again tight by [OWZ14].[2]

We achieve this result by combining the approaches of [Ahl17] and [AI06] with some new applications of dimensionality reduction to derandomizing geometric problems: We modify and extend the general approach outlined in [Ahl17] for Las Vegas approximate near neighbors data structures to the more difficult ℓ_2 case. Furthermore, we translate ball lattice hashing [AI06] to the LSF framework, construct using CountSketch [CCF04] a geometric analog of the *splitters* described in [NSS95, AMS06], and give a two-stage sequence of dimensionality reductions that allows for more efficient processing of false positives than in [Ahl17]. These techniques culminate in the following theorem and corollary:

Theorem 3.1. *There exists a Las Vegas data structure for Euclidean (r, cr)-approximate near neighbors in $\mathbb{R}^d$ with expected query time $\tilde{O}(dn^\rho)$, space usage $\tilde{O}(dn^{1+\rho})$, and preprocessing time $\tilde{O}(dn^{1+\rho})$ for $\rho = 1/c^2 + o(1)$.*

Corollary 3.2. *There exists a Las Vegas data structure for (r, cr)-approximate near neighbors in $\mathbb{R}^d$ under ℓ_p for $1 < p < 2$ with expected query time $\tilde{O}(dn^\rho)$, space usage $\tilde{O}(dn^{1+\rho})$, and preprocessing time $\tilde{O}(dn^{1+\rho})$ for $\rho = 1/c^p + o(1)$.*

For the data-dependent setting, we apply similar high-level techniques to obtain a Las Vegas version of the data-independent spherical LSF used in the data-dependent data structure of [ALRW17]. We then substitute our Las Vegas LSF family in for the spherical LSF family of [ALRW17] to obtain a Las Vegas data structure for data-dependent hashing with space-time tradeoffs:

Theorem 3.3. *Let p be such that $1 \le p \le 2$, and suppose $\rho_u, \rho_q \ge 0$ are such that*

$$c^p\sqrt{\rho_q} + (c^p - 1)\sqrt{\rho_u} \ge \sqrt{2c^p - 1}.$$

[2] Our data structure also implies a new Las Vegas construction for approximate near neighbors in Hamming space and ℓ_1, matching the bounds of [Ahl17]. This can be done via the embedding of Hamming space into Euclidean space (with distances squared) and the reduction from ℓ_1 to Hamming space, respectively.

There exists a Las Vegas (r, cr)-approximate near neighbors data structure in $\mathbb{R}^d$ under ℓ_p with expected query time $\tilde{O}(dn^{\rho_q+o(1)})$, space usage $\tilde{O}(dn^{1+\rho_u+o(1)})$, and preprocessing time $\tilde{O}(dn^{1+\rho_u+o(1)})$.

This result of Theorem 3.3 matches the optimal data-dependent hashing bounds given by [AR15, AR16, ALRW17] and resolves open questions of [Ahl17] regarding data-dependence and space-time tradeoffs for Las Vegas data structures. In particular, we have an exponent of $\rho = 1/(2c^2 - 1) + o(1)$ for Euclidean approximate near neighbors in the symmetric case $\rho_u = \rho_q$. The approach we take to make the data-independent spherical LSH of [ALRW17] into a Las Vegas filter family applies similar ideas and techniques as those of Theorem 3.1 and Corollary 3.2, but in a more technically involved way. Thus, we first describe in detail the construction for the data-independent setting before discussing how to adapt the construction to the data-dependent setting in Section 3.7.

The combination of Theorem 3.1, Corollary 3.2, and Theorem 3.3 shows that relative to Monte Carlo data structures, Las Vegas data structures for approximate near neighbors in $\mathbb{R}^d$ under the ℓ_p norm no longer have "polynomially higher" query time as noted in the survey [AI17], but rather match the best Monte Carlo constructions for all $p \in [1, 2]$ in both the data-independent and data-dependent settings.

The rest of this paper is structured as follows: In Section 3.2, we cover some preliminary notions. In Section 3.3, we provide a more technical description of how our construction works for the data-independent setting. Sections 3.4, 3.5 and 3.6 contain the details of the construction for the data-independent setting. Finally, Section 3.7 describes how to adapt the approach of the previous sections to the data-dependent setting.

3.1.1. Background and Techniques

LSH [HIM12] has been one of the most popular foundations for approximate near neighbors data structures in recent years. The basic object of this framework is a *locality-sensitive hash family*, a distribution over hash functions such that any pair of "close" points is mapped to the same bucket with probability *at least* p_1 and any pair of "distant" points is mapped to the same bucket with probability *at most* p_2. A major contribution of [HIM12] is the result that such a hash family implies a data structure for approximate near neighbors with exponent $\rho = \log(1/p_1)/\log(1/p_2)$.

Our construction roughly fits into the *locality-sensitive filters* (LSF) framework, a generalization of LSH to a setting where each point is associated to a *set* of filters instead of a single hash bucket. The basic object of this framework is a *locality-sensitive filter family* (see Section 3.2). This framework allows for greater flexibility on the algorithm designer's part [BDGL16, Chr17, ALRW17] and has been successfully used to construct Las Vegas locality-sensitive data structures for Hamming space [Ahl17]. There are also known lower bounds for LSF: [Chr17] extends the lower bound of [OWZ14] to a Monte Carlo formalization of LSF, and in this paper, we extend the results of [OWZ14, Chr17] further to LSF families with Las Vegas properties.

The construction of filters we use is inspired by the ball lattice LSH of [AI06]. In [AI06], an LSH family is constructed by covering a low-dimensional space $\mathbb{R}^b$ with lattices of balls and hashing each point to the first ball it is covered by. We apply this idea in the context of LSF,

constructing an LSF family with Las Vegas properties, in which each point is mapped to the set of balls containing it (i.e., the filters in the LSF family are the balls in the lattices of balls).

We combine this filter family with the general approach of [Ahl17] for Las Vegas approximate near neighbors data structures to obtain our data structure for Euclidean space. In [Ahl17], Ahle first constructs an LSF family with Las Vegas properties for a low-dimensional space $\{0,1\}^b$, where $b = \Theta(\log n)$. Then, a "tensoring" construction (as in [Chr17]) is used to combine LSF families for $\{0,1\}^b$ into an efficient LSF family for $\{0,1\}^B$, where $B = \Theta(\log^{1+\beta} n)$. However, directly tensoring with randomly sampled filter families is not sufficient for a Las Vegas data structure, and thus a key innovation of [Ahl17] was to apply the splitters of [NSS95, AMS06] as a derandomization tool. These splitters create a limited set of partitions of Hamming space, each of which corresponds to a way to tensor together lower-dimensional filter families such that the union of these tensored families is a filter family for $\{0,1\}^B$ with Las Vegas properties.

We show that this approach extends from Hamming space to Euclidean space with some modifications. In particular, we adapt splitters to the geometric setting of $\mathbb{R}^B$ by constructing a collection $\mathfrak{P}$ of orthogonal decompositions (see Section 3.2.3) of $\mathbb{R}^B$ such that each vector $x \in \mathbb{R}^B$ is "split" into components of almost equal length by at least one of the orthogonal decompositions in $\mathfrak{P}$. The collection $\mathfrak{P}$ is constructed with a new variation of CountSketch [CCF04] using 2-wise independent permutations [AL13]. This geometric analog of splitters, which we call "CountSketch splitters," lets us obtain a similar result as [Ahl17] for tensoring together filter families for the lower-dimensional space $\mathbb{R}^b$ into a filter family for $\mathbb{R}^B$.

The final technique of dimensionality reduction is needed to go from efficient LSF families for $\mathbb{R}^B$ to a Las Vegas locality-sensitive data structure for $\mathbb{R}^d$, where the dimension d is arbitrary. The Johnson-Lindenstrauss lemma [JL84] states that the dimension of a set of n points can be reduced to dimension $O(\varepsilon^{-2} \log n)$ while preserving pairwise distances up to a multiplicative factor of $1 \pm \varepsilon$. This idea is useful for high-dimensional approximate near neighbors and has a long history of being applied in various forms to such data structures (e.g., [Ind00, DIIM04, AI06, AC09, Ahl17, SW17]). However, as these dimensionality reduction maps are typically randomized in a Monte Carlo sense, [Ahl17, SW17] require the dimensionality reduction to have an additional "one-sided" property to preserve the Las Vegas guarantee. Our construction relies on the same one-sided property and also improves on the runtimes of [Ahl17, SW17] for this dimensionality reduction stage with a more careful two-stage sequence of reductions and an application of FastJL [AC09].

3.2. Preliminaries

3.2.1. c-Approximate Near Neighbors

If $X = \mathbb{R}^d$ and D is an ℓ_p norm, then one can assume without loss of generality that $r = 1$, in which case we also refer to (r, cr)-approximate near neighbors as *c-approximate near neighbors*. We will use this convention throughout the rest of this chapter.

3.2.2. Locality-Sensitive Filters

The basic object of the LSF framework is a *filter family*, which we define as follows:

Definition 3.1. Given a metric space (X, D), a (symmetric) *filter family* $\mathcal{F}$ is a collection of subsets of X. An element $F \in \mathcal{F}$ is known as a *filter*. For a point $x \in X$, define $\mathcal{F}(x) \coloneqq \{F \in \mathcal{F} : x \in F\}$ to be the set of all filters that contain x. A filter family can also be characterized by the values $\mathcal{F}(x)$ for all $x \in X$, i.e., as a map from X to the power set of filters.

In Section 3.7, we also consider *asymmetric* filter families in order to obtain space-time tradeoffs. Asymmetric filter families generalize (symmetric) filter families by having different filters for database points and query points.

Definition 3.2. An *asymmetric filter family* $\mathcal{F}$ for a metric space (X, D) is a collection of pairs of subsets of X, with each pair consisting of an *update filter* and a *query filter*. We refer to a such a pair collectively as an *asymmetric filter*. For each $x \in X$, define $\mathcal{F}_u(x)$ to be the set of asymmetric filters in $\mathcal{F}$ whose update filter contains x. Likewise, define $\mathcal{F}_q(x)$ to be the set of asymmetric filters in $\mathcal{F}$ whose query filter contains x.

The (symmetric) filter families of Definition 3.1 are simply asymmetric filter families such that $\mathcal{F}_u = \mathcal{F}_q$.

In [Chr17], Christiani defines $(r, cr, p_1, p_2, p_q, p_u)$-sensitive filter families as distributions over asymmetric filters with certain locality-sensitive properties. Although we will not be using this definition because we require families that have a Las Vegas guarantee, we note that the filter families considered in Sections 3.4, 3.5 and 3.7 can be converted into $(r, cr, p_1, p_2, p_q, p_u)$-sensitive filter families by defining the distributions to be sampling a filter uniformly at random from the family. This reduction extends the lower bound proven in [Chr17] to the filter families we consider, meaning our later constructions of filter families are optimal. See Appendix A for more details about the lower bound.

3.2.3. Projections and Orthogonal Decompositions

Also useful to us are orthogonal decompositions of a vector space $\mathbb{R}^d$, i.e., families of projections that express $\mathbb{R}^d$ as a direct sum of d'-dimensional subspaces. Formally, we define these families as follows:

Definition 3.3. A family $\{P_i\}_{i=1}^k$ of $d' \times d$ matrices is an *orthogonal decomposition* of $\mathbb{R}^d$ if $kd' = d$ and the set of row vectors of $P_1, \ldots, P_k$ forms an orthonormal basis of $\mathbb{R}^d$.

Given a linear map $P : \mathbb{R}^d \to \mathbb{R}^{d'}$, the *distortion* of P with respect to $x \in \mathbb{R}^d$ is $\sqrt{d/d'}\, \|Px\|_2 / \|x\|_2$. This value measures the extent to which P preserves the length of x.

3.2.4. Notation

For a point $x \in \mathbb{R}^d$, we use $\mathcal{S}(x, r)$ to denote the sphere of radius r with center x and $\mathcal{B}(x, r)$ to denote the ball of radius r with center x. We use $\mathcal{N}(\mu, \Sigma)$ to denote the normal distribution with mean μ and covariance matrix Σ.

3.3. Overview of Data-Independent Construction

As discussed in Section 3.1.1, our construction for the data-independent setting can be roughly broken down into three stages. The first stage takes place in $\mathbb{R}^b$, where $b = \Theta(\log^\alpha n)$ is a power of two and $0 < \alpha < 1$; the second stage takes place in $\mathbb{R}^B$, where $B = \Theta(\log^{1+\beta} n)$ is a power of two and $0 < \beta < \alpha$; and the third stage takes place in the original dimension $\mathbb{R}^d$. Although any α and β satisfying these constraints gives a data structure with exponent $\rho = 1/c^2 + o(1)$, the specific choice of α and β affects the $o(1)$ term in ρ. Setting $\alpha = 4/5$ and $\beta = 2/5$ gives the fastest convergence of ρ to $1/c^2$. (This choice of parameters yields an additive $o(1)$ term in the exponent that diminishes as $\tilde{O}(\log^{-1/5} n)$.)

The stages of our construction proceed as follows:

1. *Construct an LSF family that solves c-approximate near neighbors in $\mathbb{R}^b$.* We accomplish this by translating the ball lattice hashing approach of [AI06] to the LSF framework. Our construction starts by sampling a collection of ball lattices to cover $\mathbb{R}^b$, with the individual balls being the filters of a filter family $\mathcal{F}$. We then show that this sampled filter family has the desired "Las Vegas" locality-sensitive properties with probability at least $1/2$. We also describe an algorithm to verify these properties given a sampled $\mathcal{F}$. It suffices for the sampling to succeed with probability $1/2$, since if the verification fails, we can restart—the constant probability of success implies we expect to restart at most $O(1)$ times.

   More concretely, our LSF family will have the following properties: a filter family $\mathcal{F}$ can be constructed efficiently (in time $O(\text{poly}(b^b)) = n^{o(1)}$); for any $x \in \mathbb{R}^b$, $\mathcal{F}(x)$ can be computed efficiently (in time $O(\text{poly}(b^b))$); for all points $x, y \in \mathbb{R}^b$ such that $\|x - y\|_2 \le 1$, $\mathcal{F}(x) \cap \mathcal{F}(y) \neq \emptyset$; and for all points $x, y \in \mathbb{R}^b$ such that $\|x - y\|_2 \ge t$,

$$\mathop{\mathbb{E}}_{\mathcal{F}}(|\mathcal{F}(x) \cap \mathcal{F}(y)|) \le \exp(-\Omega(t^2)),$$

   with the constant implied by the big-Ω to be specified later.

2. *Construct an efficient LSF family that solves c-approximate near neighbors in $\mathbb{R}^B$.* Let $\mathcal{P}$ be an orthogonal decomposition of $\mathbb{R}^B$ into subspaces of dimension b. We can construct a filter family for $\mathbb{R}^B$ from B/b filter families for $\mathbb{R}^b$ (one for each subspace of $\mathcal{P}$) by applying a "tensoring" operation. In the tensored filter family, the set of filters containing a point $x \in \mathbb{R}^B$ is the direct product of the sets of filters containing each component of x in the subspaces of $\mathcal{P}$. We repeat this for a collection $\mathfrak{P}$ of orthogonal decompositions of $\mathbb{R}^B$ into $\mathbb{R}^b$. We have our final filter family be the union of these tensored filter families.

   The collection $\mathfrak{P}$ of orthogonal decompositions we consider has the property that every vector in $\mathbb{R}^B$ is "split" into components that are almost equal in length by some orthogonal decomposition in $\mathfrak{P}$. To construct $\mathfrak{P}$, we first describe a modification of CountSketch using 2-wise independent permutations where each element of the modified CountSketch family is an orthogonal projection. We then construct the elements of $\mathfrak{P}$ in a "divide-and-conquer" manner by composing CountSketch projections that each halve the dimension of the space. Important to us is the fact that $\mathfrak{P}$ has size $\text{poly}(B^{B/b}) = n^{o(1)}$.

   The splitting property of $\mathfrak{P}$ implies that if $x, y \in \mathbb{R}^B$ are such that $\|x - y\|_2 \le 1$, then there exists an orthogonal decomposition $\mathcal{P} \in \mathfrak{P}$ such that all components of $x - y$ under $\mathcal{P}$

have length at most $(1+\varepsilon)\sqrt{b/B}$ for $\varepsilon = \Theta(\log^{-\alpha/2} n)$. Thus, if a pair of points $x, y \in \mathbb{R}^B$ are "close," then there exists an orthogonal decomposition $\mathcal{P} \in \mathfrak{P}$ such x and y are "close" in all subspaces given by $\mathcal{P}$. Furthermore, an analogous property bounding the expected number of shared filters holds for "distant" points in $\mathbb{R}^B$. These are useful because they let the Las Vegas properties for the filter families in $\mathbb{R}^b$ transfer to the filter family in $\mathbb{R}^B$.

The result is a filter family for $\mathbb{R}^B$ with Las Vegas properties, such that the set of filters containing $x \in \mathbb{R}^B$ can be efficiently computed (because of tensoring). We can use this filter family to obtain a data structure for Euclidean c-approximate near neighbors in $\mathbb{R}^B$ with exponent $\rho = 1/c^2 + o(1)$.

3. *Reduce approximate near neighbors in $\mathbb{R}^d$ to d/B instances of approximate near neighbors in $\mathbb{R}^B$.* We reduce the original c-approximate near neighbors problem in $\mathbb{R}^d$ to d/B instances of c'-approximate near neighbors in $\mathbb{R}^B$, where $c' = (1-\varepsilon)c$ for $\varepsilon = \Theta(\log^{-\beta/2} n)$. This is done with a two-stage dimensionality reduction process that again uses orthogonal decompositions. We construct distributions over orthogonal decompositions using distributions of Johnson-Lindenstrauss maps in which all elements are projections.

   Orthogonal decompositions of $\mathbb{R}^d$ into subspaces of dimension d' have the property that if $x, y \in \mathbb{R}^d$ are such that $\|x-y\|_2 \le 1$, then at least one component of $x - y$ under the decomposition will have length at most $\sqrt{d'/d}$. Thus, if we treat each subspace as a separate instance of approximate near neighbors, then all x and y that are "close" in $\mathbb{R}^d$ will also be close in at least one of the subspaces in the orthogonal decomposition, preserving the Las Vegas guarantee.

   A caveat of applying an orthogonal decomposition is that two "distant" points could be close in some subspaces, such that solving approximate near neighbors in those subspaces produces false positives for the original problem. We must filter out these false positives to maintain the Las Vegas guarantee. To reduce the time spent checking for false positives, we break the dimensionality reduction from $\mathbb{R}^d$ to $\mathbb{R}^B$ into two steps: The first step uses FastJL to map $\mathbb{R}^d$ into $\mathbb{R}^{B'}$ for $B' = O(\text{poly}(\log(dn)))$, and the second step uses a random Gaussian projection to map $\mathbb{R}^{B'}$ into $\mathbb{R}^B$. We check for false positives after each level of projection. This process has a runtime overhead of $O(d\,\text{poly}(\log(dn)))$ per point. Hence query time, space usage, and preprocessing time are dominated by those of solving approximate near neighbors in the subspaces.

In the following three sections, we describe in detail the constructions for each stage.

3.4. Ball Lattice Filters for $\mathbb{R}^b$

In this section, the variables b, w, and δ are each functions of the number of points n, such that $b = \Theta(\log^{\alpha} n)$, $w = \Theta(\log^{\beta/2} n)$ and $\delta = \Theta(1/b)$ where α and β are as defined in Section 3.3. (One can also consider the explicit parameter setting $\alpha = 4/5$ and $\beta = 2/5$.)

We start by introducing some notation and a lemma relating to balls in $\mathbb{R}^b$. Let V_b denote the volume of a unit b-ball, let $C_b(u)$ denote the volume of the cap at distance u from the center of a

unit b-ball[3], and define $I_b(u) \coloneqq C_b(u)/V_b$ to be the relative cap volume at distance u for a b-ball.

Lemma 3.4 ([AI06]). *For all $b \geq 2$ and $0 \leq u \leq 1$,*

$$\frac{1}{\sqrt{b}}\left(1-u^2\right)^{\frac{b}{2}} \lesssim I_b(u) \leq \left(1-u^2\right)^{\frac{b}{2}}.$$

(Here, the $\lesssim$ drops a constant universal across b.)

The following proposition is the main result of this section; the distribution $\mathfrak{F}$ over filter families with Las Vegas properties that we construct will be used in the tensoring step in Section 3.5.

Proposition 3.5. *There is a distribution $\mathfrak{F}$ over filter families for $\mathbb{R}^b$ with the following properties:*

1. *A filter family $\mathcal{F}$ can be sampled from $\mathfrak{F}$ in expected time $O(\mathrm{poly}(b^b))$.*
2. *For all $x \in \mathbb{R}^b$, $\mathcal{F}(x)$ can be computed in expected time $O(\mathrm{poly}(b^b))$.*
3. *For all $x, y \in \mathbb{R}^b$ such that $\|x-y\|_2 \leq 1$, $\mathcal{F}(x) \cap \mathcal{F}(y) \neq \emptyset$.*
4. *Let $t \geq 0$ be a fixed constant. For all $x, y \in \mathbb{R}^b$ such that $\|x-y\|_2 \geq t$,*

$$\mathop{\mathbb{E}}_{\mathcal{F}\sim\mathfrak{F}}(|\mathcal{F}(x) \cap \mathcal{F}(y)|) \leq O\left(\mathrm{poly}(b)\exp\left(-\frac{b}{2}\frac{t^2-1-o(1)}{4w^2}\right)\right).$$

In particular, for $t = 0$,

$$\mathop{\mathbb{E}}_{\mathcal{F}\sim\mathfrak{F}}(|\mathcal{F}(x)|) = O\left(\mathrm{poly}(b)\exp\left(\frac{b}{2}\frac{1+o(1)}{4w^2}\right)\right).$$

Each filter in the filter families that we sample will be a ball of radius w. Like in [AI06], we consider infinite lattices of such balls in $\mathbb{R}^b$, in particular translations of the lattice of balls of radius w with centers at $3w \cdot \mathbb{Z}^b$. Each such lattice of balls thus corresponds to an offset $v \in [0, 3w]^b$.

Let $\mathfrak{F}_0$ be the distribution over filter families such that a filter family $\mathcal{F}$ is sampled by generating N offsets $v_1, \ldots, v_N$, where each offset is sampled independently and uniformly from $[0, 3w]^b$, and then defining

$$\mathcal{F} \coloneqq \bigcup_{i=1}^{N}\left\{\mathcal{B}(u, w) : u \in v_i + 3w \cdot \mathbb{Z}^b\right\}.$$

The set of filters a point belongs to is exactly the set of balls that contain it. (This differs from the ball lattice LSH of [AI06], in which a point gets hashed to the smallest i such that the ball with offset v_i contains it.)

[3] Alternatively, $C_b(u)$ is half the volume of the intersection of two unit b-balls whose centers are distance $2u$ apart.

A filter family sampled from $\mathfrak{F}_0$ does not necessarily satisfy property 3. Thus, to ensure property 3 holds, we obtain $\mathfrak{F}$ from $\mathfrak{F}_0$ by adding a verification step that drops all filter families drawn from $\mathfrak{F}_0$ that do not satisfy property 3. Checking that property 3 holds exactly is difficult, since there are infinitely many pairs $x, y \in \mathbb{R}^b$ such that $\|x - y\|_2 \le 1$. We instead verify a stronger condition and choose N so that a filter family sampled from $\mathfrak{F}_0$ satisfies this stronger condition with probability at least $1/2$. Then a filter family can be sampled from $\mathfrak{F}$ with $O(1)$ samples from $\mathfrak{F}_0$ in expectation.

The stronger condition we check is the following: Consider the lattice $L \coloneqq \delta \cdot \mathbb{Z}^b$ and a smaller radius $w' \coloneqq w - \frac{1}{2}\delta\sqrt{b}$. We check that for all $x, y \in L$ such that $\|x - y\|_2 \le 1 + \delta\sqrt{b}$, there exists some filter $F \in \mathcal{F}$ with center u such that $x, y \in \mathcal{B}(u, w')$. This condition suffices because for every pair $x', y' \in \mathbb{R}^b$ such that $\|x' - y'\|_2 \le 1$, there exist $x, y \in L$ such that $\|x' - x\|_2, \|y' - y\|_2 \le \frac{1}{2}\delta\sqrt{b}$. In particular, $\|x - y\|_2 \le 1 + \delta\sqrt{b}$, and for all $u \in \mathbb{R}^b$, $x, y \in \mathcal{B}(u, w')$ implies $x', y' \in \mathcal{B}(u, w)$. Note that this condition holds if it holds on the subset $L \cap [0, 6w]^b$ by the lattice structure of the filters. Therefore, it is enough to check $O((6w/\delta)^{2b}) = O(\mathrm{poly}(b^b))$ pairs of points in L, which can be done in $O(N\,\mathrm{poly}(b^b))$ time.

The next lemma lower bounds the probability of success for a fixed "close" pair $x, y \in L$:

Lemma 3.6. *Let $x, y \in L$ be such that $\|x - y\|_2 \le 1 + \delta\sqrt{b}$. Suppose an offset $v \in [0, 3w]^b$ is sampled uniformly at random. The probability there exists a point $u \in v + 3w \cdot \mathbb{Z}^b$ such that $x, y \in \mathcal{B}(u, w')$ is at least $\Omega(\mathrm{poly}(b^{-1}) V_b 3^{-b} \exp(-\frac{b}{2}\frac{1+o(1)}{4w^2}))$.*

Proof. Let E_1 be the event that there exists a $u \in v + 3w \cdot \mathbb{Z}^b$ such that $x, y \in \mathcal{B}(u, w')$ and E_2 be the event that there exists a $u \in v + 3w \cdot \mathbb{Z}^b$ such that $x \in \mathcal{B}(u, w')$. Because E_1 implies E_2, we can condition to obtain

$$\begin{aligned}\mathbb{P}(E_1) &= \mathbb{P}(E_2) \cdot \mathbb{P}(E_1 | E_2) \\ &\ge \left(\frac{w'}{3w}\right)^b V_b \cdot 2 I_b\left(\frac{1 + \delta\sqrt{b}}{2w'}\right).\end{aligned}$$

For the first term, assuming sufficiently large b and w:

$$\begin{aligned}\mathbb{P}(E_2) &= V_b 3^{-b} \left(1 - \frac{\delta\sqrt{b}}{2w}\right)^b \\ &= V_b 3^{-b} \exp\left(b \log\left(1 - \frac{\delta\sqrt{b}}{2w}\right)\right) \\ &\ge V_b 3^{-b} \exp\left(-\frac{b}{2}\frac{1}{4w^2} \cdot 8w\delta\sqrt{b}\right) \\ &= V_b 3^{-b} \exp\left(-\frac{b}{2}\frac{1}{4w^2} \cdot o(1)\right).\end{aligned}$$

We now bound the second term. Since $I_b(u)$ is decreasing in u, we first upper bound its argument

(for large enough b and w) as $\frac{1}{2w'}(1+\delta\sqrt{b}) \le \frac{1}{2w}(1+3\delta\sqrt{b})$. Let $\xi = \frac{1}{2w}(1+3\delta\sqrt{b})$. By Lemma 3.4,

$$\begin{aligned}
\mathbb{P}(E_1|E_2) &\ge I_b(\xi) \\
&\gtrsim \frac{1}{\sqrt{b}}\left(1-\xi^2\right)^{\frac{b}{2}} \\
&= \frac{1}{\sqrt{b}}\exp\left(\frac{b}{2}\log\left(1-\xi^2\right)\right) \\
&\ge \frac{1}{\sqrt{b}}\exp\left(-\frac{b}{2}\xi^2(1+\xi^2)\right) \\
&\ge \frac{1}{\sqrt{b}}\exp\left(-\frac{b}{2}\frac{1}{4w^2}\left(1+3\delta\sqrt{b}\right)^2\left(1+\frac{1}{w^2}\right)\right) \\
&= \mathrm{poly}(b^{-1})\exp\left(-\frac{b}{2}\frac{1}{4w^2}(1+o(1))\right).
\end{aligned}$$

The desired bound now follows from combining our bounds on $\mathbb{P}(E_1)$ and $\mathbb{P}(E_1|E_2)$. □

Let $p = \Omega(\mathrm{poly}(b^{-1})V_b 3^{-b}\exp(-\frac{b}{2}\frac{1+o(1)}{4w^2}))$ be the least probability with which two "close" points share a filter in a random ball lattice as in Lemma 3.6. Setting $N = \Theta(p^{-1}b\log(w/\delta)) = O(\mathrm{poly}(b^b))$ allows all checks for $x, y \in L \cap [0, 6w]$ to succeed with probability at least $1/2$ by the union bound. Thus it is possible to check that a filter family sampled from $\mathfrak{F}_0$ satisfies the stronger condition in $O(\mathrm{poly}(b^b))$, and property 1 follows. To get property 2, note that there are only $N = O(\mathrm{poly}(b^b))$ offsets, and thus computing the set of filters containing a given point can be done by iterating over all N offsets and finding for each offset the filter (if any) that contains the point.

It remains to verify property 4. Let $x, y \in \mathbb{R}^b$ be such that $\|x-y\|_2 \ge t \ge 0$. The construction succeeds with probability at least $1/2$. Therefore,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathcal{F}\sim\mathfrak{F}}(|\mathcal{F}(x)\cap\mathcal{F}(y)|) &= \mathop{\mathbb{E}}_{\mathcal{F}\sim\mathfrak{F}_0}(|\mathcal{F}(x)\cap\mathcal{F}(y)| \mid \text{construction succeeds}) \\
&\le 2\mathop{\mathbb{E}}_{\mathcal{F}\sim\mathfrak{F}_0}(|\mathcal{F}(x)\cap\mathcal{F}(y)|).
\end{aligned}$$

We prove a lemma that lets us bound the expected value on the right-hand side:

Lemma 3.7. *Let $x, y \in \mathbb{R}^b$ be such that $\|x-y\|_2 \ge t \ge 0$. Suppose an offset $v \in [0, 3w]^b$ is sampled uniformly at random. The probability there exists a point $u \in v + 3w\cdot\mathbb{Z}^b$ such that $x, y \in \mathcal{B}(u, w)$ is at most $V_b 3^{-b}\exp(-\frac{b}{2}\frac{t^2}{4w^2})$.*

Proof. Let E_1 be the event that there exists a $u \in v + 3w\cdot\mathbb{Z}^b$ such that $x, y \in \mathcal{B}(u, w)$ and E_2 be

the event that there exists a $u \in v + 3w \cdot \mathbb{Z}^b$ such that $x \in \mathcal{B}(u, w)$. By Lemma 3.4,

$$\begin{aligned}\mathbb{P}(E_1) &= \mathbb{P}(E_2) \cdot \mathbb{P}(E_1|E_2) \\ &\leq V_b 3^{-b} \cdot I_b\left(\frac{t}{2w}\right) \\ &\leq V_b 3^{-b}\left(1 - \frac{t^2}{4w^2}\right)^{\frac{b}{2}} \\ &\leq V_b 3^{-b} \exp\left(-\frac{b}{2}\frac{t^2}{4w^2}\right).\end{aligned}$$

□

Applying Lemma 3.7, we get that

$$\begin{aligned}\mathop{\mathbb{E}}_{\mathcal{F}\sim\mathfrak{F}_0}(|\mathcal{F}(x) \cap \mathcal{F}(y)|) &\leq N \cdot V_b 3^{-b} \exp\left(-\frac{b}{2}\frac{t^2}{4w^2}\right) \\ &= O\left(\text{poly}(b) \exp\left(-\frac{b}{2}\frac{t^2 - 1 - o(1)}{4w^2}\right)\right),\end{aligned}$$

which gives us property 4 and completes our proof of Proposition 3.5.

3.5. Tensoring Up with CountSketch Splitters

In this section, we combine a tensoring operation on LSF families with a Euclidean analog of splitters [NSS95, AMS06] to construct a distribution over filter families for $\mathbb{R}^B$. This distribution will give us an efficient data structure for Euclidean c-approximate near neighbors in $\mathbb{R}^B$. In Section 3.5.1, we construct CountSketch splitters, a collection of orthogonal decompositions with a "splitting" property. In Section 3.5.2, we show how to use CountSketch splitters, tensoring, and Proposition 3.5 to get the desired distribution over filter families for $\mathbb{R}^B$.

3.5.1. A Collection of "Splitting" Orthogonal Decompositions

In this subsection, we assume without loss of generality that both the initial dimension d and the dimension d' ($d' < d$) of the space that we decompose $\mathbb{R}^d$ into are powers of two. We can ensure this by padding zeroes to either space while increasing the dimension by at most a constant factor.

We describe how to construct a collection $\mathfrak{P}$ of orthogonal decompositions of $\mathbb{R}^d$ into $\mathbb{R}^{d'}$ such that for any vector $x \in \mathbb{R}^d$, there is an orthogonal decomposition in $\mathfrak{P}$ that "splits" x into components in $\mathbb{R}^{d'}$ that are almost equal in length. Using a modified CountSketch [CCF04], we get this splitting property while having only $\text{poly}(d^{d/d'})$ orthogonal decompositions in $\mathfrak{P}$. We call the resulting family $\mathfrak{P}$ of orthogonal decompositions *CountSketch splitters*.

Theorem 3.8 (CountSketch splitters)**.** *For all $0 < \varepsilon < 1/2$ and $d' = \Omega(1/\varepsilon^2)$, there exists a collection $\mathfrak{P}$ of orthogonal decompositions of $\mathbb{R}^d$ into $\mathbb{R}^{d'}$ such that $|\mathfrak{P}| = O(\text{poly}(d^{d/d'}))$ and for*

$$A_0 := d' \left\{ \underbrace{\begin{bmatrix} \underbrace{\sqrt{\frac{d'}{d}} \quad \sqrt{\frac{d'}{d}} \quad \cdots \quad \sqrt{\frac{d'}{d}}}_{d/d'} & & & \\ & \underbrace{\sqrt{\frac{d'}{d}} \quad \sqrt{\frac{d'}{d}} \quad \cdots \quad \sqrt{\frac{d'}{d}}}_{d/d'} & & \\ & & \ddots & \\ & & & \underbrace{\sqrt{\frac{d'}{d}} \quad \sqrt{\frac{d'}{d}} \quad \cdots \quad \sqrt{\frac{d'}{d}}}_{d/d'} \end{bmatrix}}_{d} \right.$$

Figure 3.1.: Matrix A_0.

all $x \in \mathbb{R}^d$ of unit norm, there exists a decomposition $\mathcal{P} \in \mathfrak{P}$ such that

$$\left| \sqrt{\frac{d}{d'}} \|Px\|_2 - 1 \right| < \varepsilon$$

for all $P \in \mathcal{P}$.

To construct such a collection $\mathfrak{P}$, we introduce a variation of CountSketch using 2-wise independent permutations [AL13]. The traditional CountSketch implementation [CCF04] gives a family of linear maps from $\mathbb{R}^d$ to $\mathbb{R}^{d'}$, where $d' = \Omega(1/(\varepsilon^2\delta))$, such that for all $x \in \mathbb{R}^d$, the distortion is within $1 \pm \varepsilon$ with probability at least $1 - \delta$. (See Section 3.2 for the definition of distortion.) We prove a similar result for a modified CountSketch where each element of the family is a projection. CountSketch can be derandomized with k-wise independent hash families; we do the same for our variant with 2-wise independent permutations.

Define $\varepsilon := 10/\sqrt{d'}$. Furthermore, let A_0 be a $d' \times d$ matrix with exactly one non-zero entry per column and exactly d/d' non-zero entries per row, such that all of its non-zero entries have value $\sqrt{d'/d}$. (See Figure 3.1.) Then, let H be a family of 2-wise independent permutations of $[d]$ and Σ be a family of 4-wise independent hash functions $[d] \to \{\pm 1\}$. Let $A_{h,\sigma}$ for $h \in H$ and $\sigma \in \Sigma$ be the matrix obtained by permuting the columns of A_0 by h and then multiplying the i-th column by $\sigma(i)$ for each $i \in [d]$. We define our modified CountSketch family to be $\mathcal{A} := \{A_{h,\sigma} : h \in H, \sigma \in \Sigma\}$. To analyze this family, we use the second-moment method, following the approach of [CJN18].

Lemma 3.9. *Suppose A is sampled uniformly at random from $\mathcal{A}$. For all $x \in \mathbb{R}^d$ of unit norm,*

$$\mathbb{P}\left(\left| \sqrt{\frac{d}{d'}} \|Ax\|_2 - 1 \right| > \varepsilon \right) < \frac{1}{10}.$$

Proof. Let $\eta_{r,i}$ be an indicator for whether $A_{r,i}$ is non-zero, and let σ_i be the sign of the non-zero entry of $A_{\cdot,i}$. Then for A sampled uniformly at random, the $\eta_{\cdot,j}$ vectors are drawn from a 2-wise independent permutation of d elements and the σ_i values are 4-wise independent, with the sets of values $\{\eta_{\cdot,\cdot}\}$ and $\{\sigma_\cdot\}$ themselves being independent of each other.

For the event to not occur, it suffices that

$$\frac{d}{d'}\|Ax\|_2^2 - 1 \in [-2\varepsilon + \varepsilon^2, 2\varepsilon].$$

Therefore, it is enough to analyze the random variable $Z \coloneqq \frac{d}{d'}\|Ax\|_2^2 - 1$. Rewrite

$$Z = \sum_{r=1}^{d'} \sum_{i \neq j} \eta_{r,i}\eta_{r,j}\sigma_i\sigma_j x_i x_j.$$

By Chebyshev's inequality,

$$\begin{aligned}
\mathbb{P}(|Z| > \varepsilon) &\leq \frac{1}{\varepsilon^2}\mathbb{E}(Z^2) \\
&= \frac{1}{\varepsilon^2}\mathbb{E}\left(\sum_{r=1}^{d'}\left(\sum_{i\neq j}\eta_{r,i}\eta_{r,j}\sigma_i\sigma_j x_i x_j\right)^2\right) \\
&\qquad + \frac{1}{\varepsilon^2}\mathbb{E}\left(\sum_{r\neq s}\left(\left(\sum_{i\neq j}\eta_{r,i}\eta_{r,j}\sigma_i\sigma_j x_i x_j\right)\left(\sum_{i\neq j}\eta_{s,i}\eta_{s,j}\sigma_i\sigma_j x_i x_j\right)\right)\right).
\end{aligned}$$

Due to the 4-wise independence of the σ_i and the fact that $\eta_{r,i}\eta_{s,i} = 0$ when $r \neq s$, we have that the expectations of all terms in the second summation are zero. By the same reasoning, the expectation of the first term is equal to

$$\begin{aligned}
2d'\sum_{i\neq j}\mathbb{E}(\eta_{r,i}\eta_{r,j})x_i^2x_j^2 &= 2d'\sum_{i\neq j}\frac{d/d'}{d}\frac{d/d'-1}{d}x_i^2x_j^2 \\
&\leq 2\left(\frac{1}{d'} - \frac{1}{d}\right)\left(\sum_i x_i^2\right)^2 \leq \frac{2}{d'}.
\end{aligned}$$

Thus $\mathbb{P}(|Z| > \varepsilon) \leq 2/(\varepsilon^2 d') < 1/10$. □

We now construct the collection $\mathfrak{P}$ of orthogonal decompositions of $\mathbb{R}^d$ into $\mathbb{R}^{d'}$. Let $\ell \in \mathbb{Z}$ be such that $d = 2^\ell d'$. Let $d_j = 2^j d'$ and $\varepsilon_j = 10/\sqrt{d_j}$ for $j \in \{0, \ldots, \ell\}$. Let $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$ be families of modified CountSketch projections as defined above, where $\mathcal{A}_i$ consists of linear maps $\mathbb{R}^{d_i} \to \mathbb{R}^{d_{i-1}}$.

Each orthogonal decomposition in $\mathfrak{P}$ can be constructed in the following manner: Start with some $A_0 \in \mathcal{A}_\ell$. Let $A_1 \colon \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_{\ell-1}}$ be any projection onto $\ker(A_0)$ (equivalently, a projection onto the orthogonal complement of the row space of A_0). We then choose $A_{00}, A_{10} \in \mathcal{A}_{\ell-1}$, and set A_{01} and A_{11} to be projections onto $\ker(A_{00})$ and $\ker(A_{10})$, respectively. Similarly, we choose $A_{s_1\cdots s_{i-1}0} \in \mathcal{A}_{\ell-i+1}$ for each $(s_1, \ldots, s_{i-1}) \in \{0,1\}^{i-1}$ for all $i \in [\ell]$ and set $A_{s_1\cdots s_{i-1}1}$ to be a projection onto $\ker(A_{s_1\cdots s_{i-1}0})$. For each $(s_1, \ldots, s_\ell) \in \{0,1\}^\ell$, define $P_{s_1\cdots s_\ell} \coloneqq A_{s_1\cdots s_\ell}A_{s_1\cdots s_{\ell-1}}\cdots A_{s_1s_2}A_{s_1}$. We take the set

$$\mathcal{P} \coloneqq \left\{P_{s_1\cdots s_\ell} : (s_1, \ldots, s_\ell) \in \{0,1\}^\ell\right\}$$

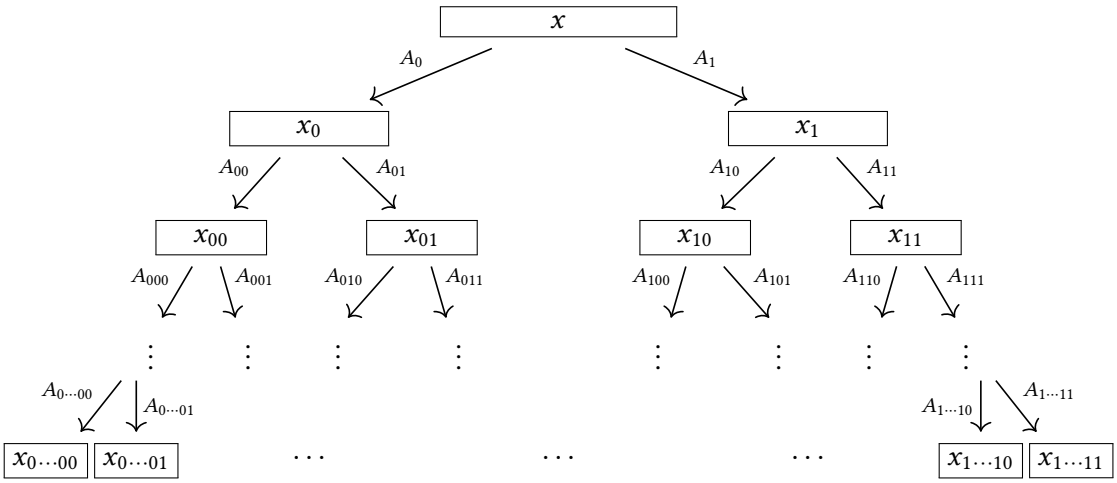

Figure 3.2.: An orthogonal decomposition $\mathcal{P} \in \mathfrak{P}$ applied to some vector $x \in \mathbb{R}^d$.

to be an element of $\mathfrak{P}$. (In particular, it is not difficult to see that $\mathcal{P}$ is an orthogonal decomposition of $\mathbb{R}^d$.) For a diagram of this process, see Figure 3.2.

We now prove that the set $\mathfrak{P}$ of all such orthogonal decompositions $\mathcal{P}$ (i.e., over all choices of elements from the $\mathcal{A}_i$) has the desired properties for Theorem 3.8. Suppose $x \in \mathbb{R}^d$ is of unit norm. By Lemma 3.9 there is some $A_0 \in \mathcal{A}_\ell$ that projects x to $x_0 \in \mathbb{R}^{d_{\ell-1}}$ with distortion $1 \pm \varepsilon_\ell$. Then A_1 also projects x to $x_1 \in \mathbb{R}^{d_{\ell-1}}$ with distortion $1 \pm \varepsilon_\ell$ by the Pythagorean theorem:

$$\|x_0\|^2 + \|x_1\|^2 = \|x\|^2 .$$

Inducting on i, we have by Lemma 3.9 that there exists some $A_{s_1\cdots s_{i-1}0} \in \mathcal{A}_i$ at each level that projects $x_{s_1\cdots s_{i-1}}$ with distortion $1 \pm \varepsilon_{\ell-i+1}$ to $x_{s_1\cdots s_{i-1}0}$. Summing the geometric series due to the exponentially increasing ε_j, observe that the total distortion of any $x_{s_1\cdots s_\ell}$ relative to x in this construction is bounded by

$$\begin{aligned}\exp\left(\log(1 \pm \varepsilon_\ell) + \log(1 \pm \varepsilon_{\ell-1}) + \cdots + \log(1 \pm \varepsilon_0)\right) &= \exp(O(\pm\varepsilon_\ell \pm \varepsilon_{\ell-1} \pm \cdots \pm \varepsilon_0)) \\ &= 1 \pm O(\varepsilon),\end{aligned}$$

since $\varepsilon_0 = 1/\sqrt{d'} = O(\varepsilon)$. Therefore, the orthogonal decomposition

$$\mathcal{P} := \{P_{s_1\cdots s_\ell} x : (s_1, \ldots, s_\ell) \in \{0,1\}^\ell\}$$

satisfies the distortion property required for Theorem 3.8.

Finally, we check that $|\mathfrak{P}|$ is small enough. Note that each element of $\mathcal{A}_i$ can be characterized by a permutation h on d elements drawn from a 2-wise independent permutation family and a function $\sigma : [d] \to \{\pm 1\}$ drawn from a 4-wise independent hash family. Both h and σ can be characterized by $O(\log d)$ bits, so $|\mathcal{A}_i| = O(\text{poly}(d))$ for all i. Each element of $\mathfrak{P}$ is defined in terms of $2^\ell - 1 \le d/d'$ selections from $\mathcal{A}_i$ families. Hence $|\mathfrak{P}| = O(\text{poly}(d)^{d/d'}) = O(\text{poly}(d^{d/d'}))$, and our construction of $\mathfrak{P}$ for Theorem 3.8 is complete.

3.5.2. Efficient Filter Families for $\mathbb{R}^B$

In this subsection, B and ε are functions of n such that $B = \Theta(\log^{1+\beta} n)$ and $\varepsilon = \Theta(\log^{-\alpha/2} n)$, where α and β are as defined in Section 3.3. (One can also consider the explicit parameter setting $\alpha = 4/5$ and $\beta = 2/5$.) Let $w = \Theta(\log^{\beta/2} n)$ be as defined in Section 3.4. Finally, we use $\mathfrak{P}$ to refer to a family of CountSketch splitters decomposing $\mathbb{R}^B$ into subspaces of dimension b.

We construct a filter family with Las Vegas properties for $\mathbb{R}^B$ via a tensoring operation. The tensoring operation produces a filter family for $\mathbb{R}^B$ by combining B/b filter families for $\mathbb{R}^b$ with respect to an orthogonal decomposition of $\mathbb{R}^B$. We take the union of several tensored filter families, one for each element of $\mathfrak{P}$, to obtain the filter family for $\mathbb{R}^B$ that we want. The splitting property of $\mathfrak{P}$ gives this filter family the desired Las Vegas properties.

We begin by defining what it means to "tensor" together filter families and then state the main result (Proposition 3.10) for this subsection.

Definition 3.4. The *tensoring* operation takes an orthogonal decomposition $\{P_i\}_{i=1}^{d/d'}$ of $\mathbb{R}^d$ into $\mathbb{R}^{d'}$ and filter families $\mathcal{F}_1, \ldots, \mathcal{F}_{d/d'}$ for $\mathbb{R}^{d'}$ and returns a filter family $\mathcal{G}$ whose filters are such that

$$\mathcal{G}(x) = \mathcal{F}_1\left(\sqrt{\frac{d}{d'}}P_1 x\right) \times \cdots \times \mathcal{F}_{d/d'}\left(\sqrt{\frac{d}{d'}}P_{d/d'} x\right).$$

In particular, the filters of $\mathcal{G}$ are implicitly characterized by defining $\mathcal{G}(x)$ for each $x \in \mathbb{R}^d$.

Proposition 3.10. *There is a distribution $\mathfrak{G}$ over filter families for $\mathbb{R}^B$ with the following properties:*

1. *A filter family $\mathcal{G}$ can be sampled from $\mathfrak{G}$ in expected time $O(\mathrm{poly}(B^{B/b} b^b))$.*
2. *For all $x \in \mathbb{R}^B$, $\mathcal{G}(x)$ can be computed in expected time $O(\mathrm{poly}(B^{B/b} b^b) + |\mathcal{G}(x)|)$.*
3. *For all $x, y \in \mathbb{R}^B$ such that $\|x - y\|_2 \le 1$, $\mathcal{G}(x) \cap \mathcal{G}(y) \neq \emptyset$.*
4. *Let $t \ge 0$ be a fixed constant. For all $x, y \in \mathbb{R}^B$ such that $\|x - y\|_2 \ge t$,*

$$\mathop{\mathbb{E}}_{\mathcal{G}\sim\mathfrak{G}}(|\mathcal{G}(x) \cap \mathcal{G}(y)|) \le O\left(\mathrm{poly}(B^{B/b}) \exp\left(-\frac{B}{2}\frac{t^2 - 1 - o(1)}{4w^2}\right)\right).$$

In particular, for $t = 0$,

$$\mathop{\mathbb{E}}_{\mathcal{G}\sim\mathfrak{G}}(|\mathcal{G}(x)|) = O\left(\mathrm{poly}(B^{B/b}) \exp\left(\frac{B}{2}\frac{1 + o(1)}{4w^2}\right)\right).$$

Let $\mathfrak{F}$ be as in Proposition 3.5 and $\mathfrak{P}$ be as in Theorem 3.8, with the elements of $\mathfrak{P}$ decomposing $\mathbb{R}^B$ into copies of $\mathbb{R}^b$. To sample a filter family from $\mathfrak{G}$, we first define $\mathcal{G}_\mathcal{P}$ as a filter family obtained by applying the tensoring operation to $\mathcal{P} \in \mathfrak{P}$ and B/b filter families drawn from $\mathfrak{F}$. Now, define $\mathcal{G}_0 := \bigcup_{\mathcal{P}\in\mathfrak{P}} \mathcal{G}_\mathcal{P}$. Our sampled family $\mathcal{G}$ is defined by scaling $\mathcal{G}_0$ so that $\mathcal{G}(x) = \mathcal{G}_0(x/(1+\varepsilon))$.

Such a $\mathcal{G}$ can be sampled in time

$$|\mathfrak{P}|\,\frac{B}{b}O(\mathrm{poly}(b^b)) = O(\mathrm{poly}(B^{B/b}b^b)),$$

which yields property 1. To compute $\mathcal{G}(x)$, we first compute $\mathcal{F}(x)$ in time $O(\mathrm{poly}(b^b))$ for each $\mathcal{F}$ sampled from $\mathfrak{F}$. There are $\mathrm{poly}(B^{B/b})$ such $\mathcal{F}$, so the time for this step is $O(\mathrm{poly}(B^{B/b}b^b))$. Finally, $\mathcal{G}(x)$ can be generated from the sets $\mathcal{F}(x)$ in $O(|\mathcal{G}(x)|)$, giving us property 2.

To see property 4, suppose $x, y \in \mathbb{R}^B$ are such that $\|x-y\|_2 \geq t \geq 0$. Let $x' = x/(1+\varepsilon)$ and $y' = y/(1+\varepsilon)$. Then for $\mathcal{G}_\mathcal{P}$, where $\mathcal{P} \coloneqq \{P_i\}_{i=1}^{B/b} \in \mathfrak{P}$ and $\mathcal{F}_1, \ldots, \mathcal{F}_{B/b}$ are sampled from $\mathfrak{F}$, we have

$$\begin{aligned}
\mathbb{E}(|\mathcal{G}_\mathcal{P}(x') \cap \mathcal{G}_\mathcal{P}(y')|) &= \prod_{i=1}^{B/b} \mathop{\mathbb{E}}_{\mathcal{F}_i \sim \mathfrak{F}} \left(\left| \mathcal{F}_i\left(\sqrt{\frac{B}{b}}P_i x'\right) \cap \mathcal{F}_i\left(\sqrt{\frac{B}{b}}P_i y'\right) \right| \right) \\
&\leq \prod_{i=1}^{B/b} \mathrm{poly}(b) \exp\left(-\frac{b}{2} \frac{\frac{B}{b}\,\|P_i x' - P_i y'\|_2^2 - 1 - o(1)}{4w^2} \right) \\
&= O\left(\mathrm{poly}(b^{B/b}) \exp\left(-\frac{B}{2} \frac{t^2 - 1 - o(1)}{4w^2} \right) \right).
\end{aligned}$$

And since

$$\mathop{\mathbb{E}}_{\mathcal{G}\sim\mathfrak{G}}(|\mathcal{G}(x) \cap \mathcal{G}(y)|) = \mathrm{poly}(B^{B/b})\,\mathbb{E}(|\mathcal{G}_\mathcal{P}(x') \cap \mathcal{G}_\mathcal{P}(y')|),$$

the above gives us property 4.

Finally, to verify property 3, suppose $x, y \in \mathbb{R}^B$ are such that $\|x-y\|_2 \leq 1$. Let $x' = x/(1+\varepsilon)$ and $y' = y/(1+\varepsilon)$. By the splitting property of $\mathfrak{P}$ (see Theorem 3.8), there exists some $\mathcal{P}$ in $\mathfrak{P}$ such that for all $P \in \mathcal{P}$,

$$\sqrt{\frac{B}{b}}\,\|Px' - Py'\|_2 \leq 1.$$

Then $\mathcal{G}(x) \cap \mathcal{G}(y) \supseteq \mathcal{G}_\mathcal{P}(x') \cap \mathcal{G}_\mathcal{P}(y') \neq \emptyset$ by the tensor definition of $\mathcal{G}_\mathcal{P}$ and property 3 of $\mathfrak{F}$. This completes our proof of Proposition 3.10

Corollary 3.11. *There exists a Las Vegas data structure for Euclidean c-approximate near neighbors in $\mathbb{R}^B$ with expected query time $O(n^\rho)$, space usage $O(n^{1+\rho})$, and preprocessing time $O(n^{1+\rho})$ for $\rho = 1/c^2 + o(1)$.*

Proof. We use the distribution $\mathfrak{G}$ constructed in Proposition 3.10. Recall our parameter settings for b and B in terms of α and β, i.e., $b = \Theta(\log^\alpha n)$ and $B = \Theta(\log^{1+\beta} n)$. Notice that $\mathrm{poly}(B^{B/b}b^b) = n^{o(1)}$. Recall also that $w = \Theta(\log^{\beta/2} n)$. For w, we specifically set

$$w \coloneqq \sqrt{\frac{B}{8\log n}}c.$$

By Proposition 3.10, for $\mathcal{G} \sim \mathfrak{G}$, the expected number of filters containing any $x \in \mathbb{R}^B$ is $O(n^{1/c^2+o(1)})$. In preprocessing, we sample $\mathcal{G}$ and compute $\mathcal{G}(x)$ for each x in $\mathcal{D}$, which takes

time $O(n^{1+1/c^2+o(1)})$. Because we need $|\mathcal{G}(x)|$ space to store the filters containing x, space usage is also $O(n^{1+1/c^2+o(1)})$. Finally, to answer queries with this data structure, we can compute $\mathcal{G}(q)$ for a query point $q \in \mathbb{R}^B$ in expected time $O(n^{1/c^2+o(1)})$. The expected number of collisions between q and distant points in $\mathcal{D}$ is $n \cdot O(n^{-1+1/c^2+o(1)}) = O(n^{1/c^2+o(1)})$. Hence the expected runtime of a query is $O(n^{1/c^2+o(1)})$. □

3.6. Projecting Down

We complete our construction of the data structure in this section by defining a two-stage sequence of dimensionality reductions that efficiently reduces the original problem into d/B instances of approximate near neighbors in B dimensions. In this reduction, it is guaranteed that any pair of near neighbors are considered near neighbors in *at least* one of the lower-dimensional problems.

Our dimensionality reduction improves on previous implementations of the "one-sided" dimensionality reduction idea [Ahl17, SW17, Wyg17] in that the overhead per point is now $\tilde{O}(d)$ instead of having a linear dependence on d^2. The data structure in [Ahl17] implements this idea for Hamming space and has $O(1)$ false positives from each lower-dimensional problem. Each false positive requires $O(d)$ time to filter out, resulting in a total cost of $O(d^2/B)$. Our two-stage dimensionality reduction lets us spend only $\tilde{O}(d)$ time handling false positives. The data structures of [SW17] and [Wyg17] implement one-sided dimensionality reduction for Euclidean space, but require $\Omega(d^2)$ time to compute the dimensionality reduction since they multiply by a random rotation in $\mathbb{R}^d$. We note that this can be improved to $\tilde{O}(d)$ by applying a slightly modified FastJL.

After applying the dimensionality reduction, we use the data structure constructed for $\mathbb{R}^B$ from Section 3.5 to get a Las Vegas data structure that solves c-approximate near neighbors for all dimensions with an exponent of $\rho = 1/c^2 + o(1)$.

3.6.1. Efficient Distributions of Orthogonal Decompositions

In this subsection, we prove two "one-sided" dimensionality reduction results (Proposition 3.12 and Lemma 3.14) using orthogonal decompositions. These results will be used to reduce the dimension of the problem from d to B.

For this subsection, we assume without loss of generality that the initial dimension d is a power of two. We can guarantee this by padding zeros, which increases d by at most a constant factor.

Proposition 3.12. *For all $0 < \varepsilon < 1/2$, $0 < \delta < 1/2$, and $d' = \Omega(\varepsilon^{-2} \log(1/(\varepsilon\delta)) \log(1/\delta))$, there exists a distribution $\mathscr{P}_1$ over orthogonal decompositions of $\mathbb{R}^d$ into $\mathbb{R}^{d'}$ such that for all $x \in \mathbb{R}^d$ of unit norm and $i \in [d/d']$, where $\mathcal{P} \coloneqq \{P_i\}_{i=1}^{d/d'}$ is sampled from $\mathscr{P}_1$,*

$$\mathop{\mathbb{P}}_{\mathcal{P}\sim\mathscr{P}_1}\left(\left|\sqrt{\frac{d}{d'}}\left\|P_i x\right\|_2 - 1\right| > \varepsilon\right) < \delta.$$

Furthermore, the set of values $\{P_i x\}_{i=1}^{d/d'}$ is computable in $O(d \log d)$ time.

To construct this distribution $\mathscr{P}_1$, we use a modified FastJL [AC09]. Let H_d be the d-dimensional Hadamard matrix, and consider the distribution $\mathcal{A}$ over $d' \times d$ matrices defined as follows: Sample a $d \times d$ diagonal matrix D whose diagonal entries are independent Rademachers and a $d' \times d$ matrix S whose rows are standard basis vectors sampled without replacement. We take $A := SH_dD$ to be the sampled element of $\mathcal{A}$. Note that A is orthogonal. We analyze $\mathcal{A}$ with the following lemma:

Lemma 3.13. *Suppose $A \in \mathcal{A}$ is sampled uniformly at random. Then for all $x \in \mathbb{R}^d$ of unit norm,*

$$\mathbb{P}\left(\left|\sqrt{\frac{d}{d'}}\,\|Ax\|_2 - 1\right| > \varepsilon\right) < \delta.$$

Proof. Let $\mathcal{A}'$ be the distribution $\mathcal{A}$ with the modification that the rows of the coordinate sampling matrix S are sampled *with* replacement. Applying [Hoe63, Theorem 4] followed by [CNW16, Theorem 9], we have for $p = \log(1/\delta)$ that

$$\begin{aligned}\mathop{\mathbb{E}}_{A\sim\mathcal{A}}\left(\left|\frac{d}{d'}\,\|Ax\|_2^2 - 1\right|^p\right) &\le \mathop{\mathbb{E}}_{A\sim\mathcal{A}'}\left(\left|\frac{d}{d'}\,\|Ax\|_2^2 - 1\right|^p\right)\\ &< \varepsilon^p\delta,\end{aligned}$$

since $x \mapsto |x|^p$ is a convex function. By Markov's inequality, the lemma follows. □

We now construct our distribution $\mathscr{P}_1$ over orthogonal decompositions by sampling $D \in \mathbb{R}^{d\times d}$ and a random permutation matrix $P \in \mathbb{R}^{d\times d}$. We define the orthogonal decomposition to be the rows of PH_dD partitioned into d/d' matrices of dimension $d' \times d$. Since PH_dD is orthogonal, the resulting family is an orthogonal decomposition. By Lemma 3.13, this distribution over orthogonal decompositions satisfies the condition of Proposition 3.12.

Lemma 3.14 ([SW17])**.** *For all $0 < \varepsilon < 1/2$, $0 < \delta < 1/2$, and $d' = \Omega(\varepsilon^{-2}\log(1/\delta))$, there exists a distribution $\mathscr{P}_2$ over orthogonal decompositions of $\mathbb{R}^d$ into $\mathbb{R}^{d'}$ such that for all $x \in \mathbb{R}^d$ of unit norm and $i \in [d/d']$, where $\mathcal{P} = \{P_i\}_{i=1}^{d/d'}$ is sampled from $\mathscr{P}_2$,*

$$\mathop{\mathbb{P}}_{\mathcal{P}\sim\mathscr{P}_2}\left(\left|\sqrt{\frac{d}{d'}}\,\|P_ix\|_2 - 1\right| > \varepsilon\right) < \delta.$$

Proof. Sample a random rotation R in d-dimensions and define the orthogonal decomposition to be the rows of R split into $\frac{d}{d'}$ matrices of dimension $d' \times d$. By the Johnson-Lindenstrauss lemma, if $d' = \Omega(\varepsilon^{-2}\log(1/\delta))$, then the property holds with the desired probability. □

3.6.2. Constructing the Data Structure for $\mathbb{R}^d$

Let ε be a function of n with $\varepsilon = \Theta(\log^{-\beta/2} n)$, where β is as defined in Section 3.3. (One can also consider the explicit parameter setting $\beta = 2/5$.)

Our algorithm for c-approximate near neighbors works by a sequence of two reductions to lower dimensional c-approximate near neighbors problems. In the first reduction, we reduce the dimension by applying an orthogonal decomposition drawn from $\mathscr{P}_1$ (as in Proposition 3.12)

for $\delta = 1/(nd)$ and ε as above to obtain d/B' problems in dimension $B' \coloneqq \Theta(\varepsilon^{-2}\log^2(nd))$. We then reduce the dimension further by applying an orthogonal decomposition drawn from $\mathscr{P}_2$ (as in Lemma 3.14) with $\delta = 1/n$ and ε as above for each B'-dimensional problem to obtain a set of B-dimensional problems. Finally, we apply the algorithm of Corollary 3.11. We describe the steps in greater detail below.

A General Reduction

We give a general reduction from c-approximate near neighbors in $\mathbb{R}^d$ to d/d' instances of c'-approximate near neighbors in $\mathbb{R}^{d'}$ for $c' \coloneqq c(1-\varepsilon)$, assuming a distribution $\mathscr{P}$ over orthogonal decompositions of $\mathbb{R}^d$ into $\mathbb{R}^{d'}$ such that for $\mathcal{P} \coloneqq \{P_i\}_{i=1}^{d/d'}$ sampled from $\mathscr{P}$,

$$\mathop{\mathbb{P}}_{\mathcal{P}\sim\mathscr{P}}\left(\left|\sqrt{\frac{d}{d'}}\left\|P_i x\right\|_2 - 1\right| > \varepsilon\right) < \delta$$

for all $x \in \mathbb{R}^d$ of unit norm.

We start by sampling an orthogonal decomposition $\mathcal{P} \sim \mathscr{P}$. Then $\mathcal{P}$ decomposes $\mathbb{R}^d$ as a direct sum of d/d' subspaces of dimension d'. For any two points $x, y \in \mathbb{R}^d$ such that $\|x-y\|_2 \le 1$, if we let $x_1, \ldots, x_{d/d'}$ and $y_1, \ldots, y_{d/d'}$ be the components of x and y under $\mathcal{P}$, respectively, then there exists some i such that $\sqrt{d/d'}\,\|x_i - y_i\|_2 \le 1$. Therefore, if x and y are near neighbors in $\mathbb{R}^d$, then they are also near neighbors in at least one of subspaces after scaling by $\sqrt{d/d'}$. That is, all pairs of near neighbors in $\mathbb{R}^d$ correspond to at least one pair of near neighbors in the subspaces.

Our only worry is that $\|x-y\|_2 > c$, but $\sqrt{d/d'}\,\|x_i - y_i\|_2 \le (1-\varepsilon)c$, i.e., we have a *false positive*. By the definition of $\mathscr{P}$, the expected number of false positives is bounded above by $\delta \cdot nd/d'$ for a query $y \in \mathbb{R}^d$. However, we can filter out these false positives out by continuing on in the algorithm for the lower-dimensional problem as if a false positive x did not share a filter with y. The cost of filtering out such a false positive is $O(d)$, since we need to check whether $\|x-y\|_2 > c$ for each candidate near neighbor x returned. Thus the total cost of false positives is $\delta nd^2/d'$.

Proof of Theorem 3.1

We now prove Theorem 3.1 by describing a data structure in which the above reduction is applied twice. We first apply the reduction to get from dimension d to dimension B' using $\mathscr{P}_1$ and $\delta \coloneqq 1/(nd)$. For this, we need time $\delta nd^2/B' = O(d/B')$ to handle false positives and time $O(d\log d)$ to compute the orthogonal decomposition of q. The space overhead of storing an element of $\mathscr{P}_1$ is $O(d)$. We apply the reduction again to get from dimension B' to B using $\mathscr{P}_2$ and $\delta \coloneqq 1/n$. This time, we need time $O(B'^2/B)$ to handle false positives and time $O(B'^2)$ to compute the orthogonal decomposition of q. The space overhead of storing an element of $\mathscr{P}_2$ is $O(B'^2)$. However, these costs are incurred d/B' times, once for each of the subproblems to which the reduction is applied.

We are finally ready to account for the costs of the entire data structure. Let $c'' = c(1-\varepsilon)^2$ be

the approximation factor for the B-dimensional subproblems. Then the total query time is

$$O(d \log d) + O\left(\frac{d}{B'}\right) + \frac{d}{B'}\left(O\left(\frac{B'^2}{B}\right) + O\left(B'^2\right) + \frac{B'}{B} \cdot n^{1/c''^2+o(1)}\right) = \tilde{O}(dn^{1/c^2+o(1)}),$$

the total space usage is

$$O(d) + \frac{d}{B'}\left(O(B'^2) + \frac{B'}{B} \cdot n^{1+1/c''^2+o(1)}\right) = \tilde{O}(dn^{1+1/c^2+o(1)}),$$

and the total preprocessing time is

$$n \cdot O(d \log d) + \frac{d}{B'}\left(n \cdot O(B'^2) + \frac{B'}{B} \cdot n^{1+1/c''^2+o(1)}\right) = \tilde{O}(dn^{1+1/c^2+o(1)}).$$

These match the desired exponent of $\rho = 1/c^2 + o(1)$, proving Theorem 3.1.

For the parameter setting $\alpha = 4/5$ and $\beta = 2/5$, the $o(1)$ term in the exponent is bounded by $\tilde{O}(\log^{-1/5} n)$.

Proof of Corollary 3.2

We first state a result of [Ngu14]:

Lemma 3.15 ([Ngu14, Theorem 116])**.** *There exists an explicit map $f\colon \mathbb{R}^d \to \mathbb{R}^{d\,\mathrm{poly}(\varepsilon^{-1}\log d)}$ such that for all $x, y \in \mathbb{R}^d$ such that $\|x-y\|_p \le d^{O(1)}$, we have*

$$(1-\varepsilon)\|x-y\|_p^p - \varepsilon \le \|f(x)-f(y)\|_2^2 \le (1+\varepsilon)\|x-y\|_p^p + \varepsilon.$$

To construct a data structure for Corollary 3.2 (for c-approximate near neighbors in ℓ_p) we consider a random shift of the lattice $d \cdot \mathbb{Z}^d$ and for each resulting grid cell, build a separate data structure for c-approximate near neighbors on the hypercube of side length $d+2$ that has the same center as the grid cell. Then, for each point in $\mathbb{R}^d$, we expect it to belong to $(1+2/d)^d = O(1)$ such data structures, adding a constant factor overhead. To see correctness, observe that there exists a data structure containing every pair of points $x, y \in \mathbb{R}^d$ such that $\|x-y\|_p \le 1$.

We finish by constructing a data structure for c-approximate near neighbors in ℓ_p for a hypercube of side length $d+2$. Notice that $\|x-y\|_p \le O(d\sqrt{d}) \le d^{O(1)}$ for all x, y in the hypercube. We may thus apply Lemma 3.15 with $\varepsilon = 1/\log n$ to obtain an embedding into an instance of c'-approximate near neighbors in ℓ_2, where $c' = (1-2\varepsilon)c^{p/2}$. Theorem 3.1 then gives us a Las Vegas data structure solving c-approximate near neighbors in ℓ_p with exponent $\rho = 1/c^p + o(1)$.

3.7. Las Vegas Data-Dependent Hashing

Our goal in this section is to construct a data-independent Las Vegas filter family for the unit sphere in $\mathbb{R}^B$, where $B = \Theta(\log n \cdot \log\log N)$, and as a consequence of [ALRW17] (which provides optimal space-time tradeoffs for Monte Carlo data-dependent hashing), prove Theorem 3.16. Our main result, Theorem 3.3, then follows as a consequence of the techniques in Section 3.6.

Theorem 3.16. *Suppose $\rho_u, \rho_q \geq 0$ are such that*

$$c^2\sqrt{\rho_q} + (c^2-1)\sqrt{\rho_u} \geq \sqrt{2c^2-1}.$$

Then there exists a Las Vegas data structure for Euclidean c-approximate near neighbors in $\mathbb{R}^B$ with expected query time $O(n^{\rho_q+o(1)})$, space usage $O(n^{1+\rho_u+o(1)})$, and preprocessing time $O(n^{1+\rho_u+o(1)})$.

We accomplish this by applying the high-level approach used in the previous sections to LSF on a sphere. Specifically, we construct a Las Vegas LSF family for the unit sphere in $\mathbb{R}^b$ by discretizing the sphere to a finite subset for which we have a "brute force" probabilistic construction. We then use the CountSketch splitters of Section 3.5.1 to extend this construction to the unit sphere in $\mathbb{R}^B$. We obtain Theorem 3.16 by plugging our Las Vegas LSF family for the unit sphere in $\mathbb{R}^B$ into the data-dependent data structure of [ALRW17] for $\mathbb{R}^B$. Since the one-sided dimensionality reductions of Section 3.6 still apply, the desired data structure for Theorem 3.3 follows.

The resulting construction matches the bounds of [ALRW17] across the entire space-time tradeoff. In particular, for the symmetric case, we match [AR15], achieving an exponent of $\rho = 1/(2c^2-1) + o(1)$, which is known to be optimal for data-dependent hashing.

3.7.1. Approximate Near Neighbors on a Sphere

The technical bulk of this section consists of solving (r, cr)-approximate near neighbor search on a sphere in $\mathbb{R}^d$. That is, we consider approximate near neighbors for the metric space (X, D), where $X \subseteq \mathbb{R}^d$ is a sphere and the metric D is Euclidean distance. We assume without loss of generality that the sphere is the unit sphere $\mathcal{S}(0, 1)$. However, note that this implies we can no longer assume that $r = 1$.

Like in the Euclidean setting, we can tackle approximate near neighbor search on a sphere using the framework of LSF. Specifically, we consider a class of LSF families specialized for the unit sphere known as *spherical LSF*. In spherical LSF, each asymmetric filter is given by a randomly sampled Gaussian in $\mathbb{R}^d$; membership in the filter is determined by inner products with the vector. We define this class of LSF families in greater detail below. Previously, this class of LSF families was studied in the Monte Carlo setting by [AR15, BDGL16, ALRW17].

As our goal is to handle space-time tradeoffs for LSF, we assume for the remainder of this section that all the filters and filter families we consider are *asymmetric* (see Section 3.2).

Spherical LSF

Spherical LSF on the unit sphere is defined in terms of two parameters: an update parameter η_u and a query parameter η_q. Each asymmetric filter T in spherical LSF corresponds to a vector $z \sim \mathcal{N}(0, I)$, such that a database point u belongs to T if $\langle z, u\rangle \geq \eta_u$ and a query point q belongs to T if $\langle z, q\rangle \geq \eta_q$. That is, both the update and query filter of T are spherical caps, with the sizes of these caps determined by η_u and η_q, respectively.

To analyze such a filter family, we define functions

$$F(\eta) \coloneqq \mathop{\mathbb{P}}_{z\sim\mathcal{N}(0,I)}(\langle z, x\rangle \geq \eta),$$

where $x \in \mathcal{S}(0,1)$ is an arbitrary point, and

$$G(s, \eta_u, \eta_q) \coloneqq \mathop{\mathbb{P}}_{z \sim \mathcal{N}(0,I)}(\langle z, u\rangle \geq \eta_u \text{ and } \langle z, q\rangle \geq \eta_q),$$

where $u, q \in \mathcal{S}(0,1)$ are arbitrary points such that $\|u - q\|_2 = s$. These functions give the probability that a point x lies in a filter with parameter η and the probability that a database point u and a query point q share a filter with parameters η_u and η_q, respectively.

The following lemmas provide estimates for F and G:

Lemma 3.17 ([ALRW17, AIL+15])**.** *For $\eta \to \infty$,*

$$F(\eta) = \exp\left(-(1 + o(1))\frac{\eta^2}{2}\right).$$

Lemma 3.18 ([ALRW17, AIL+15])**.** *For $\eta, \sigma \to \infty$ and $0 < s < 2$,*

$$G(s, \eta, \sigma) = \exp\left(-(1 + o(1))\frac{\eta^2 + \sigma^2 - 2\eta\sigma\mathfrak{c}(s)}{2\mathfrak{s}(s)^2}\right),$$

where $\mathfrak{c}(s) \coloneqq 1 - s^2/2$ and $\mathfrak{s}(s) \coloneqq \sqrt{1 - \mathfrak{c}^2(s)}$ are the cosine and sine, respectively, of the angle between two points that are distance s apart on the unit sphere.

3.7.2. Las Vegas Spherical LSF Families in $\mathbb{R}^b$

We start by constructing a spherical LSF family in dimension $b = \Theta(\log^\alpha n)$, where $\alpha = 2/3$, with a Las Vegas property. Specifically, we prove the following:

Proposition 3.19. *For all $r > 0$, there exists a distribution $\mathfrak{F}$ over spherical LSF families for $\mathcal{S}(0,1)$ in $\mathbb{R}^b$ with the following properties:*

1. *A filter family $\mathcal{F}$ can be sampled from $\mathfrak{F}$ in time $O(\text{poly}(b^b)/G(r, \eta_u, \eta_q))$.*
2. *For all $u, q \in \mathcal{S}(0,1)$, $\mathcal{F}_u(u)$ and $\mathcal{F}_q(q)$ can be computed in time $O(\text{poly}(b)/G(r, \eta_u, \eta_q))$.*
3. *For all $u, q \in \mathcal{S}(0,1)$ such that $\|u - q\|_2 \leq r$, $\mathcal{F}_u(u) \cap \mathcal{F}_q(q) \neq \emptyset$.*
4. *All filters in $\mathcal{F}$ are subsets of spherical LSF filters sampled with parameters η_u and η_q.*

As observed in [ALRW17], random sampling constructs good spherical LSF families with high probability. Our main challenge is sampling a spherical LSF family while being able to verify that it indeed has the desired properties for all points in the sphere. We apply a similar technique as in Section 3.4: We divide the space into grid cells, each containing a small portion of the sphere. The intersection of each grid cell and the sphere can be contained in a small spherical cap W, where we use the same shaped spherical cap for each grid cell. We then discretize W with a finer mesh L and round each point in W to its nearest neighbor in L. Finally, we sample spherical LSF families until we obtain a satisfactory LSF family for L. The resulting LSF family for the entire sphere is obtained by pasting together the filters for the rotations of W

that correspond to grid cells. In the remainder of this subsection, we describe this construction in detail.

For the first step our construction, we reduce the problem to spherical caps comparable in size to r; our resulting filter family will be the union of the filter families we construct for all the spherical caps we consider. To do so, we take a random shift of the lattice $rb \cdot \mathbb{Z}^b$ and consider the (axis-aligned) hypercubes with side length $r(b+2)$ centered at the lattice points. We build a separate filter family for the intersection of $\mathcal{S}(0,1)$ and each hypercube. Due to the random shift, each point in $\mathbb{R}^b$ belongs to $(1+2/b)^b = O(1)$ hypercubes in expectation. Hence this reduction adds only a constant factor overhead. Furthermore, for each pair of points that are at most distance r apart, there exists a hypercube containing both points, so this step does not separate any pairs of near neighbors.

Now, consider a hypercube centered at $O \in rb \cdot \mathbb{Z}^b$ that intersects $\mathcal{S}(0,1)$, and let O' be the point on $\mathcal{S}(0,1)$ closest to O. This hypercube is contained in $\mathcal{B}(O', r(b+2)\sqrt{b})$, so it suffices to construct a filter family for the spherical cap $W := \mathcal{S}(0,1) \cap \mathcal{B}(O', r(b+2)\sqrt{b})$. For $\delta > 0$, let L be a subset of W of size $\mathrm{poly}((b/\delta)^b)$ such that each point in W is at most distance $r\delta$ from a point in L.[4] Fixing $\delta = \Theta(1/b)$, we can round each point to its nearest neighbor in L with negligible loss in precision—points that are initially at most distance r apart remain at most distance $r(1+2\delta) = r(1+o(1))$ apart and points that are initially at least distance cr apart remain at least distance $(c-2\delta)r = cr(1-o(1))$ apart. This rounding therefore worsens the approximation factor by a factor of at most $1+o(1)$.

After rounding all database and query points in W to $L \subseteq W$, we show how to construct a spherical LSF family for W. Observe that $p := G(r, \eta_u, \eta_q)$ is a lower bound on the probability that a database point u and a query point q such that $\|u-q\|_2 \le r$ both belong to a randomly sampled filter. Thus, if we sample $N := \Theta(p^{-1} \log |L|)$ filters, the condition $\mathcal{F}_u(u) \cap \mathcal{F}_q(q) \neq \emptyset$ for all pairs $u, q \in L$ at most distance r apart holds with probability at least $1/2$ by the union bound. Let the resulting distribution over spherical LSF families for W be $\mathfrak{F}_0$. Given $\mathfrak{F}_0$, we sample a filter family such that the condition holds for all pairs $u, q \in L$ that are at most distance r apart. Because the success probability is $1/2$, we expect to sample at most $O(1)$ times. Checking whether the condition holds takes time $O(N|L|^2) = O(\mathrm{poly}(b^b)/G(r, \eta_u, \eta_q))$.

This spherical LSF family works for each spherical cap W associated to a hypercube, since each hypercube can be contained in the same-shaped spherical cap. Thus we only need to do the above sampling procedure once to obtain a spherical LSF family for the entire sphere by taking the union over all caps. Our last step is to analyze the properties of this LSF family and show that it satisfies the properties stated in Proposition 3.19.

By the definition of sampling above, we immediately have properties 1 and 3. Observe that for each cap, there are only $N = O(\mathrm{poly}(b)/G(r, \eta_u, \eta_q))$ filters. Thus to compute $\mathcal{F}_u(u)$ (and similarly for $\mathcal{F}_q(q)$), we can iterate over the set of filters and check for each whether $\langle z, u \rangle \ge \eta_u$. This gives us property 2. Finally, property 4 follows from the definition of the filter family.[5]

[4] One construction is to project the lattice $r\delta/\sqrt{b} \cdot \mathbb{Z}^b \cap \mathcal{B}(O', 2r(b+2)\sqrt{b})$ onto the sphere $\mathcal{S}(0,1)$. Then each point in W is at most distance $r\delta/2$ from a point in the lattice. Projecting the lattice onto $\mathcal{S}(0,1)$ increases distances by a factor of at most $\sqrt{2}$.

[5] We actually expect to reject up to $1/2$ of the filters sampled this way, so the distribution is slightly different than that of typical spherical LSF, but by the argument for property 4 in Proposition 3.5, this only affects upper bounds on expected values by a factor of 2.

3.7.3. Tensoring Up Spherical LSF

We show how to use a modification of the tensoring operation to construct a LSF family for $\mathcal{S}(0,1)$ in $\mathbb{R}^B$, where $B = \Theta(\log n \cdot \log\log n)$.

We start with some parameter settings. Define $K = \Theta(\log^{1/2} n)$, and suppose ρ_u and ρ_q are such that

$$(1 - \mathfrak{c}(r)\mathfrak{c}(cr))\sqrt{\rho_q} + (\mathfrak{c}(r) - \mathfrak{c}(cr))\sqrt{\rho_u} \geq \mathfrak{s}(r)\mathfrak{s}(cr).$$

By [ALRW17, Section 3.3.3], there exist η_u and η_q such that the following hold:

$$\begin{aligned}
\frac{F(\eta_u)}{G(r,\eta_u,\eta_q)} &\leq n^{(\rho_u + o(1))/K} \\
\frac{F(\eta_q)}{G(r,\eta_u,\eta_q)} &\leq n^{(\rho_q + o(1))/K} \\
\frac{G(cr,\eta_u,\eta_q)}{G(r,\eta_u,\eta_q)} &\leq n^{(\rho_q - 1 + o(1))/K}.
\end{aligned}$$

Let $\eta_u' \coloneqq \eta_u\sqrt{b/B}$ and $\eta_q' \coloneqq \eta_q\sqrt{b/B}$. Then

$$\begin{aligned}
\frac{F(\eta_u')}{G(r,\eta_u',\eta_q')} &\leq n^{\frac{1}{K}\frac{b}{B}(\rho_u + o(1))} \\
\frac{F(\eta_q')}{G(r,\eta_u',\eta_q')} &\leq n^{\frac{1}{K}\frac{b}{B}(\rho_q + o(1))} \\
\frac{G(cr,\eta_u',\eta_q')}{G(r,\eta_u',\eta_q')} &\leq n^{\frac{1}{K}\frac{b}{B}(\rho_q - 1 + o(1))}.
\end{aligned}$$

The spherical LSF parameters η_u' and η_q' are what we will use to instantiate the distribution $\mathfrak{F}$ of Proposition 3.19 for our tensoring construction. Finally, define the maximum distortion we allow to be $1 \pm \varepsilon$, where $\varepsilon = \log^{-\alpha/2} n$.

We begin our construction by noting that there exists a collection $\mathfrak{P}$ of orthogonal decompositions of $\mathbb{R}^B$ into $\mathbb{R}^b$ such that for any pair $u, q \in \mathcal{S}(0,1)$, there exists a $\mathcal{P} \in \mathfrak{P}$ such that every projection in $\mathcal{P}$ projects u, q, and $u - q$ with distortion $1 \pm \varepsilon$. Furthermore, $\mathfrak{P}$ has size bounded by $|\mathfrak{P}| \leq \mathrm{poly}(B^{B/b})$. Formally, we show that CountSketch splitters satisfy the following stronger version of Theorem 3.8:

Lemma 3.20. *For all $0 < \varepsilon < 1/2$ and $d' = \Omega(1/\varepsilon^2)$, there exists a collection $\mathfrak{P}$ of orthogonal decompositions of $\mathbb{R}^d$ into $\mathbb{R}^{d'}$ such that $|\mathfrak{P}| = O(\mathrm{poly}(d^{d/d'}))$ and for all $x_1, x_2, x_3 \in \mathbb{R}^d$ of unit norm, there exists a decomposition $\mathcal{P} \in \mathfrak{P}$ such that*

$$\left| \sqrt{\frac{d}{d'}} \left\| P x_i \right\|_2 - 1 \right| < \varepsilon$$

for all $P \in \mathcal{P}$ and all $i \in \{1,2,3\}$.

Proof. Let $\mathcal{A}$ be as in Lemma 3.9. By Lemma 3.9 and the union bound, for A sampled from $\mathcal{A}$,

$$\mathbb{P}\left(\left|\sqrt{\frac{d}{d'}}\|Ax_i\|_2 - 1\right| > \varepsilon \text{ for any } i \in \{1,2,3\}\right) < \frac{1}{2}.$$

Thus, we may apply the divide-and-conquer construction of Theorem 3.8 to obtain the desired result. At each level in the divide-and-conquer, there exists some $A \in \mathcal{A}_j$ that projects x_i for each $i \in \{1,2,3\}$ with low distortion by our preceding observation. □

Given this collection of orthogonal decompositions $\mathfrak{P}$, we prove the following:

Proposition 3.21. *For all r, c such that $c > 1$ and $cr \geq \sqrt{2}$, there exists a distribution $\mathfrak{G}$ over filter families for $\mathcal{S}(0,1)$ in $\mathbb{R}^b$ with the following properties:*

1. *A filter family $\mathcal{G}$ can be sampled from $\mathfrak{G}$ in time $O(n^{o(1)})$.*
2. *For all $u, q \in \mathcal{S}(0,1)$, $\mathcal{G}_u(u)$ can be computed in time $O(n^{o(1)} + |\mathcal{G}_u(u)|)$ and likewise for $\mathcal{G}_q(q)$.*
3. *For all $u, q \in \mathcal{S}(0,1)$ such that $\|u-q\|_2 \leq r$, $\mathcal{G}_u(u) \cap \mathcal{G}_q(q) \neq \emptyset$.*
4. *For all $u, q \in \mathcal{S}(0,1)$, we have that*

$$\mathop{\mathbb{E}}_{\mathcal{G}\sim\mathfrak{G}}(|\mathcal{G}_u(u)|) \leq \text{poly}(B^{B/b})\frac{F(\eta_u)}{G(r,\eta_u,\eta_q)} \leq n^{(\rho_u+o(1))/K}$$

and

$$\mathop{\mathbb{E}}_{\mathcal{G}\sim\mathfrak{G}}(|\mathcal{G}_q(q)|) \leq \text{poly}(B^{B/b})\frac{F(\eta_q)}{G(r,\eta_u,\eta_q)} \leq n^{(\rho_q+o(1))/K}.$$

Furthermore, if $\|u-q\|_2 > cr$, then

$$\mathop{\mathbb{E}}_{\mathcal{G}\sim\mathfrak{G}}(|\mathcal{G}_u(u) \cap \mathcal{G}_q(q)|) \leq \text{poly}(B^{B/b})\frac{G(cr,\eta_u,\eta_q)}{G(r,\eta_u,\eta_q)} \leq n^{(-1+\rho_u+o(1))/K}.$$

To construct our LSF family, we independently sample, for each orthogonal decomposition $\mathcal{P} \in \mathfrak{P}$, B/b filter families $\mathcal{F}^1, \ldots, \mathcal{F}^{B/b}$ from $\mathfrak{F}$ (of Proposition 3.19) for $\mathcal{S}(0,1) \subseteq \mathbb{R}^b$ and $r' := r(1+8\varepsilon)$. We then construct a filter family $\mathcal{G}^{\mathcal{P}}$ according to a modified version of the tensoring operation. We define a modified version of the tensoring operation because the projections in $\mathcal{P}$ do not necessarily map points on $\mathcal{S}(0,1)$ in $\mathbb{R}^B$ to points on $\mathcal{S}(0,1)$ in $\mathbb{R}^b$ (even after normalizing the projection map).

Fix $u \in \mathcal{S}(0,1)$, and define u_i' to be the projection of $P_i u$ onto the sphere $\mathcal{S}(0, \sqrt{b/B})$ in $\mathbb{R}^b$. We then define

$$\mathcal{G}_u^{\mathcal{P}}(u) := \mathcal{F}_u^1\left(\sqrt{\frac{B}{b}}u_1'\right) \times \cdots \times \mathcal{F}_u^{B/b}\left(\sqrt{\frac{B}{b}}u_{B/b}'\right)$$

if $\|u'_i - P_i u\|_2 \le \varepsilon$ for all $i \in [B/b]$ and define $\mathcal{G}_u^{\mathcal{P}}$ to be the empty set otherwise. We define $\mathcal{G}_q^{\mathcal{P}}(q)$ similarly for query points $q \in \mathcal{S}(0,1)$. The filter family $\mathcal{G}$ for $\mathcal{S}(0,1)$ that we want will be the union of the filter families $\mathcal{G}^{\mathcal{P}}$ over all $\mathcal{P} \in \mathfrak{P}$.

By the parameter setting of [ALRW17, Section 3.3.3], $G(r, \eta_u, \eta_q) \ge n^{-o(1)}$ when $cr \ge \sqrt{2}$. Thus, by Proposition 3.19, we can sample from $\mathfrak{F}$ and compute the filters containing a given point in time $O(n^{o(1)})$. The overhead of $\mathfrak{P}$ is also only $n^{o(1)}$. This gives us properties 1 and 2 for $\mathcal{G}$.

The "splitting" property of $\mathfrak{P}$ makes our filter family $\mathcal{G}$ Las Vegas locality-sensitive. For every pair $u, q \in \mathcal{S}(0,1)$ such that $\|u - q\|_2 \le r$, there exists some $\mathcal{P} \in \mathfrak{P}$ so that the three distances u, q, and $u - q$ are preserved (i.e., with distortion within $1 \pm \varepsilon$) for all projections $P \in \mathcal{P}$. Then $\mathcal{G}_u^{\mathcal{P}}(u)$ and $\mathcal{G}_q^{\mathcal{P}}(q)$ are non-empty. It follows from the Las Vegas property of filter families drawn from $\mathfrak{F}_0$ that u'_i and q'_i share a filter for each $i \in [B/b]$, since scaling $P_i u$ and $P_i q$ to u'_i and q'_i increases the distance between them by at most a factor of $1 + \varepsilon$. The accumulation of these distortions results in a distance between $\sqrt{B/b}u'_i$ and $\sqrt{B/b}q'_i$ that is at most r'. Hence $\mathcal{G}_u^{\mathcal{P}}(u) \cap \mathcal{G}_q^{\mathcal{P}}(q) \ne \emptyset$ by Proposition 3.19, which gives us property 3.

To bound the number of filters containing a point and the number of collisions, we use the fact that $\mathfrak{F}$ is a distribution over spherical LSF. Notice that for a database point u to belong to a spherical LSF filter characterized by the tuple $z \coloneqq (z_1, \ldots, z_{B/b})$, where the $z_i \in \mathbb{R}^b$ correspond to the centers of the filters in each of the B/b subspaces and z is thought of as belonging to $\mathbb{R}^B$, we must have

$$(1 \pm \varepsilon)\langle z, u\rangle = \sqrt{\frac{b}{B}}\left(\sum_{i=1}^{B/b}\left\langle z_i, \sqrt{\frac{B}{b}}u'_i\right\rangle\right) \ge \sqrt{\frac{B}{b}}\eta'_u = \eta_u.$$

Similarly, we would need $(1 \pm \varepsilon)\langle z, q\rangle \ge \eta_q$ for a query point q. To obtain average case behavior for the random variable $\langle z, q\rangle$, we can sample a random rotation of $\mathcal{S}(0,1)$ that we apply to all database and query points at the very beginning, with only $\mathrm{poly}(B)$ overhead. Then z is distributed as a spherical Gaussian, and we get by Lemma 3.17 that

$$\mathbb{P}\left((1+\varepsilon)\langle z, u\rangle \ge \eta_u\right) \le F((1-2\varepsilon)\eta_u) \le n^{o(1)/K}F(\eta_u)$$

and

$$\mathbb{P}\left((1+\varepsilon)\langle z, q\rangle \ge \eta_q\right) \le F((1-2\varepsilon)\eta_q) \le n^{o(1)/K}F(\eta_q).$$

Furthermore, if $\|q - x\|_2 > cr$, then by Lemma 3.18,

$$\begin{aligned}\mathbb{P}((1+\varepsilon)\langle z, u\rangle \ge \eta_u \text{ and } (1+\varepsilon)\langle z, q\rangle \ge \eta_q) &\le G(cr, (1-2\varepsilon)\eta_u, (1-2\varepsilon)\eta_q)\\ &\le n^{o(1)/K}G(cr, \eta_u, \eta_q).\end{aligned}$$

Since there are a total of

$$|\mathfrak{P}| \cdot \left(\frac{\mathrm{poly}(b)}{G(r, \eta'_u, \eta'_q)}\right)^{B/b} = \frac{\mathrm{poly}(B^{B/b})}{G(r, \eta_u, \eta_q)}$$

filters in a filter family sampled from $\mathfrak{G}$ and $\mathrm{poly}(B^{B/b}) = n^{o(1)/K}$, property 4 now follows. This completes our proof of Proposition 3.21.

3.7.4. Proofs of Theorems 3.3 and 3.16

Now to prove Theorem 3.16, note that the filter family that we constructed in Proposition 3.21 can be used as a drop-in replacement for the filter family used in the data-dependent data structure of [ALRW17]. By the parameter settings given in [ALRW17, Section 4.3], the filter family will always be instantiated with $cr \geq \sqrt{2}$ (our cr corresponds to r^*/R in [ALRW17]). We now show that the resulting data structure is Las Vegas. The only way randomness leads to false negatives in the data structure of [ALRW17] is through the filtering step, when two near neighbors fail to share a filter. However, this cannot happen with our LSF family by property 3 of Proposition 3.21. (Randomness is also used elsewhere, e.g., for the fast preprocessing step described in [AR15], but this randomness does not create false negatives.)

Finally, to reduce the dimension of the problem from d to B, we apply the same dimensionality reductions as in Section 3.6. To make this reduction step work, we need to slightly modify the data-dependent data structure of [ALRW17], since our reduction requires the option of continuing after encountering a false positive. In [ALRW17], at certain nodes of the decision tree, only one point is stored at a vertex as a "representative" for specific base cases of the recursion. We require the entire list of points mapped to that vertex to be stored, since it could be that the representative point is a false positive. This modification increases space usage negligibly. We finish by noting that the reduction of [Ngu14] for ℓ_p still applies, from which Theorem 3.3 follows.

Bibliography

[AC09] Nir Ailon and Bernard Chazelle. The Fast Johnson–Lindenstrauss Transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009.

[AGK06] Arvind Arasu, Venkatesh Ganti, and Raghav Kaushik. Efficient exact set-similarity joins. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, pages 918–929, 2006.

[Ahl17] Thomas Dybdahl Ahle. Optimal Las Vegas locality sensitive data structures. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 938–949, 2017.

[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, 2006.

[AI17] Alexandr Andoni and Piotr Indyk. Nearest neighbors in high-dimensional spaces. In Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry, Third Edition.*, chapter 43, pages 1135–1155. CRC Press, Boca Raton, FL, 2017.

[AIL+15] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya P. Razenshteyn, and Ludwig Schmidt. Practical and optimal LSH for angular distance. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1225–1233, 2015.

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

[AIR18] Alexandr Andoni, Piotr Indyk, and Ilya P. Razenshteyn. Approximate nearest neighbor search in high dimensions. *CoRR*, abs/1806.09823, 2018.

[AL13] Noga Alon and Shachar Lovett. Almost k-wise vs. k-wise independent permutations, and uniformity for general group actions. *Theory of Computing*, 9:559–577, 2013.

[ALRW17] Alexandr Andoni, Thijs Laarhoven, Ilya P. Razenshteyn, and Erik Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *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, 2017.

[AMS06] Noga Alon, Dana Moshkovitz, and Shmuel Safra. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms*, 2(2):153–177, 2006.

[AR15] Alexandr Andoni and Ilya P. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 793–801, 2015.

[AR16] Alexandr Andoni and Ilya P. Razenshteyn. Tight lower bounds for data-dependent locality-sensitive hashing. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, pages 9:1–9:11, 2016.

[AW15] Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 136–150. IEEE Computer Society, 2015.

[Azu67] Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Math. J. (2)*, 19(3):357–367, 1967.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 10–24, 2016.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[BGMZ97] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997.

[BKC+15] Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P Drake, Jane Landolin, and Adam M Phillippy. Assembling large genomes with single-molecule sequencing and locality sensitive hashing. *Nature biotechnology*, 33:623–630, 05 2015.

[CCF04] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

[CH67] Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory*, 13(1):21–27, 1967.

[Chr17] Tobias Christiani. A framework for similarity search with space-time tradeoffs using locality-sensitive filtering. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 31–46, 2017.

[CJN18] Michael B. Cohen, T. S. Jayram, and Jelani Nelson. Simple analyses of the Sparse Johnson-Lindenstrauss Transform. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, pages 15:1–15:9, 2018.

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

[CNW16] Michael B. Cohen, Jelani Nelson, and David P. Woodruff. Optimal approximate matrix product in terms of stable rank. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 11:1–11:14, 2016.

[CP17] Tobias Christiani and Rasmus Pagh. Set similarity search beyond MinHash. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1094–1107, 2017.

[DCLT] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

[DGJC06] Debojyoti Dutta, Rajarshi Guha, Peter C. Jurs, and Ting Chen. Scalable partitioning and exploration of chemical spaces using geometric hashing. *Journal of Chemical Information and Modeling*, 46(1):321–333, 2006.

[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, Brooklyn, New York, USA, June 8-11, 2004*, pages 253–262, 2004.

[GIM99] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 518–529, 1999.

[GPSS17] Mayank Goswami, Rasmus Pagh, Francesco Silvestri, and Johan Sivertsen. Distance sensitive Bloom filters without false negatives. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 257–269, 2017.

[HIM12] 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.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

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

[Ind00] Piotr Indyk. Dimensionality reduction techniques for proximity problems. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA.*, pages 371–378, 2000.

[JL84] William Johnson and Joram Lindenstrauss. Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 01 1984.

[KKKC16] Matti Karppa, Petteri Kaski, Jukka Kohonen, and Padraig Ó Catháin. Explicit correlation amplifiers for finding outlier correlations in deterministic subquadratic time. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 52:1–52:17, 2016.

[KN14] Daniel M. Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. *J. ACM*, 61(1):4:1–4:23, 2014.

[KOR00] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 30(2):457–474, 2000.

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

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

[Ngu14] Huy L. Nguyen. *Algorithms for High Dimensional Data.* PhD thesis, Princeton University, 2014.

[NSS95] Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 182–191, 1995.

[OWZ14] Ryan O'Donnell, Yi Wu, and Yuan Zhou. Optimal lower bounds for locality-sensitive hashing (except when q is tiny). *TOCT*, 6(1):5:1–5:13, 2014.

[Pag16] Rasmus Pagh. Locality-sensitive hashing without false negatives. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1–9, 2016.

[PP16] Ninh Pham and Rasmus Pagh. Scalability and total recall with fast CoveringLSH. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 1109–1118, 2016.

[Raz17] Ilya P. Razenshteyn. *High-dimensional similarity search and sketching: algorithms and hardness.* PhD thesis, Massachusetts Institute of Technology, Cambridge, USA, 2017.

[SDI06] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing).* The MIT Press, 2006.

[SW17] Piotr Sankowski and Piotr Wygocki. Approximate nearest neighbors search without false negatives for ℓ_2 for $c > \sqrt{\log \log n}$. In *28th International Symposium on Algorithms and Computation, ISAAC 2017, December 9-12, 2017, Phuket, Thailand*, pages 63:1–63:12, 2017.

[Wai19] Martin J. Wainwright. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint.* Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019.

[Wei19] Alexander Wei. Optimal las vegas approximate near neighbors in p. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1794–1813, 2019.

[Wyg17] Piotr Wygocki. Improved approximate near neighbor search without false negatives for ℓ_2. *CoRR*, abs/1709.10338, 2017.

Appendix

A. Las Vegas Locality-Sensitive Filter Families

In this appendix, we define "Las Vegas" data-independent filter families and show that they imply Monte Carlo families of filters as defined in [Chr17]. Combined with [Chr17, Theorem 1.5], this reduction implies analogs of the data-independent lower bounds of [OWZ14] hold for Las Vegas LSF families. It follows from these lower bounds that the exponents of Theorem 3.1 and Corollary 3.2 are optimal for LSF in the data-independent setting.

Let (X, D) be a metric space. We start with a Monte Carlo definition of LSF families as considered in [Chr17]:

Definition .1 ((Symmetric[1]) Monte Carlo filter families [Chr17])**.** A distribution $\mathscr{F}$ over filters is said to be *Monte Carlo* (r, cr, p_1, p_2, q)*-sensitive* if the following holds:

- For all $x, y \in X$ such that $D(x, y) \leq r$,
$$\mathop{\mathbb{P}}_{F \sim \mathscr{F}}(x \in F, y \in F) \geq p_1.$$
- For all $x, y \in X$ such that $D(x, y) > cr$,
$$\mathop{\mathbb{P}}_{F \sim \mathscr{F}}(x \in F, y \in F) \leq p_2.$$
- For all $x \in X$, $\mathbb{P}_{F \sim \mathscr{F}}(x \in F) \leq q$.

To obtain Las Vegas algorithms using LSF, we need a stronger notion of filter family, which we define as follows:

Definition .2 (Las Vegas filter families)**.** A distribution $\mathfrak{F}$ over filter families is said to be *Las Vegas* (r, cr, p_2, q, N)*-sensitive* if the following holds:

- For all $x, y \in X$ such that $D(x, y) \leq r$,
$$\mathop{\mathbb{P}}_{\mathcal{F} \sim \mathfrak{F}}(\mathcal{F}(x) \cap \mathcal{F}(y) \neq \emptyset) = 1.$$
- For all $x, y \in X$ such that $D(x, y) > cr$,
$$\mathop{\mathbb{E}}_{\mathcal{F} \sim \mathfrak{F}}(|\mathcal{F}(x) \cap \mathcal{F}(y)|) \leq p_2.$$

[1] In [Chr17], asymmetric filter families yielding space-time tradeoffs for LSF are also discussed. These filter families have separate "query" and "update" filters. However, since this section focuses only on the symmetric setting, we merge the (asymmetric) LSF parameters p_q and p_u of [Chr17] into the single (symmetric) parameter q.

- For all $x \in X$, $\mathbb{E}_{\mathcal{F}\sim\mathfrak{F}}(|\mathcal{F}(x)|) \leq q$.
- The number of filters in $\mathcal{F}$ sampled from $\mathfrak{F}$ is always N.

This definition is general in the sense that for a set of filters to be usable in a Las Vegas setting, it must guarantee that every pair of "close" points is contained in a filter. Furthermore, p_2 and q are analogous to those of Definition .1, and our lower bound is independent of the size N.

With this definition of LSF in a Las Vegas setting, the remaining proofs are straightforward: We first show a general reduction from Las Vegas LSF to Monte Carlo LSF, and then, for the specific case of ℓ_p spaces, we show that the lower bound of [Chr17] and [OWZ14] still holds.

Lemma .1. *Suppose a distribution $\mathfrak{F}$ over filter families is Las Vegas (r, cr, p_2, q, N)-sensitive, and define $p_1' \coloneqq 1/N$, $p_2' \coloneqq p_2/N$, and $q' \coloneqq q/N$. Then there exists a distribution $\mathscr{F}$ over filters that is Monte Carlo (r, cr, p_1', p_2', q')-sensitive.*

Proof. Let $\mathscr{F}$ be the distribution over filters where we first sample $\mathcal{F} \sim \mathfrak{F}$ and then sample a filter uniformly at random from $\mathcal{F}$. One can check that $\mathscr{F}$ is Monte Carlo (r, cr, p_1', p_2', q')-sensitive. □

Proposition .2. *Suppose $0 < p < \infty$. Every Las Vegas (r, cr, p_2, q, N)-sensitive filter family $\mathfrak{F}$ for $\mathbb{R}^d$ under the ℓ_p norm must satisfy*

$$\rho \coloneqq \frac{\log(q)}{\log(q/p_2)} \geq \frac{1}{c^p} - o_d(1)$$

when $p_2/q \geq 2^{-o(d)}$. (In particular, this lower bound depends only on p_2 and q and not on N.)

Proof. By the reduction of Lemma .1 and [Chr17, Theorem 1.4], we have that

$$\rho = \frac{\log(q'/p_1')}{\log(q'/p_2')} \geq \frac{1}{c^p} - o_d(1).$$

□