High-Dimensional Similarity Search and Sketching: Algorithms and Hardness

by

Ilya Razenshteyn

B.S., Lomonosov Moscow State University (2012)
S.M., Massachusetts Institute of Technology (2014)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2017

Author .....
Department of Electrical Engineering and Computer Science
August 31, 2017

Certified by .....
Piotr Indyk
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

High-Dimensional Similarity Search and Sketching:

Algorithms and Hardness

by

Ilya Razenshteyn

Submitted to the Department of Electrical Engineering and Computer Science
on August 31, 2017, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

We study two fundamental problems that involve massive high-dimensional datasets: approximate near neighbor search (ANN) and sketching. We obtain a number of new results including:

- An algorithm for the ANN problem over the ℓ_1 and ℓ_2 distances that, for the first time, improves upon the Locality-Sensitive Hashing (LSH) framework. The key new insight is to use random space partitions that *depend on the dataset.*
- An implementation of the core component of the above algorithm, which is released as FALCONN: a new C++ library for high-dimensional similarity search.
- An efficient algorithm for the ANN problem over *any* distance that can be expressed as a *symmetric norm.*
- For norms, we establish the equivalence between the existence of short and accurate sketches and good embeddings into ℓ_p spaces for $0 < p \leq 2$. We use this equivalence to show the first sketching lower bound for the Earth Mover's Distance (EMD).

Thesis Supervisor: Piotr Indyk
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

Throughout my life I have been extremely lucky to be surrounded with incredible people: mentors, teachers, colleagues, classmates, friends and family. Without your support and investment in me I would not be where I am now.

Let me start with my Ph. D. advisor Piotr Indyk. Being Piotr's student for the past five years has been both an honor and a pleasure. In addition to being a brilliant researcher, Piotr has all the other qualities that a great advisor needs: a warm and enthusiastic personality, a great sense of care for all of his students, extremely high ethical standards when it comes to collaborations and publishing papers, and a perfect balance between being hands-on and giving enough freedom. I would also like to point out Piotr's talent of finding "just right" problems for each particular student (with an amazing precision!), and a strong much needed support during my academic job search. Last but not least, as a well-established professor, Piotr has built a developed ecosystem of current and former students and postdocs, with many of whom I had a privilege to work with...

...which brings me to the second person I would like to thank: Alex Andoni, a former student of Piotr. Alex has effectively been my informal advisor. Both scientifically: for instance, all the papers this thesis is based on are co-authored with Alex, and more generally: he has given me lots of academic and life advice. Alex's cheerful personality made our five-year long (extremely fruitful) collaboration very enjoyable, even during harder times when not all the problems were being solved to our satisfaction. In fact, I met Alex for the first time even before Piotr: back in Summer 2011 at Microsoft Research, where I was an intern. Our short conversation and Alex's (unintended?) pitch for MIT and Piotr were decisive later when I needed to choose a school and an advisor for my Ph. D.

My fascination with Theory started during my undergraduate studies at Moscow State University, and the two people responsible for this are Maxim Babenko and Sasha Shen, who were kind enough to advise a clueless and naive but eager undergrad. Both Maxim and Sasha served as great role models both in academic and non-academic

matters. Among many other things, I would like to thank Maxim for providing me with a part-time job at Yandex, when I needed money, and Sasha for teaching me (by many examples) how to give good talks and lectures and write papers well. Uncharacteristically for Russian researchers, both Maxim and Sasha encouraged me to go abroad for a graduate school, which in retrospect was essentially the only viable option.

Let me turn to my summer internship mentors: Andrew Goldberg, Renato Werneck, Vahab Mirrokni, Silvio Lattanzi, Alex Andoni, David Woodruff, and Konstantin Makarychev. Between 2010 and 2016 I spent only one summer outside of a research lab, and all of the remaining summers were formative for me as a researcher. I would like to thank Andrew and Renato for strongly influencing my research taste[1], and Andrew for all of the support throughout the years. I thank Vahab and Silvio for expanding my research horizons. I interned with Alex and Konstantin at Microsoft Research, and both of these summers were perhaps the most productive time in my life so far. Finally, I thank David for being a very dedicated mentor.

Summer internships are not as much fun without fellow interns. I would like to thank Andreea, Avi, Cyrus, Daniel, Dimitris, Fan, Gopi, Irina, Jana, Konstantina, Melanie, Mika, Mohsen, Nina, Thomas, Tomas, Li-Yang, Rad, Reza, Sadra, Sahil, Tom, Tselil, and Zhao for great time we spent together. On a loosely related note, I would like to thank Clement, Dimitris and Erik for a great social atmosphere in a place, which I am not allowed to name here.

I would like to thank all my co-authors and collaborators, past and present: Thomas Ahle, Zeyuan Allen-Zhu, Alex Andoni, Misha Andreev, Maxim Babenko, Arturs Backurs, Daniel Delling, Daniel Fleischman, Rati Gelashvili, Andrew Goldberg, Alexey Gusakov, Piotr Indyk, Michael Kapralov, Ignat Kolesnichenko, Robi Krauthgamer, Thijs Laarhoven, Silvio Lattanzi, Stefano Leonardi, Sepideh Mahabadi, Konstantin Makarychev, Yury Makarychev, Vahab Mirrokni, Huy Nguyen, Sasho Nikolov, Rasmus Pagh, Eric Price, Ruslan Savchenko, Ludwig Schmidt, Negev Shekel Nosatzki, Sasha Shen, Francesco Silvestri, Zhao Song, Ameya Velingker, Tal Wagner, Erik Waingarten,

[1] In particular with their passion for algorithms implementations.

Renato Werneck, and David Woodruff. I have learnt a lot from each of you, and all the collaborations were not only productive, but also very enjoyable.

MIT and especially the Theory group of CSAIL have become my second home for the past five years. I would like to thank fellow graduate students who made this place particularly cozy for me: Adrian, Arturs, Badih, Fereshte, G, Govind, Henry, Madalina, Nicholas, Nishanth, Rati, Sepideh, Tal, and Zeyuan. No matter what format our interactions took—going for hikes, walks, runs or drinks, organizing parties at conferences, participating in ICPC contests or writing papers together—it was all great fun. I thank Henry for hosting me during the EECS visit day back in 2012 and showing me what a wonderful place MIT is.

I would like to thank my friends back from Russia: Alexey, Boris, Ilya, Maxim, Polina, Sasha and Sasha. You are all over the world, and I may not be meeting you that often, but whenever we meet, I have this feeling that we never really parted.

I thank my sister Eleonora and my parents Lyudmila and Petr for believing in me. Without your love, support and passion for education and science, nothing would be possible.

My wife Oksana has been the main person in my life and the best friend for the past eight years. Thank you for everything!

The person who introduced the world of computer science and programming to me is Vladimir Denisovich Lelyukh, my school informatics teacher. It all started when my parents brought me, a ten-year old, to his class almost seventeen years ago. Vladimir Denisovich was an amazing teacher, a charismatic mentor, and a good friend to all of his many students. He passed away four years ago, but I still can't fully believe it happened. I dedicate this thesis to him.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

In this thesis, we show new algorithms and hardness results for fundamental problems that involve massive high-dimensional data processing. Such datasets are pervasive in many applications. Let us list just a few immediate examples.

- **Feature vectors** A popular approach in data analysis is to map a dataset into a *feature vector space,* and reduce a given task to a geometric computational problem. The simplest example of this sort is the *bag of words* representation: one maps a text document into a vector of word counts (see Figure 1-1). In many applications, both the number of data items and features is huge. For instance, Twitter [1] receives around *200 billion* tweets per year. Thus, if we wanted to analyze them using the bag of words approach, we would need to process 200B vectors with hundreds of thousands of dimensions, since the number of coordinates is equal to the number of distinct words.

- **Signals** Typically, signals of various nature (sound, images, etc.) can be modeled by real or complex vectors after appropriate sampling/discretization. As in the previous example, one often needs to deal with extremely high-dimensional vectors. For instance, if we sample audio at 44.1 kHz, one minute corresponds to a vector with more than *two million* dimensions.

1	everything
0	excluding
1	in
1	including
2	moderation
0	out

Figure 1-1: The bag of words representation of the sentence "Everything in moderation, including moderation."

- **Data streams** When processing a long stream of data items that one can not even store, thinking of the stream as a sequence of updates to a high-dimensional vector is often useful. For instance, if one needs to process a stream of network packets and detect potential DDoS attacks, one can imagine a vector indexed by all possible 2^{32} IP addresses[1]. Each packet is interpreted as an update to this vector ("increase the corresponding coordinate by 1"), and the goal is to keep track of its largest coordinates, which correspond to high-traffic source/destination addresses.

These and other applications motivate the field of *high-dimensional computational geometry*, which this thesis contributes to. It should be contrasted with the classical, "low-dimensional" computational geometry, where most of the efficient algorithms have running time *exponential* in the dimension, thus making their applicability to the high-dimensional regime limited.

1.1.1 Overview of the problems

This thesis consists of two parts. Each part is devoted to a separate problem, which we briefly and informally introduce below.

- **Near neighbor search** (see Figure 1-2) Given a dataset of n points (vectors) in a d-dimensional space $\mathbb{R}^d$, we would like to preprocess it and build a data structure for answering *nearest neighbor queries.* Each query is a d-dimensional point, and the goal is to return the data point closest to it. In a simpler version of

[1] for IPv4, or 2^{128} for IPv6

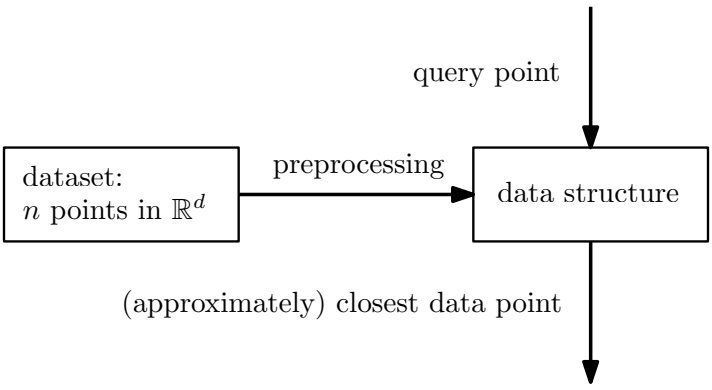


Figure 1-2: Near neighbor search

the problem, we allow answers to be approximate. The parameters we are trying to optimize are: space used by a data structure, query time, and preprocessing time.

A naïve solution is the linear scan: omit the preprocessing stage altogether, and, whenever given a query, try all the data points and choose the closest one. However, this is too slow for large datasets, and one typically uses the preprocessing stage to compute and store auxiliary information which allows to speed up the query procedure.

Perhaps the most "obvious" application of near neighbor search is similarity search over various kinds of data, which corresponds to nearest neighbor queries in the feature space [162].

- **Sketching (succinct summarization)** (see Figure 1-3) Sketching involves (lossy) compression of a massive object into a short summary which can be used as a proxy in further computations. Sketching has been used for similarity search [85, 108, 171, 172], fast algorithms for numerical linear algebra [178, 154], processing data streams [136, 5], and speeding up other algorithms [22, 99].

  In this thesis, we study sketching of high-dimensional vectors with the goal of estimating a given *distance function* between two vectors from their (succinct) summaries.

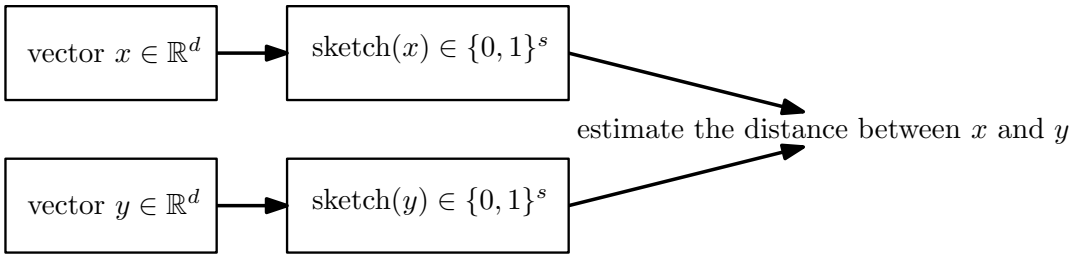


Figure 1-3: Sketching high-dimensional vectors; here $s \ll d$

Superficially, the common theme among these problems is working with high-dimensional vectors/datasets. However, the relation is more fundamental. First of all, the problems are closely related on a technical level: sketching can be used for near neighbor search. Second, both of these problems build on a coherent toolbox of mathematical/algorithmic primitives, which can be called "efficient representations of data": randomized hashing, dimension reduction, metric embeddings, etc. Hence, this thesis can be seen as a step towards building a unified theory of such efficient representations.

1.1.2 Main contributions: a bird's eye view

Here we briefly describe the main contributions of the thesis. A more detailed description can be found in the further sections of the introduction.

Near neighbor search We present a number of new algorithms for the near neighbor search problem. Most notably, our algorithms provide the *first* improvement over Locality-Sensitive Hashing (LSH), a popular framework in theory and applications. We implement a core component of one of the new algorithms and release it as a part of FALCONN: a new open-source C++ library for similarity search over high-dimensional data. In addition to algorithms, we show several *impossibility results*, which we hope will guide further algorithmic research on the problem.

This part is based on the following papers:

- Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, Ludwig Schmidt, *Practical and Optimal LSH for Angular Distance,* appeared at NIPS

2015 [17],

- Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, Erik Waingarten, *Optimal Hashing-Based Time–Space Trade-offs for Approximate Near Neighbors,* appeared at SODA 2017 [24], invited to the special issue of ACM Transactions on Algorithms; this result heavily builds on:
  - Alexandr Andoni, Piotr Indyk, Huy Nguyen, Ilya Razenshteyn, *Beyond Locality-Sensitive Hashing,* appeared at SODA 2014 [18],
  - Alexandr Andoni, Ilya Razenshteyn, *Optimal Data-Dependent Hashing for Approximate Near Neighbors,* appeared at STOC 2015 [26],
- Alexandr Andoni, Huy Nguyen, Aleksandar Nikolov, Ilya Razenshteyn, Erik Waingarten, *Approximate Near Neighbors for General Symmetric Norms,* appeared at STOC 2017 [25].

Sketching Various distance functions (metrics) behave very differently when it comes to sketching: for some distances there are extremely efficient compression techniques (such as the celebrated Johnson–Lindentrauss lemma), for other distances essentially nothing is possible. For a large class of distances, we characterize completely when efficient sketching is possible. As a byproduct of this characterization, we obtain lower bounds on sketches for several distances, including the Earth Mover's Distance (EMD). This part of the thesis is based on the paper:

- Alexandr Andoni, Robert Krauthgamer, Ilya Razenshteyn, *Sketching and Embedding are Equivalent for Norms,* appeared at STOC 2015 [23], accepted to the special issue of SIAM Journal on Computing.

1.2 Near neighbor search (NNS)

The Near Neighbor Search problem (NNS) is defined as follows. Given a dataset of n points, preprocess it to prepare for answering queries of the following kind: given

a query point, find a data point within distance r from the query provided that it exists. Here r is a parameter that is known from the beginning and is the same for all queries.

One can define the NNS problem with respect to *any* distance function. However, we will mostly consider the case when the dataset and queries lie in a d-dimensional vector space $\mathbb{R}^d$ equipped with the ℓ_1 or ℓ_2 distances[2]. A useful special case of the ℓ_1 scenario is when the dataset and queries lie in the hypercube $\{0,1\}^d$. This corresponds to the Hamming distance[3]. An important special case of the ℓ_2 setting is when data points and queries lie on a unit sphere $S^{d-1} \subset \mathbb{R}^d$: this is equivalent to the near neighbor search with respect to the *cosine similarity*.

The NNS problem has many applications. Perhaps the most "obvious" one is similarity search for various types of data: text documents, images, audio files, proteins, etc. A typical approach is to take a dataset and map it into $\mathbb{R}^d$ by computing a certain feature representation [162]. This step typically requires significant domain expertise and often utilizes techniques from machine learning and/or optimization. But after the feature vectors are computed, the similarity search directly corresponds to executing the NNS queries in the feature space. A sample of other applications includes faster algorithms for cryptanalysis [109], optimization [67, 79, 8], and training neural networks [165], to name a few.

The simplest NNS algorithm is the linear scan: omit the preprocessing stage, and whenever given a query, try all the data points and choose the closest one. However, linear scan ends up being too slow for large datasets and we will be looking for algorithms that have query time *sublinear* in the number of points n. In order for this to be possible, we are going to utilize the preprocessing stage to compute some auxiliary information that will be useful during the query stage.

Three most important parameters of an NNS data structure are:

- *Space* the data structure occupies. Storing the dataset itself requires $O(dn)$ space,

[2]The ℓ_1 distance is defined as $\|x-y\|_1 = \sum_{i=1}^{d} |x_i - y_i|$, while the ℓ_2 distance is $\sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$.

[3]The Hamming distance between two binary vectors is defined as the number of positions where they differ.

but auxiliary information computed during the preprocessing takes additional space to store;

- *Time* it takes to answer a query. Ideally, processing a query should take time $O(d)$, but usually the query time depends on n. In any case, the query time must be strongly sublinear in n (say, at most $n^{0.99}$);

- Time it takes to *build* a data structure. Clearly, it cannot be smaller than the amount of space used, but typically the preprocessing time can be made to be *near-linear* in the space.

The NNS problem exhibits the so-called "curse of dimensionality". Namely, all the known data structures with query time strongly sublinear in n require space $n^{\Omega(d)}$ (see, e.g., [59, 125]), which is super-polynomial as soon as d is super-constant. Moreover, modulo a plausible hardness assumption, there is no data structure for NNS with polynomial in n and d preprocessing time, and query time that is polynomial in d and strongly sublinear in n [175][4]. To overcome the curse of dimensionality, we will relax the NNS problem somewhat and then observe that the techniques we develop for the relaxed version can be used to solve the exact NNS problem on many real-world datasets.

1.2.1 Approximate near neighbor search (ANN)

The relaxed version is called the Approximate Near Neighbor Search problem (ANN) and is defined similarly to the NNS. However, whenever we receive a query with a data point within distance r from it, we are now allowed to return any data point within distance cr from the query (see Figure 1-4). The new parameter $c > 1$ is the approximation we are willing to tolerate. Allowing answers to be approximate overcomes the curse of dimensionality and enables very efficient data structures with polynomial dependence on the dimension d, space polynomial in n and query time sublinear in n (see [77] for a survey of the state of affairs back in 2012).

[4]Technically, to get this hardness result, we need to modify the reduction from [175] slightly. A variant of this modification can be found in [4], but even before it had been a folklore result.

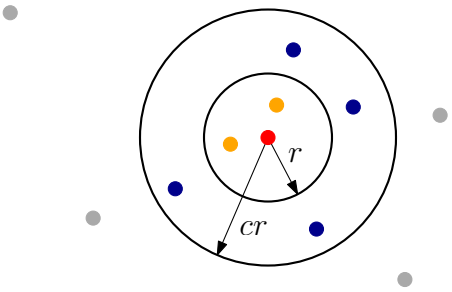


Figure 1-4: Illustration for the definition of the ANN problem. The red point is a query. We are guaranteed that the smaller ball is non-empty, but are satisfied with any point from the larger ball.

Thus, for the ANN problem we will be studying the trade-off between approximation c, space, query time and preprocessing time. We would hope to achieve for every $c > 1$:

- Space polynomial (ideally, near-linear) in n and d;
- Query time polynomial (ideally, near-linear) in d and strongly sublinear in n;
- Preprocessing time near-linear in the space used.

How good are approximate answers in practice? It turns out that for many real-world datasets, approximation is not really necessary and one can quickly find *exact* near neighbors. The reason is that oftentimes the following "gap phenomenon" holds: most of the data points are much further from the query than the nearest neighbor[5] (see Figure 1-5 for an example). Under this gap assumption, most of the ANN algorithms (including the ones described in this thesis) can be shown to return exact nearest neighbors in time sublinear in n.

For the ANN problem over ℓ_1 and ℓ_2 distances, we can assume that[6] $d = \widetilde{O}(\log n)$. Indeed, for the ℓ_2 distance we can apply a random projection onto $\widetilde{O}(\log n)$ dimensions: by the Johnson–Lindenstrauss lemma [93, 65], this procedure incurs distortion of

[5] One could go even further and argue that if there is no such a gap, then the ANN instance is not very interesting to start with, since the nearest neighbors are not distinguishable from the rest of the points.

[6] From now on, $\widetilde{O}(f)$ is defined as $O(f \log^{O(1)} f)$.

distances by a multiplicative factor $1 + o(1)$. For the ℓ_1 distance the situation is a bit more delicate, since the dimension reduction as above is known to be impossible [46]. However, since we care only about a *single* distance scale ("r vs. cr"), there are known procedures for the dimension reduction for this relaxed setting [108]. From now on, we will be implicitly assuming that $d = \widetilde{O}(\log n)$ unless stated otherwise.

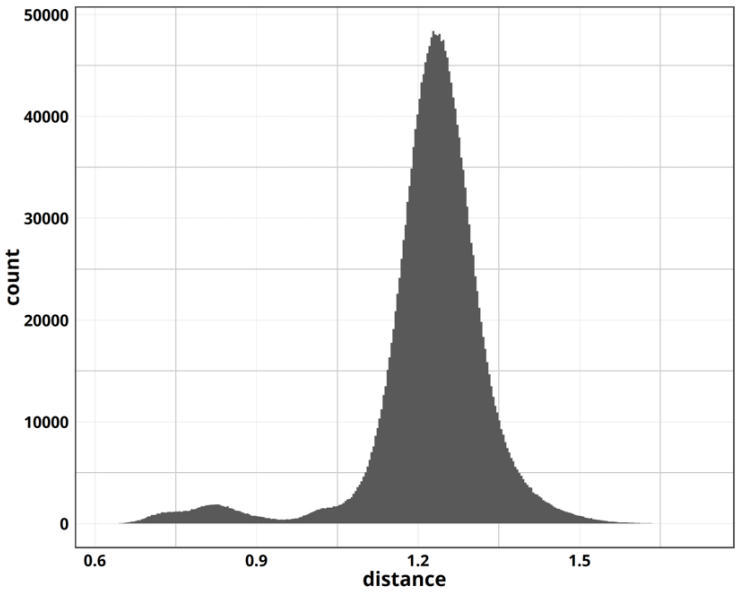


Figure 1-5: A distribution of distances in a dataset of GloVe word embeddings [148]; a random data point used as a query.

1.2.2 Locality-sensitive hashing (LSH)

One of the main techniques for the high-dimensional ANN problem is locality-sensitive hashing (LSH) [85, 77]. The idea is to design and use a random partition $\mathcal{P}$ of the ambient space $\mathbb{R}^d$ with the following properties[7]. For every $x, y \in \mathbb{R}^d$ the following conditions are satisfied:

- If the distance between x and y is at most r, then

$$\Pr_{\mathcal{P}}[x \text{ and } y \text{ end up in the same part of } \mathcal{P}] \geq p_1; \tag{1.1}$$

[7] There is a one-to-one correspondence between partitions and hash functions $h\colon \mathbb{R}^d \to \mathbb{Z}$ if we only care about collisions. We will be using both "languages" interchangeably.

- If the distance between x and y is more than cr, then

$$\Pr_{\mathcal{P}}[x \text{ and } y \text{ end up in the same part of } \mathcal{P}] \leq p_2. \tag{1.2}$$

Here p_1 and p_2 are two parameters that characterize the quality of an LSH partition. Of course, we must have $p_1 > p_2$, and the bigger the gap between p_1 and p_2 is, the higher quality a partition is of.

More precisely, as has been shown in [85], the existence of an "efficient"[8] partition $\mathcal{P}$ with parameters p_1 and p_2 implies a data structure for ANN with preprocessing time and space $n^{1+\rho+o(1)}/p_1$ and query time $n^{\rho+o(1)}/p_1$, where

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

Thus, the bigger the gap between p_1 and p_2 is, the smaller the value of ρ is, and the better bounds on the data structure parameters one is getting. Typically, $\rho = \rho(c)$ is equal to 1 for $c = 1$ and $\rho(c) \to 0$ as $c \to \infty$.

Having the above reduction in mind, we can try to optimize the value of ρ for an LSH partition. Of course, the optimal value crucially depends on the underlying distance metric, and, as it turns out, the best partitions for the ℓ_1 and ℓ_2 distances are known. Namely, for ℓ_1 the original LSH paper [85] gives $\rho \leq \frac{1}{c} + o(1)$, while for the ℓ_2 distance a longer line of work [85, 66, 13] leads to the value $\rho \leq \frac{1}{c^2} + o(1)$. Translating these bounds into query times, we can get, for instance, query time $n^{1/2+o(1)}$ for ℓ_1 and $n^{1/4+o(1)}$ for ℓ_2 for approximation $c = 2$. Moreover, the above values of ρ are *optimal* for both ℓ_1 and ℓ_2 [129, 140]. Thus, we understand LSH almost completely. However:

- A data structure for ANN does not have to fit the LSH framework. Can we get an ANN data structure that bypasses the limitations of LSH?

- For many important metrics beyond ℓ_1 and ℓ_2, LSH with non-trivial guarantees does not exist; a good example is the ℓ_∞ distance[9] [81]. How tractable the ANN

[8] Given a point, we can quickly find a part of $\mathcal{P}$, which it belongs to.
[9] Defined as $\|x - y\|_\infty = \max_{i=1}^{d} |x_i - y_i|$.

problem is for such metrics?

- The optimal LSH for ℓ_2 from [13], which gives the exponent $\rho \leq \frac{1}{c^2} + o(1)$, is highly impractical. Is there a practical analog of it?

In this thesis we make significant progress on all of the above questions.

1.2.3 Data-dependent LSH

In [18, 26] we show new data structures for high-dimensional ANN over ℓ_1 and ℓ_2 that overcome the above LSH barrier. The new space and time bounds are still of the form $n^{1+\rho+o(1)}$ and $n^{\rho+o(1)}$, but the new values of ρ are:

$$\rho \leq \frac{1}{2c-1} + o(1) \tag{1.3}$$

for the ℓ_1 distance, and

$$\rho \leq \frac{1}{2c^2-1} + o(1) \tag{1.4}$$

for ℓ_2. For instance, for approximation $c = 2$, the new data structures give query time $n^{1/3+o(1)}$ for ℓ_1 and $n^{1/7+o(1)}$ for ℓ_2. At the same time, we now require less memory than the best possible LSH-based approaches.

The main idea behind these results is to allow (random) partitions to *depend on the dataset.* It turns out that we do not need the full power of conditions (1.1) and (1.2) for *every* pair of points $x, y \in \mathbb{R}^d$ from the ambient d-dimensional space that the universal LSH constructions provide. Instead, we can try to design a random partition that works well for a *given* dataset of n points we know in advance.

At a high level, the new algorithms consist of two parts. First, we design and analyze *Voronoi LSH:* a new LSH family that yields improved exponents (1.3) and (1.4) under the assumption that a dataset is *random*[10]. In Voronoi LSH, one samples a number of random landmark points and hashes a point to the closest landmark. Second, to handle the general case, we show how to reduce a worst-case dataset to several pieces

[10]In case of ℓ_1, it is assumed to be a random subset of $\{0,1\}^d$, while for ℓ_2, it must be a random subset of a unit sphere $S^{d-1} \subset \mathbb{R}^d$, in both cases queries are generated so that there is one data point that is c times closer than all the other ones.

that look random enough so that Voronoi LSH works well on them. This is done by removing dense clusters of points iteratively and handling them separately. The overall algorithm can be seen as a random collection of $n^{\rho+o(1)}$ decision trees; during the query stage, we query all of them.

Let us make two remarks. First, data-dependent hashing (partitioning) has been ubiquitous in practice (see, e.g., [171, 172]), but, to the best of our knowledge, all the known constructions implicitly or explicitly exploit some nice structure hidden in a dataset. In contrast to these, our algorithms give a provable improvement upon LSH for *worst-case* datasets. Second, by being data-dependent, we seemingly sacrifice one important property of LSH: it supports adding and removing data points for free. However, we can utilize a generic data structure technique of dynamization [142] and support insertions/deletions of data points in time $n^{\rho+o(1)}$.

In a follow-up work [27] we show *optimality* of the above bounds (1.3) and (1.4) for any data structure based on data-dependent LSH. We need to be careful which partitions we allow and which we do not. After all, for every dataset its *Voronoi diagram* gives an "unreasonably high-quality" partition, but it is not very useful algorithmically, since point location for the Voronoi diagram is equivalent to an exact NNS query. Ideally, we would hope to prove a lower bound against partitions that are algorithmically efficient. However, this would require showing data structure lower bounds that are way beyond reach of the currently known techniques. Instead, we require partitions to have strongly sublinear in n *description complexity*. Indeed, it is the case for all the known algorithms, including [18, 26], and overcoming this barrier is a notorious open problem[11]. The hard distribution used to prove the lower bound is simply a random instance as introduced above.

However, in this thesis we do not describe the above new results, since after publishing the results [18, 26, 27], we significantly extended the data-dependent LSH framework. We introduce this extension next.

[11] In all the known constructions of high-quality partitions, the point location time is proportional to space occupied by a partition.

1.2.4 Time–space trade-offs for ANN

Data-independent and data-dependent LSH both provide a very specific relation between space and query time: one gets space $n^{1+\rho+o(1)}$ and query time $n^{\rho+o(1)}$, where ρ is a function of the approximation c. However, one can ask if it is possible to trade space for query time and vice versa. For example, in practice space is often a scarcer resource than time, and one may require the space used by a data structure to be $n^{1+o(1)}$. Can we still get sublinear query time in this space-constrained regime?

Researchers have studied various time–space trade-offs for ANN [85, 108, 80, 144, 13, 97]. The culmination of the prior work has been the paper [97] by Kapralov, which shows how to smoothly interpolate between: the space-constrained regime (space $n^{1+o(1)}$, sublinear query time), the balanced regime (space $n^{1+\rho+o(1)}$, query time $n^{\rho+o(1)}$ as for the LSH framework), for which it essentially matches the best data-independent LSH constructions (from [85] and [13]), and the fast queries regime (query time $n^{o(1)}$, polynomial space).

In Chapter 2, which is based on the paper [24], we describe how to extend the data-dependent LSH framework and show how to get an improved time–space trade-off for ANN which:

- Interpolates between the near-linear space regime, the balanced regime, and the fast queries regime;
- In the balanced regime (space $n^{1+\rho+o(1)}$, query time $n^{\rho+o(1)}$), matches the best data-dependent LSH data structures (achieving (1.3) and (1.4) for ℓ_1 and ℓ_2, respectively);
- In all the other regimes, significantly improves upon all the previous work, including the result from [97].

More precisely, we show how to get space $n^{1+\rho_u+o(1)}$ and query time $n^{\rho_q+o(1)}$ for every $\rho_u, \rho_q \geq 0$ such that:

$$c\sqrt{\rho_q} + (c-1)\sqrt{\rho_u} = \sqrt{2c-1} \tag{1.5}$$

for ℓ_1 and

$$c^2\sqrt{\rho_q} + (c^2 - 1)\sqrt{\rho_u} = \sqrt{2c^2 - 1} \tag{1.6}$$

for ℓ_2.

For example, for approximation $c = 2$ and the ℓ_1 distance, we can interpolate between:

- Space $n^{1+o(1)}$, query time $n^{3/4+o(1)}$;
- Space $n^{4/3+o(1)}$, query time $n^{1/3+o(1)}$;
- Space $n^{4+o(1)}$, query time $n^{o(1)}$.

Similarly, for the ℓ_2 distance, we can get:

- Space $n^{1+o(1)}$, query time $n^{7/16+o(1)} = n^{0.4375+o(1)}$;
- Space $n^{8/7+o(1)}$, query time $n^{1/7+o(1)}$;
- Space $n^{16/9+o(1)} = n^{1.77\ldots}$, query time $n^{o(1)}$.

(See Figure 1-6 for the illustration).

In Chapter 6 (also based on [24]), we show that the time–space trade-offs (1.5) and (1.6) are optimal in two settings:

- When a data structure fits a certain general "hashing framework", which all the known data structures fit;
- If a data structure is arbitrary, but is allowed to inspect at most *two* memory locations during a query stage;

In both cases we use random instances mentioned in Section 1.2.3 as a hard distribution. The latter lower bound shows and uses an interesting connection between ANN data structures and locally-decodable codes (LDC) (see, e.g., [179] for a survey). Namely, we show that if there had been a too-good-to-be-true ANN data structure that touches t memory locations, we would have gotten a too-good-to-be-true LDC that performs t queries. Such a reduction allows us to utilize a strong lower-bound

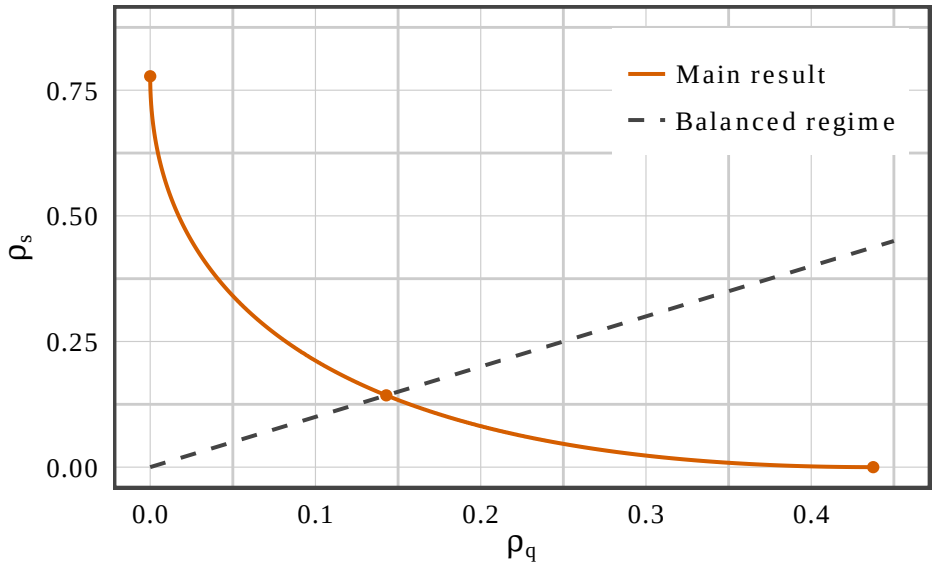


Figure 1-6: Illustration of the new time–space trade-off for the ℓ_2 distance and approximation $c = 2$. We can interpolate between fast queries ($\rho_q = 0$, $\rho_s = 7/9 = 0.77\ldots$), the balanced regime ($\rho_q = \rho_s = 1/7 = 0.14\ldots$) , and the near-linear space regime ($\rho_q = 7/16 = 0.43\ldots$, $\rho_s = 0$).

for two-query LDCs [101], which gives the desired space lower bound for the ANN problem. However, this approach breaks down completely for $t \geq 3$, since there are strong LDC constructions that make three or more queries. In light of this transition, one can not help but ask if LDCs might be useful for obtaining *upper bounds* that would bypass the abovementioned lower bounds for the "hashing framework", where the bounds (1.5) and (1.6) are tight.

1.2.5 FALCONN: practical and optimal LSH for unit sphere

The next natural question is how practical the new algorithms are. After all, data-dependent partitioning is very popular in practice, so it would be natural to try to eliminate the gap between the theory and the practice completely. Unfortunately, in full generality the new algorithms are impractical, especially the reduction from worst-case datasets to "random looking" pieces[12].

However, at the core of the above algorithms there is a simple construction of

[12]Though, see [28], where we make a first step in making such a reduction practical for the Hamming space.

Voronoi LSH, which is a *data-independent* LSH partition that gives an improved exponent ρ for random instances. We notice that in order for Voronoi LSH to work well, a dataset does not have to be *completely* random. Rather, it is enough that the dataset lies on a sphere and is not too concentrated in any small region of it. As it turns out, the latter is a reasonable assumption for many real-world datasets, thus it makes sense to investigate if Voronoi LSH is practical.

Voronoi LSH can also be seen as a (theoretical) improvement upon the celebrated and widely-used Hyperplane LSH [55]. Moreover, Voronoi LSH can be shown to give guarantees similar to the optimal LSH construction for ℓ_2 from [13] for worst-case data in $\mathbb{R}^d$. Thus, the question of practicality of Voronoi LSH becomes especially pressing.

But, as it turns out, even Voronoi LSH by itself is not practical due to prohibitively high hashing time. In Chapter 3, which is based on the paper [17], we show how to modify it to make it practical, while essentially preserving the theoretical guarantees.

The new construction can be seen as a combination of three ingredients:

- Cross-Polytope LSH from [168], where it was proposed but not analyzed. We show that the Cross-Polytope LSH, which can be seen as a *structured version* of the Voronoi LSH, gives the same theoretical guarantees as the vanilla Voronoi LSH.

- Fast pseudo-random rotations based on the Fast Hadamard Transform (FHT) from [6]. Unlike the first ingredient, this step is heuristic: we conjecture that the new LSH scheme achieves the same theoretical guarantees as the pure Cross-Polytope LSH or Voronoi LSH, but we currently do not know how to show it. However, there have been follow-up works [100, 43], where this issue has been (partially) addressed.

- If a dataset is extremely high-dimensional and very sparse (e.g., bag of words for text documents), then it pays off to first perform feature hashing [173] down to the intermediate dimension and then apply the above fast variant of the Cross-Polytope LSH.

In practice, the space bound $n^{1+\rho+o(1)}$ is usually too high. This is the reason why we need to adapt the technique of Multiprobe LSH [118] to the Cross-Polytope LSH, which reduces the space usage at a cost of somewhat increased query time. Multiprobe LSH can be seen as a practical analog of the space-constrained algorithm from Section 1.2.4.

We implement the resulting algorithm and release it as a part of FALCONN: a new C++ library with Python bindings for similarity search over high-dimensional data [152]. The implementation requires quite a bit of careful engineering: in particular, we extensively use SIMD to speed up various operations, and we spend lots of effort to make sure that all the necessary data structures occupy as little space as possible. We release our implementation of the Fast Hadamard Transform separately [153]: as it turns out, it is faster than what one can get out of FFTW [75], and we hope it might be useful for other implementations.

We perform a set of experiments on various kinds of data and conclude that FALCONN indeed significantly outperforms Hyperplane LSH as the theory predicts, and is competitive with the state of the art *heuristics* for high-dimensional similarity search.

1.2.6 ANN for general symmetric norms

In light of the above results, the ANN problem for ℓ_1 and ℓ_2 distances has been studied pretty thoroughly. However, our understanding of ANN for more general classes of distances is still very rudimentary.

Since for distances other than ℓ_1 and ℓ_2 there are no known dimension reduction results, we cannot assume that $d = \widetilde{O}(\log n)$ anymore. However, it is convenient to assume that $d = n^{o(1)}$, which means that we are fine with *any* polynomial dependence on the dimension d.

For instance, for the ℓ_∞ distance, the best known algorithm from [81] achieves for every $\varepsilon > 0$ space $n^{1+\varepsilon}$, query time $n^{o(1)}$ and approximation $O_\varepsilon(\log \log d)$. This is qualitatively different from ℓ_1/ℓ_2, where in this regime we would be getting approximation $O_\varepsilon(1)$. Somewhat unexpectedly, the $O_\varepsilon(\log \log d)$ approximation turns out to

be tight for certain restricted models of computation [11, 98].

In Chapter 4, which is based on the paper [25], we present a new general algorithm that solves ANN for any *symmetric* norm[13] (that is, a norm invariant under permutations of coordinates and sign flips), which achieves space $n^{1+o(1)}$, query time $n^{o(1)}$ and approximation $(\log \log n)^{O(1)}$.

The approach that does not quite work is to observe that any norm embeds into ℓ_∞ with distortion $1+\varepsilon$ for every $\varepsilon > 0$[14] [176], and then use the data structure for the ℓ_∞ distance from [81]. The problem with this reduction is that the required dimension of ℓ_∞ can be as high as $2^{\Omega(d)}$ even if the original norm is as simple as ℓ_2 [32].

The alternative approach is to design an embedding into a space more complicated than ℓ_∞, but which is nevertheless tractable. This would allow us to save on the dimension making it only polynomial in d. Indeed, we show that any symmetric norm embeds into a $d^{O(1)}$-dimensional normed space that is a *direct sum* of certain sufficiently simple spaces. Then, we invoke a general result from [82, 10] that gives an ANN data structure for such direct sums.

Finally, we show that for *general,* not necessarily symmetric norms a similar approach would not work: any universal normed space with dimension $d^{O(1)}$ that could potentially "host" all the d-dimensional normed spaces must incur distortion $d^{1/2-o(1)}$.

1.3 Characterization of sketchable distances

Sketching is a general notion of compression of a massive object while preserving its useful properties. It is typically used either to reduce the amount of space necessary to store a dataset (in particular, for streaming algorithms [136, 5] or similarity search [171, 172]) or to speed-up various computations [22, 99, 178].

[13] A norm is a function $\|\cdot\|\colon \mathbb{R}^d \to \mathbb{R}_+$ that satisfies:

- $\|x\| = 0$ iff $x = 0$;
- $\|\alpha x\| = |\alpha|\|x\|$;
- $\|x+y\| \le \|x\| + \|y\|$.

[14] That is, there is a map $f\colon \mathbb{R}^d \to \mathbb{R}^{d'}$ such that for every vector $x \in \mathbb{R}^d$, one has $\|f(x)\|_\infty \in (1 \pm \varepsilon) \cdot \|x\|$.

In this thesis, we are concerned with sketching *distance functions.* Given a metric[15] on $\mathbb{R}^d$, can we compress vectors so that we can accurately estimate the distance between two vectors given only their compressed versions (*sketches*)? It turns out that the answer to this question depends crucially on a metric at hand.

For the ℓ_2 distance and for every $\varepsilon > 0$, one can compress vectors into $O(1/\varepsilon^2)$ bits in a randomized manner such that from these sketches we can distinguish ℓ_2 distance at most 1 vs. more than $1 + \varepsilon$ with high probability. This result follows from the Johnson–Lindenstrauss Lemma [93, 65] together with a discretization trick from [66]. Note that the sketch size is completely independent of the original dimension d and is determined solely by a desired level of accuracy $\varepsilon > 0$. The main idea is to project $\mathbb{R}^d$ on a random subspace of dimension $O(1/\varepsilon^2)$; with high probability, this preserves the Euclidean distance between a given pair of points up to a multiplicative factor of $(1 \pm \varepsilon)$.

For the ℓ_p distances for $0 < p < 2$ (including ℓ_1), the random projections approach has been generalized by Indyk [84], who showed how to achieve the same bound (sketch size $O(1/\varepsilon^2)$ for distinguishing distances at most 1 vs. more than $1 + \varepsilon$). For every $0 < p \leq 2$, the bound $O(1/\varepsilon^2)$ on the sketch size is, in fact, tight [177].

On the other hand, for ℓ_p distances with $p > 2$, the situation is not as nice. It has been shown [9, 34, 87] that in this case the sketch size must be $\widetilde{\Theta}(d^{1-2/p})$ for any constant ε. The lower bound is shown using communication complexity. In particular, the result for ℓ_∞ follows from the celebrated linear lower bound for the disjointness problem [96]. Thus, even the simplest case of ℓ_p distances is quite delicate and mysterious, which justifies the further study of the problem.

The next observation is that a metric admits efficient sketches if it *embeds* into an ℓ_p space with $0 < p \leq 2$ with a small *distortion* D. That is, there exists a map $f\colon \mathbb{R}^d \to \mathbb{R}^{d'}$ such that for every $x_1, x_2 \in \mathbb{R}^d$ it is true that $\|f(x_1) - f(x_2)\|_p$ is within a multiplicative factor of D from the distance between x_1 and x_2 with respect to the

[15] A metric on a set X is a symmetric function $d\colon X \times X \to \mathbb{R}_+$ such that:

- $d(x, y) = 0$ iff $x = y$;
- $d(x, z) \leq d(x, y) + d(y, z)$.

metric. If there exists such an embedding, we can sketch our metric with approximation $O(D)$ and constant sketch size by applying the embedding and then using sketches for ℓ_p from [84].

In Chapter 5, which is based on the paper [23], we show that these are essentially the only possibilities if the metric is a *norm.* More precisely, we show that if a norm $\|.\|$ admits sketches of size s bits and approximation D, then one can embed $\|.\|$ into $\ell_{1-\varepsilon}$ with distortion $O(sD/\varepsilon)$ for every $\varepsilon > 0$. This provides a complete characterization of "efficiently sketchable" norms. The proof of this result uses a combination of information-theoretic and analytic tools.

If we take the result in the contrapositive, we conclude that for a norm non-embeddability into ℓ_p spaces for $0 < p < 2$ implies a lower bound on general sketches. This is very useful, since there are many known tools to show non-embeddability statements, and our result allows to lift them automatically to lower bounds for sketches.

As two concrete applications, we show first sketching lower bounds for the Earth Mover's Distance (EMD) [155] using a non-embeddability result from [134], and for the trace norm (a. k. a. the nuclear norm) using a result from [149].

1.4 Preliminaries and notation

We denote by $\mathbb{R}_+$ the set of non-negative real numbers. Let us denote $\langle \cdot, \cdot \rangle$ the standard dot product on $\mathbb{R}^d$. For a positive integer n, we define $[n]$ to be the set $\{1, 2, \ldots, n\}$. Whenever we use the notation $o(\cdot)$, $O(\cdot)$, $\omega(\cdot)$ or $\Omega(\cdot)$, we write all the parameters the implicit constant depends on as subscripts.

1.4.1 Metric and normed spaces

A *metric space* is a pair $M = (X, d)$, where X is an arbitrary set and $d \colon X \times X \to \mathbb{R}_+$ is a distance function that satisfies the following properties:

- For every $x, y \in X$, $d(x, y) = 0$ iff $x = y$;

- For every $x, y \in X$, $d(x, y) = d(y, x)$;
- For every $x, y, z \in X$, $d(x, z) \leq d(x, y) + d(y, z)$.

We denote $B_M(x, R)$ a closed ball in a metric space M centered in x and of radius R. More formally, $B_M(x, R) = \{x' \in X \mid d_X(x, x') \leq R\}$. Oftentimes, we will omit the subscript if the metric space is clear from the context.

An *embedding* of a metric space (X, d_X) into a metric space (Y, d_Y) with distortion $D \geq 1$ is a function $f\colon X \to Y$ such that there exists a constant $C > 0$ such that for every $x_1, x_2 \in X$ one has:

$$C \cdot d_X(x_1, x_2) \leq d_Y(f(x_1), f(x_2)) \leq DC \cdot d_X(x_1, x_2).$$

A *normed space* is a pair $(\mathbb{R}^d, \|\cdot\|)$, where the function $\|\cdot\|\colon \mathbb{R}^d \to \mathbb{R}_+$ is called a norm and satisfies the following properties:

- For every $x \in \mathbb{R}^d$, $\|x\| = 0$ iff $x = 0$;
- For every $\alpha \in \mathbb{R}$ and $x \in \mathbb{R}^d$, $\|\alpha x\| = |\alpha| \cdot \|x\|$;
- For every $x, y \in \mathbb{R}^d$, $\|x + y\| \leq \|x\| + \|y\|$.

It is immediate to check that any normed space defines a metric on $\mathbb{R}^d$ via the formula $d(x, y) = \|x - y\|$.

If we consider embeddings of a normed space into another normed space, we may want to restrict ourselves to *linear* embeddings, for which the mapping is given by a linear operator $A\colon \mathbb{R}^d \to \mathbb{R}^{d'}$.

If $X = (\mathbb{R}^d, \|\cdot\|)$ is a normed space, we can define the *dual* space $X^* = (\mathbb{R}^d, \|\cdot\|^*)$ as follows:

$$\|y\|^* = \sup_{\|x\| \leq 1} |\langle x, y \rangle|.$$

It is immediate to check that $\|\cdot\|^*$ is indeed a norm. Moreover, one has $(X^*)^* = X$. This follows from a finite-dimensional version of the Hahn–Banach theorem [156].

For a normed space X, we denote B_X the unit ball $B_X(0,1)$, which is a centrally-symmetric convex body. Finally, for a ball $B(x,R)$ in a normed space, we denote by $\partial B(x,R)$ its boundary.

1.4.2 Examples of metrics and norms

For every $p \geq 1$ and $d \geq 1$ we can define the ℓ_p norm on $\mathbb{R}^d$ as follows:

$$\|x\|_p = \left(\sum_{i=1}^{d} |x_i|^p \right)^{1/p}.$$

We can also define the ℓ_∞ norm as follows:

$$\|x\|_\infty = \max_{1 \leq i \leq d} |x_i|.$$

The proof of the fact that the above functions are indeed norms can be found in any introductory textbook in real or functional analysis.

For $0 < p < 1$, the function $\|\cdot\|_p$ is not a norm, since it might be the case that $\|x+y\|_p > \|x\|_p + \|y\|_p$. However, one has:

$$\|x+y\|_p \leq 2^{\frac{1}{p}-1} \cdot (\|x\|_p + \|y\|_p).$$

Such functions are called *quasi-norms*. We denote ℓ_p^d the space $\mathbb{R}^d$ equipped with the ℓ_p (quasi-)norm.

The dual to the ℓ_p^d space for $p \geq 1$ is the ℓ_q^d space, where $q \geq 1$ is such that

$$\frac{1}{p} + \frac{1}{q} = 1.$$

For $1 \leq k \leq d$, we define the *top-k norm* of a vector to be the sum of k largest absolute values of the coordinates. The top-1 norm coincides with the ℓ_∞ norm, while the top-d norm coincides with the ℓ_1 norm.

For $1 \leq p \leq \infty$, a *Schatten-p norm* of a square matrix $A \in \mathbb{R}^{n \times n}$ is defined as the

ℓ_p norm of the vector of the singular values of A. The Schatten-1 norm is also called the trace or the nuclear norm. The Schatten-2 norm coincides with the Frobenius norm, and the Schatten-∞ norm is also called the spectral or operator norm of a matrix.

Let $f\colon [n]^2 \to \mathbb{R}$ be a function such that $\sum_{i,j} f((i,j)) = 0$ (such functions form an (n^2-1)-dimensional vector space). Then the Lipschitz norm of f is defined as:

$$\|f\|_{\text{Lip}} = \max_{(i_1,j_1)\neq(i_2,j_2)} \frac{|f((i_1,j_1)) - f((i_2,j_2))|}{|i_1 - i_2| + |j_1 - j_2|}.$$

The *Earth Mover's Distance (EMD) norm* is defined to be the dual of the Lipschitz norm (see [134] for more information).

For metric spaces $M_1 = (X_1, d_1)$, $M_2 = (X_2, d_2)$, ..., $M_n = (X_n, d_n)$ and every $p \geq 1$, one can define an ℓ_p-*direct sum* of M_1, ..., M_n as follows. It is a metric space denoted by $M_1 \bigoplus_{\ell_p} M_2 \bigoplus_{\ell_p} \ldots \bigoplus_{\ell_p} M_n$ whose ground set is $M_1 \times M_2 \times \ldots \times M_n$ and the distance function is defined as follows:

$$d\left((x_1, x_2, \ldots, x_n), (y_1, y_2, \ldots, y_n)\right) = \left\|(d_1(x_1, y_1), d_2(x_2, y_2), \ldots, d_n(x_n, y_n))\right\|_p.$$

The ℓ_p-direct sum of normed spaces is also a normed space. Confusingly enough, ℓ_p-direct sums are sometimes called ℓ_p-direct products or ℓ_p-products.

1.4.3 Approximate near neighbor search (ANN)

Here we introduce one of the two central problems we study in this thesis formally.

Definition 1.4.1 (The (c, r)-ANN problem)**.** Given an n-point dataset P lying in a metric space $M = (X, d_X)$, the goal is to preprocess it to answer the following queries. Given a query point $q \in X$ such that there exists a data point within distance r from q, return a data point within distance cr from q. All distances are computed in M using the function $d_X(\cdot, \cdot)$.

Both approximation c and the distance scale r are known from the beginning rather than being a part of a query. Usually the ANN problem is studied for the case

when M is a normed d-dimensional space, where $d = n^{o(1)}$. We assume that the norm of a vector $x \in \mathbb{R}^d$ can be computed in time $d^{O(1)} = n^{o(1)}$.

1.4.4 Locality-sensitive hashing (LSH)

Here we introduce locality-sensitive hashing, one of the canonical tools to solve the ANN problem. Next comes the central definition.

Definition 1.4.2. We say that a hash family $\mathcal{H}$ on a metric space $M = (X, d_X)$ is (r_1, r_2, p_1, p_2)-*sensitive*, if for every $p, q \in X$ one has $\Pr_{h\sim\mathcal{H}}[h(x) = h(y)] \geq p_1$ if $d_X(x, y) \leq r_1$, and $\Pr_{h\sim\mathcal{H}}[h(x) = h(y)] \leq p_2$ if $d_X(x, y) > r_2$,

It is known [85, 77] that an *efficient* (r, cr, p_1, p_2)-sensitive hash family on a $n^{o(1)}$-dimensional normed space implies a data structure for (c, r)-ANN with space $n^{1+\rho+o(1)}/p_1$ and query time $n^{\rho+o(1)}/p_1$, where $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$.

Note that sometimes it is more convenient to talk about *random partitions* of the metric space M. Clearly, random partitions and hash families are equivalent in this context.

Let us briefly sketch this reduction, see [77] for the further details. The reduction depends on two integer parameters: a number of tables L and a number of hash functions in a single table K. In each of the L tables, we sample K hash functions $h_1, h_2, \ldots, h_K$ from the LSH family, and hash the dataset using the composite hash function $p \mapsto \big(h_1(p), h_2(p), \ldots, h_K(p)\big)$. Now, to answer a query q, we iterate over all the L tables, and in each table we retrieve[16] the data points that hash to the same bucket as q; as soon as we find a point within distance cr from the query, we stop immediately.

Let us show how to set the parameters K and L. First, we choose K to be the smallest integer such that p_2^K is at most $1/n$. Then, for a fixed query and a given table, there will be at most $n \cdot p_2^K \leq 1$ far points that fall into the same bucket with the query. Thus, the expected query time is $n^{o(1)} \cdot L$ given that we can evaluate hash functions from the LSH family efficiently. It remains to choose L. Since we are a data

[16] This can be implemented efficiently using classical hash tables.

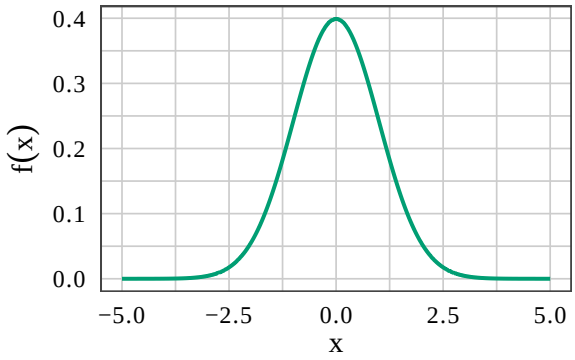


Figure 1-7: Density of $N(0,1)$.

point within distance r from the query, the probability of success for a single table is at least $p_1^K = p_1^{\left\lceil \frac{\log n}{\log(1/p_2)} \right\rceil} \geq n^{-\rho} \cdot p_1$. Thus, to get a high probability of success, one needs to choose $L = O(n^\rho/p_1)$.

1.4.5 Gaussian distribution

Our algorithms crucially use the *Gaussian distribution* $N(a, \sigma^2)$, which is defined as the distribution over the reals $\mathbb{R}$ with density:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-a)^2}{2\sigma^2}}$$

(see Figure 1-7). The mean and the variance of $N(a, \sigma^2)$ are a and σ^2, respectively. We denote by $N(a, \sigma^2)^d$ a distribution over $\mathbb{R}^d$ where each coordinate is an i.i.d. sample from $N(a, \sigma^2)$.

A property of the Gaussian distribution most useful for us is its *spherical symmetry*. Namely, the density of $N(0,1)^d$ in point $v \in \mathbb{R}^d$ depends only on the norm $\|v\|_2$. In particular, if a vector distributed according to $N(0,1)^d$ is multiplied by a fixed orthogonal matrix, the resulting vector is also distributed according to $N(0,1)^d$. Spherical symmetry implies 2-*stability:* if v is distributed according to $N(0,1)^d$ and $u \in \mathbb{R}^d$ is an arbitrary vector, then $\langle u, v \rangle$ is distributed according to $N(0, \|u\|_2^2)$. Indeed,

$$\langle u, v \rangle = \langle Au, Av \rangle$$

for any fixed orthogonal matrix A. We can choose A so that $Au = (\|u\|_2, 0, \ldots, 0)$, and remembering that, by the spherical symmetry, Av has the same distribution as v, we get the desired 2-stability property.

Chapter 2

New ANN algorithms for ℓ_1 and ℓ_2

2.1 Problem definition

In this chapter, we will show new algorithms for the Approximate Nearest Neighbor search problem for the ℓ_p distances for $1 \leq p \leq 2$. Recall that the ℓ_p distance between two vectors $x, y \in \mathbb{R}^d$ is defined as follows:

$$\|x - y\|_p = \begin{cases} \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{1/p} & 1 \leq p < \infty, \\ \max\limits_{1 \leq i \leq d} |x_i - y_i| & p = \infty. \end{cases}$$

For applications, the two most important cases are ℓ_1 (a.k.a. Manhattan distance) and ℓ_2 (a.k.a. Euclidean distance).

First, let us define the actual problem we will be solving in this chapter.

Definition 2.1.1 (The (c, r)-ANN problem)**.** Given an n-point dataset P in $\mathbb{R}^d$ with $d = n^{o(1)}$, the goal is to preprocess it to answer the following queries. Given a query point $q \in \mathbb{R}^d$ such that there exists a data point within ℓ_p distance r from q, return a data point within ℓ_p distance cr from q.

Both approximation c and the distance scale r are known from the beginning rather than being a part of a query.

As we pointed out before, the two most important ℓ_p distances, both for theory

and for applications, are ℓ_1 and ℓ_2. There are two further important special cases:

- The (c, r)-ANN problem for the ℓ_1 distance, when the dataset and queries are promised to lie in the hypercube $\{0, 1\}^d$. The hypercube equipped with the ℓ_1 distance is often called the *Hamming space*. Sometimes, it will be convenient for us to work with $\{-1, 1\}^d$ instead of $\{0, 1\}^d$.

- The (c, r)-ANN problem for the ℓ_2 distance, when the dataset and queries are promised to lie on the unit sphere $S^{d-1} = \{x \in \mathbb{R}^d : \|x\|_2 = 1\} \subset \mathbb{R}^d$.

For the (c, r)-ANN problem over the whole $\mathbb{R}^d$, we can assume w.l.o.g. that $r = 1$ by rescaling the dataset and queries. However, this is not the case for the Hamming space and the unit sphere settings. As we will see later in this chapter, for both of these settings, the (c, r)-ANN problem becomes easier as r increases[1].

2.1.1 Parameters and guarantees

Data structures for the ANN problem have several characteristics we would like to optimize over:

- *Space* the data structure occupies;

- *Query time:* time it takes to answer a single query;

- *Preprocessing time:* time it takes to build the data structure;

- *Insertion/deletion time:* in some scenarios we might allow the dataset to change, and we would like to minimize time it takes to add/remove a single point.

As it turns out, the two most fundamental parameters are *space* and *query time*: we will be mostly focusing on those, briefly addressing the remaining two parameters towards the end of the chapter. Finally, all the data structures we present are *randomized*. We guarantee that we return a correct answer for a given query with probability, say, 0.9. This probability is taken over both preprocessing and query

[1] As an extreme example, consider the case of the unit sphere and $cr \geq 2$. Then, the (c, r)-ANN problem becomes trivial, since any data point is an answer to any valid query.

stages. Such a guarantee is similar to, say, what *universal hashing* [51] gives for the dictionary problem.

2.2 Prior work

In this section we briefly survey the state of the art for the ANN algorithms over the ℓ_1 and ℓ_2 distances prior to the work from this thesis.

2.2.1 Locality-sensitive hashing (LSH)

A classic technique for ANN over ℓ_1 and ℓ_2 is *Locality-Sensitive Hashing* (LSH), introduced in 1998 [85, 77]. The main idea is to use *random space partitions* of $\mathbb{R}^d$, for which a pair of close points (at a distance at most r) is more likely to belong to the same part than a pair of far points (at a distance more than cr). Given such a partition, the data structure splits the dataset P according to the partition, and, given a query, checks all the data points which belong to the same part as the query. In order to return a near neighbor with high probability, one needs to maintain several partitions and to check all of them during the query stage. The LSH approach yields data structures with space $n^{1+\rho+o(1)}$ and query time $n^{\rho+o(1)}$, where ρ is the key quantity measuring the quality of a space partition for a particular distance function and approximation $c \geq 1$. Usually, $\rho = 1$ for $c = 1$ and $\rho \to 0$ as $c \to \infty$.

More specifically, the value of ρ is equal to $\frac{\log(1/p_1)}{\log(1/p_2)}$, where:

- p_1 is the *lower bound* on the probability that two points within distance *at most* r end up in the same part of a random partition;
- p_2 is the *upper bound* on a similar probability for pairs within distance *more than* cr.

For a sketch of the reduction see Section 1.4.4.

Since the introduction of LSH in [85], subsequent research established optimal values of the LSH exponent ρ for several metrics of interest, including ℓ_1 and ℓ_2. For the ℓ_1 distance, the optimal value is $\rho = \frac{1}{c} \pm o(1)$ [85, 129, 140]. For the ℓ_2 case, the

best possible exponent is $\rho = \frac{1}{c^2} \pm o(1)$ [85, 66, 13, 129, 140]. Since the LSH approach has been really successful both in theory and in practice, it has become a pressing open problem to break through this barrier and obtain algorithms for the ANN problem that are better than the best possible LSH-based algorithms.

2.2.2 Time–space trade-offs

Since the early results on LSH, a natural question has been whether one can obtain query time vs. space trade-offs for a fixed approximation c. Indeed, data structures with *polynomial* space and *polylogarithmic* query time were introduced [85, 108] simultaneously with LSH. In practice, the most important regime is that of *near-linear* space, since space is usually a harder constraint than time: see, e.g., [118]. This regime has been studied since [80], with subsequent improvements in [144, 13, 97].

For the ℓ_2 distance, the work of Kapralov [97] shows a smooth time–space trade-off that interpolates between all the three abovementioned regimes: near-linear space, balanced LSH regime, and the regime of the low query time. In the LSH regime, it almost matches the best LSH construction by [13], while in other regimes it improves upon the prior work significantly.

2.3 The main result

The following is the main result of the present chapter.

Theorem 2.3.1. *For every $1 \le p \le 2$, $c > 1$ and $\rho_s, \rho_q \ge 0$ such that:*

$$c^p \cdot \sqrt{\rho_q} + (c^p - 1) \cdot \sqrt{\rho_s} \ge \sqrt{2c^p - 1},$$

there exists a data structure for the c-ANN problem over the ℓ_p distance and n-point datasets in $\mathbb{R}^d$ with $d = n^{o(1)}$ that has the following parameters:

- *space $n^{1+\rho_s+o(1)}$,*
- *query time $n^{\rho_q+o(1)}$,*

- *preprocessing time* $n^{1+\rho_s+o(1)}$,
- *insertion/deletion time* $n^{\rho_s+o(1)}$.

In Figure 2-1, we illustrate the main result by instantiating it for various settings. See Figure 2-2 for the plot showing the trade-off between ρ_s and ρ_q we achieve for $c = 2$ and the ℓ_2 distance.

2.3.1 Comparison with the prior work

Balanced regime. For the balanced regime (space $n^{1+\rho}$, query time n^{ρ}), the new result is the first to bypass the LSH barrier. For the ℓ_1 distance, the best possible LSH construction from [85] obtains the exponent $\rho = \frac{1}{c} + o(1)$, while the new exponent is $\frac{1}{2c-1} + o(1)$. Similarly, for the ℓ_2 distance the best possible LSH from [13] gives $\rho = \frac{1}{c^2} + o(1)$, while the new bound is $\frac{1}{2c^2-1} + o(1)$. Thus, in the balanced regime we obtain a polynomial improvement both in space and in query time.

The main insight that allows us to overcome the LSH lower bounds [129, 140] is to allow random partitions to *depend on the dataset.* The definition of LSH requires certain conditions (close pairs collide with probability at least p_1, far pairs collide with probability at most p_2) to hold for every pair of points from the ambient space $\mathbb{R}^d$. However, for the ANN application, it would be enough if these conditions held for pairs, where one of the points is one of the n dataset points, which we know in advance. That is exactly the route we take. Data-dependent partitions for similarity search and related problems are ubiquitous in practice [171, 172], but the main contribution of this work is to establish their utility for getting improved *worst-case bounds.* Informally speaking, we show that every dataset has some structure to exploit.

Other regimes. In other regimes we also significantly improve upon the state of the art. In particular, for the near-linear space regime, the new algorithm is the first which achieves sub-linear in n query time for *every* approximation $c > 1$. The previous best result from [97] gets sub-linear time only for $c > \sqrt{3}$.

Distance	Approximation	Regime	Space	Query time
ℓ_1	General $c > 1$	Low space	$n^{1+o(1)}$	$n^{\frac{2}{c}-\frac{1}{c^2}+o(1)}$
		Balanced	$n^{1+\frac{1}{2c-1}+o(1)}$	$n^{\frac{1}{2c-1}+o(1)}$
		Fast queries	$n^{\frac{c^2}{(c-1)^2}+o(1)}$	$n^{o(1)}$
	$c = 1+\varepsilon$, $\varepsilon \to 0$	Low space	$n^{1+o(1)}$	$n^{1-\varepsilon^2+O(\varepsilon^3)}$
		Balanced	$n^{2-2\varepsilon+O(\varepsilon^2)}$	$n^{1-2\varepsilon+O(\varepsilon^2)}$
		Fast queries	$n^{\frac{1}{\varepsilon^2}+O\left(\frac{1}{\varepsilon}\right)}$	$n^{o(1)}$
	$c = 2$	Low space	$n^{1+o(1)}$	$n^{\frac{3}{4}+o(1)} = n^{0.75+o(1)}$
		Balanced	$n^{\frac{4}{3}+o(1)} \approx n^{1.34}$	$n^{\frac{1}{3}+o(1)} \approx n^{0.34}$
		Fast queries	$n^{4+o(1)}$	$n^{o(1)}$
	$c \to \infty$	Low space	$n^{1+o(1)}$	$n^{\frac{2}{c}+O\left(\frac{1}{c^2}\right)}$
		Balanced	$n^{1+\frac{1}{2c}+O\left(\frac{1}{c^2}\right)}$	$n^{\frac{1}{2c}+O\left(\frac{1}{c^2}\right)}$
		Fast queries	$n^{1+\frac{2}{c}+O\left(\frac{1}{c^2}\right)}$	$n^{o(1)}$
ℓ_2	General $c > 1$	Low space	$n^{1+o(1)}$	$n^{\frac{2}{c^2}-\frac{1}{c^4}+o(1)}$
		Balanced	$n^{1+\frac{1}{2c^2-1}+o(1)}$	$n^{\frac{1}{2c^2-1}+o(1)}$
		Fast queries	$n^{\frac{c^4}{(c^2-1)^2}+o(1)}$	$n^{o(1)}$
	$c = 1+\varepsilon$, $\varepsilon \to 0$	Low space	$n^{1+o(1)}$	$n^{1-4\varepsilon^2+O(\varepsilon^3)}$
		Balanced	$n^{2-4\varepsilon+O(\varepsilon^2)}$	$n^{1-4\varepsilon+O(\varepsilon^2)}$
		Fast queries	$n^{\frac{1}{4\varepsilon^2}+O\left(\frac{1}{\varepsilon}\right)}$	$n^{o(1)}$
	$c = 2$	Low space	$n^{1+o(1)}$	$n^{\frac{7}{16}+o(1)} \approx n^{0.44}$
		Balanced	$n^{\frac{8}{7}+o(1)} \approx n^{1.15}$	$n^{\frac{1}{7}+o(1)} \approx n^{0.15}$
		Fast queries	$n^{\frac{16}{9}+o(1)} \approx n^{1.78}$	$n^{o(1)}$
	$c \to \infty$	Low space	$n^{1+o(1)}$	$n^{\frac{2}{c^2}+O\left(\frac{1}{c^4}\right)}$
		Balanced	$n^{1+\frac{1}{2c^2}+O\left(\frac{1}{c^4}\right)}$	$n^{\frac{1}{2c^2}+O\left(\frac{1}{c^4}\right)}$
		Fast queries	$n^{1+\frac{2}{c^2}+O\left(\frac{1}{c^4}\right)}$	$n^{o(1)}$

Figure 2-1: Specializations of the main result (Theorem 2.3.1) for various settings. This includes: ℓ_1 and ℓ_2 distances; general approximation c, c close to 1, $c = 2$, large c; the regime of space $n^{1+o(1)}$, the balanced regime (space $n^{1+\rho}$, query time n^{ρ}) and the regime of fast query time $n^{o(1)}$.

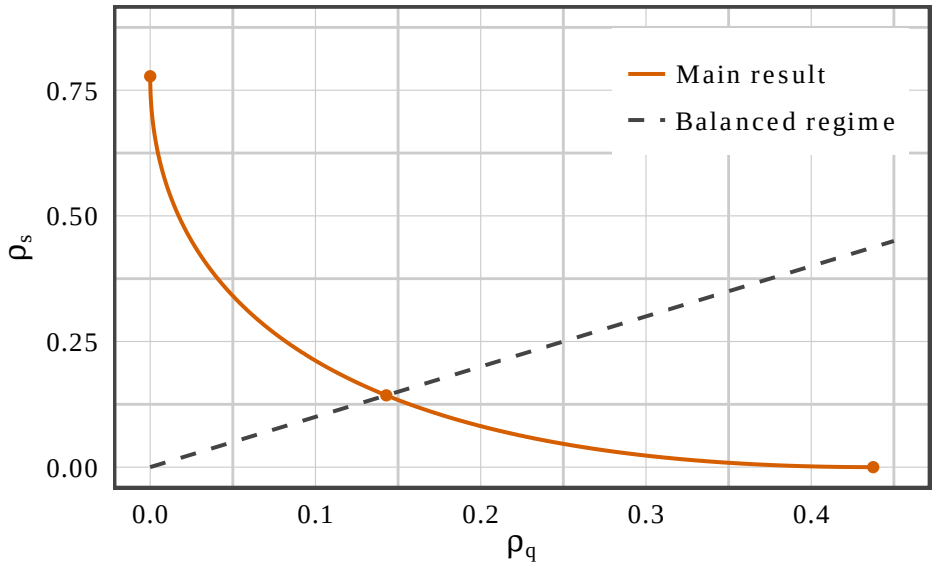


Figure 2-2: Illustration of the main result (Theorem 2.3.1) for the ℓ_2 distance and approximation $c = 2$. The time–space trade-off interpolates between fast queries ($\rho_q = 0$, $\rho_s = 7/9 = 0.77\ldots$), the balanced regime ($\rho_q = \rho_s = 1/7 = 0.14\ldots$) , and the near-linear space regime ($\rho_q = 7/16 = 0.43\ldots$, $\rho_s = 0$).

We compare the new algorithm with the prior state of the art ([13] in the balanced regime, and [97] in other regimes) for the ℓ_2 distance and $c = 2$ on Figure 2-3.

2.4 Overview of the algorithm

We now describe the proof of Theorem 2.3.1 at a high level. First of all, in Section 2.5, using by now standard tools, we reduce the general (c, r)-ANN problem for the ℓ_p distances to the $\big(c - o(1), r\big)$-ANN problem for the ℓ_2 distance on the unit sphere $S^{d-1} \subset \mathbb{R}^d$, where $d = \log^{1+o(1)} n$ and $r = \frac{1}{\log\log n}$. For this case it is enough to achieve the following trade-off:

$$c^2 \cdot \sqrt{\rho_q} + (c^2 - 1) \cdot \sqrt{\rho_s} \geq \sqrt{2c^2 - 1}, \tag{2.1}$$

The new algorithm for the unit sphere case consists of two major stages. In the first stage (Section 2.8), we give an algorithm for *random* instances (introduced formally in Section 2.6). To generate such an instance, we sample a dataset uniformly at

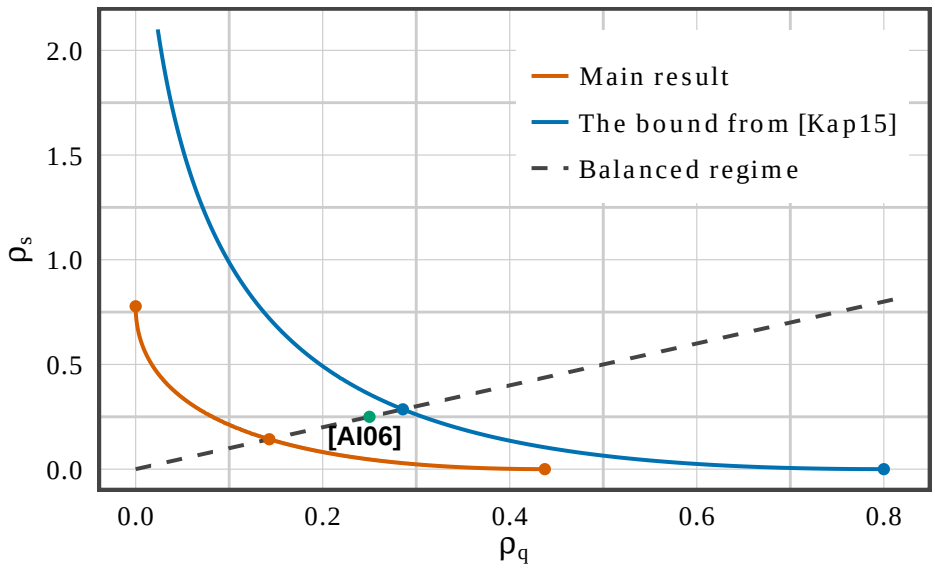


Figure 2-3: Comparison of the main result (Theorem 2.3.1) with the prior state of the art for the ℓ_2 distance and $c = 2$. In the balanced regime, the best prior bound follows from the work of [13], while in all the other regimes, the best bound has been shown in [97]. For $\rho_q = 0$, the algorithm from [97] gives $\rho_s = 4$ (this point does not fit on the plot).

random on a unit sphere $S^{d-1} \subset \mathbb{R}^d$ and plant a query at random within distance $\sqrt{2}/c$ from a randomly chosen data point. In the second stage, we show the claimed result for the *worst-case* instances by combining the algorithm from the first stage with data-dependent partitioning.

Data-independent partitions. To handle random instances, we use a certain *data-independent* partitioning process, which we briefly introduce below. It can be seen as a modification of spherical Locality-Sensitive Filtering from [38], and is related to a data structure from [146]. While this data-independent approach can be extended to *worst case* instances, it gives a bound worse than the desired (2.1).

We now describe the partitioning process which produces a decision tree to solve an ANN instance on the unit sphere S^{d-1}. We take our initial dataset $P \subset S^{d-1}$ and sample T i.i.d. standard Gaussian d-dimensional vectors $z_1, z_2, \ldots, z_T \in \mathbb{R}^d$. The sets

$P_i \subseteq P$ (not necessarily disjoint) are defined for each z_i as follows:

$$P_i = \{p \in P \mid \langle z_i, p \rangle \geq \eta_s\}.$$

We then recurse and repeat the above procedure for each non-empty P_i. We stop the recursion once we reach depth K. The above procedure generates a tree of depth K and degree at most T, where each leaf explicitly stores the corresponding subset of the dataset. To answer a query $q \in S^{d-1}$, we start at the root and descend into (potentially multiple) P_i's for which $\langle z_i, q \rangle \geq \eta_q$. When we eventually reach the K-th level, we iterate through all the points stored in the leaf searching for a near neighbor.

The parameters T, K, η_s and η_q depend on the distance threshold r, the approximation factor c, as well as the desired space and query time exponents ρ_s and ρ_q. The special case of $\eta_s = \eta_q$ corresponds to the balanced regime $\rho_s = \rho_q$; $\eta_s < \eta_q$ corresponds to the "fast queries" regime $\rho_q < \rho_s$ (the query procedure is more selective); and $\eta_s > \eta_q$ corresponds to the "low memory" regime $\rho_s < \rho_q$. The analysis of this algorithm relies on bounds on the Gaussian area of certain two-dimensional sets (see Section 2.7).

This algorithm has two important consequences. First, we obtain the desired trade-off (2.1) for *random instances* by setting $r = \frac{\sqrt{2}}{c}$. Second, we obtain a worse trade-off trade-off for $r = \frac{1}{\log \log n}$. Namely, we get:

$$(c^2 + 1)\sqrt{\rho_q} + (c^2 - 1) \cdot \sqrt{\rho_s} \geq 2c. \tag{2.2}$$

Even though it is worse than the desired bound from (2.1)[2], it is already non-trivial. In particular, (2.2) is already better than all the prior work on time–space trade-offs for ANN.

Data-dependent partitions. We then improve upon (2.2) for worst-case instances and obtain the final result, Theorem 2.3.1. For this we employ data-dependent partitioning.

[2]See Figure 2-4 for comparison for the case $c = 2$.

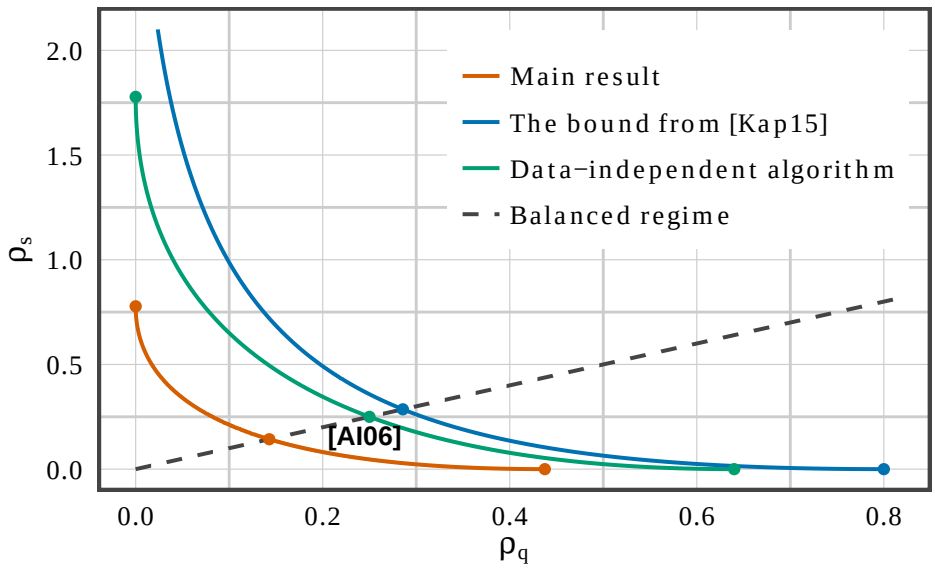


Figure 2-4: Illustration of the data-independent bound (2.2) for $c = 2$. In the balanced regime, it matches the best LSH from [13], while in other regimes it improves upon the work of [97]. The final bound is polynomially better than the data-independent one.

If pairwise distances between data points are distributed roughly like for a random instance, we could apply the data-independent procedure. In absence of such a guarantee, we manipulate the dataset in order to reduce it to a random-looking case. Namely, we search for low-diameter clusters that contain many data points. We extract these clusters, and we enclose each of them in a ball of radius non-trivially smaller than one, and we recurse on each cluster. For the remaining points, which do not lie in any cluster, we perform one step of the data-independent algorithm: we sample T Gaussian vectors, form T subsets of the dataset, and recurse on each subset. Overall, we make progress in two ways: for the clusters, we make them a bit more isotropic after re-centering and shrinking, which, after several re-centerings, makes the instance amenable to the data-independent algorithm, and for the remainder of the points, we can show that the absence of dense clusters makes the data-independent algorithm work for a single level of the tree (though, when recursing into P_i's, dense clusters may re-appear, which we will need to extract again).

While the above intuition is very simple and, in hindsight, natural, the actual execution requires a good amount of work. For example, we need to formalize "low-

diameter", "lots of points", "more isotropic", etc.

Finally, getting fast preprocessing time $n^{1+\rho_s+o(1)}$ and fast insertion/deletion times $n^{\rho_s+o(1)}$ requires additional work. For the former, we show how to quickly find dense cluster using sampling (see Section 2.9.5). For the latter we employ a standard dynamization technique [142] (see Section 2.9.6).

2.5 Simplification of the problem

It can be shown that in order to prove Theorem 2.3.1, it is enough to establish the following special case of it.

Theorem 2.5.1. *For every $c > 1$ and $\rho_s, \rho_q \geq 0$ such that:*

$$c^2 \cdot \sqrt{\rho_q} + (c^2 - 1) \cdot \sqrt{\rho_s} \geq \sqrt{2c^2 - 1}, \tag{2.3}$$

there exists a data structure for the (c, r)-ANN problem over the ℓ_2 distance with n-point datasets and queries lying on the unit sphere $S^{d-1} \subset \mathbb{R}^d$ with $d = \log^{1+o(1)} n$ and $r = \frac{1}{\sqrt{\log\log n}}$ that has the following parameters:

- *space $n^{1+\rho_s+o(1)}$,*
- *query time $n^{\rho_q+o(1)}$,*
- *preprocessing time $n^{1+\rho_s+o(1)}$,*
- *insertion/deletion times $n^{\rho_s+o(1)}$.*

In short, it is enough to handle the (c, r)-ANN problem over the ℓ_2 distance on the unit sphere in $\mathbb{R}^d$ for $d = \log^{1+o(1)} n$ and $r = \frac{1}{\log\log n} = o(1)$.

The proof of the fact that Theorem 2.5.1 implies Theorem 2.3.1 is by now standard if a bit tedious. Let us list the necessary steps and provide pointers to the literature.

- We reduce the (c, r)-ANN problems over the ℓ_p distance to the $(c^{p/2}, 1)$-ANN problem over the ℓ_2 distance. This reduction is described in detail in [137].

- We reduce the dimension to $d' = \log^{1+o(1)} n$ by using the dimension reduction lemma of Johnson and Lindenstrauss [93, 65]. This step introduces multiplicative distortion $1 \pm o(1)$ for pairwise distances, which is acceptable for us.

- Next, we reduce the diameter of the dataset to $O\Big((\log\log n)^{1/4}\Big)$. This can be done by partitioning the dataset using LSH family from [66] and querying the part, where the query belongs. We need to repeat this procedure $n^{O\Big(\frac{1}{(\log\log n)^{1/4}}\Big)} = n^{o(1)}$ times to get high probability of success.

- Finally, we reduce the problem to the unit sphere case with $r = \frac{1}{\sqrt{\log\log n}}$. This reduction can be found in [169].

Furthermore, from now on we will mostly focus on bounding the space and query time, postponing the preprocessing and insertion/deletion times until Sections 2.9.5 and 2.9.6, respectively.

2.6 Random ANN instances

Let us look at the complexity of (c, r)-ANN on the unit sphere S^{d-1} when c is fixed. As we saw in Section 2.5, the case $r = o(1)$ is the hardest, since it implies a c-ANN algorithm for the whole $\mathbb{R}^d$ with essentially the same guarantees. At the other extreme is the case $r \geq \frac{2}{c}$. Since the diameter of the sphere is 2, if $cr \geq 2$, any data point is an answer for any valid query. Hence, in this case, the (c, r)-ANN problem is trivial.

As it turns out, there exists an intermediate regime: $r = \frac{\sqrt{2}}{c}$, which corresponds to the following *random instances*.

- A dataset $P \subset S^{d-1}$ is given by n unit vectors, where each vector is drawn independently and uniformly at random from S^{d-1}. We assume that $d = \omega(\log n)$.

- A query $q \in S^{d-1}$ is drawn by first choosing a dataset point $p \in P$ uniformly at random, and then choosing q uniformly at random from all points in S^{d-1} within distance $\frac{\sqrt{2}}{c}$ from p.

- The goal of the data structure is to preprocess P in order to recover the data point p from the query point q quickly.

Any $\left(c - o(1), \frac{\sqrt{2}}{c}\right)$-ANN data structure for the unit sphere must be able to find p given q with high probability. Indeed, the distance between p and q is at most $\frac{\sqrt{2}}{c}$, while all the distances from q to other data points are at least $\sqrt{2} - o(1)$ with high probability, since $d = \omega(\log n)$[3].

As it turns out, handling these random instances are crucial for getting improved ANN algorithms for the general case. Namely, the proof of Theorem 2.5.1 proceeds in two big steps:

- We show how to get the desired bounds (2.1) on space and query time for $\left(c - o(1), r\right)$-ANN with $r = \frac{\sqrt{2}}{c}$, and, as discussed above, this setting is enough to handle random instances;

- Then, we will see how to reduce the setting of Theorem 2.5.1, $r = o(1)$, to the above case of $r = \frac{\sqrt{2}}{c}$ by finding subsets of the dataset that "look random".

Section 2.8 is devoted to the first part, while (a much more involved) Section 2.9 describes the second part.

2.7 Gaussians and tail bounds

Here we develop a bit the theory of the Gaussian distribution introduced in Section 1.4.5.

For $0 \leq s \leq 2$, let $\alpha(s) = 1 - \frac{s^2}{2}$ be the cosine of the angle between two points on a unit Euclidean sphere S^{d-1} with distance s between them, and $\beta(s) = \sqrt{1 - \alpha^2(s)}$ be the sine of the same angle (see Figure 2-5).

We introduce two functions that will be useful later. First, for $\rho > 0$, let

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

[3]This follows from the fact that two random vectors on a high-dimension sphere are nearly orthogonal with high probability. See, e.g., [32, 121].

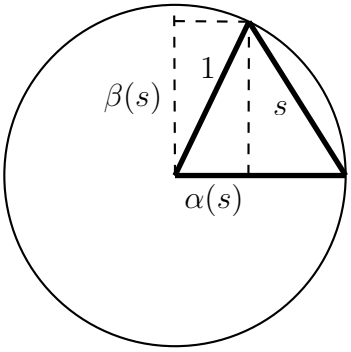


Figure 2-5: Illustration of the definition of $\alpha(s)$ and $\beta(s)$.

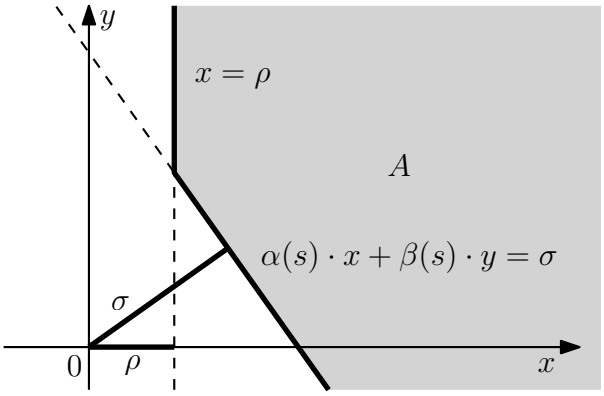


Figure 2-6: Illustration of the definition of A.

where $u \in S^{d-1}$ is an arbitrary point on the unit sphere. Note that $F(\rho)$ does not depend on the specific choice of u due to the spherical symmetry of $N(0,1)^d$.

Second, for $0 \leq s \leq 2$ and $\rho, \sigma > 0$, let

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

where $u, v \in S^{d-1}$ are arbitrary points from the unit sphere with $\|u - v\|_2 = s$. As with F, the value of $G(s, \rho, \sigma)$ does not depend on the specific choice of u and v; it only depends on the distance $s = \|u - v\|_2$ between them. Clearly, $G(s, \rho, \sigma)$ is non-increasing in s, for fixed ρ and σ.

We state two useful bounds on $F(\cdot)$ and $G(\cdot, \cdot, \cdot)$. The first is a standard tail bound for the Gaussian distribution and the second can be seen as a certain two-dimensional generalization of the first.

Lemma 2.7.1 ([74]). *For $\rho \to \infty$,*

$$F(\rho) = e^{-\left(1 \pm O\left(\frac{\log \rho}{\rho^2}\right)\right) \cdot \frac{\rho^2}{2}}.$$

Proof. By the 2-stability,

$$F(\rho) = \Pr_{z \sim N(0,1)^d} [\langle z, u \rangle \geq \rho] = \Pr_{x \sim N(0,1)} [x \geq \rho] = \frac{1}{\sqrt{2\pi}} \int_{\rho}^{\infty} e^{-x^2/2}\, dx.$$

We can bound the latter integral as follows:

$$\begin{aligned}
\left(\frac{1}{\rho} - \frac{1}{\rho^3}\right) \cdot \frac{e^{-\rho^2/2}}{\sqrt{2\pi}} &= \frac{1}{\sqrt{2\pi}} \int_{\rho}^{\infty} \left(1 - \frac{3}{x^4}\right) e^{-x^2/2}\, dx \\
&\leq \frac{1}{\sqrt{2\pi}} \int_{\rho}^{\infty} e^{-x^2/2}\, dx \\
&\leq \frac{1}{\sqrt{2\pi}} \int_{\rho}^{\infty} \left(1 + \frac{1}{x^2}\right) e^{-x^2/2}\, dx = \frac{1}{\rho} \cdot \frac{e^{-\rho^2/2}}{\sqrt{2\pi}},
\end{aligned}$$

which implies the required statement. □

Lemma 2.7.2. *If $\rho, \sigma \to \infty$, then, for every $0 < s < 2$, one has:*

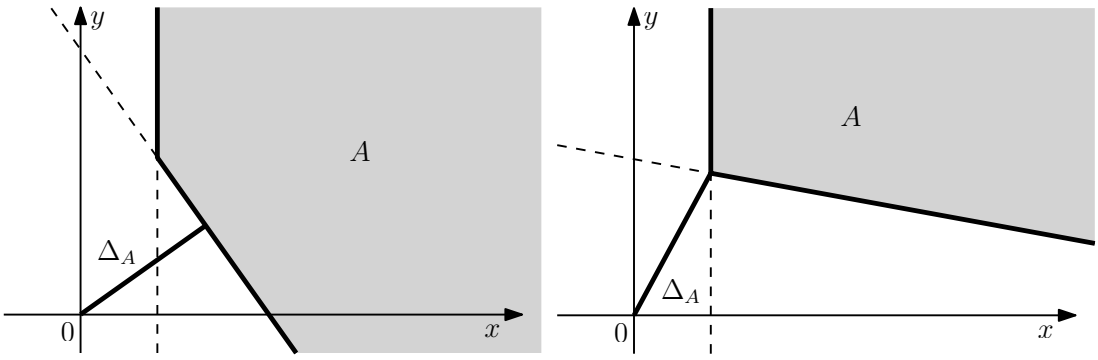
$$G(s, \rho, \sigma) = e^{-\left(1 \pm O\left(\frac{\log \Delta}{\Delta^2}\right)\right) \cdot \frac{\Delta^2}{2}},$$

where:

$$\Delta^2 = \begin{cases} \max\{\rho^2, \sigma^2\}, & \text{if } \frac{\min\{\rho,\sigma\}}{\max\{\rho,\sigma\}} < \alpha(s) \leq 1; \\ \frac{\rho^2 + \sigma^2 - 2\alpha(s) \cdot \rho\sigma}{\beta^2(s)}, & \text{otherwise.} \end{cases} \tag{2.4}$$

Proof. By the spherical symmetry,

$$\begin{aligned}
G(s, \rho, \sigma) &= \Pr_{z \sim N(0,1)^d} [\langle z, u \rangle \geq \rho \text{ and } \langle z, v \rangle \geq \sigma] \\
&= \Pr_{x,y \sim N(0,1)} [x \geq \rho \text{ and } \alpha(s) \cdot x + \beta(s) \cdot y \geq \sigma].
\end{aligned}$$



(a) The closest point lies "on a side". (b) The closest point is a "corner".

Figure 2-7: Two cases for Δ_A correspond to the two cases in the statement of Lemma 2.7.2.

The latter quantity is the Gaussian measure $\mathcal{G}(A)$ of the two-dimensional set

$$A = \{(x, y) \in \mathbb{R}^2 \mid x \geq \rho \text{ and } \alpha(s) \cdot x + \beta(s) \cdot y \geq \sigma\}$$

(see Figure 2-6). Let us denote Δ_A the distance from the origin to A. It is not hard to check that Δ_A is equal to Δ from the statement of the lemma. The two cases in (2.4) correspond to Figure 2-7a and Figure 2-7b, respectively. Thus, it remains to show that if $\rho, \sigma \to \infty$, then

$$\mathcal{G}(A) = e^{-\left(1 \pm O\left(\frac{\log \Delta_A}{\Delta_A^2}\right)\right) \cdot \frac{\Delta_A^2}{2}}.$$

For $r > 0$, let us denote $0 \leq \mu_A(r) \leq 1$ the normalized measure of the intersection $A \cap \partial B(0, r)$:

$$\mu_A(r) = \frac{\mu(A \cap \partial B(0, r))}{2\pi r},$$

where μ is the standard one-dimension Lebesgue measure (see Figure 2-8). Then, integrating in the polar coordinates, we get:

$$\mathcal{G}(A) = \int_{\Delta_A}^{\infty} \mu_A(r) \cdot r e^{-r^2/2} \, dr. \tag{2.5}$$

We can upper bound (2.5) as follows:

$$\int_{\Delta_A}^{\infty} \mu_A(r) \cdot re^{-r^2/2}\,dr \leq \int_{\Delta_A}^{\infty} re^{-r^2/2}\,dr = e^{-\Delta_A^2/2}.$$

On the lower bound side, since $\mu_A(r)$ is non-decreasing as r increases, we get for every $\varepsilon > 0$:

$$\int_{\Delta_A}^{\infty} \mu_A(r)\cdot re^{-r^2/2}\,dr \geq \mu_A\Big((1+\varepsilon)\Delta_A\Big) \int_{(1+\varepsilon)\Delta_A}^{\infty} re^{-r^2/2}\,dr = \mu_A\Big((1+\varepsilon)\Delta_A\Big)e^{-(1+\varepsilon)^2\Delta_A^2/2}.$$

Thus, to get the desired bound, it is enough to show that as ε tends to zero, we have:

$$\mu_A\Big((1+\varepsilon)\Delta_A\Big) \geq \left(\frac{\varepsilon}{\Delta_A}\right)^{O(1)}.$$

This is a pretty straightforward computation. If we are in the situation of Figure 2-7b, then one can show that:

$$\begin{aligned}\mu\Big((1+\varepsilon)\Delta_A\Big) &= \frac{1}{2\pi}\cdot\left(\arccos\left(\frac{\rho}{(1+\varepsilon)\Delta_A}\right) - \arccos\left(\frac{\rho}{\Delta_A}\right)\right.\\ &\quad\left. + \arccos\left(\frac{\sigma}{(1+\varepsilon)\Delta_A}\right) - \arccos\left(\frac{\sigma}{\Delta_A}\right)\right)\\ &\geq \Omega\left(\frac{\varepsilon}{\Delta_A^2}\right)\end{aligned}$$

as required. Otherwise, the situation is like on Figure 2-7a, and we have $\mu\Big((1+\varepsilon)\Delta_A\Big) = \Omega(\varepsilon^{1/2})$, again, as required. □

2.8 Data-independent partitions

Here we fulfill the first step of the program outlined in Section 2.4. We show a simple algorithm based on data-independent partitioning that gives the trade-off (2.2) for worst-case instances of (c, r)-ANN on the unit sphere and a better trade-off (2.1) for random instances (as defined in Section 2.6). Later we will show how to achieve

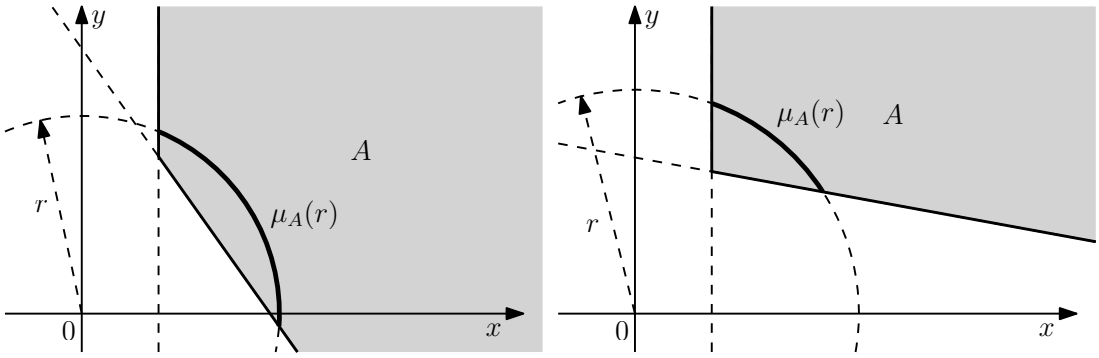


Figure 2-8: Illustration of the definition of $\mu_A(r)$.

the trade-off (2.1) for worst-case instances, which will imply Theorem 2.5.1 and, by Section 2.5, the final result (Theorem 2.3.1). See Figure 2-4 for the comparison of the trade-offs for $c = 2$.

2.8.1 Results

Below is the main theorem we prove in Section 2.8.

Theorem 2.8.1. *Let $\varepsilon_0 > 0$ be a (small) fixed positive constant. For every $c > 1$, $\frac{1}{\log\log n} \le r \le \sqrt{2} - \varepsilon_0$, $\rho_q \ge 0$ and $\rho_s \ge 0$ such that $cr \le 2 - \varepsilon_0$ and*

$$\big(1 - \alpha(r)\alpha(cr)\big) \cdot \sqrt{\rho_q} + \big(\alpha(r) - \alpha(cr)\big) \cdot \sqrt{\rho_s} \ge \beta(r)\beta(cr), \tag{2.6}$$

there exists a data structure for (c, r)-ANN on a unit sphere $S^{d-1} \subset \mathbb{R}^d$, where $d = n^{o(1)}$, with space $n^{1+\rho_s+o(1)}$ and query time $n^{\rho_q+o(1)}$.

Recall that $\alpha(\cdot)$ and $\beta(\cdot)$ are the functions defined in the beginning of Section 2.7. Let us see what this theorem gives for $r = o(1)$ and $r = \frac{\sqrt{2}}{c}$.

Corollary 2.8.2. *If $\frac{1}{\log\log n} \le r = o(1)$, then Theorem 2.8.1 holds as soon as:*

$$(c^2 + 1) \cdot \sqrt{\rho_q} + (c^2 - 1) \cdot \sqrt{\rho_s} \ge 2c. \tag{2.7}$$

Proof. We use the following estimates that are easy to check:

$$1 - \alpha(r)\alpha(cr) = \frac{(c^2+1)r^2}{2} + O_c(r^4)$$
$$\alpha(r) - \alpha(cr) = \frac{(c^2-1)r^2}{2}$$
$$\beta(r)\beta(cr) = cr^2 + O_c(r^3).$$

Plugging these estimates into (2.6), we obtain the result. □

Thus, using the algorithm from this section, one can immediately obtain a version of Theorem 2.5.1 with the trade-off (2.7) instead of the required (2.1).

Corollary 2.8.3. *If $r = \frac{\sqrt{2}}{c}$, then Theorem 2.8.1 holds as soon as:*

$$c^2 \cdot \sqrt{\rho_q} + (c^2 - 1) \cdot \sqrt{\rho_s} \geq \sqrt{2c^2 - 1}. \tag{2.8}$$

Proof. Since $c > 1$, one has $r \leq \sqrt{2} - \Omega(1)$, so we can apply Theorem 2.8.1. We use the following identities: $\alpha(r) = 1 - \frac{1}{c^2}$, $\alpha(cr) = \alpha(\sqrt{2}) = 0$, $\beta(r) = \sqrt{\frac{2}{c^2} - \frac{1}{c^4}}$, $\beta(cr) = \beta(\sqrt{2}) = 1$, which, as one can check, imply the required bound. □

This corollary implies that the algorithm from this section applied to random instances gives the trade-off (2.8), which we would like to get in full generality to conclude the proof of Theorem 2.5.1. The remainder of Section 2.8 is devoted to proving Theorem 2.8.1.

2.8.2 Description

Fix K and T to be positive integers, we determine their exact value later. Our data structure is a rooted tree where each node corresponds to a spherical cap. The tree consists of $K + 1$ levels of nodes where each node has out-degree at most T. We will index the levels by 0, 1, ..., K, where the 0-th level consists of the root denoted by v_0, and each node up to the $(K-1)$-th level has at most T children. Therefore, there are at most T^K nodes at the K-th level.

For every node v in the tree, let $\mathcal{L}_v$ be the set of nodes on the path from v to the root v_0 excluding the root (but including v). Each node v, except for the root, stores a random Gaussian vector $z_v \sim N(0,1)^d$. For each node v, we define the following subset of the dataset $P_v \subseteq P$:

$$P_v = \{p \in P \mid \forall v' \in \mathcal{L}_v \ \ \langle z_{v'}, p\rangle \geq \eta_s\},$$

where $\eta_s > 0$ is a parameter to be chosen later.

At the root node v_0, $P_{v_0} = P$, since $\mathcal{L}_{v_0} = \emptyset$. Intuitively, each set P_v corresponds to a subset of the dataset lying in the intersection of spherical caps centered around $z_{v'}$ for all $v' \in \mathcal{L}_v$. Every leaf ℓ at the level K stores the subset P_ℓ explicitly.

We build the tree recursively. For a given node v in levels $0, \ldots, K-1$, we first sample T i.i.d. Gaussian vectors $g_1, g_2, \ldots, g_T \sim N(0,1)^d$. Then, for every i such that $\{p \in P_v \mid \langle g_i, p\rangle \geq \eta_s\}$ is non-empty, we create a new child v' with $z_{v'} = g_i$ and recursively process v'. At the K-th level, each node v stores P_v as a list of points.

In order to answer a query $q \in S^{d-1}$, we start from the root v_0 and descend down the tree. We consider every child v of the root for which $\langle z_v, q\rangle \geq \eta_q$, where $\eta_q > 0$ is another parameter to be chosen later[4]. After identifying all such children, we proceed down the children recursively. If we reach leaf ℓ at level K, we scan through all the points in P_ℓ and compute their distance to the query q. If a point lies at a distance at most cr from the query, we return it and stop immediately.

We provide pseudocode for the data structure above in Figure 2-9. The procedure BUILD(P, 0, $\perp$) builds the data structure for dataset P and returns the root of the tree, v_0. The procedure QUERY(q, v_0) queries the data structure with root v_0 at point q.

[4]Note that η_s may not be equal to η_q. It is exactly this discrepancy that will govern the time–space trade-off.

```
function BUILD(P', l, z)
    create a tree node v
    store l as v.l
    store z as v.z
    if l = K then
        store P' as v.P
    else
        for i ← 1...T do
            sample a Gaussian vector z' ~ N(0,1)^d
            P'' ← {p ∈ P' | ⟨z', p⟩ ≥ η_s}
            if P'' ≠ ∅ then
                add BUILD(P'', l + 1, z') as a child of v
    return v
function QUERY(q, v)
    if v.l = K then
        for p ∈ v.P do
            if ‖p − q‖ ≤ cr then
                return p
    else
        for v' : v' is a child of v do
            if ⟨v'.z, q⟩ ≥ η_q then
                p ← QUERY(q, v')
                if p ≠⊥ then
                    return p
    return ⊥
```

Figure 2-9: Pseudocode for data-independent partitions

2.8.3 Analysis

In this section we analyze the above data structure. During the analysis, we will use functions $F(\cdot)$ and $G(\cdot, \cdot, \cdot)$ defined in Section 2.7.

Probability of success. We first analyze the probability of success of the data structure. We assume that a query q has some $p \in P$ where $\|p - q\|_2 \leq r$. The data structure succeeds when QUERY(q, v_0) returns some point $p' \in P$ with $\|q - p'\|_2 \leq cr$.

Lemma 2.8.4. *If*

$$T \geq \frac{3}{G\left(r, \eta_s, \eta_q\right)},$$

then with probability at least 0.9, QUERY(q, v_0) *finds some point within distance* cr

from q.

Proof. We prove the lemma by induction on the depth of the tree. Let $q \in S^{d-1}$ be a query point and $p \in P$ its near neighbor. Suppose we are within the recursive call QUERY(q, v) for some node v in the tree. Suppose we have not yet failed, that is, $p \in P_v$. We would like to prove that—if the condition of the lemma is met—the probability that this call returns *some* point within distance cr is at least 0.9.

When v is a node in the last level K, the algorithm enumerates P_v and, since we assume $p \in P_v$, some good point will be discovered (though not necessarily p itself). Therefore, this case is trivial. Now suppose that v is not from the K-th level. Using the inductive assumption, suppose that the statement of the lemma is true for all T potential children of v, i.e., if $p \in P_{v'}$, then with probability 0.9, QUERY(q, v') returns some point within distance cr from q. Then,

$$\begin{aligned}\Pr[\text{failure}] &\leq \prod_{i=1}^{T} \left(1 - \Pr_{g \sim N(0,1)^d}\left[\langle g, p\rangle \geq \eta_s \text{ and } \langle g, q\rangle \geq \eta_q\right] \cdot 0.9\right) \\ &\leq \left(1 - G\left(r, \eta_s, \eta_q\right) \cdot 0.9\right)^T \leq e^{-G(r,\eta_s,\eta_q)\cdot 0.9 \cdot T} \leq 0.1,\end{aligned}$$

where the first step follows from the inductive assumption and independence between the children of v during the preprocessing phase. The second step follows by monotonicity of $G(s, \rho, \sigma)$ in s, and the fourth is due to the assumption of the lemma. □

Space. We now analyze the space consumption of the data structure.

Lemma 2.8.5. *The expected space consumption of the data structure is at most*

$$n^{1+o(1)} \cdot K \cdot \left(TF(\eta_s)\right)^K.$$

Proof. We compute the expected total size of the sets P_ℓ for leaves ℓ at K-th level. There are at most T^K such nodes, and for a fixed point $p \in P$ and a fixed leaf ℓ the probability that $p \in P_\ell$ is equal to $F(\eta_s)^K$. Thus, the expected total size is at most $n \cdot \left(TF(\eta_s)\right)^K$. Since we only store a node v if P_v is non-empty, the number of nodes

stored is at most $K+1$ times the number of points stored at the leaves. The Gaussian vectors stored at each node require space $O(d)$, which is at most $n^{o(1)}$. □

Query time. Finally, we analyze the query time.

Lemma 2.8.6. *If $TF(\eta_q) \geq 3$, then the expected query time is at most*

$$n^{o(1)} \cdot T \cdot (TF(\eta_q))^K + n^{1+o(1)} \cdot (TG(cr, \eta_s, \eta_q))^K . \tag{2.9}$$

Proof. First, we compute the expected query time spent going down the tree, without scanning the leaves. The expected number of *nodes* the query procedure reaches is:

$$1 + TF(\eta_q) + (TF(\eta_q))^2 + \ldots + (TF(\eta_q))^K = O(1) \cdot (TF(\eta_q))^K ,$$

since we assume that $TF(\eta_q) \geq 3$. In each of node, we spend time $n^{o(1)} \cdot T$. The product of the two expressions gives the first term in (2.9).

The expected time spent scanning points in the leaves is at most $n^{o(1)}$ times the number of points scanned at the leaves reached. The number of points scanned is always at most one more than the number of *far* points, i.e., lying a distance greater than cr from q, that reached the same leaf. There are at most $n-1$ far points and T^K leaves. For each far point p' and each leaf ℓ the probability that both p' and q end up in P_ℓ is at most $G(cr, \eta_s, \eta_q)^K$. For each such pair, we spend time at most $n^{o(1)}$ processing the corresponding p'. This gives the second term in (2.9). □

2.8.4 Setting parameters: idealized version

We end the section by describing how to set parameters T, K, η_s and η_q to prove Theorem 2.8.1. We first show how we would do it under the following simplifying assumptions:

- Lemma 2.7.1 holds in the form $F(\rho) = e^{-\rho^2/2}$;
- Lemma 2.7.2 holds as $G(s, \rho, \sigma) = e^{-\Delta^2/2}$;

- There is no additional factor of T in the first term of (2.9): that is, the query time is $n^{o(1)} \cdot (TF(\eta_q))^K + n^{1+o(1)} \cdot (TG(cr, \eta_s, \eta_q))^K$.

After we will be done under these assumptions, we will argue that in reality we can achieve similar guarantees by essentially setting parameters to their "ideal" values. For the idealized setting, the value of K will not be too important, we will only need to have $K = o(\log n)$. For $\tau_q, \tau_s > 0$ (which we fix later), we choose η_q, η_s such that: $F(\eta_q)^K = n^{-\tau_q}$ and $F(\eta_s)^K = n^{-\tau_s}$. By the idealized version of Lemma 2.7.1 ($F(\rho) = e^{-\rho^2/2}$), this means the following:

$$\eta_s = \sqrt{\frac{2\tau_s \ln n}{K}}, \tag{2.10}$$

$$\eta_q = \sqrt{\frac{2\tau_q \ln n}{K}}. \tag{2.11}$$

We will set $\tau_s, \tau_q > 0$ such that:

$$\alpha(cr) \le \alpha(r) \le \frac{\min\{\eta_q, \eta_s\}}{\max\{\eta_q, \eta_s\}} = \frac{\min\left\{\sqrt{\tau_s}, \sqrt{\tau_q}\right\}}{\max\left\{\sqrt{\tau_s}, \sqrt{\tau_q}\right\}} \tag{2.12}$$

(the last step follows from (2.10) and (2.11)), so the idealized version of Lemma 2.7.2 ($G(s, \rho, \sigma) = e^{-\frac{\Delta^2}{2}}$) combined with (2.10) and (2.11) gives:

$$G(r, \eta_s, \eta_q)^K = e^{-\frac{\eta_s^2 + \eta_q^2 - 2\alpha(r)\cdot\eta_s\eta_q}{2\beta^2(r)}\cdot K} = n^{-\frac{\tau_s + \tau_q - 2\alpha(r)\cdot\sqrt{\tau_s\tau_q}}{\beta^2(r)}}, \tag{2.13}$$

$$G(cr, \eta_s, \eta_q)^K = e^{-\frac{\eta_s^2 + \eta_q^2 - 2\alpha(cr)\cdot\eta_s\eta_q}{2\beta^2(cr)}\cdot K} = n^{-\frac{\tau_s + \tau_q - 2\alpha(cr)\cdot\sqrt{\tau_s\tau_q}}{\beta^2(cr)}}. \tag{2.14}$$

Now let us check that we set the parameters such that the terms in (2.9) are approximately the same. Namely, we aim at satisfying the following:

$$F(\eta_q)^K = n \cdot G(cr, \eta_s, \eta_q)^K, \tag{2.15}$$

which by (2.14) and the relation between η_q and τ_q means that:

$$n^{-\tau_q} = n^{1 - \frac{\tau_s + \tau_q - 2\alpha(cr)\cdot\sqrt{\tau_s\tau_q}}{\beta^2(cr)}},$$

or, looking at the exponents:

$$-\tau_q = 1 - \frac{\tau_s + \tau_q - 2\alpha(cr) \cdot \sqrt{\tau_s \tau_q}}{\beta^2(cr)}.$$

Remembering that $1 - \beta^2(cr) = \alpha^2(cr)$, we get that, in order to balance the terms in the idealized version of (2.9) we need to have:

$$\left|\sqrt{\tau_s} - \alpha(cr)\sqrt{\tau_q}\right| = \beta(cr). \tag{2.16}$$

We set $T = \frac{3}{G(r,\eta_s,\eta_q)}$. By Lemma 2.8.4 it means that the probability of success of the data structure is at least 0.9. By (2.13) and $K = o(\log n)$, we get:

$$T^K = n^{\frac{\tau_s+\tau_q-2\alpha(r)\cdot\sqrt{\tau_s\tau_q}}{\beta^2(r)}+o(1)}. \tag{2.17}$$

Now let us look at space and query time. By Lemma 2.8.5, the space can be upper bounded as follows:

$$\begin{aligned} n^{1+o(1)} \cdot \big(TF(\eta_s)\big)^K &= n^{1+\frac{\tau_s+\tau_q-2\alpha(r)\cdot\sqrt{\tau_s\tau_q}}{\beta^2(r)}-\tau_s+o(1)} \\ &= n^{1+\frac{\left(\sqrt{\tau_q}-\alpha(r)\cdot\sqrt{\tau_s}\right)^2}{\beta^2(r)}+o(1)}, \end{aligned}$$

where the first step follows from (2.13), (2.17) and the relation between η_s and τ_s and the second step follows from $1 - \beta^2(r) = \alpha^2(r)$. Thus, we get the following identity for the space exponent:

$$\sqrt{\rho_s} = \frac{\left|\sqrt{\tau_q} - \alpha(r)\sqrt{\tau_s}\right|}{\beta(r)}. \tag{2.18}$$

Finally, let us look at the query time. By the assumed idealized version of the Lemma 2.8.6 (no factor T in the first term of (2.9)) and (2.15), we get the following upper bound on the query time:

$$\begin{aligned} n^{o(1)} \cdot \big(TF(\eta_q)\big)^K &= n^{\frac{\tau_s+\tau_q-2\alpha(r)\cdot\sqrt{\tau_s\tau_q}}{\beta^2(r)}-\tau_q+o(1)} \\ &= n^{1+\frac{\left(\sqrt{\tau_s}-\alpha(r)\cdot\sqrt{\tau_q}\right)^2}{\beta^2(r)}+o(1)}, \end{aligned}$$

where the first step follows from (2.13), (2.17) and the relation between η_q and τ_q and the second step follows from $1-\beta^2(r)=\alpha^2(r)$. Thus, we get the following identity for the query time exponent:

$$\sqrt{\rho_q}=\frac{|\sqrt{\tau_s}-\alpha(r)\sqrt{\tau_q}|}{\beta(r)}. \tag{2.19}$$

Now by varying positive $\tau_s,\tau_q>0$ such that (2.16) and (2.12) hold, we can vary the exponents ρ_s,ρ_q according to (2.18) and (2.19), respectively. The equation (2.12) restricts us to the range:

$$\frac{\sqrt{\tau_s}}{\sqrt{\tau_q}}\in\left[\alpha(r);\frac{1}{\alpha(r)}\right]. \tag{2.20}$$

Note that $\alpha(r)>0$, since $r\leq\sqrt{2}-\Omega(1)$. But for this range, (2.16), (2.18), and (2.19) turn into:

$$\begin{aligned}\sqrt{\tau_s}&=\alpha(cr)\cdot\sqrt{\tau_q}+\beta(cr),\\ \sqrt{\rho_s}&=\frac{\sqrt{\tau_q}-\alpha(r)\sqrt{\tau_s}}{\beta(r)},\\ \sqrt{\rho_q}&=\frac{\sqrt{\tau_s}-\alpha(r)\sqrt{\tau_q}}{\beta(r)},\end{aligned}$$

respectively. Thus, if we move $\sqrt{\rho_q}$ between 0 and $\frac{\beta(r)\beta(cr)}{1-\alpha(r)\alpha(cr)}>0$, we get:

$$\sqrt{\tau_s}=\frac{\alpha(r)\beta(cr)-\alpha(cr)\beta(r)\sqrt{\rho_q}}{\alpha(r)-\alpha(cr)}, \tag{2.21}$$

$$\sqrt{\tau_q}=\frac{\beta(cr)-\beta(r)\sqrt{\rho_q}}{\alpha(r)-\alpha(cr)}, \tag{2.22}$$

$$\sqrt{\rho_s}=\frac{\beta(r)\beta(cr)-(1-\alpha(r)\alpha(cr))\sqrt{\rho_q}}{\alpha(r)-\alpha(cr)}. \tag{2.23}$$

We need to show that for this range of $\sqrt{\rho_q}$ the constraint (2.20) holds and, moreover, the right-hand sides of (2.21) and (2.22) are positive. Let us start with the former. We have:

$$\frac{\sqrt{\tau_s}}{\sqrt{\tau_q}}=\frac{\alpha(r)\beta(cr)-\alpha(cr)\beta(r)\sqrt{\rho_q}}{\beta(cr)-\beta(r)\sqrt{\rho_q}}. \tag{2.24}$$

If we treat $\sqrt{\rho_q}$ as a formal variable, then the derivative of (2.24) with respect to it

is positive. At the boundary points, the ratio is equal to $\alpha(r)$ and $\frac{1}{\alpha(r)}$. Thus, (2.20) holds. Now let us check that the right-hand sides of (2.21) and (2.22) are indeed positive. They are linear functions of $\sqrt{\rho_q}$, so it is enough to check non-negativity for $\sqrt{\rho_q} \in \left\{0, \frac{\beta(r)\beta(cr)}{1-\alpha(r)\alpha(cr)}\right\}$. For $\sqrt{\rho_q} = 0$, we have:

$$\sqrt{\tau_s} = \frac{\alpha(r)\beta(cr)}{\alpha(r) - \alpha(cr)},$$
$$\sqrt{\tau_q} = \frac{\beta(cr)}{\alpha(r) - \alpha(cr)}.$$

For $\sqrt{\rho_q} = \frac{\beta(r)\beta(cr)}{1-\alpha(r)\alpha(cr)}$, we have:

$$\sqrt{\tau_s} = \frac{\beta(cr)}{1 - \alpha(r)\alpha(cr)},$$
$$\sqrt{\tau_q} = \frac{\alpha(r)\beta(cr)}{1 - \alpha(r)\alpha(cr)}.$$

All four values are positive, since $\alpha(r) > 0$ due to $r \leq \sqrt{2} - \Omega(1)$, and $\beta(cr) > 0$, since $0 < cr < 2$. The smallest of these values is:

$$\frac{\alpha(r)\beta(cr)}{1 - \alpha(r)\alpha(cr)},$$

since $\alpha(r) - \alpha(cr) \leq 1 - \alpha(r)\alpha(cr)$ and $\alpha(r) \leq 1$. The largest is:

$$\frac{\beta(cr)}{\alpha(r) - \alpha(cr)}.$$

Finally, (2.23) gives the desired trade-off (2.6). When $\sqrt{\rho_q}$ changes between 0 and $\frac{\beta(r)\beta(cr)}{1-\alpha(r)\alpha(cr)}$, $\sqrt{\rho_s}$ changes from $\frac{\beta(r)\beta(cr)}{\alpha(r)-\alpha(cr)}$ to 0 (and remains non-negative).

2.8.5 Setting parameters: final version

Here we get rid of the (unrealistic) assumptions we made before. Namely, we need to make sure that the identities (2.13) and (2.14) (approximately) hold, and that $T = n^{o(1)}$ if we set all the parameters appropriately. We start with fixing $K = \sqrt{\ln n}$. Suppose that the query exponent we would like to achieve is $0 \le \sqrt{\rho_q} \le \frac{\beta(r)\beta(cr)}{1-\alpha(r)\alpha(cr)}$. Let us choose $\sqrt{\tau_s}$ and $\sqrt{\tau_q}$ according to the formulas (2.21) and (2.22), respectively. From Section 2.8.4, we know that:

$$\frac{\alpha(r)\beta(cr)}{1-\alpha(r)\alpha(cr)} \le \sqrt{\tau_s}, \sqrt{\tau_q} \le \frac{\beta(cr)}{\alpha(r)-\alpha(cr)}. \tag{2.25}$$

The lower bound in (2.25) is at least $\Omega_c(\varepsilon_0^{3/2})$, since $r \le \sqrt{2} - \varepsilon_0$ and $cr \le 2 - \varepsilon_0$, while the upper bound is at most $O_c(\log\log n)$, since $r \ge \frac{1}{\log\log n}$. Thus, we have:

$$\Omega_c(\varepsilon_0^3) \le \tau_s, \tau_q \le O_c(\log^2 \log n). \tag{2.26}$$

Now we need to choose $\eta_s, \eta_q > 0$. Let us choose them as in the idealized scenario:

$$\eta_s = \sqrt{\frac{2\tau_s \ln n}{K}} = \sqrt{2\tau_s} \cdot \ln^{1/4} n, \tag{2.27}$$

$$\eta_q = \sqrt{\frac{2\tau_q \ln n}{K}} = \sqrt{2\tau_q} \cdot \ln^{1/4} n. \tag{2.28}$$

Now, we have by Lemma 2.7.1:

$$\begin{aligned} F(\eta_s)^K &= n^{-\tau_s} \cdot e^{\pm O_c(K \cdot \log\log n)} \\ &\in n^{-\tau_s} \cdot e^{\pm O_c(\sqrt{\log n} \cdot \log\log n)}, \end{aligned} \tag{2.29}$$

$$\begin{aligned} F(\eta_q)^K &= n^{-\tau_q} \cdot e^{\pm O_c(K \cdot \log\log n)} \\ &\in n^{-\tau_q} \cdot e^{\pm O_c(\sqrt{\log n} \cdot \log\log n)}, \end{aligned} \tag{2.30}$$

since $\tau_s, \tau_q = O_c(\log^2 \log n)$ by (2.26). Finally, as in the idealized scenario, we have:

$$\frac{\min\{\eta_s, \eta_q\}}{\max\{\eta_s, \eta_q\}} = \frac{\min\{\sqrt{\tau_s}, \sqrt{\tau_q}\}}{\max\{\sqrt{\tau_s}, \sqrt{\tau_q}\}} \ge \alpha(r) \ge \alpha(cr),$$

thus we can use Lemma 2.7.2 the same way as in Section 2.8.4. Namely, if we denote:

$$\Delta_r^2 = \frac{\eta_s^2 + \eta_q^2 - 2\alpha(r) \cdot \eta_s \eta_q}{\beta^2(r)},$$
$$\Delta_{cr}^2 = \frac{\eta_s^2 + \eta_q^2 - 2\alpha(cr) \cdot \eta_s \eta_q}{\beta^2(cr)},$$

we get:

$$G(r, \eta_s, \eta_q) = e^{-\frac{\Delta_r^2}{2} \pm O(\log \Delta_r)},$$
$$G(cr, \eta_s, \eta_q) = e^{-\frac{\Delta_{cr}^2}{2} \pm O(\log \Delta_{cr})}.$$

Since $\Delta_r^2, \Delta_{cr}^2 \leq O_c(\sqrt{\log n} \cdot \log^4 \log n)$, we have $\log \Delta_r, \log \Delta_{cr} \leq O_c(\log \log n)$. Thus,

$$G(r, \eta_s, \eta_q)^K = n^{-\frac{\tau_s + \tau_q - 2\alpha(r) \cdot \sqrt{\tau_s \tau_q}}{\beta^2(r)}} \cdot e^{\pm O_c(\sqrt{\log n} \cdot \log \log n)}, \tag{2.31}$$
$$G(cr, \eta_s, \eta_q)^K = n^{-\frac{\tau_s + \tau_q - 2\alpha(cr) \cdot \sqrt{\tau_s \tau_q}}{\beta^2(cr)}} \cdot e^{\pm O_c(\sqrt{\log n} \cdot \log \log n)}. \tag{2.32}$$

Thus, using (2.29), (2.30), (2.31) and (2.32), we can bound the query and the space exponents exactly the same way as in the idealized setting. The only thing remains to be checked is that $T = n^{o(1)}$. But this is immediate:

$$T = \frac{3}{G(r, \eta_s, \eta_q)} = O(1) \cdot e^{\frac{\Delta_r^2}{2} \pm O(\log \Delta_r)} \leq e^{O_c(\sqrt{\log n} \cdot \log^4 \log n)} \leq n^{o(1)}.$$

Finally, to prepare for the next section, let us state the relation between parameters we can use as a black box later.

Lemma 2.8.7. *If $c > 1$ and $\frac{1}{\log \log n} \leq r \leq \sqrt{2} - \Omega(1)$, then for every $\rho_s, \rho_q \geq 0$ such that (2.6) holds there exist $\eta_s, \eta_q > 0$ such that:*

- $F(\eta_s)/G(r, \eta_s, \eta_q) \leq n^{\left(\rho_s + o(1)\right)/K}$;
- $F(\eta_q)/G(r, \eta_s, \eta_q) \leq n^{\left(\rho_q + o(1)\right)/K}$;

- $G(cr, \eta_s, \eta_q)/G(r, \eta_s, \eta_q) \leq n^{\left(\rho_q - 1 + o(1)\right)/K}$;
- $1/G(r, \eta_s, \eta_q) \leq n^{o(1)}$,

where $K = \sqrt{\ln n}$.

2.9 Data-dependent partitions

In this section we prove Theorem 2.5.1, which we restate below:

Theorem 2.9.1. *For every* $c > 1$, $\rho_q \geq 0$ *and* $\rho_s \geq 0$ *such that*

$$c^2\sqrt{\rho_q} + \left(c^2 - 1\right)\sqrt{\rho_s} \geq \sqrt{2c^2 - 1}, \tag{2.33}$$

there exists a data structure for (c, r)*-ANN for the unit sphere* S^{d-1} *with* $r = \frac{1}{\sqrt{\log \log n}}$ *and* $d = \log^{1+o(1)} n$ *with space* $n^{1+\rho_s+o(1)}$ *and query time* $n^{\rho_q+o(1)}$.

2.9.1 Overview

We start with a high-level overview. Consider a dataset P_0 of n points. We partition P_0 into various components: s *dense* components, denoted by C_1, C_2, $\ldots$, C_s, and one *pseudo-random* component, denoted by $\widetilde{P}$. The partition is designed to satisfy the following properties. Each dense component C_i satisfies $|C_i| \geq \tau \cdot |P_0 \setminus (C_1 \cup C_2 \cup \ldots \cup C_{i-1})|$ and can be covered by a spherical cap of radius $\sqrt{2} - \varepsilon$ (see Figure 2-10). Here $\tau, \varepsilon > 0$ are small quantities to be chosen later. One should think of C_i's as clusters consisting of many points which are closer than random points would be. The pseudo-random component $\widetilde{P}$ consists of the remaining points without any dense clusters inside.

We process each C_i and $\widetilde{P}$ separately. We enclose every dense component C_i in a smaller ball E_i of radius $1 - \Omega(\varepsilon^2)$ (see Figure 2-10). For simplicity, we first ignore the fact that C_i does not necessarily lie on the boundary ∂E_i. Once we enclose each dense cluster with a smaller ball, we recurse on each resulting spherical instance of radius $1 - \Omega(\varepsilon^2)$. We treat the pseudo-random component $\widetilde{P}$ similarly to the

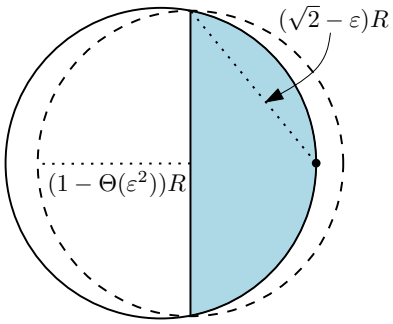


Figure 2-10: Covering a spherical cap of radius $(\sqrt{2}-\varepsilon)R$ on a ball of radius R.

random instance described in Section 2.6. Namely, we sample T Gaussian vectors $z_1, z_2, \ldots, z_T \sim N(0,1)^d$, and form T subsets of $\widetilde{P}$:

$$\widetilde{P}_i = \{p \in \widetilde{P} \mid \langle z_i, p\rangle \geq \eta_s R\},$$

where $\eta_s > 0$ is a parameter to be chosen later (for each pseudo-random remainder separately). Then, we recurse on each $\widetilde{P}_i$. Note that after we recurse, new dense clusters may appear in some $\widetilde{P}_i$ since it becomes easier to satisfy the minimum size constraint.

During the query procedure, we recursively query *each* C_i[5] with the query point q. For the pseudo-random component $\widetilde{P}$, we identify all i's such that $\langle z_i, q\rangle \geq \eta_q R$, and query all the corresponding children recursively. Here $\eta_q > 0$ is a new parameter that need to be chosen carefully (for each pseudo-random remainder separately).

Our algorithm makes progress in two ways. For dense clusters, we reduce the radius of the enclosing sphere by a factor of $(1-\Omega(\varepsilon^2))$. Since initially $r = \frac{1}{\sqrt{\log\log n}}$, in $O(\log\log\log n/\varepsilon^2)$ iterations we would arrive to the regime, where Corollary 2.8.3 begins to apply. For the pseudo-random component $\widetilde{P}$, most points will lie within a distance at least $\sqrt{2}-\varepsilon$ from each other. In particular, a typical inter-point distance is approximately $\sqrt{2}$, exactly like for the random case. For this reason, we call $\widetilde{P}$ pseudo-random. In this setting, the data structure from Section 2.8 performs well, at least for one step. However, after we recurse, we will need to extract dense clusters

[5]There will be at most $O\left(\frac{\log n}{\tau}\right) = n^{o(1)}$ of C_i's.

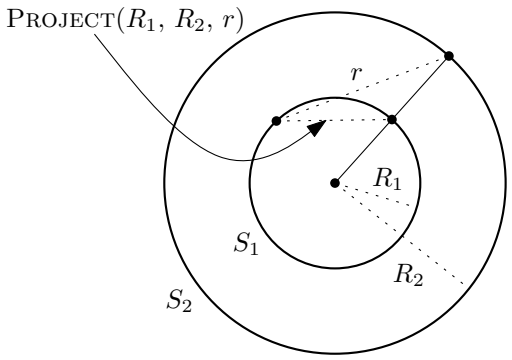


Figure 2-11: The definition of PROJECT

again.

We now address the issue deferred in the above high-level description: a dense component C_i does not generally lie on ∂E_i, but rather can occupy the interior of E_i. In this case, we partition E_i into thin annuli of carefully chosen width and treat each annulus as a sphere. This discretization adds to the complexity of the analysis, but does not seem to be too fundamental. However, now it becomes harder to argue about clusters making progress, since the value of r may decrease. As a result, we need quite a delicate potential function argument to reason about it. In particular, we need to make sure that r does not drop below $\frac{1}{\log\log n}$ in order for the techniques from Section 2.8 to be applicable.

2.9.2 Description

We are now ready to describe the data structure formally. It depends on the (small positive) parameters τ, ε, ε' and δ, an integer parameter $K \sim \sqrt{\ln n}$ and a large parameter $\gamma_{\text{stop}} > 1$. We also need to choose parameters T, $\eta_s > 0$, $\eta_q > 0$ for each pseudo-random remainder separately. Figure 2-13 provides pseudocode for the algorithm.

Preprocessing. Our preprocessing algorithm consists of the following functions:

- PROCESSSPHERE(P, r_1, r_2, o, R, l) builds the data structure for a dataset P lying on a sphere $\partial B(o, R)$, assuming we need to solve ANN with distance

thresholds r_1 and r_2. Moreover, we are guaranteed that queries will lie on the sphere $\partial B(o, R)$ as well. The parameter l counts the number of non-cluster nodes in the recursion stack we have encountered so far. Similarly to the data-independent algorithm from Section 2.8, we stop as soon as we encounter K of them. We start the preprocessing with calling ProcessSphere(P_0, r, cr, 0, 1, 0).

- ProcessBall(P, r_1, r_2, o, R, l) builds the data structure for a dataset P lying inside the ball $B(o, R)$, assuming we need to solve ANN with distance thresholds r_1 and r_2. Unlike ProcessSphere, here queries can be arbitrary. The parameter l has the same meaning as for ProcessSphere.

- Project(R_1, R_2, r) is an auxiliary function used when we discretize a ball into annuli. Suppose we have two spheres S_1 and S_2 with a common center and radii R_1 and R_2, respectively. Suppose there are points $p_1 \in S_1$ and $p_2 \in S_2$ with $\|p_1 - p_2\| = r$. Project(R_1, R_2, r) returns the distance between p_1 and the point $\widetilde{p_2}$ that lies on S_1 and is the closest to p_2 (see Figure 2-11). This is implemented by a formula.

We now elaborate on the above descriptions of ProcessSphere and ProcessBall.

ProcessSphere. We consider three base cases.

1. If $l = K$, we stop and store P explicitly. This corresponds to having reached a leaf in the algorithm from Section 2.8.

2. If $r_2 \geq 2R$, then we may only store one point, since any point in P is a valid answer to any query made on a sphere of radius R containing P.

3. If $r_2^2/r_1^2 \geq \gamma_{\text{stop}}$, then the instance can be handled by using an LSH-based data structure from [66]. We will set γ_{stop} to be super-constant, thus the data structure occupies space $|P| \cdot n^{o(1)}$, and allows query time $n^{o(1)}$.

If none of the above cases apply, we proceed by removing the dense components and then handling the pseudo-random remainder. The dense components C_i are clusters of at least $\tau|P \setminus (C_1 \cup C_2 \cup \ldots \cup C_{i-1})|$ points lying in a ball of radius $(\sqrt{2}-\varepsilon)R$ with its center on $\partial B(o,R)$. These balls can be enclosed by smaller balls of radius $\widetilde{R} \leq (1-\Omega(\varepsilon^2))R$ (with unrestricted centers). For each of these smaller balls, we invoke ProcessBall with the same l.

After we extract the dense components, we consider two cases. If $r_2 \geq (\sqrt{2}+\varepsilon') \cdot R$ (where $\varepsilon' = \Theta(\varepsilon)$), we are essentially done. Indeed, in this case for every query either an answer lies in one of the dense components, or $\Omega(1)$-fraction of the remainder can serve as a valid answer. We can check the latter (during the query stage) using uniform subsampling.

Finally, if $r_2 < (\sqrt{2}+\varepsilon') \cdot R$, then we build a single level of the tree from Section 2.8 for the remaining pseudo-random remainder. We pick the appropriate $\eta_s, \eta_q > 0$ and T and recurse on each part with ProcessSphere with l increased by 1.

ProcessBall. Similarly to ProcessSphere, if $r_1 + 2R \leq r_2$, then any point from $B(o,R)$ is a valid answer to any query in $B(o, R+r_2)$.

If we are not in the trivial setting above, we reduce the ball to the spherical case via a discretization of the ball $B(o,R)$ into thin annuli of width δr_1. First, we round all distances from points and queries to o to a multiple of δr_1. This rounding can change the distance between any pair of points by at most $2\delta r_1$ by the triangle inequality.

Then, we handle each non-empty annulus separately. In particular, for a fixed annulus at distance $\delta i r_1$ from o, a possible query can lie at most a distance $\delta j r_1$ from o, where $\delta r_1 |i-j| \leq r_1 + 2\delta r_1$. For each such case, we recursively build a data structure with ProcessSphere. However, when projecting points, the distance thresholds of r_1 and r_2 change, and this change is computed using Project.

Overall, the preprocessing creates a rooted tree. The root corresponds to the first ProcessSphere call, and subsequent nodes correspond to procedures ProcessSphere and ProcessBall. We refer to the tree nodes correspondingly, using the

labels in the description of the query algorithm from below.

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

- For a ProcessSphere node, we first handle the three base cases in a straightforward way. If $l = K$, then we just enumerate the whole P until we find a point within distance r_2. If $r_2 \geq 2R$, we simply return an arbitrary data point. If $r_2^2/r_1^2 \geq \gamma_{\text{stop}}$, then we query the data structure from [66] we build for this case and stop.

  If none of the base cases apply, we start with querying the data structures corresponding to the clusters recursively.

  Then, if $r_2 \geq (\sqrt{2}+\varepsilon')R$, we check $O(1)$ uniformly random points from P against the query point q. If $r_2 < (\sqrt{2}+\varepsilon')R$, we locate q in the spherical caps (with threshold η_q, like in Section 2.8), and query data structures we built for the corresponding subsets of P. If one of these calls finds a near neighbor, we return.

- In ProcessBall, we first consider the base case, where we just return the stored point if it is close enough. In general, we check whether $\|q-o\|_2 \leq R+r_1$. If not, we return with no neighbor, since each dataset point lies within a ball of radius R from o, but the query point is at least $R + r_1$ away from o. If $\|q - o\|_2 \leq R + r_1$, we round q so the distance from o to q is a multiple of δr_1 and enumerate all possible distances from o to the potential near neighbor we are looking for. For each possible distance, we query the corresponding ProcessSphere children after projecting q on the sphere with a tentative near neighbor using Project.

2.9.3 Setting parameters

We complete the description of the data structure by setting the remaining parameters. Recall that the dimension is $d = \log^{1+o(1)} n$ and $K \sim \sqrt{\ln n}$. We set $\varepsilon, \varepsilon', \delta, \tau, \gamma_{\text{stop}}$ as

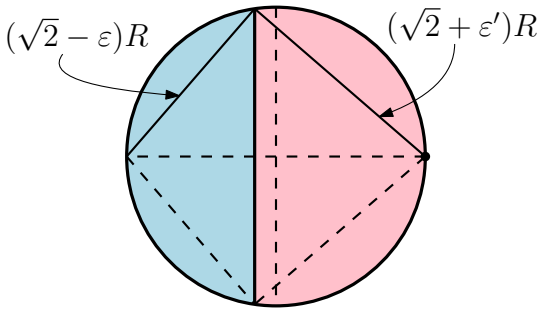


Figure 2-12: The definition of $\varepsilon' > 0$.

follows:

- $\varepsilon = \frac{1}{(\log\log\log n)^{0.01}}$;
- $\varepsilon' = \Theta(\varepsilon)$ according to Figure 2-12;
- $\delta = \frac{1}{\log n}$;
- $\tau = \frac{1}{\exp\left(\log^{2/3} n\right)}$;
- $\gamma_{\text{stop}} = (\log\log\log n)^{0.01}$.

Now we specify how to set $\eta_s, \eta_q > 0$ and T for each pseudo-random remainder. The idea will be to try to replicate the parameter settings of Section 2.8.4 corresponding to the random instance. Since we remove all the dense components, we can use $(\sqrt{2}-\varepsilon)R$ as an "effective r_2." In particular, we let

$$T = \frac{3}{G(\sqrt{2}-\varepsilon, \eta_s, \eta_q)}$$

in order to achieve a high probability of success. Then we let η_s and η_q such that

- $F(\eta_s)/G(r_1/R, \eta_s, \eta_q) \le n^{(\rho_s + o_c(1))/K}$;
- $F(\eta_q)/G(r_1/R, \eta_s, \eta_q) \le n^{(\rho_q + o_c(1))/K}$;
- $G(\sqrt{2}-\varepsilon, \eta_s, \eta_q)/G(r_1/R, \eta_s, \eta_q) \le n^{(\rho_q - 1 + o_c(1))/K}$;
- $1/G(r_1/R, \eta_s, \eta_q) \le n^{o_c(1)}$.

which correspond to the parameter settings achieving the trade-off of Section 2.8.3. It is not immediate why we can accomplish this, we will show this later in Lemma 2.9.7 (namely, we will need to show that $\frac{1}{\log\log n} \leq \frac{r_1}{R} \leq (1 + o_c(1)) \cdot \frac{\sqrt{2}}{c}$, and then we simply invoke Lemma 2.8.7).

A crucial relation between the parameters is that τ should be much smaller than $G(\sqrt{2} - \varepsilon, \eta_s, \eta_q)$. This implies that the "effective r_2" is equal to $(\sqrt{2} - \varepsilon)R$, at least for the sake of a single step of the random partition.

We collect some basic facts from the data structure which will be useful for the analysis. These facts follow trivially from the pseudocode in Figure 2-13.

- The children to ProcessSphere may contain at most $O\left(\frac{\log n}{\tau}\right)$ many calls to ProcessBall, corresponding to cluster nodes, and T calls to ProcessSphere. Each ProcessBall call of ProcessSphere handles a disjoint subset of the dataset. Points can be replicated in the pseudo-random remainder, when a point lies in the intersection of two or more caps.

- A ProcessBall node has many children, all of which are ProcessSphere which do not increment l. Each of these children corresponds to a call for a specific annulus of width δr_1 as well as a possible distance for a query. For each annulus, there are at most $\frac{2}{\delta} + 4 = O\left(\frac{1}{\delta}\right)$ "notable distances": after rounding by δr_1, a valid query can be at most $r_1 + 2\delta r_1$ away from a particular annulus in both directions, thus, each point gets duplicated at most $O\left(\frac{1}{\delta}\right) = O(\log n)$ many times.

- For each possible point $p \in P$, we may consider the subtree of nodes which process that particular point. We make the distinction between two kinds of calls to ProcessSphere: calls where p lies in a dense cluster, and calls where p lies in a pseudo-random remainder. If p lies in a dense cluster, l is not incremented; if p lies in the pseudo-random remainder, l is incremented. The point p may be processed by various rounds of calls to ProcessBall and ProcessSphere without incrementing l; however, there will be a moment when p is not in a dense cluster and will be part of the pseudo-random remainder. In that setting,

p will be processed by a call to ProcessSphere which increments l.

2.9.4 Analysis

In this section we will analyze the above data structure. We start with a somewhat technical part.

Lemma 2.9.2. *Suppose we are within a call to* ProcessSphere *and none of the three base cases took place. Then:*

- $\frac{r_2}{r_1} \geq \big(1 - o_c(1)\big) \cdot c$;
- $\frac{r_1}{R} \geq \frac{1}{\log\log n}$;
- *The number of calls to* ProcessBall *in the recursion stack is at most*

$$O_c\left((\log\log\log n)^2\right).$$

The values of r_1, r_2 and R can only change when we call ProcessBall (from which ProcessSphere is called): namely, R changes when we call ProcessBall, and then all three parameters change during a subsequent ProcessSphere call.

Let us outline the proof of Lemma 2.9.2. Suppose that we are within a call to ProcessSphere such that none of the three base cases took place. Consider the sequence of the triples $(r_1^{(i)}, r_2^{(i)}, R^{(i)})$ as they evolved from the root of the tree to the current node (the total number of the triples is equal to the number of the ProcessBall calls in the recursion stack). For this sequence, we will consider a new "idealized" sequence $(\widetilde{r_1}^{(i)}, \widetilde{r_2}^{(i)}, R^{(i)})$ defined as follows. We rerun the same branch of the recursion, but in ProcessBall we do not add δr_1 to r_1 and do not subtract δr_1 from r_2 when computing new thresholds (but we use the same values of i and j). Note that at some point the idealized sequence may stop being well-defined, but the actual sequence still is: for instance, when in the idealized sequence $r_1 < \delta|j-i|r_1 \leq (1+2\delta)r_1$. Now the proof of Lemma 2.9.2 consists of two steps. First, we show it for the idealized sequence by keeping track of the following quantities: $\gamma = \frac{r_2^2}{r_1^2}$ and $\xi = \frac{r_2^2}{R^2}$. Then, we

function ProcessSphere(P, r_1, r_2, o, R, l)
  if $l = K$ **then**
    store P explicitly
    return
  if $r_2 \geq 2R$ **then**
    store any point from P
    return
  if $r_2^2/r_1^2 \geq \gamma_{\text{stop}}$ **then**
    build a data structure for P from [66]
    return
  $\widehat{R} \leftarrow (\sqrt{2} - \varepsilon)R$
  while $\exists\, x \in \partial B(o, R) : |B(x, \widehat{R}) \cap P| \geq \tau|P|$ **do**
    $B(\widetilde{o}, \widetilde{R}) \leftarrow$ the SEB for $P \cap B(x, \widehat{R})$
    ProcessBall($P \cap B(x, \widehat{R})$, r_1, r_2, $\widetilde{o}$, $\widetilde{R}$, l)
    $P \leftarrow P \setminus B(x, \widehat{R})$
  if $r_2 \geq (\sqrt{2} + \varepsilon')R$ **then**
    return ▷ We will compare queries against a (small) uniform subsample of P.
  choose η_s and η_q such that:

- $F(\eta_s)/G(r_1/R, \eta_s, \eta_q) \leq n^{(\rho_s + o(1))/K}$;
- $F(\eta_q)/G(r_1/R, \eta_s, \eta_q) \leq n^{(\rho_q + o(1))/K}$;
- $G(\sqrt{2} - \varepsilon, \eta_s, \eta_q)/G(r_1/R, \eta_s, \eta_q) \leq n^{(\rho_q - 1 + o(1))/K}$;
- $1/G(r_1/R, \eta_s, \eta_q) \leq n^{o(1)}$.

  $T \leftarrow 3/G(r_1/R, \eta_s, \eta_q)$
  for $i \leftarrow 1 \dots T$ **do**
    sample $z \sim N(0, 1)^d$
    $P' \leftarrow \{p \in P \mid \langle z, p \rangle \geq \eta_s R\}$
    if $P' \neq \emptyset$ **then**
      ProcessSphere(P', r_1, r_2, o, R, $l + 1$)
function ProcessBall(P, r_1, r_2, o, R, l)
  if $r_1 + 2R \leq r_2$ **then**
    store any point from P
    return
  $P \leftarrow \{o + \delta r_1 \lceil \frac{\|p-o\|}{\delta r_1} \rceil \cdot \frac{p-o}{\|p-o\|} \mid p \in P\}$
  for $i \leftarrow 1 \dots \lceil \frac{R}{\delta r_1} \rceil$ **do**
    $\widetilde{P} \leftarrow \{p \in P \colon \|p - o\| = \delta i r_1\}$
    if $\widetilde{P} \neq \emptyset$ **then**
      for $j \leftarrow 1 \dots \lceil \frac{R + r_1 + 2\delta r_1}{\delta r_1} \rceil$ **do**
        if $\delta|i - j| \leq r_1 + 2\delta r_1$ **then**
          $\widetilde{r_1} \leftarrow$ Project($\delta i r_1$, $\delta j r_1$, $r_1 + 2\delta r_1$)
          $\widetilde{r_2} \leftarrow$ Project($\delta i r_1$, $\delta j r_1$, $r_2 - 2\delta r_1$)
          ProcessSphere($\widetilde{P}$, $\widetilde{r_1}$, $\widetilde{r_2}$, o, $\delta i r_1$, l)
function Project(R_1, R_2, r)
  return $\sqrt{R_1(r^2 - (R_1 - R_2)^2)/R_2}$

Figure 2-13: Pseudocode of the data structure (SEB stands for *smallest enclosing ball*)

show that the actual sequence can not diverge too much from the idealized one (or stop being well-defined) unless one of the base cases in ProcessSphere is triggered for the actual sequence.

Let us start with the idealized sequence. First of all, the initial values of γ and ξ are c^2 and $\frac{c^2}{\log\log n}$, respectively. Second, the largest γ and ξ can be are γ_{stop} and 4, respectively, since otherwise base cases in ProcessSphere are triggered. Now supposed we call ProcessBall followed by ProcessSphere for some values i and j. Suppose that γ and ξ evolved into $\widetilde{\gamma}$ and $\widetilde{\xi}$, respectively. We have the following claims which immediately follow from the formula for Project.

Claim 2.9.3.

$$\widetilde{\gamma} = \frac{r_2^2 - \delta^2(i-j)^2 r_1^2}{r_1^2 - \delta^2(i-j)^2 r_1^2}.$$

Claim 2.9.4.

$$\widetilde{\xi} = \frac{r_2^2 - \delta^2(i-j)^2 r_1^2}{\delta^2 ij r_1^2}.$$

We observe that γ never decreases. On the other hand, ξ can decrease but not drastically.

Claim 2.9.5.

$$\frac{\widetilde{\xi}}{\xi} \geq \Omega_c(1).$$

Proof. We have: $\delta i r_1 \leq (1 - \Omega(\varepsilon^2)) \cdot R + \delta r_1 \leq O(R)$, since $r_1 \leq O(R)$ and $\delta = O(1)$. Besides that, $\delta j r_1 \leq \delta i r_1 + r_1 \leq O(R)$. Thus, by Claim 2.9.4, since γ is non-decreasing and starts with c^2, and since $\delta^2(i-j)^2 \leq 1$,

$$\frac{\widetilde{\xi}}{\xi} = \frac{(r_2^2 - \delta^2(i-j)^2 r_1^2)/\delta^2 ij r_1^2}{r_2^2/R^2} \geq \frac{(r_2^2 - r_1^2)/O(R^2)}{r_2^2/R^2} \geq \Omega(c^2 - 1) \geq \Omega_c(1).$$

□

The following claim is the key to the overall analysis for the idealized case.

Claim 2.9.6. *If* $\frac{\widetilde{\xi}}{\xi} \leq 1 + \Theta(\varepsilon^2)$, *then* $\frac{\widetilde{\gamma}}{\gamma} \geq 1 + \Omega_c(\varepsilon^4)$.

Proof. By Claim 2.9.4 one has:

$$\frac{\widetilde{\xi}}{\xi} = \frac{(r_2^2 - \delta^2(i-j)^2 r_1^2)\, R^2}{\delta^2 ijr_1^2 r_2^2}.$$

We have $\delta i r_1 \le (1 - \Omega(\varepsilon^2) + \delta)\, R$ and

$$\delta j r_1 \le \delta i r_1 + \delta |i-j| r_1 \le \left(1 - \Omega(\varepsilon^2) + \delta\right) R + \delta |i-j| r_1.$$

Thus,

$$\frac{\widetilde{\xi}}{\xi} \ge \frac{1 - \frac{\delta^2 (i-j)^2 r_1^2}{r_2^2}}{(1 - \Omega(\varepsilon^2) + \delta)\left(1 - \Omega(\varepsilon^2) + \delta + \frac{\delta |i-j| r_1}{R}\right)}.$$

Since $\delta \ll \varepsilon^2$, $r_2 \ge r_1$, and $R \ge \Omega(r_1)$, we have that $\frac{\widetilde{\xi}}{\xi} \le 1 + \Theta(\varepsilon^2)$ implies

$$\delta |i-j| = \Omega_c(\varepsilon^2).$$

But this implies by Claim 2.9.3:

$$\frac{\widetilde{\gamma}}{\gamma} = \frac{1 - \delta^2 (i-j)^2 \cdot \frac{r_1^2}{r_2^2}}{1 - \delta^2 (i-j)^2} \ge 1 + \Omega_c(\varepsilon^4),$$

since $\frac{r_1}{r_2} \le \frac{1}{c} < 1$. □

Thus, the number of times ξ increases by at most $1 + \Theta(\varepsilon^2)$ is at most the number of times γ increases by at least $1 + \Omega_c(\varepsilon^4)$. Since γ is non-decreasing, starts at c^2 and grows until γ_{stop}, the number of such steps is at most $O_c\left(\frac{\log \gamma_{\text{stop}}}{\varepsilon^4}\right)$. Since ξ decreases by at most $O_c(1)$ each step, in the worst case it can go down to:

$$\frac{c^2}{\log\log n} \cdot 2^{-O_c\left(\frac{\log \gamma_{\text{stop}}}{\varepsilon^4}\right)} \ge \frac{1}{\log\log n} \cdot 2^{-O_c\left((\log\log\log n)^{0.05}\right)} \gg \frac{\gamma_{\text{stop}}}{\log^2 \log n}.$$

Thus,

$$\frac{r_1^2}{R^2} \ge \frac{\xi}{\gamma_{\text{stop}}} \ge \frac{1}{\log^2 \log n}.$$

Since during the remaining steps, ξ increases by a factor at least $1 + \Omega(\varepsilon^2)$, and can

be at most 4, the number of the remaining iterations is at most:

$$O_c\left(\frac{1}{\varepsilon^2}\cdot\left(\frac{1}{\varepsilon^4}+\log\gamma_{\text{stop}}+\log\log\log n\right)\right)\ll O_c\left((\log\log\log n)^2\right).$$

Thus, we proved Lemma 2.9.2 for the *idealized sequence*. Let us now argue about the actual sequence. In order to do this, we will consider both sequences side by side. Namely, let (r_1, r_2, R) be an element of the idealized sequence, and let $(\widetilde{r_1}, \widetilde{r_2}, R)$ be the corresponding element of the real sequence. Let us assume that we have already shown that:

- $\widetilde{r_1} \in (1 \pm \tau_1)\cdot r_1$, where $\tau_1 = o(1)$;
- $\widetilde{r_2} \in (1 \pm \tau_2)\cdot r_2$, where $\tau_2 = o(1)$;
- In particular, $\frac{\widetilde{r_2}}{\widetilde{r_1}} \geq \big(1 - o(1)\big)\cdot c$.

Let us see how these conditions evolve during one step. Let us show that the new gap between $\widetilde{r_1}$ and r_1 is $\big(1\pm\tau_2\big)\big(1\pm O(\sqrt{\delta})\big)$ unless a base case within ProcesssSphere is triggered. One of the two undesired things can happen:

- First, it can be the case that $\delta|i-j|>1$. In this case, the next step for the idealized sequence is not defined.
- One has

$$\frac{\textsc{Project}(\delta i r_1, \delta j r_1, r_1)^2}{\textsc{Project}(\delta i\widetilde{r_1}, \delta j\widetilde{r_1}, \widetilde{r_1}+2\delta\widetilde{r_1})^2}\notin\left(1\pm\Theta(\sqrt{\delta})\right)\cdot\frac{r_1^2}{\widetilde{r_1}^2}. \tag{2.34}$$

  In this case the idealized and the real r_1's diverge too much.

We will show that in both of these cases, for the real sequence, during the next iteration a base case will be triggered. First, we will show that in both cases, $\delta|i-j|\geq\left(1-O\left(\sqrt{\delta}\right)\right)$. For the first case, this is immediate. Let us handle the case of (2.34) now:

$$\frac{\textsc{Project}(\delta i r_1, \delta j r_1, r_1)^2}{\textsc{Project}(\delta i\widetilde{r_1}, \delta j\widetilde{r_1}, \widetilde{r_1}+2\delta\widetilde{r_1})^2}=\frac{r_1^2-\delta^2(i-j)^2r_1^2}{\widetilde{r_1}^2(1+2\delta)^2-\delta^2(i-j)^2\widetilde{r_1}^2}\notin\left(1\pm\Theta(\sqrt{\delta})\right)\cdot\frac{r_1^2}{\widetilde{r_1}^2}$$

iff

$$\frac{1-\delta^2(i-j)^2}{(1+2\delta)^2-\delta^2(i-j)^2} \notin \left(1 \pm \Theta(\sqrt{\delta})\right)$$

iff

$$\delta^2(i-j)^2 \geq 1 - O\left(\sqrt{\delta}\right)$$

iff

$$\delta|i-j| \geq 1 - O\left(\sqrt{\delta}\right).$$

But if $\delta|i-j| \geq 1 - O\left(\sqrt{\delta}\right)$, then at the next iteration the new value of γ is:

$$\begin{aligned}\frac{\textsc{Project}(\delta i\widetilde{r_1}, \delta j\widetilde{r_1}, \widetilde{r_2} - 2\delta\widetilde{r_1})^2}{\textsc{Project}(\delta i\widetilde{r_1}, \delta j\widetilde{r_1}, \widetilde{r_1} + 2\delta\widetilde{r_1})^2} &= \frac{(\widetilde{r_2} - 2\delta\widetilde{r_1})^2 - \delta^2(i-j)^2\widetilde{r_1}^2}{(\widetilde{r_1} + 2\delta\widetilde{r_1})^2 - \delta^2(i-j)^2\widetilde{r_1}^2} \\ &= \Omega_c\left(\frac{1}{\sqrt{\delta}}\right) = \Omega_c(\sqrt{\log n}) \gg \gamma_{\text{stop}},\end{aligned}$$

where we use that $\frac{\widetilde{r_2}^2}{\widetilde{r_1}^2} \geq (1 - o(1))c$.

Overall, the gap between r_1 and $\widetilde{r_1}$ grows by at most a factor of $\left(1 \pm O(\sqrt{\delta})\right)$.

Now let us handle the relation between r_2 and $\widetilde{r_2}$.

$$\frac{\textsc{Project}(\delta i r_1, \delta j r_1, r_2)^2}{\textsc{Project}(\delta i\widetilde{r_1}, \delta j\widetilde{r_1}, \widetilde{r_2} - 2\delta\widetilde{r_1})^2} = \frac{r_2^2 - \delta^2(i-j)^2 r_1^2}{(\widetilde{r_2} - 2\delta\widetilde{r_1})^2 - \delta^2(i-j)^2\widetilde{r_1}^2} \in 1 \pm O_c(\tau_2 + \delta).$$

Thus, after all the $O_c(\log\log\log n)$ iterations we will have:

$$\frac{r_1}{\widetilde{r_1}} = \left(1 + O(\sqrt{\delta})\right)^{O_c(\log\log\log n)} = 1 + o_c(1) \tag{2.35}$$

and

$$\frac{r_2}{\widetilde{r_2}} = 1 + \delta \cdot 2^{O_c(\log\log\log n)} = 1 + o_c(1). \tag{2.36}$$

Thus,

$$\frac{\widetilde{r_2}}{\widetilde{r_1}} \geq (1 - o_c(1)) \cdot c$$

as required. Other desired statements follow from their counterparts for the idealized sequence, (2.35) and (2.36).

Lemma 2.9.7. *During the algorithm we are always able to choose η_s and η_q such*

that:

- $F(\eta_s)/G(r_1/R, \eta_s, \eta_q) \le n^{(\rho_s + o_c(1))/K}$;
- $F(\eta_q)/G(r_1/R, \eta_s, \eta_q) \le n^{(\rho_q + o_c(1))/K}$;
- $G(\sqrt{2} - \varepsilon, \eta_s, \eta_q)/G(r_1/R, \eta_s, \eta_q) \le n^{(\rho_q - 1 + o_c(1))/K}$;
- $1/G(r_1/R, \eta_s, \eta_q) = n^{o_c(1)}$.

Proof. If we need to choose η_s and η_q, then $r_2 \le (\sqrt{2}+\varepsilon')R$. Since by Lemma 2.9.2, $\frac{r_2}{r_1} \ge (1 - o_c(1))c$, we have $r_1 \le \frac{(1+o_c(1))(\sqrt{2}+\varepsilon')R}{c}$ (in particular, it is at most $(\sqrt{2} - \Omega_c(1))R$). Besides, by Lemma 2.9.2, we have $r_1 \ge \frac{R}{\log\log n}$. This allows us to choose the required η_s, η_q by using Lemma 2.8.7, since $\varepsilon, \varepsilon' = o(1)$. □

Lemma 2.9.8. *The probability of success of the data structure is at least* 0.9.

Proof. There are three places the preprocessing and query procedures are not deterministic.

First, when we build a data structure from [66] (in case $r_2^2/r_1^2 \ge \gamma_{\text{stop}}$). In this case the probability of success is at least 0.9 by construction.

Second, when in ProcessSphere we have $r_2 \ge (\sqrt{2}+\varepsilon')R$, and, after querying the dense components, we check the query against a uniform subsample of the remainder. Let us show that if a near neighbor is in the reminder, we succeed with high probability. Indeed, if in this case we have more than τ-fraction of the remainder further than $(\sqrt{2} + \varepsilon')R$ from the query, it means that there is a dense component (see Figure 2-12), which contradicts the definition of a remainder. Thus, a $(1 - \tau)$-fraction of the remainder can serve as a valid answer to the query, which means that uniform sampling finds it with probability at least $1 - \tau > 0.9$.

Third, when we perform a single step of the algorithm from Section 2.8. In this case, we can repeat the proof of Lemma 2.8.4 and conclude that we succeed with probability at least 0.9, since $T = 3/G(r_1/R, \eta_s, \eta_q)$. □

Lemma 2.9.9. *The total space the data structure occupies is at most* $n^{1+\rho_s+o(1)}$ *in expectation.*

Proof. We will prove that the total number of explicitly stored points (when $l = K$) is at most $n^{1+\rho_s+o(1)}$. We will count the contribution from each point separately, and use linearity of expectation to sum up the contributions. In particular, for a point $p \in P_0$, we want to count the number of lists where p appears in the data structure. Each root to leaf path of the tree has at most K calls to ProcessSphere which increment l, and at most $O_c((\log\log\log n)^2)$ calls to ProcessBall, and thus $O_c((\log\log\log n)^2)$ calls to ProcessSphere which do not increment l. Thus, once we count the number of lists, we may multiply by $K + O_c((\log\log\log n)^2) = n^{o(1)}$ to count the size of the whole tree.

For each point, we will consider the subtree of the data structure where the point was processed. In particular, we may consider the tree corresponding to calls to ProcessSphere and ProcessBall which process p. As discussed briefly in Section 2.9.3, we distinguish between calls to ProcessSphere which contain p in a dense cluster, and calls to ProcessSphere which contain p in the pseudo-random remainder. We increment l only when p lies in the pseudo-random remainder.

Claim 2.9.10. *It suffices to consider the data structure where each node is a function call to* ProcessSphere *which increments l, i.e., when p lies in the pseudo-random remainder, since the total amount of duplication of points corresponding to other nodes is $n^{o(1)}$.*

We will account for the duplication of points in calls to ProcessBall and calls to ProcessSphere which do not increment l. Consider the first node v in a path from the root which does not increment l, this corresponds to a call to ProcessSphere which had p in some dense cluster. Consider the subtree consisting of descendants of v where the leaves correspond to the first occurrence of ProcessSphere which increments l. We claim that every internal node of the tree corresponds to alternating calls to ProcessBall and ProcessSphere which do not increment l. Note that calls to ProcessSphere which do not increment l never replicate p. Each call to ProcessBall replicates p in $b := O(1/\delta) = O(\log n)$ many times. We may consider contracting the tree and at edge, multiplying by the number of times we encounter

ProcessBall.

Note that p lies in a dense cluster if and only if it does not lie in the pseudo-random remainder. Thus, our contracted tree looks like a tree of K levels, each corresponding to a call to ProcessSphere which contained p in the pseudo-random remainder.

The number of children of some nodes may be different; however, the number of times ProcessBall is called in each branch of computation is

$$U := O_c\left((\log\log\log n)^2\right),$$

the total amount of duplication of points due to ProcessBall is at most $b^U = n^{o(1)}$. Now, the subtree of nodes processing p contains K levels with each T children, exactly like the data structure for Section 2.8.

Claim 2.9.11. *A node v corresponding to* ProcessSphere(P, r_1, r_2, o, R, l) *has, in expectation, p appearing in $n^{((K-l)\rho_s+o_c(1))/K}$ many lists in the subtree of v.*

The proof is an induction over the value of l in a particular node. For our base case, consider some node v corresponding to a function call of ProcessSphere which is a leaf, so $l = K$, in this case, each point is only stored at most once, so the claim holds.

Suppose for the inductive assumption the claim holds for some l, then for a particular node at level $l-1$, consider the point when p was part of the pseudo-random remainder. In this case, p is duplicated in

$$T \cdot F(\eta_s) = \frac{3 \cdot F(\eta_s)}{G(r_1/R, \eta_s, \eta_q)} \leq n^{(\rho_s+o_c(1))/K}$$

many children, and in each child, the point appears $n^{((K-l)\rho_s+o(1))/K}$ many times. Therefore, in a node v, p appears in $n^{((K-l+1)\rho_s+o(1))/K}$ many list in its subtree. Letting $l = 0$ for the root gives the desired outcome. □

Lemma 2.9.12. *The expected query time is at most $n^{\rho_q+o(1)}$.*

Proof. We need to bound the expected number of nodes we traverse as well as the number of points we enumerate for nodes with $l = K$.

We first bound the number of nodes we traverse. Let $A(u, l)$ be an upper bound on the expected number of visited nodes when we start in a ProcessSphere node such that there are u ProcessBall nodes in the stack and l non-cluster nodes. By Lemma 2.9.2,

$$u \le U := O_c\left((\log\log\log n)^2\right),$$

and from the description of the algorithm, we have $l \le K$. We will prove $A(0,0) \le n^{\rho_q + o_c(1)}$, which corresponds to the expected number of nodes we touch starting from the root.

We claim

$$A(u,l) \le \exp(\log^{2/3+o(1)} n) \cdot A(u+1,l) + n^{(\rho_q+o_c(1))/K} \cdot A(u,l+1). \tag{2.37}$$

There are at most $O(\log n/\tau) = \exp(\log^{2/3+o(1)} n)$ cluster nodes, and in each node, we recurse on $O(1/\delta) = \exp(\log^{o(1)} n)$ possible annuli with calls to ProcessSphere nodes where u increased by 1 and l remains the same. On the other hand, there are

$$T \cdot F(\eta_q) = \frac{3 \cdot F(\eta_q)}{G(r_1/R, \eta_s, \eta_q)} \le n^{(\rho_q+o_c(1))/K}$$

caps, where the query falls, in expectation. Each calls ProcessSphere where u remains the same and l increased by 1.

Solving (2.37):

$$A(0,0) \le \binom{U+K}{K} \exp(U \cdot \log^{2/3+o(1)} n) \cdot n^{\rho_q+o_c(1)} \le n^{\rho_q+o_c(1)}.$$

We now give an upper bound on the number of points the query algorithm will test at level K. Let $B(u, l)$ be an upper bound on the expected fraction of the dataset in the current node that the query algorithm will eventually test at level K (where we count multiplicities). The variables u and l have the same meaning as discussed above.

We claim

$$B(u,l) \leq O\left(\frac{\log n}{\tau}\right) \cdot B(u+1,l) + n^{(\rho_q - 1 + o_c(1))/K} \cdot B(u,l+1)$$

The first term comes from recursing down dense clusters. The second term, which corresponds to random spherical caps, is a bit more subtle. We have T spherical caps. For each cap, where the query belongs to, we have some number of close points (closer than $(\sqrt{2}-\varepsilon)R$ to the query) and some number of far points. Thus, the expected number on the fraction of the tested points in this case is at most:

$$\begin{aligned}
&T \cdot \Big(F(\eta_q) \cdot \tau + G(\sqrt{2}-\varepsilon, \eta_s, \eta_q)\Big) \cdot B(u,l+1) \\
&\leq \frac{3 \cdot \big(F(\eta_q) \cdot \tau + G(\sqrt{2}-\varepsilon, \eta_s, \eta_q)\big)}{G(r_1/R, \eta_s, \eta_q)} \cdot B(u,l+1) \\
&\leq \frac{4 \cdot G(\sqrt{2}-\varepsilon, \eta_s, \eta_q)}{G(r_1/R, \eta_s, \eta_q)} \cdot B(u,l+1) \\
&\\
&\leq n^{(\rho_q - 1 + o_c(1))/K} \cdot B(u,l+1),
\end{aligned}$$

where the second step follows from the fact that

$$G(\sqrt{2}-\varepsilon, \eta_s, \eta_q) = 2^{-O_c(\sqrt{\log n} \cdot \log^{O(1)} \log n)}$$

and that $\tau = 2^{-\Theta(\log^{2/3} n)}$. Unraveling the recursion, we note that $u \leq U$ and $l \leq K \sim \sqrt{\ln n}$. Additionally, we have that $B(u,K) \leq 1$, since we do not store duplicates in the last level. Therefore,

$$B(0,0) \leq \binom{U+K}{U} O\left(\frac{\log n}{\tau}\right)^U \cdot \left(n^{(\rho_q - 1 + o(1))/K}\right)^K = n^{\rho_q - 1 + o_c(1)}.$$

□

2.9.5 Fast preprocessing

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

Near-quadratic time

To get preprocessing time $n^{2+\rho_s+o_c(1)}$ we need to observe that during the clustering step in ProcessSphere we may look only for balls with centers being points from P. We build upon the following lemma.

Proposition 2.9.13 (van der Corput Lemma)**.** *For any $v^*, v_1, v_2, \ldots, v_n \in S^{d-1}$ one has*

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

Proof. We have

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

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

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

Claim 2.9.14. *Fix $\varepsilon > 0$. Suppose that $U \subset S^{d-1}$ with $|U| = n$. Suppose that there exists $u^* \in S^{d-1}$ such that $\|u^* - u\| \leq \sqrt{2} - \varepsilon$ for every $u \in U$. Then, there exists $u_0 \in U$ such that*

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

Proof. First, observe that $\|u^* - u\| \leq \sqrt{2} - \varepsilon$ iff $\langle u^*, u\rangle \geq \Omega(\varepsilon)$. By van der Corput Lemma (Proposition 2.9.13),

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

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

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

This implies

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

which is equivalent to

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

□

It means that if in ProcessSphere we search for clusters of radius $\sqrt{2} - \Omega(\varepsilon^2)$ centered in data points that cover at least $\Theta(\varepsilon^2\tau)$-fraction of the remaining data points, then after we remove all of them, we are sure that there are no clusters of radius $\sqrt{2} - \varepsilon$ with *arbitrary* centers that cover at least τ-fraction of the remaining points. It is immediate to check that we can adjust the analysis of the data structure accordingly to accommodate this change.

It is easy to see that by reducing the time of each clustering step to near-quadratic, we reduce the total preprocessing time to $n^{2+\rho_s+o_c(1)}$. This follows from the proof of the space bound (intuitively, each point participates in $n^{o_c(1)}$ instances of the clustering subroutine).

Near-linear time

To get $n^{1+\rho_s+o_c(1)}$ preprocessing time, we just subsample the dataset before finding each dense component. Indeed, since we care about the clusters with at least $\varepsilon^2\tau = n^{-o_c(1)}$ fraction of the remaining points, we can sample $n^{o_c(1)}$ points from the dataset and find dense components for the sample. Then, using the fact that the VC-dimension for balls in $\mathbb{R}^d$ is $O(d)$, we can argue that this sample is accurate enough with probability at least $1 - n^{-10\rho_s}$. Thus, with high probability, all the clustering steps are correct.

2.9.6 Handling insertions and deletions

The paper [142] shows how to convert a static data structure into a dynamic one if the following two conditions hold:

- The problem we are solving is *decomposable*. This means that if we partition a dataset into two parts and answer a query for both of the parts, we can compute the answer for the whole dataset efficiently. This condition clearly holds for the ANN problem.
- The preprocessing time of the static data structure must be small.

Then, [142] shows how to obtain a dynamic data structure whose query time is the static query time times $\log n$, and insertion/deletion time is the preprocessing time times $\frac{\log n}{n}$. This gives query time $n^{\rho_q+o_c(1)}$ and insertion/deletion time $n^{\rho_s+o(1)}$.

Chapter 3

FALCONN: practical and optimal LSH for unit sphere

3.1 Introduction

In Chapter 2 we developed a (c, r)-ANN data structure over the ℓ_p distance with strong guarantees (Theorem 2.3.1). A natural question is how practical the resulting data structure is. More specifically, we will focus on the case of the unit sphere $S^{d-1} \subset \mathbb{R}^d$, thus asking for a practical counterpart of Theorem 2.5.1. Indeed, in theory Theorem 2.5.1 implies Theorem 2.3.1, while in practice, the unit sphere case corresponds to similarity search with respect to the *cosine similarity*, which is widely used in applications such as comparing image feature vectors [91], speaker representations [158], and tf-idf data sets [166].

Unfortunately, the data structure we developed to prove Theorem 2.5.1 (for an overview, see Section 2.4) is far from being practical, the biggest problem being the "worst case to random" reduction from Section 2.9[1]. But can we at least make the data structure based on data-independent partitions (Theorem 2.8.1 or, more specifically, Figure 2-9) practical? This is the problem we address in the present chapter.

As it turns out, as described in Section 2.8, even the simple data-independent approach is impractical. One immediate problem is the following: in order to get good

[1]To get an idea why this is the case, see Section 2.9.3 or the proof of Lemma 2.9.2.

space and query exponents ρ_s, ρ_q, we must allow tree nodes to have many[2] children, and during the query stage, we have to enumerate all of them.

Let us focus on the balanced regime (space $n^{1+\rho+o(1)}$ and query time $n^{\rho+o(1)}$). In this regime, the data-independent approach from Section 2.8 gives the bound:

$$\sqrt{\rho} = \frac{\beta(r)\beta(cr)}{\big(1+\alpha(r)\big)\big(1-\alpha(cr)\big)},$$

where functions $\alpha(\cdot)$ and $\beta(\cdot)$ are defined in the beginning of Section 2.7, or, expanding $\alpha(\cdot)$ and $\beta(\cdot)$,

$$\rho = \frac{4-c^2r^2}{4-r^2} \cdot \frac{1}{c^2}. \tag{3.1}$$

Thus, we can ask: can one achieve the exponent (3.1) with a *practical* data structure?

3.1.1 Our results

In this chapter we obtain a practical data structure that achieves the exponent (3.1) for the (c, r)-ANN on the unit sphere. In particular, when $r = o(1)$, we get $\rho = \frac{1}{c^2} + o_c(1)$ matching the best possible data-independent LSH for ℓ_2 from [13]. Matching the guarantee from [13] by a practical algorithm has been an open problem since 2006 and here we essentially resolve it.

New practical LSH family for the unit sphere

The new data structure fits the Locality-Sensitive Hashing (LSH) framework outlined in Section 1.2.2 and Section 2.2.1. We describe the relation between LSH and the ANN problem in more detail in Section 1.4.4, but for now we just recall the definition of LSH partitions. We say that a random partition $\mathcal{P}$ of the unit sphere S^{d-1} is (r, cr, p_1, p_2)-*sensitive*, if for every $x, y \in S^{d-1}$ one has:

- if $\|x-y\|_2 \le r$, then $\Pr_{\mathcal{P}}[x \text{ and } y \text{ fall into the same part of } \mathcal{P}] \ge p_1$;
- if $\|x-y\|_2 > cr$, then $\Pr_{\mathcal{P}}[x \text{ and } y \text{ fall into the same part of } \mathcal{P}] \le p_2$.

[2] More quantitatively, we need to set $T = 2^{\Omega(\sqrt{\log n})}$ in order for the exponents ρ_s, ρ_q to deviate from their limits by $O\left(\frac{1}{\sqrt{\log n}}\right)$. See Section 2.8.4 and Section 2.8.5 for the details.

It is known [85, 77] that an efficient (r, cr, p_1, p_2)-sensitive LSH family implies a data structure for (c, r)-ANN with space $n^{1+\rho+o(1)}/p_1$ and query time $n^{\rho+o(1)}/p_1$, where:

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

In light of this reduction, it is enough to construct an efficient (c, cr, p_1, p_2)-sensitive LSH family for the unit sphere, for which

$$\frac{\log(1/p_1)}{\log(1/p_2)} \leq \frac{4 - c^2 r^2}{4 - r^2} \cdot \frac{1}{c^2} + o_{c,r}(1). \tag{3.2}$$

This is essentially what we do in this chapter. Let us compare the new LSH family with the existing families.

- **Voronoi LSH** from [18, 26] also achieves (3.2), but it is impractical for more or less the same reason as the data-independent approach from Section 2.8. Thus, our result can be seen as a practical counterpart of the Voronoi LSH.

- **Hyperplane LSH** from [55] gives the exponent worse than (3.2). See Figure 3-1 for the comparison for the case $c = 2$. Nevertheless, Hyperplane LSH is very simple and is widely used (see, e.g., [118, 166]). The new LSH family described in this chapter significantly outperforms Hyperplane LSH in practice.

- **Cross-polytope LSH** introduced in [168] (and then appeared in [72]) is one of the main building blocks of the new family. To sample a cross-polytope LSH partition, we consider a randomly rotated set $\{\pm e_i\}_{i=1}^d$, where e_i are the standard basis vectors in $\mathbb{R}^d$; then we use their Voronoi diagram as a partitions. Prior to this work, this LSH construction has not been analyzed, but in this chapter we analyze it rigorously and show that it achieves (3.2). However, by itself, the Cross-polytope LSH is still impractical, and we need to improve it using additional algorithmic ideas.

Thus, the new LSH family can be seen as the Cross-polytope LSH "on steroids".

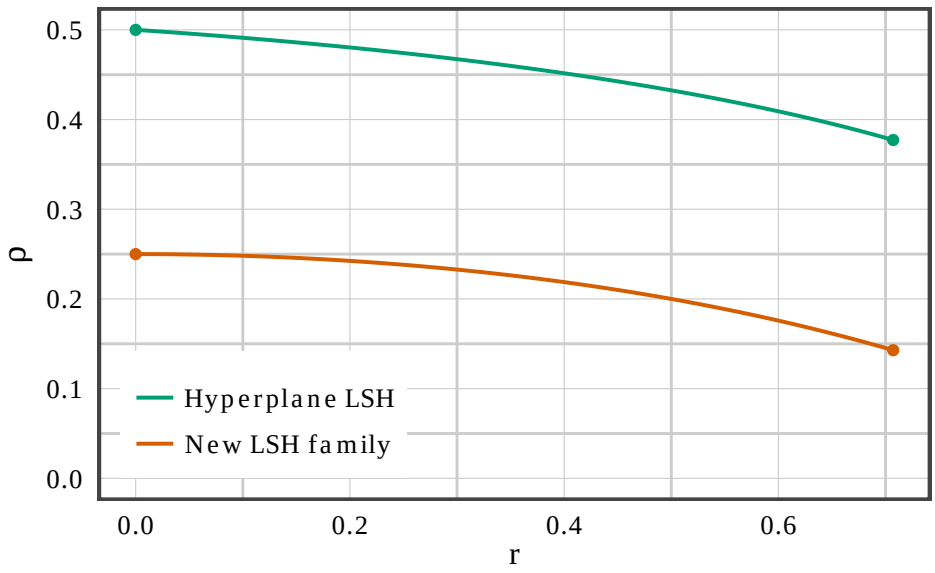


Figure 3-1: The dependence of the exponent ρ on r for the Hyperplane LSH and the LSH family developed in this chapter for $c = 2$. The dots on the left correspond to $r = 0$, while the ones on the right correspond to $r = \frac{\sqrt{2}}{2}$ (the regime of random instances).

Fine-grained lower bound for LSH for unit sphere

Let us note that the bound on the exponent (3.2) is known to be optimal for LSH over the sphere in the case $r = \frac{\sqrt{2}}{c} \pm o(1)$, which corresponds to the random case [27]. But in this chapter we obtain a somewhat finer lower bound.

To highlight the difficulty of obtaining optimal *and* practical LSH schemes, we prove the first *non-asymptotic* lower bound on the trade-off between the collision probabilities p_1 and p_2 for $r = \frac{\sqrt{2}}{c} \pm o(1)$. So far, the optimal LSH upper bound $\rho = \frac{1}{2c^2-1}$ (Voronoi LSH from [18, 26] and the Cross-polytope LSH from this chapter) attain this bound only in the limit, as $p_1, p_2 \to 0$. Very small p_1 and p_2 are undesirable since the hash evaluation time is often proportional to $1/p_2$. Our lower bound proves this is unavoidable: if we require p_2 to be large, ρ has to be suboptimal.

This result has two important implications for designing practical hash functions. First, it shows that the trade-offs achieved by the cross-polytope LSH and the Voronoi LSH from [18, 26] are essentially optimal. Second, the lower bound guides design of future LSH functions: if one is to significantly improve upon the cross-polytope LSH,

one has to design a hash function that is computed more efficiently than by explicitly enumerating its range (see Section 3.4 for a more detailed discussion).

Multiprobe scheme for the cross-polytope LSH

The space complexity of an LSH data structure is sub-*quadratic*, but even this is often too large (i.e., strongly super-*linear* in the number of points), and several methods have been proposed to address this issue. Empirically, the most efficient scheme is multiprobe LSH [118], which leads to a significantly reduced memory footprint for the hyperplane LSH. In order to make the cross-polytope LSH competitive in practice with the multiprobe hyperplane LSH, we propose a novel multiprobe scheme for the cross-polytope LSH.

This part of the algorithm can be seen as a heuristic analogue of the time–space trade-off from Chapter 2 (rather, the half of the trade-off, which corresponds to $\rho_s \leq \rho_q$).

Experimental evaluation

We complement these contributions with an experimental evaluation on both real and synthetic data (SIFT vectors, tf-idf data, and a random point set). In order to make the cross-polytope LSH practical, we combine it with fast pseudo-random rotations [6] via the Fast Hadamard Transform, and feature hashing [173] to exploit sparsity of data. Our results show that for data sets with around 10^5 to 10^8 points, our multiprobe variant of the cross-polytope LSH is up to $10\times$ faster than an efficient implementation of the hyperplane LSH, and up to $700\times$ faster than a linear scan. To the best of our knowledge, our combination of techniques provides the first "exponent-optimal" algorithm that empirically improves over the hyperplane LSH in terms of query time for an *exact* nearest neighbor search.

We also conducted preliminary comparison with other practical ANN algorithms and concluded that the algorithm from this chapter is competitive with the state of the art. However, we will report on these experiments in future work.

Finally, we release the implementation as a new C++ library with Python bindings called FALCONN [153].

3.1.2 Related work

The cross-polytope LSH functions were originally proposed in [168]. However, the analysis in that paper was mostly experimental. Specifically, the probabilities p_1 and p_2 of the proposed LSH functions were estimated empirically using the Monte Carlo method. Similar hash functions were later proposed in [72]. The latter paper also uses DFT to speed-up the random matrix-vector matrix multiplication operation. Both of the aforementioned papers consider only the *single-probe* algorithm.

There are several works that show lower bounds on the quality of LSH hash functions [129, 70, 140, 27]. However, those papers provide only a lower bound on the ρ parameter for asymptotic values of p_1 and p_2, as opposed to an actual trade-off between these two quantities. In this paper we provide such a trade-off, with implications as outlined in the introduction.

Finally, let us point out that the algorithm developed in this chapter has strong parallels with the algorithm from Chapter 2 as well as with Product Quantization [91]. We will elaborate on these similarities in the future work.

3.2 Preliminaries

In this chapter we use $\|.\|$ to denote the Euclidean (a.k.a. ℓ_2) norm on $\mathbb{R}^d$. The Gaussian distribution with mean zero and variance of one is denoted by $N(0,1)$. Let μ be a normalized Haar measure on S^{d-1} (that is, $\mu(S^{d-1}) = 1$). Note that μ it corresponds to the uniform distribution over S^{d-1}. We also let $u \sim S^{d-1}$ be a point sampled from S^{d-1} uniformly at random. For $\eta \in \mathbb{R}$ we denote

$$\Phi_c(\eta) = \Pr_{X \sim N(0,1)}[X \geq \eta] = \frac{1}{\sqrt{2\pi}} \int_\eta^\infty e^{-t^2/2}\, dt.$$

This chapter relies heavily on the definition of Locality-Sensitive Hashing (LSH)

and the relation between LSH and the ANN problem. See Section 1.4.4 or [77] for the introduction.

3.3 Cross-polytope LSH

In this section, we describe the cross-polytope LSH, analyze it, and show how to make it practical. First, we recall the definition of the cross-polytope LSH [168]: Consider the following hash family $\mathcal{H}$ for points on a unit sphere $S^{d-1} \subset \mathbb{R}^d$. Let $A \in \mathbb{R}^{d \times d}$ be a random matrix with i.i.d. Gaussian entries ("a random rotation"). To hash a point $x \in S^{d-1}$, we compute $y = Ax/\|Ax\| \in S^{d-1}$ and then find the point closest to y from $\{\pm e_i\}_{1 \le i \le d}$, where e_i is the i-th standard basis vector of $\mathbb{R}^d$. We use the closest neighbor as a hash of x.

The following theorem bounds the collision probability for two points under the above family $\mathcal{H}$.

Theorem 3.3.1. *Suppose that $p, q \in S^{d-1}$ are such that $\|p-q\| = \tau$, where $0 < \tau < 2$. Then,*

$$\ln \frac{1}{\Pr_{h \sim \mathcal{H}}\left[h(p) = h(q)\right]} = \frac{\tau^2}{4-\tau^2} \cdot \ln d + O_\tau(\ln \ln d) \; .$$

Before we show how to prove this theorem, we briefly describe its implications. Theorem 3.3.1 shows that the cross-polytope LSH achieves almost the same bounds on the collision probabilities as the (theoretically) optimal LSH for the sphere from [26] (see Section "Spherical LSH" there). In particular, substituting the bounds from Theorem 3.3.1 for the cross-polytope LSH into the standard reduction from Near Neighbor Search to LSH (see [77] or Section 1.4.4), we obtain the following data structure with sub-quadratic space and sublinear query time for Near Neighbor Search on a sphere.

Corollary 3.3.2. *The (c, r)-ANN on a unit sphere S^{d-1} can be solved in space $O(n^{1+\rho} + dn)$ and query time $O(d \cdot n^\rho)$, where $\rho = \frac{1}{c^2} \cdot \frac{4-c^2r^2}{4-r^2} + o(1)$.*

We now outline the proof of Theorem 3.3.1. For the full proof, see Section 3.8.

Due to the spherical symmetry of Gaussians, we can assume that $p = e_1$ and $q = \alpha e_1 + \beta e_2$, where α, β are such that $\alpha^2 + \beta^2 = 1$ and $(\alpha - 1)^2 + \beta^2 = \tau^2$. Then, we expand the collision probability:

$$\begin{aligned}
&\Pr_{h\sim\mathcal{H}}[h(p) = h(q)] \\
&= 2d \cdot \Pr_{h\sim\mathcal{H}}[h(p) = h(q) = e_1] \\
&= 2d \cdot \Pr_{u,v\sim N(0,1)^d}[\forall i \ \ |u_i| \le u_1 \text{ and } |\alpha u_i + \beta v_i| \le \alpha u_1 + \beta v_1] \\
&= 2d \cdot \mathop{\mathrm{E}}_{X_1,Y_1}\left[\Pr_{X_2,Y_2}\left[|X_2| \le X_1 \text{ and } |\alpha X_2 + \beta Y_2| \le \alpha X_1 + \beta Y_1\right]^{d-1}\right], \qquad (3.3)
\end{aligned}$$

where $X_1, Y_1, X_2, Y_2 \sim N(0, 1)$. Indeed, the first step is due to the spherical symmetry of the hash family, the second step follows from the above discussion about replacing a random orthogonal matrix with a Gaussian one and that one can assume w.l.o.g. that $p = e_1$ and $q = \alpha e_1 + \beta e_2$; the last step is due to the independence of the entries of u and v.

Thus, proving Theorem 3.3.1 reduces to estimating the right-hand side of (3.3). Note that the probability $\Pr[|X_2| \le X_1$ and $|\alpha X_2 + \beta Y_2| \le \alpha X_1 + \beta Y_1]$ is equal to the Gaussian area of the planar set S_{X_1,Y_1} shown in Figure 3-2a. The latter is *heuristically* equal to $1 - e^{-\Delta^2/2}$, where Δ is the distance from the origin to the complement of S_{X_1,Y_1}, which is easy to compute (see Section 3.7 for the precise statement of this argument). Using this estimate, we compute (3.3) by taking the outer expectation.

3.3.1 Making the cross-polytope LSH practical

As described above, the cross-polytope LSH is not quite practical. The main bottleneck is sampling, storing, and applying a random rotation. In particular, to multiply a random Gaussian matrix with a vector, we need time proportional to d^2, which is infeasible for large d.

Pseudo-random rotations. To rectify this issue, we instead use *pseudo-random rotations*. Instead of multiplying an input vector x by a random Gaussian matrix,

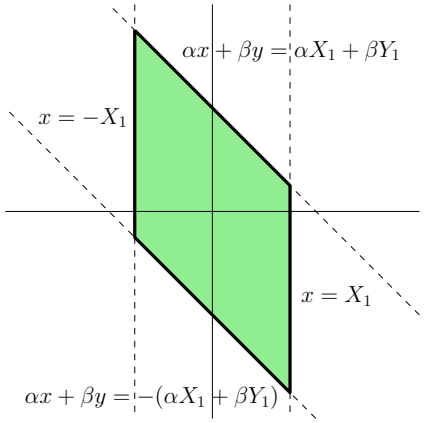
we apply the following linear transformation: $x \mapsto HD_3HD_2HD_1x$, where H is the Hadamard transform, and D_i for $i \in \{1, 2, 3\}$ is a random diagonal ± 1-matrix. Clearly, this is an orthogonal transformation, which one can store in space $O(d)$ and evaluate in time $O(d \log d)$ using the Fast Hadamard Transform. This is similar to pseudo-random rotations used in the context of LSH [64], dimensionality reduction [6], or compressed sensing [7]. While we are currently not aware how to prove rigorously that such pseudo-random rotations perform as well as the fully random ones, empirical evaluations show that three applications of HD_i are exactly equivalent to applying a true random rotation (when d tends to infinity). We note that only *two* applications of HD_i are not sufficient.

Feature hashing. While we can apply a pseudo-random rotation in time $O(d \log d)$, even this can be too slow. E.g., consider an input vector x that is *sparse*: the number of non-zero entries of x is s much smaller than d. In this case, we can evaluate the hyperplane LSH from [55] in time $O(s)$, while computing the cross-polytope LSH (even with pseudo-random rotations) still takes time $O(d \log d)$. To speed-up the cross-polytope LSH for sparse vectors, we apply feature hashing [173]: before performing a pseudo-random rotation, we reduce the dimension from d to $d' \ll d$ by applying a linear map $x \mapsto Sx$, where S is a random sparse $d' \times d$ matrix, whose columns have *one* non-zero ± 1 entry sampled uniformly. This way, the evaluation time becomes $O(s + d' \log d')$. [3]

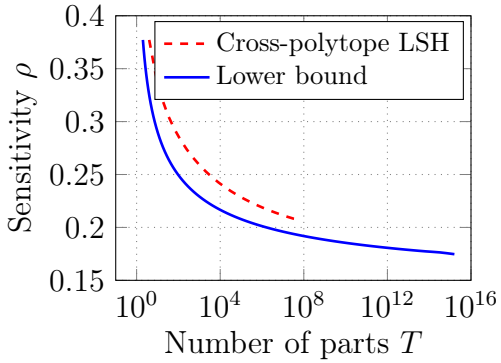
"Partial" cross-polytope LSH. In the above discussion, we defined the cross-polytope LSH as a hash family that returns the closest neighbor among $\{\pm e_i\}_{1 \leq i \leq d}$ as a hash (after a (pseudo-)random rotation). In principle, we do not have to consider all d basis vectors when computing the closest neighbor. By restricting the hash to $d' \leq d$ basis vectors instead, Theorem 3.3.1 still holds for the new hash family (with d replaced by d') since the analysis is essentially dimension-free. This slight

[3] Note that one can apply Lemma 2 from the arXiv version of [173] to claim that—after such a dimension reduction—the distance between *any* two points remains sufficiently concentrated for the bounds from Theorem 3.3.1 to still hold (with d replaced by d').

Figure 3-2



(a) The set appearing in the analysis of the cross-polytope LSH: $S_{X_1Y_1} = \{|x| \leq X_1 \text{ and } |\alpha x + \beta y| \leq \alpha X_1 + \beta Y_1\}$.



(b) Trade-off between ρ and the number of parts for distances $\sqrt{2}/2$ and $\sqrt{2}$ (approximation $c = 2$); both bounds tend to $1/7$ (see discussion in Section 3.4).

generalization of the cross-polytope LSH turns out to be useful for experiments (see Section 3.6). Note that the case $d' = 1$ corresponds to the hyperplane LSH.

3.4 Lower bound

Let $\mathcal{H}$ be a hash family on S^{d-1}. For $0 < r_1 < r_2 < 2$ we would like to understand the trade-off between p_1 and p_2, where p_1 is the *smallest* probability of collision under $\mathcal{H}$ for points at distance *at most* r_1 and p_2 is the *largest* probability of collision for points at distance *at least* r_2. We focus on the case $r_2 \approx \sqrt{2}$ because setting r_2 to $\sqrt{2} - o(1)$ (as d tends to infinity) allows us to replace p_2 with the following quantity that is somewhat easier to handle:

$$p_2^* = \Pr_{\substack{h \sim \mathcal{H} \\ u,v \sim S^{d-1}}} [h(u) = h(v)].$$

This quantity is at most $p_2 + o(1)$, since the distance between two random points on a unit sphere S^{d-1} is tightly concentrated around $\sqrt{2}$. So for a hash family $\mathcal{H}$ on a unit sphere S^{d-1}, we would like to understand the upper bound on p_1 in terms of p_2^* and $0 < r_1 < \sqrt{2}$.

For $0 \leq \tau \leq \sqrt{2}$ and $\eta \in \mathbb{R}$, we define $\Lambda(\tau, \eta)$ to be:

$$\Pr_{X,Y \sim N(0,1)} \left[X \geq \eta \text{ and } \left(1 - \frac{\tau^2}{2}\right) \cdot X + \sqrt{\tau^2 - \frac{\tau^4}{4}} \cdot Y \geq \eta \right] \Big/ \Pr_{X \sim N(0,1)} [X \geq \eta] \, .$$

We are now ready to formulate the main result of this section.

Theorem 3.4.1. *Let $\mathcal{H}$ be a hash family on S^{d-1} such that every function in $\mathcal{H}$ partitions the sphere into at most T parts of measure at most $1/2$. Then we have $p_1 \leq \Lambda(r_1, \eta) + o(1)$, where $\eta \in \mathbb{R}$ is such that $\Phi_c(\eta) = p_2^*$ and $o(1)$ is a quantity that depends on T and r_1 and tends to 0 as d tends to infinity.*

The idea of the proof is first to reason about one part of the partition using the isoperimetric inequality from [73], and then to apply a certain averaging argument by proving concavity of a function related to Λ using a delicate analytic argument. For the full proof, see Section 3.9.

We note that the above requirement of all parts induced by $\mathcal{H}$ having measure at most $1/2$ is only a technicality. We conjecture that Theorem 3.4.1 holds without this restriction. In any case, as we will see below, in the interesting range of parameters this restriction is essentially irrelevant.

One can observe that if every hash function in $\mathcal{H}$ partitions the sphere into at most T parts, then $p_2^* \geq \frac{1}{T}$ (indeed, p_2^* is precisely the average sum of squares of measures of the parts). This observation, combined with Theorem 3.4.1, leads to the following interesting consequence. Specifically, we can numerically estimate Λ in order to give a lower bound on $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$ for any hash family $\mathcal{H}$ in which every function induces at most T parts of measure at most $1/2$. See Figure 3-2b, where we plot this lower bound for $r_1 = \sqrt{2}/2$,[4] together with an upper bound that is given by the cross-polytope LSH[5] (for which we use numerical estimates for (3.3)). We can make several conclusions from this plot. First, the cross-polytope LSH gives an almost optimal trade-off between ρ and T. Given that the evaluation time for the

[4] The situation is qualitatively similar for other values of r_1.

[5] More specifically, for the "partial" version from Section 3.3.1, since T should be constant, while d grows

cross-polytope LSH is $O(T \log T)$ (if one uses pseudo-random rotations), we conclude that in order to improve upon the cross-polytope LSH substantially in practice, one should design an LSH family with ρ being close to optimal and evaluation time that is *sublinear in* T. We note that none of the known LSH families for a sphere has been shown to have this property. This direction looks especially interesting since the convergence of ρ to the optimal value (as T tends to infinity) is extremely slow (for instance, according to Figure 3-2b, for $r_1 = \sqrt{2}/2$ and $r_2 \approx \sqrt{2}$ we need more than 10^5 parts to achieve $\rho \leq 0.2$, whereas the optimal ρ is $1/7 \approx 0.143$).

3.5 Multiprobe LSH for the cross-polytope LSH

We now describe our multiprobe scheme for the cross-polytope LSH, which is a method for reducing the number of independent hash tables in an LSH data structure. Given a query point q, a "standard" LSH data structure considers only a *single* cell in each of the L hash tables (the cell is given by the hash value $h_i(q)$ for $i \in [L]$). In multiprobe LSH, we consider candidates from *multiple* cells in each table [118]. The rationale is the following: points p that are close to q but fail to collide with q under hash function h_i are still likely to hash to a value that is close to $h_i(q)$. By probing multiple hash locations close to $h_i(q)$ in the same table, multiprobe LSH achieves a given probability of success with a smaller number of hash tables than "standard" LSH. Multiprobe LSH has been shown to perform well in practice [118, 164].

The main ingredient in multiprobe LSH is a probing scheme for generating and ranking possible modifications of the hash value $h_i(q)$. The probing scheme should be computationally efficient and ensure that more likely hash locations are probed first. For a single cross-polytope hash, the order of alternative hash values is straightforward: let x be the (pseudo-)randomly rotated version of query point q. Recall that the "main" hash value is $h_i(q) = \operatorname{argmax}_{j \in [d]} |x_j|$.[6] Then it is easy to see that the second highest probability of collision is achieved for the hash value corresponding to the

[6] In order to simplify notation, we consider a slightly modified version of the cross-polytope LSH that maps both the standard basis vector $+e_j$ and its opposite $-e_j$ to the same hash value. It is easy to extend the multiprobe scheme defined here to the "full" cross-polytope LSH from Section 3.3.

coordinate with the second largest absolute value, etc. Therefore, we consider the indices $i \in [d]$ sorted by their absolute value as our probing sequence or "ranking" for a single cross-polytope.

The remaining question is how to combine multiple cross-polytope rankings when we have more than one hash function. As in the analysis of the cross-polytope LSH (see Section 3.3, we consider two points $q = e_1$ and $p = \alpha e_1 + \beta e_2$ at distance R. Let $A^{(i)}$ be the i.i.d. Gaussian matrix of hash function h_i, and let $x^{(i)} = A^{(i)}e_1$ be the randomly rotated version of point q. Given $x^{(i)}$, we are interested in the probability of p hashing to a certain combination of the individual cross-polytope rankings. More formally, let $r^{(i)}_{v_i}$ be the index of the v_i-th largest element of $|x^{(i)}|$, where $v \in [d]^k$ specifies the alternative probing location. Then we would like to compute

$$\begin{aligned}&\Pr_{A^{(1)},\dots,A^{(k)}}\left[h_i(p) = r^{(i)}_{v_i} \text{ for all } i \in [k] \mid A^{(i)}q = x^{(i)}\right] \\ &\quad = \prod_{i=1}^{k} \Pr_{A^{(i)}}\left[\operatorname{argmax}_{j\in[d]} \left|(\alpha \cdot A^{(i)}e_1 + \beta \cdot A^{(i)}e_2)_j\right| = r^{(i)}_{v_i} \;\middle|\; A^{(i)}e_1 = x^{(i)}\right].\end{aligned}$$

If we knew this probability for all $v \in [d]^k$, we could sort the probing locations by their probability. We now show how to approximate this probability efficiently for a single value of i (and hence drop the superscripts to simplify notation). WLOG, we permute the rows of A so that $r_v = v$ and get

$$\begin{aligned}&\Pr_{A}\left[\operatorname{argmax}_{j\in[d]} \left|(\alpha x + \beta \cdot Ae_2)_j\right| = v \;\middle|\; Ae_1 = x\right] \\ &= \Pr_{y\sim N(0,I_d)}\left[\operatorname{argmax}_{j\in[d]} \left|(x + \frac{\beta}{\alpha} \cdot y)_j\right| = v\right].\end{aligned}$$

The RHS is the Gaussian measure of the set $S = \{y \in \mathbb{R}^d \mid \operatorname{argmax}_{j\in[d]} \left|(x+\frac{\beta}{\alpha}y)_j\right| = v\}$. Similar to the analysis of the cross-polytope LSH, we approximate the measure of S by its distance to the origin. Then the probability of probing location v is proportional to $\exp(-\|y_{x,v}\|^2)$, where $y_{x,v}$ is the shortest vector y such that $\operatorname{argmax}_j |x + y|_j = v$. Note that the factor β/α becomes a proportionality constant, and hence the probing scheme does not require to know the distance R. For computational performance and

simplicity, we make a further approximation and use $y_{x,v} = (\max_i |x_i| - |x_v|) \cdot e_v$, i.e., we only consider modifying a single coordinate to reach the set S.

Once we have estimated the probabilities for each $v_i \in [d]$, we incrementally construct the probing sequence using a binary heap, similar to the approach in [118]. For a probing sequence of length m, the resulting algorithm has running time $O(L \cdot d \log d + m \log m)$. In our experiments, we found that the $O(L \cdot d \log d)$ time taken to sort the probing candidates v_i dominated the running time of the hash function evaluation. In order to circumvent this issue, we use an incremental sorting approach that only sorts the relevant parts of each cross-polytope and gives a running time of $O(L \cdot d + m \log m)$.

3.6 Experiments

We now show that the cross-polytope LSH, combined with our multiprobe extension, leads to an algorithm that is also efficient in practice and improves over the hyperplane LSH on several data sets. The focus of our experiments is the query time for an *exact* nearest neighbor search. Since hyperplane LSH has been compared to other nearest-neighbor algorithms before [158], we limit our attention to the relative speed-up compared with hyperplane hashing.

We evaluate the two hashing schemes on three types of data sets. We use a synthetic data set of randomly generated points because this allows us to vary a single problem parameter while keeping the remaining parameters constant. We also investigate the performance of our algorithm on real data: two tf-idf data sets [113] and a set of SIFT feature vectors [91]. We have chosen these data sets in order to illustrate when the cross-polytope LSH gives large improvements over the hyperplane LSH, and when the improvements are more modest. See Section 3.10 for a more detailed description of the data sets and our experimental setup (implementation details, CPU, etc.).

In all experiments, we set the algorithm parameters so that the empirical probability of successfully finding the exact nearest neighbor is at least 0.9. Moreover, we set the

number of LSH tables L so that the amount of additional memory occupied by the LSH data structure is comparable to the amount of memory necessary for storing the data set. We believe that this is the most interesting regime because significant memory overheads are often impossible for large data sets. In order to determine the parameters that are not fixed by the above constraints, we perform a grid search over the remaining parameter space and report the best combination of parameters. For the cross-polytope hash, we consider "partial" cross-polytopes in the last of the k hash functions in order to get a smooth trade-off between the various parameters (see Section 3.3.1).

Multiprobe experiments. In order to demonstrate that the multiprobe scheme is critical for making the cross-polytope LSH competitive with hyperplane hashing, we compare the performance of a "standard" cross-polytope LSH data structure with our multiprobe variant on an instance of the random data set ($n = 2^{20}$, $d = 128$). As can be seen in Table 3.2 (Section 3.10), the multiprobe variant is about 13× faster in our memory-constrained setting ($L = 10$). Note that in all of the following experiments, the speed-up of the multiprobe cross-polytope LSH compared to the multiprobe hyperplane LSH is less than 11×. Hence without our multiprobe addition, the cross-polytope LSH would be slower than the hyperplane LSH, for which a multiprobe scheme is already known [118].

Experiments on random data. Next, we show that the better time complexity of the cross-polytope LSH already applies for moderate values of n. In particular, we compare the cross-polytope LSH, combined with fast rotations (Section 3.3.1) and our multiprobe scheme, to a multi-probe hyperplane LSH on random data. We keep the dimension $d = 128$ and the distance to the nearest neighbor $R = \sqrt{2}/2$ fixed, and vary the size of the data set from 2^{20} to 2^{28}. The number of hash tables L is set to 10. For 2^{20} points, the cross-polytope LSH is already 3.5× faster than the hyperplane LSH, and for $n = 2^{28}$ the speedup is 10.3× (see Table 3.3 in Section 3.10). Compared to a linear scan, the speed-up achieved by the cross-polytope LSH ranges from 76×

Data set	Method	Query time (ms)	**Speed-up vs HP**	Best k	Number of candidates	Hashing time (ms)	Distances time (ms)
NYT	HP	120 ms		19	57,200	16	96
NYT	CP	35 ms	**3.4×**	2 (64)	17,900	3.0	30
pubmed	HP	857 ms		20	1,480,000	36	762
pubmed	CP	213 ms	**4.0×**	2 (512)	304,000	18	168
SIFT	HP	3.7 ms		30	18,628	0.2	3.0
SIFT	CP	3.1 ms	**1.2×**	6 (1)	13,000	0.6	2.2

Table 3.1: Average running times for a single nearest neighbor query with the hyperplane (HP) and cross-polytope (CP) algorithms on three real data sets. The cross-polytope LSH is faster than the hyperplane LSH on all data sets, with significant speed-ups for the two tf-idf data sets NYT and pubmed. For the cross-polytope LSH, the entries for k include both the number of individual hash functions per table and (in parenthesis) the dimension of the last of the k cross-polytopes.

for $n = 2^{20}$ to about $700\times$ for $n = 2^{28}$.

Experiments on real data. On the SIFT data set ($n = 10^6$ and $d = 128$), the cross-polytope LSH achieves a modest speed-up of $1.2\times$ compared to the hyperplane LSH (see Table 3.1). On the other hand, the speed-up is is $3 - 4\times$ on the two tf-idf data sets, which is a significant improvement considering the relatively small size of the NYT data set ($n \approx 300,000$). One important difference between the data sets is that the typical distance to the nearest neighbor is smaller in the SIFT data set, which can make the nearest neighbor problem easier (see Section 3.10). Since the tf-idf data sets are very high-dimensional but sparse ($d \approx 100,000$), we use the feature hashing approach described in Section 3.3.1 in order to reduce the hashing time of the cross-polytope LSH (the standard hyperplane LSH already runs in time proportional to the sparsity of a vector). We use 512 and 2048 as feature hashing dimensions for NYT and pubmed, respectively.

3.7 Appendix: Gaussian measure of a planar set

In this Section we formalize the intuition that the standard Gaussian measure of a closed subset $A \subseteq \mathbb{R}^2$ behaves like $e^{-\Delta_A^2/2}$, where Δ_A is the distance from the origin

to A, unless A is quite special.

For a closed subset $A \subseteq \mathbb{R}^2$ and $r > 0$ denote $0 \le \mu_A(r) \le 1$ the normalized measure of the intersection $A \cap rS^1$ (A with the circle centered in the origin and of radius r):

$$\mu_A(r) := \frac{\mu(A \cap rS^1)}{2\pi r};$$

here μ is the standard one-dimensional Lebesgue measure (see Figure 3-3a). Denote $\Delta_A := \inf\{r > 0 : \mu_A(r) > 0\}$ the (essential) distance from the origin to A. Let $\mathcal{G}(A)$ be the standard Gaussian measure of A.

Lemma 3.7.1. *Suppose that $A \subseteq \mathbb{R}^2$ is a closed set such that $\mu_A(r)$ is non-decreasing. Then,*

$$\sup_{r>0}\Big(\mu_A(r) \cdot e^{-r^2/2}\Big) \le \mathcal{G}(A) \le e^{-\Delta_A^2/2}.$$

Proof. For the upper bound, we note that

$$\mathcal{G}(A) = \int_0^\infty \mu_A(r) \cdot re^{-r^2/2}\,dr \le \int_{\Delta_A}^\infty re^{-r^2/2}\,dr = e^{-\Delta_A^2/2}.$$

For the lower bound, we similarly have, for every $r^* > 0$,

$$\mathcal{G}(A) = \int_0^\infty \mu_A(r) \cdot re^{-r^2/2}\,dr \ge \mu_A(r^*) \cdot \int_{r^*}^\infty re^{-r^2/2}\,dr = \mu_A(r^*)e^{-(r^*)^2/2},$$

where we use that $\mu_A(r^*)$ is non-decreasing. □

Now we derive two corollaries of Lemma 3.7.1.

Lemma 3.7.2. *Let $K \subseteq \mathbb{R}^2$ be the complement of an open convex subset of the plane that is symmetric around the origin. Then, for every $0 < \varepsilon < 1/3$,*

$$\Omega\Big(\varepsilon^{1/2} \cdot e^{-(1+\varepsilon)\cdot\Delta_K^2/2}\Big) \le \mathcal{G}(K) \le e^{-\Delta_K^2/2}.$$

Proof. This follows from Lemma 3.7.1: indeed, due to the convexity of the complement of K, $\mu_K(r)$ is non-decreasing. It is easy to check that

$$\mu_K\Big((1+\varepsilon)\Delta_K\Big) = \Omega\Big(\varepsilon^{1/2}\Big),$$

again, due to the convexity (see Figure 3-3b). Thus, the required bounds follow. □

Lemma 3.7.3. *Let $K \subseteq \mathbb{R}^2$ be an intersection of two closed half-planes such that:*

- *K does not contain a line;*
- *the "corner" of K is the closest point of K to the origin;*
- *the angle between half-planes equals to $0 < \alpha < \pi$.*

Then, for every $0 < \varepsilon < 1/2$,

$$\Omega_\alpha\left(\varepsilon \cdot e^{-(1+\varepsilon)\cdot\Delta_K^2}\right) \leq \mathcal{G}(K) \leq e^{-\Delta_K^2/2}.$$

Proof. This, again, follows from Lemma 3.7.1. The second condition implies that $\mu_K(r)$ is non-decreasing, and an easy computation shows that

$$\mu_K((1+\varepsilon)\Delta_K) \geq \Omega_\alpha(\varepsilon)$$

(see Figure 3-3c). □

3.8 Appendix: Proof of Theorem 3.3.1

In this section we complete the proof of Theorem 3.3.1, following the outline from Section 3.3. Our starting point is the collision probability bound from Eqn. (3.3).

For $u, v \in \mathbb{R}$ with $u \geq 0$ and $\alpha u + \beta v \geq 0$ define,

$$\sigma(u,v) = \Pr_{X_2,Y_2\sim N(0,1)}[|X_2| \leq u \text{ and } |\alpha X_2 + \beta Y_2| \leq \alpha u + \beta v].$$

Then, the right-hand side of (3.3) is equal to

$$2d \cdot \mathop{\mathrm{E}}_{X_1,Y_1\sim N(0,1)}[\sigma(X_1,Y_1)^{d-1}].$$

Let us define

$$\Delta(u,v) = \min\{u, \alpha u + \beta v\}.$$

Lemma 3.8.1. *For every* $0 < \varepsilon < 1/3$,

$$1 - e^{-\Delta(u,v)^2/2} \le \sigma(u,v) \le 1 - \Omega\left(\varepsilon^{1/2} \cdot e^{-(1+\varepsilon)\Delta(u,v)^2/2}\right).$$

Proof. This is a combination of Lemma 3.7.2 together with the following obvious observation: the distance from the origin to the set $\{(x,y) : |x| \ge u \text{ or } |\alpha x + \beta y| \ge \alpha u + \beta v\}$ is equal to $\Delta(u,v)$ (see Figure 3-2a). □

Lemma 3.8.2. *For every* $t \ge 0$ *and* $0 < \varepsilon < 1/3$,

$$\Omega_\tau\left(\varepsilon \cdot e^{-(1+\varepsilon)\cdot\frac{4}{4-\tau^2}\cdot\frac{t^2}{2}}\right) \le \Pr_{X_1,Y_1\sim N(0,1)}[\Delta(X_1,Y_1) \ge t] \le e^{-\frac{4}{4-\tau^2}\cdot\frac{t^2}{2}}.$$

Proof. Similar to the previous lemma, this is a consequence of Lemma 3.7.3 together with the fact that the squared distance from the origin to the set $\{(x,y)\colon x \ge t \text{ and } \alpha x + \beta y \ge t\}$ is equal to $\frac{4}{4-\tau^2} \cdot t^2$. □

3.8.1 Idealized proof

Let us expand (3.3) further, assuming that the "idealized" versions of Lemma 3.8.1 and Lemma 3.8.2 hold. Namely, we assume that

$$\sigma(u,v) = 1 - e^{-\Delta(u,v)^2/2}; \tag{3.4}$$

and

$$\Pr_{X_1,Y_1\sim N(0,1)}[\Delta(X_1,Y_1) \ge t] = e^{-\frac{4}{4-\tau^2}\cdot\frac{t^2}{2}}. \tag{3.5}$$

In the next section we redo the computations using the precise bounds for $\sigma(u,v)$ and $\Pr[\Delta(X_1,Y_1) \ge t]$.

Expanding Eqn. (3.3), we have

$$\begin{aligned}
\mathop{\mathrm{E}}_{X_1,Y_1\sim N(0,1)}[\sigma(X_1,Y_1)^{d-1}] &= \int_0^1 \Pr_{X_1,Y_1\sim N(0,1)}[\sigma(X_1,Y_1) \geq t^{\frac{1}{d-1}}]\,dt \\
&= \int_0^1 \Pr_{X_1,Y_1\sim N(0,1)}[e^{-\Delta(X_1,Y_1)^2/2} \leq 1 - t^{\frac{1}{d-1}}]\,dt \\
&= \int_0^1 (1-t^{\frac{1}{d-1}})^{\frac{4}{4-\tau^2}}\,dt \\
&= (d-1)\cdot\int_0^1 (1-u)^{\frac{4}{4-\tau^2}}\,u^{d-2}\,dt \\
&= (d-1)\cdot B\left(\frac{8-\tau^2}{4-\tau^2}; d-1\right) \\
&= \Theta_\tau(1)\cdot d^{-\frac{4}{4-\tau^2}}, \qquad (3.6)
\end{aligned}$$

where:

- the first step is a standard expansion of an expectation;
- the second step is due to (3.4);
- the third step is due to (3.5);
- the fourth step is a change of variables;
- the fifth step is a definition of the Beta function;
- the sixth step is due to the Stirling approximation.

Overall, substituting (3.6) into (3.3), we get:

$$\ln\frac{1}{\Pr_{h\sim\mathcal{H}}[h(p)=h(q)]} = \frac{\tau^2}{4-\tau^2}\cdot\ln d \pm O_\tau(1).$$

3.8.2 Real proof

We now perform calculations using the bounds (involving ε) from Lemma 3.8.1 and Lemma 3.8.2. We set $\varepsilon = 1/d$ and obtain the following asymptotic statements:

$$\sigma(u,v) = 1 - d^{\pm O(1)}\cdot e^{-(1\pm d^{-\Omega(1)})\cdot\Delta(u,v)^2/2};$$

and

$$\Pr_{X,Y\sim N(0,1)}[\Delta(X,Y)\geq t] = d^{\pm O(1)}\cdot e^{-(1\pm d^{-\Omega(1)})\cdot\frac{4}{4-\tau^2}\cdot\frac{t^2}{2}}.$$

Then, we can repeat the "idealized" proof (see Eqn. (3.6)) verbatim with the new estimates and obtain the final form of Theorem 3.3.1:

$$\ln\frac{1}{\Pr_{h\sim\mathcal{H}}[h(p)=h(q)]} = \frac{\tau^2}{4-\tau^2}\cdot\ln d \pm O_\tau(\ln\ln d).$$

Note the difference in the low order term between idealized and the real version. As we argue in Section 3.4, the latter $O_\tau(\ln\ln d)$ is, in fact, tight.

3.9 Appendix: Proof of Theorem 3.4.1

Lemma 3.9.1. *Let $A\subset S^{d-1}$ be a measurable subset of a sphere with $\mu(A)=\mu_0\leq 1/2$. Then, for $0<\tau<\sqrt{2}$, one has*

$$\begin{aligned}&\Pr_{u,v\sim S^{d-1}}\left[v\in A \;\middle|\; u\in A, \|u-v\|\leq\tau\right]\\ &= \frac{\Pr_{X,Y\sim N(0,1)}[X\geq\eta \text{ and } \alpha X+\beta Y\geq\eta]+o(1)}{\Pr_{X\sim N(0,1)}[X\geq\eta]+o(1)},\end{aligned}\tag{3.7}$$

where:

- $\alpha = 1-\frac{\tau^2}{2}$;
- $\beta=\sqrt{\tau^2-\frac{\tau^4}{4}}$;
- $\eta\in\mathbb{R}$ *is such that* $\Pr_{X\sim N(0,1)}[X\geq\eta]=\mu_0$.

In particular, if $\mu_0=\Omega(1)$, then

$$\Pr_{u,v\sim S^{d-1}}\left[v\in A \;\middle|\; u\in A, \|u-v\|\leq\tau\right] = \Lambda(\tau,\Phi_c^{-1}(\mu_0))+o(1).$$

Proof. First, the left-hand side of (3.7) is maximized by a spherical cap of measure μ_0. This follows from Theorem 5 of [73]. So, from now on we assume that A is a spherical

cap.

Second, one has

$$\begin{aligned}
&\Pr_{u,v\sim S^{d-1}}\left[v\in A \;\middle|\; u\in A, \|u-v\|\le\tau\right]\\
&=\Pr_{u,v\sim S^{d-1}}\left[v\in A \;\middle|\; u\in A, \|u-v\|=\tau\pm o(1)\right]+o(1)\\
&=\frac{\Pr_{u\sim S^{d-1}}\left[u_1\ge\widetilde{\eta} \text{ and } (\alpha\pm o(1))u_1+(\beta\pm o(1))u_2\ge\widetilde{\eta}\right]}{\Pr_{u\sim S^{d-1}}[u_1\ge\widetilde{\eta}]}+o(1)\\
&=\frac{\Pr_{X,Y\sim N(0,1)}[X\ge\eta \text{ and } \alpha X+\beta Y\ge\eta]+o(1)}{\Pr_{X\sim N(0,1)}[X\ge\eta]+o(1)},
\end{aligned}$$

where $\widetilde{\eta}$ is such that $\Pr_{u\sim S^{d-1}}[u_1\ge\widetilde{\eta}]=\mu_0$ and:

- the first step is due to the concentration of measure on the sphere;
- the second step is expansion of the conditional probability;
- the third step is due to the fact that a $O(1)$-dimensional projection of the uniform measure on a sphere of radius $\sqrt{d}$ in $\mathbb{R}^d$ converges in total variation to a standard Gaussian measure [68].

□

Lemma 3.9.2. *For every $0<\tau<\sqrt{2}$, the function $\mu\mapsto\Lambda(\tau,\Phi_c^{-1}(\mu))$ is concave for $0<\mu<1/2$.*

Proof. Abusing notation, for this proof we denote $\Lambda(\eta)=\Lambda(\tau,\eta)$ and

$$I(\eta)=\Pr_{X,Y\sim N(0,1)}[X\ge\eta \text{ and } \alpha X+\beta Y\ge\eta]$$

(that is, $\Lambda(\eta)=I(\eta)/\Phi_c(\eta)$). One has $\Phi_c'(\eta)=-\frac{e^{-\eta^2/2}}{\sqrt{2\pi}}$ and

$$I'(\eta)=-\sqrt{\frac{2}{\pi}}\cdot e^{-\eta^2/2}\cdot\Phi_c\left(\frac{(1-\alpha)\eta}{\beta}\right).$$

Combining, we get

$$\Lambda'(\eta) = \frac{e^{-\eta^2/2}}{\sqrt{2\pi}} \cdot \frac{I(\eta) - 2\Phi_c(\eta)\Phi_c\left(\frac{(1-\alpha)\eta}{\beta}\right)}{\Phi_c(\eta)^2}$$

and

$$\frac{d\Lambda(\Phi_c^{-1}(\mu))}{d\mu} = \frac{2\Phi_c(\eta^*)\Phi_c\left(\frac{(1-\alpha)\eta^*}{\beta}\right) - I(\eta^*)}{\Phi_c(\eta^*)^2} =: \Pi(\eta^*),$$

where $\eta^* = \eta^*(\mu) = \Phi_c^{-1}(\mu)$. It is sufficient to show that $\Pi(\eta^*)$ is non-decreasing in η^* for $\eta^* \geq 0$.

We have

$$\Pi'(\eta) = \sqrt{\frac{2}{\pi}} \cdot \frac{e^{-\eta^2/2}}{\Phi_c(\eta)^3}\left(2 \cdot \Phi_c(\eta)\Phi_c\left(\frac{(1-\alpha)\eta}{\beta}\right) - I(\eta) - \frac{1-\alpha}{\beta} \cdot e^{\frac{\alpha(1-\alpha)}{\beta^2}\cdot\eta^2}\Phi_c(\eta)^2\right)$$
$$=: \sqrt{\frac{2}{\pi}} \cdot \frac{e^{-\eta^2/2}}{\Phi_c(\eta)^3} \cdot \Omega(\eta).$$

We need to show that $\Omega(\eta) \geq 0$ for $\eta \geq 0$. We will do this by showing that $\Omega'(\eta) \leq 0$ for $\eta \geq 0$ and that $\lim_{\eta\to\infty} \Omega(\eta) = 0$. The latter is obvious, so let us show the former.

$$\Omega'(\eta) = -\frac{2\alpha(1-\alpha)^2}{\beta^3} \cdot e^{\frac{\alpha(1-\alpha)}{\beta^2}\eta^2} \cdot \Phi_c(\eta)^2 \cdot \eta \leq 0$$

for $\eta \geq 0$. □

Now we are ready to prove Theorem 3.4.1. Let us first assume that all the parts have measure $\Omega(1)$. Later we will show that this assumption can be removed. W.l.o.g.

we assume that functions from the family have subsets integers as a range. We have,

$$\begin{aligned}
p_1 &\leq \Pr_{\substack{u,v\sim S^{d-1}\\ h\sim\mathcal{H}}}\left[h(u)=h(v) \;\middle|\; \|u-v\|\leq\tau\right] \\
&= \mathop{\mathrm{E}}_{h\sim\mathcal{H}}\left[\sum_i \mu(h^{-1}(i))\Pr[v\in h^{-1}(i) \mid u\in h^{-1}(i), \|u-v\|\leq\tau]\right] \\
&\leq \mathop{\mathrm{E}}_{h\sim\mathcal{H}}\left[\sum_i \mu(h^{-1}(i))\Lambda(\tau,\Phi_c^{-1}(\mu(h^{-1}(i))))\right]+o(1) \\
&\leq \Lambda\left(\tau,\Phi_c^{-1}\left(\mathop{\mathrm{E}}_{h\sim\mathcal{H}}\left[\sum_i \mu(h^{-1}(i))^2\right]\right)\right)+o(1) \\
&\leq \Lambda(\tau,\Phi_c^{-1}(p^*(\mathcal{H})))+o(1),
\end{aligned}$$

where:

- the first step is by the definition of p_1;
- the third step is due to the condition $\mu(h^{-1}(i))=\Omega(1)$ and Lemma 3.9.1;
- the fourth step is due to Lemma 3.9.2 and the assumption $\mu(h^{-1}(i))\leq 1/2$;
- the final step is due to the definition of $p^*(\mathcal{H})$.

To get rid of the assumption that a measure of every part is $\Omega(1)$ observe that all parts with measure at most ε contribute to the expectation at most $\varepsilon\cdot T$, since there are at most T pieces in total. Note that if $\varepsilon=o(1)$, then $\varepsilon\cdot T=o(1)$, since we assume T being fixed.

3.10 Appendix: Further description of experiments

In order to compare meaningful running time numbers, we have written fast C++ implementations of both the cross-polytope LSH and the hyperplane LSH. This enables a fair comparison since both implementations have been optimized by us to the same degree. In particular, hyperplane hashing can be implemented efficiently using a matrix-vector multiplication sub-routine for which we use the Eigen library (Eigen is

also used for all other linear algebra operations). For the fast pseudo-random rotation in the cross-polytope LSH, we have written a SIMD-optimized version of the Fast Hadamard Transform (FHT). We compiled our code with g++ 4.9 and the -O3 flag. All experiments except those in Table 3.3 ran on an Intel Core i5-2500 CPU (3.3 - 3.7 GHz, 6 MB cache) with 8 GB of RAM. Since 8 GB of RAM was too small for the larger values of n, we ran the experiments in Table 3.3 on a machine with an Intel Xeon E5-2690 v2 CPU (3.0 GHz, 25 MB cache) and 512 GB of RAM.

In our experiments, we evaluate the performance of the cross-polytope LSH on the following data sets. Figure 3-4 shows the distribution of distances to the nearest neighbor for the four data sets.

random For the random data sets, we generate a set of n points uniformly at random on the unit sphere. In order to generate a query, we pick a random point q' from the data set and generate a point at distance R from q' on the unit sphere. In our experiments, we vary the dimension of the point set between 128 and 1,024. Experiments with the random data set are useful because we can study the impact of various parameters (e.g., the dimension d or the number of points n) while keeping the remaining parameters constant.

pubmed / NYT The pubmed and NYT data sets contain bag-of-words representations of medical paper abstracts and newspaper articles, respectively [113]. We convert this representation into standard tf-idf feature vectors with dimensionality about 100,000. The number of points in the pubmed data set is about 8 million, for NYT it is 300,000. Before setting up the LSH data structures, we set 1000 data points aside as query vectors. When selecting query vectors, we limit our attention to points for which the inner product with the nearest neighbor is between 0.3 and 0.8. We believe that this is the most interesting range since near-duplicates (inner product close to 1) can be identified more efficiently with other methods, and points without a close nearest neighbor (inner product less than 0.3) often do not have a semantically meaningful match.

SIFT We use the standard data set of one million SIFT feature vectors from [91],

which also contains a set of 10,000 query vectors. The SIFT feature vectors have dimension 128 and (approximately) live on a sphere. We normalize the feature vectors to unit length but keep the original nearest neighbor assignments—this is possible because only a very small fraction of nearest neighbors changes through normalization. We include this data set as an example where the speed-up of the cross-polytope LSH is more modest.

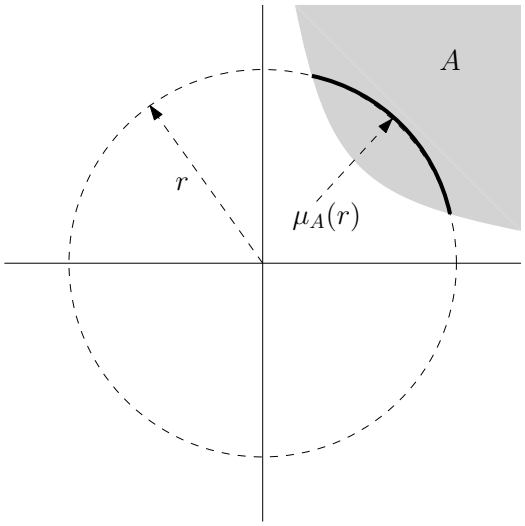
Method	k	Last CP dimension	Extra probes	**Query time (ms)**	Number of candidates	CP hashing time (ms)	Distances time (ms)
Single-probe	1	128	0	**6.7**	39800	0.01	6.3
Multiprobe	3	16	896	**0.51**	867	0.22	0.16

Table 3.2: Comparison of "standard" LSH using the cross-polytope (CP) hash vs. our multiprobe variant ($L = 10$ in both cases). On a random data set with $n = 2^{20}$, $d = 128$, and $R = \sqrt{2}/2$, the single-probe scheme requires 13× more time per query. Due to the larger value of k, the multiprobe variant performs fewer distance computations, which leads to a better trade-off between the hash computation time and the time spent on computing distances to candidates from the hash tables.

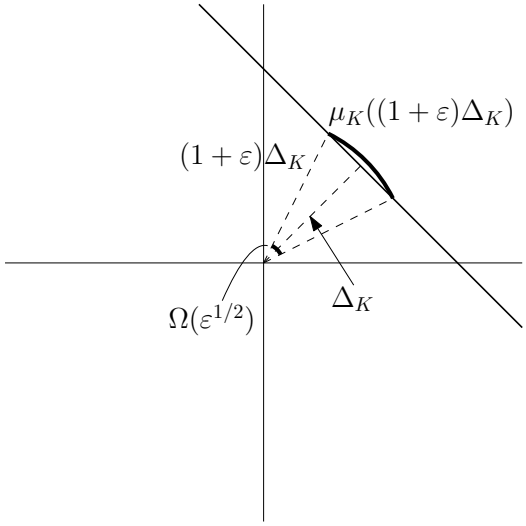
Data set size n	2^{20}	2^{22}	2^{24}	2^{26}	2^{28}
HP query time (ms)	2.6	7.4	25	63	185
CP query time (ms)	0.75	1.4	3.1	8.8	18
Speed-up	**3.5×**	**5.3×**	**8.1×**	**7.2×**	**10.3×**
k for CP	3 (16)	3 (64)	3 (128)	4 (2)	4 (64)

Table 3.3: Average running times for a single nearest neighbor query with the hyperplane (HP) and cross-polytope (CP) algorithms on a random data set with $d = 128$ and $R = \sqrt{2}/2$. The cross-polytope LSH is up to 10× faster than the hyperplane LSH. The last row of the table indicates the optimal choice of k for the cross-polytope LSH and (in parenthesis) the dimension of the last of the k cross-polytopes; all other cross-polytopes have full dimension 128. Note that the speed-up ratio is not monotonically increasing because the cross-polytope LSH performs better for values of n where the optimal setting of k uses a last cross-polytope with high dimension.

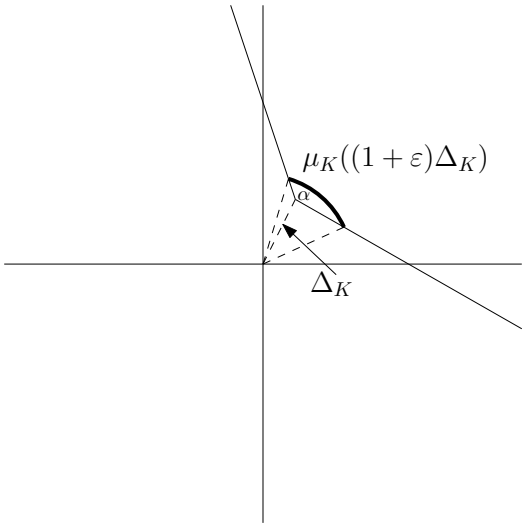
Figure 3-3: Gaussian measure of planar sets



(a) Defintion of $\mu_A(r)$



(b) For Lemma 3.7.2



(c) For Lemma 3.7.3

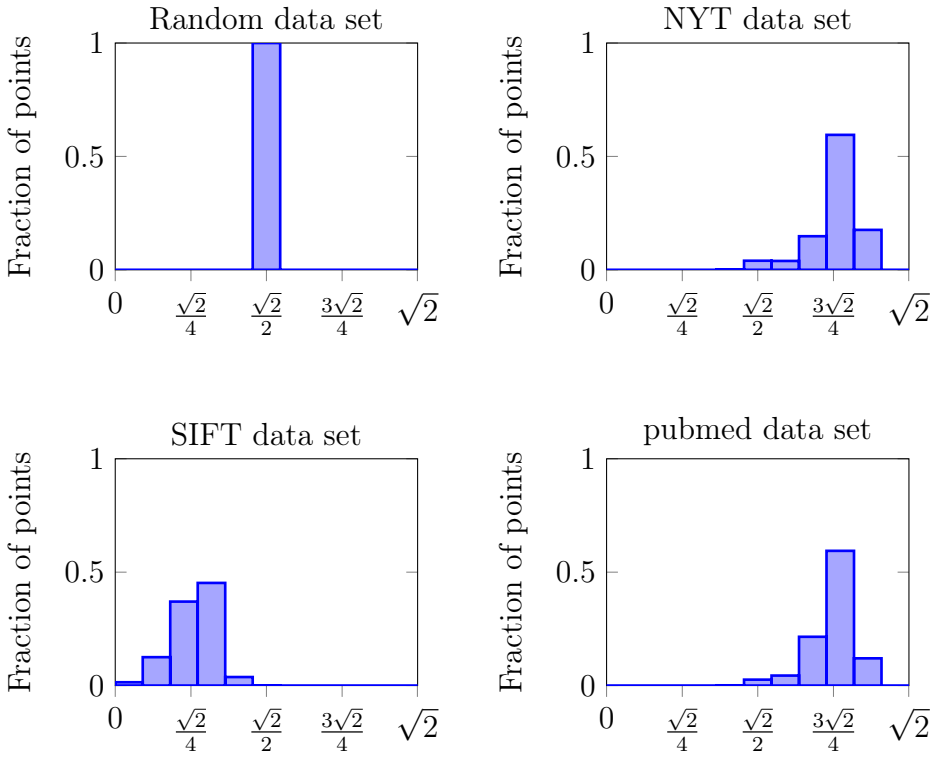


Figure 3-4: Distance to the nearest neighbor for the four data sets used in our experiments. The SIFT data set has the closest nearest neighbors.

Chapter 4

ANN algorithms for general symmetric norms

4.1 Introduction

In this chapter we will study the ANN problem for a general class of metrics. The best-studied metrics are the Hamming (ℓ_1) and the Euclidean (ℓ_2) distances. There are good reasons for this: ℓ_1 and ℓ_2 are very common in applications and admit very efficient algorithms based on *hashing*, in particular, Locality-Sensitive Hashing (LSH) [85, 13] and its data-dependent versions [18, 26]. Hashing-based algorithms for ANN over ℓ_1/ℓ_2 have now been the subject of a two-decade-long line of work, leading to a very good understanding of algorithms and their limitations. All such algorithms for c-approximate ANN obtain space $n^{1+\rho_u+o(1)}$ and query time $n^{\rho_q+o(1)}$ for some exponents ρ_u and $\rho_q < 1$ dependent on c; e.g., the most recent algorithm from Chapter 2 gives *tight* time–space trade-offs for every approximation factor $c > 1$.[1] We point the reader the introduction and Chapter 2, which summarize the state of affairs of the high-dimensional ANN over ℓ_1/ℓ_2. A practical perspective is presented in the surveys [171, 172].

Beyond ℓ_1 and ℓ_2, the landscape of ANN is much more mysterious, despite having

[1] The exact dependence, for ℓ_2, is that one can achieve any $\rho_u, \rho_q \geq 0$ satisfying $c^2\sqrt{\rho_q} + (c^2 - 1)\sqrt{\rho_u} = \sqrt{2c^2 - 1}$.

received significant attention. In 1998, [81] showed an efficient data structure for ℓ_∞ for $c = O(\log \log d)$ approximation. There are a few extensions of this result to other metrics, some of which proceed via *embedding* a metric into ℓ_∞ (see Section 4.1.3). However, we are still very far from having a general recipe for ANN data structures for *general* metrics with a non-trivial approximation; this is in stark contrast with the success of the low-dimensional regime. This state of affairs motivates the following broad question.

Problem 4.1.1. For a given approximation $c > 1$, which metric spaces allow efficient ANN algorithms?

An algorithm for general metrics is highly desirable both in theory and in practice. From the theoretical perspective, we are interested in a common theory of ANN algorithms for a wide class of distances. Such a theory would yield data structures (or impossibility results) for a variety of important distance measures for which we still do not know efficient ANN algorithms (e.g., matrix norms, the Earth Mover's Distance (EMD), the edit distance, etc.). Perhaps even more tantalizing is understanding what exactly makes some distances harder than others, and how to quantify that hardness. From the practical perspective, it is also desirable to have a generic algorithm: one that either uses the underlying distance measure as a black box, or provides a "knob" to easily specialize to any desired distance. In practice, one must oftentimes tune the distance to the specifics of the application, and hence algorithms that allow such tuning without major re-implementations are preferred.

In this paper, we focus on the following important case of Problem 4.1.1.

Problem 4.1.2. Solve Problem 4.1.1 for high-dimensional *normed* spaces.

(See Section 1.4.1 for the definition of a norm).

Norms are important for two reasons. First, most metric spaces arising in applications are actually norms (e.g., the Earth-Mover Distance [134]). Second, norms are geometrically nicer than general metrics, so there is hope for a coherent theory (e.g., for the problems of *sketching* and *streaming* norms, see the generic results of Chapter 5 and [42]). Using embeddings into ℓ_2 [92, 32], one can solve ANN for *any norm* with

approximation $O\left(\sqrt{d/\varepsilon}\right)$, space $n^{1+\varepsilon}$, and query time n^{ε}, where $0 < \varepsilon < 1/2$ is a constant; however, no better results are known in general.

4.1.1 The main result

In this paper we nearly settle Problem 4.1.2 for *symmetric* norms, i.e., norms that are invariant under all permutations and changes of signs of the coordinates of a vector. We show the following general result:

Theorem 4.1.3. *For every n, $d = n^{o(1)}$, and every d-dimensional symmetric norm $\|\cdot\|$, there exists a data structure for ANN over $\|\cdot\|$ for n-point datasets with approximation $(\log\log n)^{O(1)}$ space $n^{1+o(1)}$, and query time $n^{o(1)}$.*

We note that the techniques behind Theorem 4.1.3 cannot be extended to *general* norms; see details in Section 4.1.6.

4.1.2 Why symmetric norms?

The class of symmetric norms is, in some sense, a sweet spot. On the one hand, symmetric norms are mathematically nice and, as we show, allow for a clean characterization that leads to an efficient ANN data structure (see the proof overview from Section 4.1.4). On the other hand, symmetric norms vastly generalize ℓ_p distances and enable many new interesting examples, some of which arise in applications. We first consider the following two examples of symmetric norms, which are crucial for the subsequent discussion.

The first important example is the *top-k norm:* the sum of k largest absolute values of the coordinates of a vector; $k = 1$ corresponds to ℓ_∞, while $k = d$ corresponds to ℓ_1. Another rich set of examples is that of *Orlicz norms:* for any non-zero convex function $G : \mathbb{R}_+ \to \mathbb{R}_+$ such that $G(0) = 0$, we define the *unit ball* of a norm $\|\cdot\|_G$ to be:

$$\left\{x \in \mathbb{R}^d \;\middle|\; \sum_{i=1}^{d} G\big(|x_i|\big) \le 1\right\}.$$

Clearly, for $1 \le p < \infty$ the ℓ_p norm is Orlicz via $G(t) = t^p$.

In statistics and machine learning, Orlicz norms are known as *M-estimators* (for the case of convex losses) [60]. A specific example is the *Huber loss*. Even though *non-convex* losses do not correspond to norms, our algorithm still can handle them (see Section 4.3).

Other examples of symmetric norms used in applications include:

- *k-support norm* [29] used for the sparse regression problem; its unit ball is the convex hull of $\{x \mid x \text{ is } k\text{-sparse}, \|x\|_2 \le 1\}$,
- *box-Θ norm* [123] (again, used for sparse regression), defined for $0 < a < b \le c$ and $\Theta = \{\theta \in [a,b]^d \mid \|\theta\|_1 \le c\}$ as $\|x\| = \min_{\theta\in\Theta} \left(\sum_{i=1}^d \frac{x_i^2}{\theta_i}\right)^{1/2}$, and its dual;
- *K-functional* [69] used to show tight tail bounds, defined for $t > 0$ as $\|x\| = \min\left\{\|x_1\|_1 + t\cdot\|x_2\|_2 \;\middle|\; x_1 + x_2 = x\right\}$,
- *$\|\cdot\|_{1,2,s}$ norms* [105] used for dimension reduction, defined as $\|x\| = \left(\sum_i \|x_{S_i}\|_1^2\right)^{1/2}$, where S_1 is the set of s largest absolute values of coordinates of x, S_2 is the set of next s largest coordinates, etc.

Finally, we show two simple ways to construct many interesting examples of symmetric norms. Let $0 = a_0 \le a_1 \le a_2 \le \ldots \le a_d$ be a non-decreasing sub-additive[2] sequence. We can define two norms associated with it [40]: a *minimal* norm is defined as

$$\|x\| = \max_{1\le k\le d} a_k \cdot (\text{average of the largest } k \text{ absolute values of the coordinates of } x)\,,$$

and a *maximal* norm is equal to

$$\|x\| = \sum_{k=1}^{d} (a_k - a_{k-1}) \cdot (k\text{-th largest absolute value of a coordinate of } x)\,.$$

The minimal norm is the *smallest* norm such that for every k one has:

$$\left\|(\underbrace{1,1,\ldots,1}_{k},0,0,\ldots,0)\right\| = a_k.$$

[2]For every n, m, one has $a_{n+m} \le a_n + a_m$.

Similarly, the maximal norm is the *largest* such norm. Minimal norms will provide hard examples of symmetric norms that preclude some simple(r) approaches to ANN (see Section 4.8.1). We also note that the dual (with respect to the standard dot product) of any symmetric norm is symmetric as well.

4.1.3 Prior work: ANN for norms beyond ℓ_1 and ℓ_2

For norms beyond ℓ_1 and ℓ_2, the cornerstone result in ANN is a data structure for ℓ_∞ due to Indyk [81]. For every $\varepsilon > 0$, the data structure achieves space $n^{1+\varepsilon}$, query time $n^{o(1)}$, and approximation $O_\varepsilon(\log \log d)$. This is a *doubly-exponential* improvement over embeddings of ℓ_∞ into ℓ_1/ℓ_2 which require distortion $\Omega\big(\sqrt{d}\big)$.

It is well-known [176] that *any* d-dimensional normed space embeds into ℓ_∞ with distortion $(1+\varepsilon)$, which raises the question: can we combine this embedding with the result from [81] to solve ANN for any norm? It turns out that the answer is negative: accommodating a norm of interest may require embedding into a very high-dimensional ℓ_∞. In the worst case, we need $2^{O_\varepsilon(d)}$ dimensions, and this bound is known to be tight [32], even for spaces as simple as ℓ_2. Even though this approach would give a non-trivial approximation of $O(\log \log 2^{O(d)}) = O(\log d)$, the resulting data structure has query time which is *exponential* in d; thus, this approach is interesting only for the low-dimensional regime $d = o(\log n)$.

The result of [81] has been extended as follows. In [82, 83, 16, 10] it was shown how to build data structures for ANN over arbitrary ℓ_p-*products* of metrics given that there exists an ANN data structure for every factor. Recall that the ℓ_p-product of metric spaces M_1, M_2, ..., M_k is a metric space with the ground set $M_1 \times M_2 \times \ldots \times M_k$ and the following distance function:

$$d\big((x_1, x_2, \ldots, x_k), (y_1, y_2, \ldots, y_k)\big) = \Big\| \big(d_{M_1}(x_1, y_1), d_{M_2}(x_2, y_2), \ldots, d_{M_k}(x_k, y_k)\big) \Big\|_p .$$

In a nutshell, if we can build efficient ANN data structures for every M_i with approximation c, there exist an efficient data structure for ANN over the product space with approximation $O(c \cdot \log \log n)$. Note that the above also implies ANN for the standard

ℓ_p, though for this case a better approximation $O(\log \log d)$ is possible via randomized embeddings into ℓ_∞ [10]. For small values of p, one can also get approximation $c = 2^{O(p)}$ [133, 36] for ANN for the ℓ_p distance using different techniques.

4.1.4 Overview of the proof of Theorem 4.1.3

We prove Theorem 4.1.3 in three steps.

- First, we build a data structure for d-dimensional top-k norms. We proceed by constructing a *randomized* embedding into d-dimensional ℓ_∞ with constant distortion, and then invoke the data structure for ANN over ℓ_∞ from [81].

  Our embedding is a refinement of the technique of *max-p-stable distributions* used in [10] to embed ℓ_p into ℓ_∞. Surprisingly, the technique turns out to be very general, and can handle top-k norms as well an *arbitrary* Orlicz norm.

  While this technique can handle even arbitrary *symmetric norms* (see Section 4.8), there exist symmetric norms, for which this approach leads to merely a $\log^{\Omega(1)} d$-approximation, which is exponentially worse than the bound we are aiming at (see Section 4.8.1).

- To bypass the above limitation and obtain the desired $(\log \log n)^{O(1)}$-approximation, we show the following structural result: *any* d-dimensional symmetric norm allows a constant-distortion (deterministic) embedding into a low-dimensional *iterated product* of top-k norms. More specifically, the host space Y is an ℓ_∞-product of $d^{O(1)}$ copies of the ℓ_1-product of $X_1, X_2, \ldots, X_d$, where X_k is $\mathbb{R}^d$ equipped with the top-k norm.

  The dimension of Y is $d^{O(1)}$ which is significantly better than the bound $2^{\Omega(d)}$ necessary to embed symmetric norms (even ℓ_2) into ℓ_∞. It is exactly this improvement over the naïve approach that allows us to handle any dimension $d = n^{o(1)}$ as opposed to the trivial $o(\log n)$.

- Finally, we use known results [82, 10], which allow us to construct a data structure for ANN over a product space if we have ANN data structures for the individual factors. Each such step incurs an additional $\log \log n$ factor in the resulting approximation.

Since we have built a data structure for top-k norms, and can embed a symmetric norm into an iterated product of top-k norms, we are done!

Embeddings into iterated product spaces have been successfully used before for constructing data structures for ANN over Fréchet distance [82], edit distance [83], and Ulam distance [16]. Theorem 4.1.3 gives yet another confirmation of the power of the technique.

4.1.5 Optimality of Theorem 4.1.3

There remains one aspects of Theorem 4.1.3 that can potentially be improved: the approximation factor $(\log\log n)^{O(1)}$.

One bottleneck for our algorithm is the ANN data structure for ℓ_∞ from [81], which gives $O(\log\log d)$ approximation. This bound is known to be tight [11, 98] for certain models of computation (in particular, for decision trees, which captures the result of [81]). Thus, going beyond approximation $\Omega(\log\log d)$ in Theorem 4.1.3 might be hard; however, it remains entirely possible to improve the approximation from $(\log\log n)^{O(1)}$ to $O(\log\log d)$, which we leave as an open question.

4.1.6 Lower bounds for general norms

The second step of the proof of Theorem 4.1.3 (see Section 4.1.4) shows how to embed any d-dimensional symmetric norm into a *universal* normed space of dimension $d^{O(1)}$ with a constant distortion. In contrast, we show that for *general* norms a similar universal construction is impossible. More formally, for a fixed $0 < \varepsilon < 1/3$, suppose U is a normed space such that for every d-dimensional normed space X there exists a *randomized* linear embedding of X into U with distortion $O(d^{1/2-\varepsilon})$. Then, U must have dimension at least $\exp\left(d^{\Omega_\varepsilon(1)}\right)$. By John's theorem [92], d-dimensional ℓ_2 is a universal space for distortion $\sqrt{d}$, so our lower bound is tight up to sub-polynomial factors. See Section 4.6 for details.

To take this a step further, it would be highly desirable to prove stronger hardness results for ANN over general norms. One approach would be to show that such a

norm X has high *robust expansion*, which is a property used to deduce ANN lower bounds [146, 24]. There exist *metrics* M that have high robust expansion, such as the shortest path metric of a spectral expander (see Section 4.9). To obtain a hard norm, it suffices to embed such an N-point metric M into a $\log^{O(1)} N$-dimensional norm with a constant distortion. The result of [120] shows that there exist N-point metrics M which *cannot* be embedded into any norm of dimension $N^{o(1)}$. However, these metrics are not expanders, and for expanders such a dimension reduction procedure might be possible. [3]

4.1.7 Other related work: dealing with general norms

The recent result of [42] completely characterizes the *streaming* complexity of any symmetric norm. Even though many symmetric norms (including ℓ_∞) are hard in the streaming model, the state of affairs with ANN is arguably much nicer. In particular, our results imply that all symmetric norms have *highly efficient* ANN data structures. We also point out that streaming algorithms for the special case of *Orlicz* norms have been studied earlier [45].

Another related work is [23], which shows that for norms, the existence of good *sketches* is equivalent to *uniform* embeddability into ℓ_2. Sketches are known to imply efficient ANN data structures, but since many symmetric norms do not embed into ℓ_2 uniformly, we conclude that ANN is provably easier than sketching for a large class of norms.

Finally, we also mention the work of [2], who study ANN under the class of high-dimensional distances which are *Bregman divergences*. These results are somewhat disjoint since the Bregman divergences are not norms.

[3] In a recent work, Naor [132] showed that this approach is impossible. He shows that embedding an N-point spectral expander with constant distortion into any normed space requires $\text{poly}(N)$ dimensions.

4.2 Preliminaries

4.2.1 Norms and products

For any subset $A \subseteq \mathbb{R}$, we let $\chi_A \colon \mathbb{R} \to \{0,1\}$ be the indicator function of A. For a vector $x \in \mathbb{R}^d$ we define $|x| = (|x_1|, |x_2|, \ldots, |x_d|)$ to be the vector of the absolute values of the coordinates of x.

Definition 4.2.1. For any vector $x \in \mathbb{R}^d$, we let $x^* = P|x|$ be the vector obtained by applying the permutation matrix P to $|x|$ so coordinates of x^* are sorted in non-increasing absolute value.

Definition 4.2.2 (Symmetric norm)**.** A norm $\|\cdot\|_X \colon \mathbb{R}^d \to \mathbb{R}$ is *symmetric* if for every $x \in \mathbb{R}^d$, $\|x\|_X = \big\||x|\big\|_X = \|x^*\|_X$.

See the introduction for examples of symmetric norms. We note once again that the dual norm of a symmetric norm is also symmetric.

A natural way to combine norms is via *product spaces*, which we will heavily exploit in this paper.

Definition 4.2.3 (Product space)**.** Let $1 \le p \le \infty$. Let (X_1, d_{X_1}), (X_2, d_{X_2}), $\ldots$, (X_k, d_{X_k}) be metric spaces. We define the *ℓ_p-product space*, denoted $\bigoplus_{\ell_p} X_i$, to be a metric space whose ground set is $X_1 \times X_2 \times \ldots \times X_k$, and the distance function is defined as follows: the distance between $(x_1, x_2, \ldots, x_k)$ and $(x'_1, x'_2, \ldots, x'_k)$ is defined as the ℓ_p norm of the vector $\big(d_{X_1}(x_1, x'_1), d_{X_2}(x_1, x'_2), \ldots, d_{X_k}(x_k, x'_k)\big)$.

Next we define the top-k norm:

Definition 4.2.4. For any $k \in [d]$, the *top-k norm*, $\|\cdot\|_{T(k)} \colon \mathbb{R}^d \to \mathbb{R}$, is the sum of the absolute values of the top k coordinates. In other words,

$$\|x\|_{T(k)} = \sum_{i=1}^{k} |x^*_i|,$$

where x^* is the vector obtained in Definition 4.2.1.

Definition 4.2.5. Given vectors $x, y \in \mathbb{R}^d$, we say x *weakly majorizes* y if for all $k \in [d]$,

$$\sum_{i=1}^{k} |x_i^*| \geq \sum_{i=1}^{k} |y_i^*|.$$

Lemma 4.2.6 (Theorem B.2 in [119])**.** *If $x, y \in \mathbb{R}^d$ where x weakly majorizes y, then for any symmetric norm $\|\cdot\|_X$,*

$$\|x\|_X \geq \|y\|_X.$$

Definition 4.2.7. For $i \in [d]$, let $\xi^{(i)} \in \mathbb{R}^d$ be the vector

$$\xi^{(i)} = (\underbrace{1, \ldots, 1}_{i}, \underbrace{0, \ldots, 0}_{d-i})$$

consisting of exactly i 1's, and $d - i$ 0's.

4.2.2 ANN for ℓ_∞ and ℓ_∞-products

We will crucially use the following two powerful results of Indyk. The first result is for the standard d-dimensional ℓ_∞ space.

Theorem 4.2.8 ([81, Theorem 1])**.** *For any $\varepsilon \in (0, 1/2)$, there exists a data structure for ANN for n-points datasets in the ℓ_∞^d space with approximation $O\left(\frac{\log\log d}{\varepsilon}\right)$, space $O(d \cdot n^{1+\varepsilon})$, and query time $O(d \cdot \log n)$.*

The second is a generalization of the above theorem, which applies to an ℓ_∞-product of k metrics $X_1, \ldots X_k$, and achieves approximation $O(\log\log n)$. It only needs black-box ANN schemes for each metric X_i.

Theorem 4.2.9 ([82, Theorem 1])**.** *Let $X_1, X_2, \ldots, X_k$ be metric space, and let $c > 1$ be a real number. Suppose that for every $1 \leq i \leq k$ and every n there exists a data structure for ANN for n-point datasets from X_i with approximation c, space $S(n) \geq n$, query time $Q(n)$, and probability of success 0.99. Then, for every $\varepsilon > 0$, there exists ANN under $\bigoplus_{\ell_\infty}^k \mathcal{M}$ with:*

- $O(\varepsilon^{-1} \log\log n)$ *approximation,*
- $O(Q(n) \log n + dk \log n)$ *query time, where* d *is the time to compute distances in each* X_i*, and*
- $S(n) \cdot O(kn^{\varepsilon})$ *space/preprocessing.*

Strictly speaking, we need to impose a technical condition on the ANN for each X_i — that it reports the point with the smallest *priority* — which is satisfied in all our scenarios; see [82, Section 2] for details. Also, the original statement of [82] gave a somewhat worse space bound. The better space results simply from a better analysis of the algorithm, as was observed in [16]; we include a proof in Section 4.7.

4.3 ANN for Orlicz and top-k norms

Before showing a data structure for general symmetric norms, we give an algorithm for general Orlicz norms. We then show how to apply these ideas to top-k norms. This restricted setting has a simple analysis and illustrates one of the main techniques used in the rest of the paper. A similar approach was used in prior work to construct randomized embeddings of ℓ_p norms into ℓ_∞, and solve the ANN search problem; here we show that these techniques are in fact applicable in much greater generality.

Lemma 4.3.1. *Let* $\|\cdot\|_G$ *be an Orlicz norm. For every* $D, \alpha > 1$ *and every* $\mu \in (0, 1/2)$ *there exists a randomized linear map* $f \colon \mathbb{R}^d \to \mathbb{R}^d$ *such that for every* $x \in \mathbb{R}^d$*:*

- *if* $\|x\|_G \le 1$*, then* $\Pr_f\left[\left\|f(x)\right\|_\infty \le 1\right] \ge \mu$*;*
- *if* $\|x\|_G > \alpha D$*, then* $\Pr_f\left[\left\|f(x)\right\|_\infty > D\right] \ge 1 - \mu^\alpha$*.*

Proof. Let the distribution $\mathcal{D}$ over $\mathbb{R}_+$ have the following CDF $F : \mathbb{R}_+ \to [0, 1]$:

$$F(t) = \Pr_{u \sim \mathcal{D}}[u \le t] = 1 - \mu^{G(t)}.$$

Consider the following randomized linear map $f : \mathbb{R}^d \to \mathbb{R}^d$:

$$(x_1, x_2, \ldots, x_d) \overset{f}{\mapsto} \left(\frac{x_1}{u_1}, \frac{x_2}{u_2}, \ldots, \frac{x_d}{u_d} \right)$$

where $u_1, \ldots, u_d \sim \mathcal{D}$ are i.i.d. samples from $\mathcal{D}$. Suppose that $\|x\|_G \leq 1$. Then, $\sum_{i=1}^d G(|x_i|) \leq 1$. This, in turn, implies:

$$\Pr_f \Big[\|f(x)\|_\infty \leq 1 \Big] = \prod_{i=1}^d \Pr_{u_i \sim \mathcal{D}} \left[\left| \frac{x_i}{u_i} \right| \leq 1 \right] = \prod_{i=1}^d \mu^{G(|x_i|)} = \mu^{\sum_{i=1}^d G(|x_i|)} \geq \mu.$$

Now suppose that $\|x\|_G > \alpha D$. This, together with the convexity of $G(\cdot)$, implies:

$$\sum_{i=1}^d G\left(\frac{|x_i|}{D} \right) \geq (1-\alpha) G(0) + \alpha \cdot \sum_{i=1}^d G\left(\frac{|x_i|}{\alpha D} \right) \geq \alpha.$$

Thus, we have:

$$\Pr_f \Big[\|f(x)\|_\infty \leq D \Big] = \prod_{i=1}^d \Pr_{u_i \sim \mathcal{D}} \left[\left| \frac{x_i}{u_i} \right| \leq D \right] = \prod_{i=1}^d \mu^{G(|x_i|/D)} = \mu^{\sum_{i=1}^d G(|x_i|/D)} \leq \mu^\alpha.$$

□

Theorem 4.3.2. *For every d-dimensional Orlicz norm $\|\cdot\|_G$ and every $\varepsilon \in (0, 1/2)$, there exists a data structure for ANN over $\|\cdot\|_G$, which achieves approximation $O\left(\frac{\log\log d}{\varepsilon^2}\right)$ using space $O\left(dn^{1+\varepsilon}\right)$ and query time $O\left(dn^\varepsilon\right)$.*

Proof. Let $P \subset \mathbb{R}^d$ be a dataset of n points. Consider the data structure which does the following:

1. For all $1 \leq i \leq n^\varepsilon$, we independently apply the randomized linear map f from Lemma 4.3.1 with parameters $\mu = n^{-\varepsilon}$, $D = O\left(\frac{\log\log d}{\varepsilon}\right)$, and $\alpha = \frac{2}{\varepsilon}$. We define

   $$P_i = \{f_i(x) \mid x \in P\}$$

   to be the image of the dataset under f_i, where f_i is the i-th independent copy of f.

2. For each $1 \le i \le n^{\varepsilon}$, we use Theorem 4.2.8 to build a data structure for ANN over ℓ_∞ with approximation D for dataset P_i. We refer to the i-th data structure as T_i.

Each T_i occupies space $O(dn^{1+\varepsilon})$ and achieves approximation D with query time $O(d \log n)$. To answer a query $q \in \mathbb{R}^d$, we query T_i with $f_i(q)$ for each $i \in [n^{\varepsilon}]$. Let x_i be the point returned by T_i, and let $p_i \in P$ be the pre-image of x_i under f_i, so that $f_i(p_i) = x_i$. If for some T_i, the point returned satisfies $\|p_i - q\|_G \le \alpha D$, then we return p_i.

- If there exists some $p \in P$ with $\|p - q\|_G \le 1$, then by Lemma 4.3.1, with probability $1 - (1 - n^{-\varepsilon})^{n^{\varepsilon}} \ge \frac{3}{5}$, some f_i has $\|f_i(p-q)\|_\infty \le 1$. Since f_i is linear, $\|f_i(p) - f_i(q)\|_\infty \le 1$ as well.

- Let $i \in [n^{\varepsilon}]$ be an index where some $p \in P$ with $\|p - q\|_G \le 1$ has $\|f_i(p) - f_i(q)\|_\infty \le 1$. Every other $p' \in P$ with $\|p' - q\|_G \ge \alpha D$ satisfies

$$\Pr\Big[\|f_i(p') - f_i(q)\|_\infty \le D\Big] \le \frac{1}{n^2}.$$

  A union bound over at most n points with distance greater than αD to q shows that except with probability at most $\frac{1}{n}$, T_i returns some $p_i \in P$ with $\|p_i - q\|_G \le \alpha D$. Thus, the total probability of success of the data structure is at least $\frac{3}{5} - \frac{1}{n}$.

The total query time is $O\left(dn^{\varepsilon} \cdot \log n\right)$ and the total space used is $O\left(dn^{1+2\varepsilon}\right)$. This data structure achieves approximation $\alpha D = O\left(\frac{\log \log d}{\varepsilon^2}\right)$. Decreasing ε by a constant factor, we get the desired guarantees. □

Remark. The construction of the randomized embedding in Lemma 4.3.1 and the data structure from Theorem 4.3.2 work in a somewhat more general setting, rather than just for Orlicz norms. For a fixed norm $\|\cdot\|$, we can build a randomized map $f\colon \mathbb{R}^d \to \mathbb{R}^d$ with the guarantees of Lemma 4.3.1 if there exists a non-decreasing $G\colon \mathbb{R}_+ \to \mathbb{R}_+$ where $G(0) = 0$, $G(t) \to \infty$ as $t \to \infty$, and for every $x \in \mathbb{R}^d$:

- if $\|x\| \le 1$, then $\sum_{i=1}^{d} G(|x_i|) \le 1$, and
- if $\|x\| \ge \alpha D$, then $\sum_{i=1}^{d} G\left(\frac{|x_i|}{D}\right) \ge \alpha$.

The data structure itself just requires the existence of a randomized linear map satisfying the conditions of Lemma 4.3.1.

We now describe how to obtain a data structure for ANN for any top-k norm.

Lemma 4.3.3. *Fix any $k \in [d]$. For every $D, \alpha > 1$ and every $\mu \in (0, 1/2)$, there exists a randomized linear map $f\colon \mathbb{R}^d \to \mathbb{R}^d$ such that for every $x \in \mathbb{R}^d$:*

- *if $\|x\|_{T(k)} \le 1$, then $\Pr_f\left[\|f(x)\|_\infty \le 1\right] \ge \mu$;*
- *if $\|x\|_{T(k)} > \alpha D$, then $\Pr_f\left[\|f(x)\|_\infty > D\right] \ge 1 - \mu^{\alpha-1}$.*

Proof. We define $G\colon \mathbb{R}_+ \to \mathbb{R}_+$ where for every $x \in \mathbb{R}^d$,

$$G(t) = t \cdot \chi_{[\frac{1}{k}, \infty)}(t)$$

If $\|x\|_{T(k)} \le 1$, there are at most k coordinates where $|x_i| \ge \frac{1}{k}$. Therefore,

$$\sum_{i=1}^{d} G(|x_i|) \le \|x\|_{T(k)} \le 1.$$

If $\|x\|_{T(k)} \ge \alpha D$, then $\sum_{i=1}^{k} |x_i^*| \ge \alpha D$. Therefore, $\sum_{i=1}^{d} G\left(\frac{|x_i^*|}{D}\right) \ge \sum_{i=1}^{k} G\left(\frac{|x_i^*|}{D}\right) \ge \alpha - 1$. The proof now follows in the same way as Lemma 4.3.1. □

Lemma 4.3.3 gives us a data structure for any top-k norm with approximation $O(\log \log d)$ applying Theorem 4.3.2.

One could imagine using a similar argument to design an algorithm for *general* symmetric norms. This idea indeed works and yields an algorithm with approximation $\widetilde{O}(\log d)$ for a general symmetric norm (see Section 4.8 for a detailed analysis of this approach). However, we show this strategy cannot achieve an approximation better than $\Omega(\sqrt{\log d})$ (see the end of the same Section 4.8).

4.4 Embedding into product spaces

In this section, we construct an embedding of general symmetric norms into product spaces of top-k norms. To state the main result of this section, we need the following definition.

Definition 4.4.1. For any $c_1, \dots, c_d \geq 0$, let $\bigoplus_{\ell_1}^d T^{(c)} \subset \mathbb{R}^{d^2}$ be the space given by the seminorm $\|\cdot\|_{T,1}^{(c)} \cdot \mathbb{R}^{d^2} \to \mathbb{R}$ where for $x = (x_1, \dots, x_d) \in \mathbb{R}^{d^2}$ and $x_1, \dots, x_d \in \mathbb{R}^d$:

$$\|x\|_{T,1}^{(c)} = \sum_{k=1}^{d} c_k \|x_k\|_{T(k)}.$$

Theorem 4.4.2 (Embedding into a product space)**.** *For any constant $\delta \in (0, 1/2)$, any symmetric norm $\|\cdot\|_X \colon \mathbb{R}^d \to \mathbb{R}$ can be embedded linearly with distortion $1+\delta$ into $\bigoplus_{\ell_\infty}^t \bigoplus_{\ell_1}^d T^{(c)}$ where $t = d^{O(\log(1/\delta)\delta^{-1})}$. In particular, there exists $c \in \mathbb{R}_+^{t \times d}$ such that for every $x \in \mathbb{R}^d$,*

$$(1-\delta)\|x\|_X \leq \max_{i \in [t]} \left(\sum_{k=1}^{d} c_{i,k} \|x\|_{T(k)} \right) \leq (1+\delta)\|x\|_X. \tag{4.1}$$

The vectors in $\bigoplus_{\ell_\infty}^t \bigoplus_{\ell_1}^d T^{(c)} \subset \mathbb{R}^{td^2}$ can be broken up into td blocks of d coordinates each. The embedding referenced above will simply map $x \in \mathbb{R}^d$ into $\mathbb{R}^{td^2}$ by making each of the td many blocks equal to a copy of x. The non-trivial part of the above theorem is setting the constants $c_{i,k}$ for $i \in [t]$ and $k \in [d]$ so (4.1) holds. Before going on to give the proof of Theorem 4.4.2, we establish some definitions and propositions which will be used in the proof. For the subsequent sections, let $\beta \in (1, 2)$ be considered a constant close to 1.

Definition 4.4.3 (Levels and Level Vectors)**.** For any fixed vector $x \in \mathbb{R}^d$ and any $k \in \mathbb{Z}$, we define *level k with respect to x* as $B_k(x) = \{i \in [d] \mid \beta^{-k-1} < |x_i| \leq \beta^{-k}\}$. Additionally, we let $b_k(x) = |B_k(x)|$ be the *size of level k with respect to x*. The *level*

vector of x, $V(x) \in \mathbb{R}^d$ is given by

$$V(x) = (\underbrace{\beta^k, \dots, \beta^k}_{b_{-k}(x) \text{ times}}, \underbrace{\beta^{k-1}, \dots, \beta^{k-1}}_{b_{-k+1}(x) \text{ times}}, \dots, \underbrace{\beta^{-k}, \dots, \beta^{-k}}_{b_k(x) \text{ times}}, 0, \dots 0)$$

where k is some integer such that all non-zero coordinates lie in some level between $-k$ and k. We say the i-th *level vector* $V_i(x) \in \mathbb{R}^d$ is given by

$$V_i(x) = (\underbrace{\beta^{-i}, \dots, \beta^{-i}}_{b_i(x) \text{ times}}, 0, \dots, 0).$$

The notation used for level vectors appears in [42]; however, we refer to level k as the coordinates of x lying in $(\beta^{-k-1}, \beta^{-k}]$; whereas [42] refers to level k as the coordinates of x lying in $[\beta^{k-1}, \beta^k)$.

Definition 4.4.4. Fix some $\tau > 0$. For any vector $x \in \mathbb{R}^d$, let $C(x) \in \mathbb{R}^d$ be the vector where each $i \in [d]$ sets

$$C(x)_i = \begin{cases} x_i & |x_i| \geq \tau \\ 0 & |x_i| < \tau \end{cases}.$$

Proposition 4.4.5 (Proposition 3.4 in [42])**.** *Let $\|\cdot\|_X$ be any symmetric norm and $x \in \mathbb{R}^d$ be any vector. Then*

$$\frac{1}{\beta}\|V(x)\|_X \leq \|x\|_X \leq \|V(x)\|_X.$$

Proposition 4.4.6. *Let $\|\cdot\|_X$ be any symmetric norm. For any vector $x \in \mathbb{R}^d$,*

$$\|x\|_X - \tau d \leq \|C(x)\|_X \leq \|x\|_X.$$

Proof. Note that x weakly majorizes $C(x)$, so $\|C(x)\|_X \leq \|x\|_X$. For the other direction, let $v = x - C(x)$. Then v has all coordinates with absolute value at most τ,

so $\tau d\xi^{(1)}$ weakly majorizes v. Therefore,

$$\|x\|_X \leq \|C(x)\|_X + \|v\|_X \leq \|C(x)\|_X + \tau d.$$

□

Intuitively, the above two propositions say that up to multiplicative loss β and additive loss τd in the norm of the vector, we may assume that all coordinates are exactly β^j for $j \geq \log_\beta(\tau)$. Thus, if $x \in \mathbb{R}^d$, then

$$\|x\|_X - \tau d \leq \|V(C(x))\|_X \leq \beta\|x\|_X.$$

If additionally, we let $\tau = \frac{\beta}{d^2}$, so when $\|x\|_X \leq 1$ there are at most $2\log_\beta d$ non-empty levels in $V(C(x))$.

Definition 4.4.7 (Rounded counts vector)**.** Fix any level vector $x \in \mathbb{R}^d$. The *rounded counts vector* of x, $W(x) \in \mathbb{R}^d$ is given by y where the $y \in \mathbb{R}^d$ is constructed using the following procedure:

Initialize $y = (0, \ldots, 0) \in \mathbb{R}^d$ and $c = d$.
for $k = -\infty, \ldots, 2\log_\beta(d) - 1$ **do**
    if $b_k(x) \geq 0$ **then**
        Let $j \in \mathbb{Z}_+$ be the integer where $\beta^{j-1} < b_k(x) \leq \beta^j$.
        if $c \geq \lfloor \beta^j \rfloor$ **then**
            Set the first $\lfloor \beta^j \rfloor$ zero-coordinates of y with β^{-k}. Update $c \leftarrow c - \lfloor \beta^{-k} \rfloor$.
Return y

Intuitively, $W(x)$ represents the level vector of x where we ignore coordinates smaller than $\frac{\beta}{d^2}$, and additionally, we round the counts of coordinates to powers of β.

Lemma 4.4.8. *For every vector $x \in \mathbb{R}^d$ and any symmetric norm $\|\cdot\|_X$,*

$$\|x\|_X - \tau d \leq \|W(V(C(x)))\|_X \leq \beta^2\|x\|_X.$$

Proof. The bound $\|x\|_X - \tau d \leq \|W(V(C(x)))\|_X$ follows by combining Proposition 4.4.5 and Proposition 4.4.6, as well as the monotonicity of norms. The bound $\|W(V(C(x)))\|_X \leq \beta^2 \|x\|_X$ follows from Proposition 4.4.5, Proposition 4.4.6, as well as Lemma 3.5 from [42]. □

In order to simplify notation, we let $R\colon \mathbb{R}^d \to \mathbb{R}^d$ given by $R(x) = W(V(C(x)))$.

Definition 4.4.9. Let the set $\mathcal{L} \subset \mathbb{R}^d_+$ be given by

$$\mathcal{L} = \{y \in \mathbb{R}^d_+ \mid y_1 \geq \dots y_d \geq 0\}.$$

Additionally, for an arbitrary symmetric norm $\|\cdot\|_X$ with dual norm $\|\cdot\|_{X^*}$, we let the set $\mathcal{R} \subset \mathcal{L}$ be given by

$$\mathcal{R} = \{R(y) \in \mathbb{R}^d_+ \mid y \in \mathcal{L} \cap B_{X^*}\}.$$

Definition 4.4.10. Fix a vector $y \in \mathcal{L} \setminus \{0\}$ (y has non-negative, non-increasing coordinates). Let the *maximal seminorm* with respect to y, $\|\cdot\|_y\colon \mathbb{R}^d \to \mathbb{R}$ be the seminorm where for every $x \in \mathbb{R}^d$,

$$\|x\|_y = \langle |x^*|, y\rangle.$$

We first show there exists some setting of $c \in \mathbb{R}^d$ such that we may compute $\|x\|_y$ as $\oplus^d_{\ell_1} T^{(c)}$.

Lemma 4.4.11. *For every vector $y \in \mathcal{L} \setminus \{0\}$, there exists $c_1, \dots, c_d \geq 0$ where for all $x \in \mathbb{R}^d$,*

$$\|x\|_y = \|x\|^{(c)}_{T,1}.$$

Proof. For $k \in [d]$, we let $c_k = y_k - y_{k+1}$, where $y_{d+1} = 0$.

$$\langle |x^*|, y\rangle = \sum_{i=1}^d |x_i^*| y_i = \sum_{i=1}^d |x_i^*| \left(\sum_{k=i}^d c_k\right) = \sum_{k=1}^d c_k \left(\sum_{i=1}^k |x_i^*|\right) = \sum_{k=1}^d c_k \|x\|_{T(k)}$$

□

Given Lemma 4.4.11, it suffices to show that for an arbitrary symmetric norm $\|\cdot\|_X$, we may compute $\|x\|_X$ (with some distortion) as a maximum over many maximal seminorms. In the following lemma, we show that taking the maximum over maximal norms from $\mathcal{R}$ suffices, but gives sub-optimal parameters. We then improve the parameters to prove Theorem 4.4.2.

Lemma 4.4.12. *Let $\|\cdot\|_X$ be an arbitrary symmetric norm and let $\|\cdot\|_{X^*}$ be its dual norm. Then for any $\|x\|_X \le 1$,*

$$\|x\|_X - \tau d \le \max_{y\in\mathcal{R}} \|x\|_y \le \beta^2 \|x\|_X.$$

Proof. Without loss of generality, we rescale the norm so that $\|e_1\|_{X^*} = 1$, where e_1 is the first standard basis vector. Consider any $x \in \mathbb{R}^d$ with $\|x\|_X \le 1$. Then since $\|\cdot\|_X$ is symmetric, we may assume without loss of generality that all coordinates of x are non-negative and in non-increasing order. Thus for each $y \in \mathcal{L} \cap \{0\}$, we have $\|x\|_y = \langle x, y\rangle$.

The lower bound simply follows from the fact that $R(z)$, other than coordinates less than τ, is monotonically above z, and all coordinates in x are non-negative. More specifically,

$$\|x\|_X = \sup_{z\in\mathcal{L}\cap B_{X^*}} \langle x, z\rangle \le \sup_{z\in\mathcal{L}\cap B_{X^*}} \langle x, R(z)\rangle + \tau d = \max_{y\in\mathcal{R}} \langle x, y\rangle + \tau d,$$

where τd comes from the fact that because $\|x\|_X \le 1$, every coordinate of x is at most 1. On the other hand, we have

$$\max_{y\in\mathcal{R}} \langle x, y\rangle = \beta^2 \max_{y\in\mathcal{R}} \langle x, \frac{y}{\beta^2}\rangle \le \beta^2 \sup_{z\in B_{X^*}} \langle x, z\rangle = \beta^2 \|x\|_X,$$

where we used the fact that $\|\frac{y}{\beta^2}\|_{X^*} \le 1$ by Lemma 4.4.8. □

Given Lemma 4.4.12, it follows that we may linearly embed X into $\oplus^t_{\ell_\infty} \oplus^d_{\ell_1} T^{(c)}$ where $t = |\mathcal{R}|$, with distortion $\frac{\beta^2}{1-\tau d} \le \beta^3$ (where we used the fact $\tau = \frac{\beta}{d}$ and that $1 + \beta/d \le \beta$ for a large enough d). The embedding follows by copying the vector x

into the t spaces $\oplus_{\ell_1}^d T^{(c)}$ corresponding to each vector $y \in \mathcal{R}$ given in Lemma 4.4.11. The one caveat is that this embedding requires t copies of $\oplus_{\ell_1}^d T^{(c)}$, and t is as large as $\left(\log_\beta d + 1\right)^{2\log_\beta d} = d^{O(\log\log d)}$. This is because there are at most $2\log_\beta d$ many levels, and each contains has number of coordinates being some value in $\{\beta^i\}_{i=0}^{\log_\beta d}$. Thus, our algorithm becomes inefficient once $d \geq 2^{\omega\left(\frac{\log n}{\log\log n}\right)}$.

In order to avoid this problem, we will make the embedding more efficient by showing that we do not need all of $\mathcal{R}$, but rather a fine net of $\mathcal{R}$. In addition, our net will be of polynomial size in the dimension, which gives an efficient algorithm for all $\omega(\log n) \leq d \leq n^{o(1)}$. We first show that it suffices to consider fine nets of $\mathcal{R}$, and then build a fine net of $\mathcal{R}$ of size $\mathrm{poly}(d)$.

Lemma 4.4.13. *Fix an $\gamma \in (0, 1/2)$. Let $\|\cdot\|_X$ be an arbitrary symmetric norm and $\|\cdot\|_{X^*}$ be its dual norm. If N is a γ-net of $\mathcal{R}$ with respect to distance given by $\|\cdot\|_{X^*}$, then*

$$(1 - \gamma - \tau d)\|x\|_X \leq \max_{y \in N} \|x\|_y \leq (\beta^2 + \gamma)\|x\|_X$$

Proof. Since the embedding we build is linear, it suffices to show that every vector $x \in \mathbb{R}^d$ with $\|x\|_X = 1$ has

$$1 - \gamma - \tau d \leq \max_{y \in N} \|x\|_y \leq (\beta^2 + \gamma).$$

Consider a fixed vector $x \in \mathbb{R}^d$ with $\|x\|_X = 1$. Additionally, we may assume the coordinates of x are non-negative and in non-increasing order. We simply follow the computation:

$$\begin{aligned}\|x\|_X = \||x^*|\|_X &\leq \max_{y \in \mathcal{R}} \langle |x^*|, y\rangle + \tau d \\ &= \max_{y \in N} \left(\langle |x^*|, y\rangle + \langle |x^*|, v\rangle\right) + \tau d \leq \max_{y \in N} \langle |x^*|, y\rangle + \gamma\|x\|_X + \tau d,\end{aligned}$$

where we used Lemma 4.4.12 and the fact that $\|v\|_{X^*} \leq \gamma$ in a γ-net of $\mathcal{R}$ with respect to the distance given by $\|\cdot\|_{X^*}$. On the other hand,

$$\max_{y \in N} \|x\|_y = \max_{y \in \mathcal{R}} \left(\langle |x^*|, y\rangle + \langle |x^*|, v\rangle\right) \leq \max_{y \in \mathcal{R}} \|x\|_y + \gamma\|x\|_X \leq (\beta^2 + \gamma)\|x\|_X,$$

where again, we used Lemma 4.4.12 and the fact that $\|v\|_{X^*} \leq \gamma$. $\square$

Finally, we conclude the theorem by providing a γ-net for $\mathcal{R}$ of size $d^{O(\log(1/\gamma)\gamma^{-1})}$.

Lemma 4.4.14. *Fix any symmetric space X with dual X^*. There exists an $8(\beta-1)$-net of size $d^{O(\log(1/(\beta-1))/\log\beta)}$ for $\mathcal{R}$ with respect to distances given by $\|\cdot\|_{X^*}$.*

We defer the proof of Lemma 4.4.14 to the next section. The proof of Theorem 4.4.2 follows by combining Lemma 4.4.11, Lemma 4.4.13, and Lemma 4.4.14. In particular, given a $\frac{\beta-1}{8}$-net of $\mathcal{R}$, we get an embedding with distortion at most $(\beta^2 + 8(\beta - 1))(1 + (8(\beta-1) + \tau d)^2)$ from Lemma 4.4.13. We let $\tau = \frac{\beta}{d^2}$ and $\beta = \sqrt{1+\delta/100}$ to get the desired linear embedding with distortion $1+\delta$. We now proceed to proving Lemma 4.4.14, which gives the desired upper bound on the size of the net.

4.4.1 Proof of Lemma 4.4.14: bounding the net size

We now give an upper bound on the size of a fine net of $\mathcal{R}$. We proceed by constructing a further simplification of $R(x)$. Intuitively we show that one can ignore the higher levels if there are fewer coordinates in the higher levels than some lower level.

Lemma 4.4.15. *Let $\|\cdot\|_X$ be a symmetric norm. Consider any nonnegative vector $x \in \mathbb{R}^d_+$ as well as two indices $u, v \in [d]$. Let $y \in \mathbb{R}^d_+$ be the vector with:*

$$y_k = \begin{cases} x_k & k \in [d] \setminus \{u, v\} \\ x_u + x_v & k = u \\ 0 & k = v \end{cases}.$$

Then $\|y\|_X \geq \|x\|_X$.

Proof. Consider the vector $z \in B_{X^*}$ where $\langle x, z\rangle = \|x\|_X$. Now, we let $z' \in \mathbb{R}^d$ be given by

$$z'_k = \begin{cases} z_k & k \in [d] \setminus \{u, v\} \\ \max\{z_u, z_v\} & k = u \\ \min\{z_u, z_v\} & k = v \end{cases}$$

Note that z' is a permutation of z, so $z' \in B_{X^*}$. Now,

$$\langle y, z' \rangle = (x_u + x_v) \max\{z_u, z_v\} + \sum_{k \in [d] \setminus \{u,v\}} x_k z_k \geq \sum_{k \in [d]} x_k z_k = \langle x, z \rangle = \|x\|_X.$$

□

Definition 4.4.16. Consider a vector $x \in \mathcal{R}$. We define the *simplified rounded vector* $S(x)$ as the vector returned by the following procedure.

1: Initialize $z = x$
2: **for** $k = 0, 1, \ldots, 2\log_\beta(d) - 1$ **do**
3: **if** $b_k(z) \leq \max_{j < k+3\log_\beta(\beta-1)} b_j(z)$ **then**
4: Set all coordinates of z of value β^{-k} to 0 i.e. set $b_k(z) = 0$.
5: Sort the coordinates of z in non-increasing order and return z.

Next we show that the simplified rounded vector is close to the rounded counts vector.

Lemma 4.4.17. *Let $\|\cdot\|_X$ be a symmetric norm and let $x \in \mathcal{R}$. Then $\|S(x) - x\|_X \leq 2(\beta - 1)\|x\|_X$.*

Proof. Consider some $k \in [2\log_\beta d - 1]$ and let $C_k(x) \subset [d]$ be set of coordinates where x is at level k and does not equal $S(x) = z$, i.e.,

$$C_k(x) = \{i \in [d] \mid x_i = \beta^{-k} \text{ and } x_i \neq z_i\}.$$

Additionally, for each $k \in [2\log_\beta d - 1]$, let $T_k \subset [d]$ be the coordinates at level k in x which trigger line 3 of $S(x)$, and thus become 0's in z (we do not need to consider the case $k = 0$ since line 3 never triggers, in fact, we do not need to consider $k \in [-3\log_\beta(\beta - 1)]$ either). In other words,

$$T_k(x) = \{i \in [d] \mid x_i = \beta^{-k} \text{ and at iteration } k \text{ of } S(x), b_k(z) \leq \max_{j < k+3\log_\beta(\beta-1)} b_j(z)\}.$$

Note that $T_1(x), \ldots, T_{2\log_\beta d - 1}(x)$ are all disjoint, and $|C_k(x)| \leq \sum_{j \in [k]} |T_j(x)|$, since whenever we zero out coordinates in levels less than or equal to k, $S(x)$ will shift

the coordinates when we sort, causing $x_i \neq z_i$. Thus, we may consider an injection $s_k \colon C_k(x) \to \bigcup_{j \in [k]} T_j(x)$, which charges coordinates in $C_k(x)$ to coordinates which were zeroed out in line 3 of $S(x)$.

Additionally, for each $j \in [2\log_\beta d - 1]$ where $T_j(x) \neq \emptyset$, we let q_j be the integer between 0 and $j + 3\log_\beta(\beta - 1)$ which triggered line 3 of $S(x)$ at $k = j$. More specifically, $0 \leq q_j \leq j + 3\log_\beta(\beta - 1)$ is the smallest integer for which $b_j(x) \leq b_{q_j}(x)$.

Finally, for each $j \in [2\log_\beta d - 1]$ where $T_j(x) \neq \emptyset$, we let $g_j \colon T_j(x) \to B_{q_j}(x)$ (recall that $B_{q_j}(x) \subset [d]$ are the indices of x at level q_j) be an arbitrary injection. Such an injection exists because $b_j(x) \leq b_{q_j}(x)$. We may consider the mapping $F \colon \bigcup_{k \in [2\log_\beta d - 1]} C_k(x) \to [d]$ where

$$F(i) = g_j(s_k(i)) \qquad \text{where } k \text{ and } j \text{ are such that } i \in C_k(x) \text{ and } s_k(i) \in T_j(x).$$

Let y be the vector where we "aggregate" coordinates of $\bigcup_{k \in [2\log_\beta(d) - 1]} C_k(x)$ of x according to the map F according to Lemma 4.4.15. In particular, we define $y \in \mathbb{R}^d$ where for $i \in [d]$, we let

$$y_i = \sum_{i' \in F^{-1}(i)} x_{i'}.$$

Note that for each $i \in [d]$, $0 \leq (x - z)_i \leq x_i$, and $\bigcup_{k \in [2\log_\beta(d) - 1]} C_k(x) \subset [d]$ are the non-zero coordinates of $x - z$. Thus, from Lemma 4.4.15, we conclude that $\|x - z\|_X \leq \|y\|_X$. We now turn to upper-bounding $\|y\|_X$.

Fix some $i \in [d]$ where $x_i = \beta^{-j}$. Then

$$y_i = \sum_{k > j - 3\log_\beta(\beta - 1)} \left(\sum_{k' \geq k} x_{s_{k'}^{-1}(g_k^{-1}(i))} \right),$$

where we interpret $x_{s_{k'}^{-1}(g_k^{-1}(i))}$ as 0 when $g_k^{-1}(i) = \emptyset$, or $s_{k'}^{-1}(g_k^{-1}(i)) = \emptyset$. Note that

whenever $x_{s_{k'}^{-1}(g_k^{-1}(i))} \neq 0$, $x_{s_{k'}^{-1}(g_k^{-1}(i))} = \beta^{-k'}$. Thus,

$$\begin{aligned} y_i &\leq \sum_{k > j - 3\log_\beta(\beta-1)} \left(\sum_{k' \geq k} \beta^{-k'} \right) \\ &\leq \sum_{k > j - 3\log_\beta(\beta-1)} \frac{\beta^{1-k}}{\beta - 1} \leq \frac{\beta^{1-j+3\log_\beta(\beta-1)}}{(\beta-1)^2} \leq \beta(\beta-1) \cdot \beta^{-j}. \end{aligned}$$

Recall that $x_i = \beta^{-j}$, so $\beta(\beta-1)x$ weakly majorizes y, and thus

$$\|x - z\|_X \leq \|y\|_X \leq \beta(\beta-1) \cdot \|x\|_X.$$

Hence, when $\beta \leq 2$, we have $\|x - z\|_X \leq 2(\beta-1)\|x\|_X$. □

Proof of Lemma 4.4.14. We now prove the theorem by showing that the set

$$N = \{S(x) \in \mathbb{R}^d \mid x \in \mathcal{R}\}$$

is a γ-net of $\mathcal{R}$, and we give an upper bound on the size. By Lemma 4.4.17, $\|x - S(x)\|_X \leq 2(\beta-1)\|x\|_X \leq 8(\beta-1)$. So it suffices to give an upper bound on the size of N.

We bound from above the number of net points by an encoding argument. Let $z = S(x)$ and let

$$t_k = \frac{b_k(z)}{\max_{j < k + 3\log_\beta(\beta-1)} b_j(z)}.$$

Let $k^* \in \{0, \ldots, 2\log_\beta d - 1\}$ be the smallest level k with non-zero $b_k(z)$. For all $j > k^* - 3\log_\beta(\beta-1)$, we either have $t_j(z) = 0$ or $t_j(z) \geq 1$. Additionally, z has d coordinates, so

$$\prod_{j = k^* - 3\log_\beta(\beta-1)}^{2\log_\beta d - 1} \max(t_j, 1) \leq d^{-3\log_\beta(\beta-1)},$$

since terms of the product "cancel" except for at most $-3\log_\beta(\beta-1)$, which are each at most d.

We will encode $z \in N$ in three steps. In the first step, we use $2\log_\beta d - 1$ bits

in order to encode whether $b_k(z) = 0$ or $b_k(z) > 0$. In the second step, we then encode the values $b_{k^*+j}(z)$ for $j \in \{0, \dots, 3\log_\beta(1/(\beta-1))\}$. Finally, we go through $j > k^* + 3\log_\beta\left(\frac{1}{\beta-1}\right)$, and encode t_i using a prefix-free code, where writing t_i uses at most $O(\log\max(t_i, 1))$ many bits. Thus, in total, the number of bits we use is

$$\begin{aligned} &O\left(\log_\beta d + \log d \log_\beta\left(\frac{1}{\beta-1}\right) + \sum_{j=k^*-3\log_\beta(\beta-1)}^{2\log_\beta d - 1} \log\max(t_j, 1)\right) \\ &\quad = O\left(\frac{\log d \cdot \log\left(\frac{1}{\beta-1}\right)}{\log\beta} + \log\left(\prod_{j=k^*-3\log_\beta(\beta-1)}^{2\log_\beta d-1} \max(t_j, 1)\right)\right) \\ &\quad = O\left(\frac{\log d \cdot \log\left(\frac{1}{\beta-1}\right)}{\log\beta}\right). \end{aligned}$$

Thus, we obtain N is a $8(\beta-1)$-net, and the size of N is $d^{O(\log(1/(\beta-1))/\log\beta)}$. □

4.5 Proof of the main theorem

We now prove our main result, Theorem 4.1.3. The algorithm here achieves approximation

$$O\left(\frac{\log^2\log n \cdot \log\log d}{\varepsilon^5}\right).$$

We proceed by giving an algorithm for $\bigoplus_{\ell_\infty}^t \bigoplus_{\ell_1}^d T^{(c)}$ using Theorem 4.2.8, Theorem 5.1.2 from [10], and Theorem 4.2.9.

Lemma 4.5.1. *Fix some $c_1, \dots, c_d \geq 0$. Let $\bigoplus_{\ell_\infty} T^{(c)}$ be the space with*

$$\|\cdot\|_{T,\infty}^{(c)} \colon \mathbb{R}^{d^2} \to \mathbb{R}$$

seminorm where for every $x = (x_1, \dots, x_d) \in \mathbb{R}^{d^2}$,

$$\|x\|_{T,\infty}^{(c)} = \max_{k\in[d]} c_k \|x_k\|_{T(k)}.$$

For every $\varepsilon \in (0, 1/2)$, there exists a data structure for ANN over $\|\cdot\|_{T,\infty}^{(c)}$ which

achieves approximation $O\left(\frac{\log\log n \cdot \log\log d}{\varepsilon^3}\right)$ *using space* $O\left(d^2 \cdot n^{1+\varepsilon}\right)$ *and query time* $O\left(d^2 \cdot n^{\varepsilon}\right)$.

Proof. Given the randomized embedding from Lemma 4.3.3, we can build a data structure for $c_k\|\cdot\|_{T(k)}$ achieving approximation $O(\frac{\log\log d}{\varepsilon^2})$ using space $O(d^2 n^{1+\varepsilon/2})$ and query time $O(d^2 n^{\varepsilon/2})$. This data structure works in the same way as in the proof of Theorem 4.3.2. We handle the constant c_k by rescaling the norm, and since the embeddings are linear, it does not affect the correctness of the data structure. Then we apply Theorem 4.2.8. □

Lemma 4.5.2. *Fix some* $c_1, \ldots, c_d \geq 0$. *Let* $\bigoplus_{\ell_1} T^{(c)}$ *be the space with* $\|\cdot\|_{T,1}^{(c)} \colon \mathbb{R}^{d^2} \to \mathbb{R}$ *seminorm where* $x = (x_1, \ldots, x_m) \in \mathbb{R}^{d^2}$,

$$\|x\|_{T,1}^{(c)} = \sum_{k=1}^{d} c_k \|x_k\|_{T(k)}.$$

For every $\varepsilon \in (0, 1/2)$, *there exists a data structure for ANN over* $\|\cdot\|_{T,1}^{(c)}$ *which achieves approximation* $O\left(\frac{\log\log n \cdot \log\log d}{\varepsilon^4}\right)$ *using space* $O(d^2 \cdot n^{1+\varepsilon})$ *and query time* $O(d^2 \cdot n^{\varepsilon})$.

Proof. The proof follows from Theorem 5.1.2 in [10] and Lemma 4.5.1. □

Finally, we are combine the above results to get an improved algorithm for general symmetric norms.

Theorem 4.5.3. *For every* d*-dimensional symmetric norm* $\|\cdot\|_X$ *and every* $\varepsilon \in (0, 1/2)$, *there exists a data structure for ANN over* $\|\cdot\|_X$ *which achieves approximation* $O(\frac{\log^2 \log n \log\log d}{\varepsilon^5})$ *using space* $d^{O(1)} \cdot O(n^{1+\varepsilon})$ *and query time* $d^{O(1)} \cdot O(n^{\varepsilon})$.

Proof. Given Theorem 4.4.2, we embed $\|\cdot\|_X$ into $\bigoplus_{\ell_\infty} \bigoplus_{\ell_1} T^{(c)}$ with approximation $(1 \pm \frac{1}{10})$. The result from Lemma 4.5.2 allows we to apply Theorem 4.2.9 to obtain the desired data structure. □

Theorem 4.5.3 implies our main result Theorem 4.1.3 stated in the introduction.

4.6 Lower bounds

In this section, we show that our techniques do not extend to general norms. In particular, we show there does not exist a *universal* norm U for which any norm embeds (possibly randomized) with constant distortion, unless the blow-up in dimension is exponential. Hence the result from below applies to cases of $U = \ell_\infty$ as well as an (low-dimensional) product spaces.

Theorem 4.6.1. *For any $\varepsilon > 0$, let U be a d'-dimensional normed space such that for any d-dimensional normed space X, there exists a distribution $\mathcal{D}$ supported on linear embeddings $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ where for every $x \in \mathbb{R}^d$,*

$$\|x\|_X \leq \|f(x)\|_U \leq D\|x\|_U$$

holds with probability at least $\frac{2}{3}$ over the draw of $f \sim \mathcal{D}$, for $D = O(d^{1/2-\varepsilon})$. Then $d' = \exp\left(\Omega(d^{2\varepsilon})\right)$.

We will prove the above theorem by showing that if there exists a universal normed space U satisfying the conditions of Theorem 4.6.1 above, then two parties, call them Alice and Bob, can use the embeddings to solve the communication problem Index with only a few bits. Let U be a proposed d'-dimensional normed space satisfying the conditions of Theorem 4.6.1. By the John's theorem [32], we may apply a linear transform so that:

$$B_{\ell_2} \subset B_U \subset \sqrt{d'} B_{\ell_2}$$

Lemma 4.6.2. *For any $\varepsilon > 0$, there exists a set of $\exp\left(\Omega(d^{2\varepsilon})\right)$ many points on the unit sphere S^{d-1} such that pairwise inner-products are at most $\frac{1}{d^{1/2-\varepsilon}}$. In fact, these points may consist of points whose coordinates are $\pm\frac{1}{\sqrt{d}}$.*

Proof. Consider picking two random points $x, y \in S^{d-1}$ where each entry is $\pm\frac{1}{\sqrt{d}}$. Then by Bernstein's inequality,

$$\Pr_{x,y}\left[|\langle x, y\rangle| \geq \frac{1}{d^{1/2-\varepsilon}}\right] \leq 2\exp\left(-\Omega(d^{2\varepsilon})\right)$$

We may pick $\exp(\Omega(d^{2\varepsilon}))$ random points and union bound over the probability that some pair has large inner product. $\square$

Fix $\varepsilon > 0$ and $C = d^{1/2-\varepsilon}$, and let P be set a set of unit vectors with pairwise inner-product at most $\frac{1}{C}$ of size $\exp(\Omega(d^{2\varepsilon}))$. For each $a \in \{0,1\}^P$ consider the following norm:

$$\|x\|_a = C \cdot \max_{y \in P : a_y = 1} |\langle x, y \rangle|.$$

Assume there exists a randomized linear embedding $f \colon \mathbb{R}^d \to \mathbb{R}^{d'}$ with the following guarantees:

- For every $x \in \mathbb{R}^d$, $\|x\|_a \leq \|f(x)\|_U \leq D\|x\|_a$ with probability at least $\frac{2}{3}$.

Note the embedding f can be described by M, a $d' \times d$ matrix of real numbers. Additionally, we consider rounding each entry of M by to the nearest integer multiple of $\frac{1}{\text{poly}(d)}$ to obtain M'. For each $x \in S^{d-1}$, $\|(M - M')x\|_U \leq \|(M - M')x\|_2 \leq \frac{1}{\text{poly}(d)}$. Thus, we may assume each entry of M is an integer multiple of $\frac{1}{\text{poly}(d)}$, and lose $(1 \pm \frac{1}{\text{poly}(d)})$ factor in the distortion of the embedding for vectors in B_2.

We now show that the existence of the randomized embedding implies a one-way randomized protocol for the communication problem Index. We first describe the problem. In an instance of Index:

- Alice receives a string $a \in \{0,1\}^n$.
- Bob receives an index $i \in [n]$.
- Alice communicates with Bob so that he can output a_i.

Theorem 4.6.3 ([107]). *The randomized one-way communication complexity of* Index *is* $\Omega(n)$.

We give a protocol for Index:

1. Suppose Alice has input $a \in \{0,1\}^P$. She will generate the norm $\|\cdot\|_a$ described above. Note that $f \sim \mathcal{D}$ has that for each $x \in \mathbb{R}^d$, the embedding preserves the norm of x up to D with probability $\frac{2}{3}$. In particular, if Bob's input is $i \in |P|$,

corresponding to point y, then an embedding $f \in \mathcal{D}$, which we represent as a $d' \times d$ matrix M, satisfies:

$$\|y\|_a \leq \|My\|_U \leq D\|y\|_a$$

with probability $\frac{2}{3}$. In particular, with probability $\frac{2}{3}$:

- If $a_i = 0$, then $\|y\|_a \leq 1$, which implies $\|My\|_U \leq D$.
- If $a_i = 1$, then $\|y\|_a \geq C$, which implies $\|My\|_U \geq C$.

Alice computes the set $P_c \subset P$ of vectors which satisfy the above property (i.e. the embedding M preserves increases the norm by at most a factor D).

2. Alice finds a subset $B \subset P_c$ of linearly independent vectors such that every $x \in P_c$ we have $x \in \text{span}(B)$. Note that $|B| \leq d$ and for all $x \in B$, $\|Mx\|_2 \leq \sqrt{d'}\|Mx\|_U \leq C \cdot D \cdot \sqrt{d'}$. Therefore, each $Mx \in \mathbb{R}^{d'}$ can be written with $\tilde{O}(d')$ bits. So Alice sends the set B, as well as Mx for each $x \in B$ using $\tilde{O}(dd')$ bits.

3. In order for Bob to decode a_i, he first checks whether $y \in \text{span}(B)$, and if not, he guesses. If $y \in \text{span}(B)$, which happens with probability $\frac{2}{3}$, then Bob writes

$$y = \sum_{b_i \in B} c_i b_i$$

and $My = \sum_{b_i \in B} c_i Mb_i$. If $\|My\|_U \leq D$, then $a_i = 0$ and if $\|My\|_U \geq C$ then $a_i = 1$. Thus, if $D < \frac{C}{2}$, Bob can recover a_i with probability $\frac{2}{3}$.

Alice communicates $\tilde{O}(dd')$ bits, and Bob is able to recover a_i with probability $\frac{2}{3}$. By Theorem 4.6.3, $dd' \geq \tilde{\Omega}(|P|)$, which in turn implies $d' \geq \exp(\Omega(d^{2\varepsilon}))$.

4.7 Appendix: Bounding space in Theorem 4.2.9

Here we justify the space bound of the algorithm from Theorem 4.2.9 (from [82]). We note that the improved bound was also claimed in [16], albeit without a proof.

First of all, as suggested at the end of Section 3 of [82], one modifies the algorithm to obtain space of the form of $n^{1+\varepsilon}$, at the expense of increasing the approximation to $O(\varepsilon^{-1} \log\log n)$. This is done by replacing the conditions in Case 2 and 3 by respectively:

$$\left[\frac{|B(s, R(s)+c+1) \cap S_{|i}|}{|S_{|i}|}\right]^{1+\varepsilon} < \frac{|B(s, R(s)) \cap S_{|i}|}{|S_{|i}|},$$

and

$$\left[\frac{|S_{|i} - B(p, R')|}{|S|}\right]^{1+\varepsilon} < \frac{|S_{|i} - B(s, R'+2)|}{|S|}.$$

With the remaining algorithm being precisely the same, our only task here is to argue the space bound. First of all we bound *the sum of the number of points stored in all the leaves*. For a tree with m nodes, let $L(m)$ be an upper bound on this count. We would like to prove that $L(m) \leq m^{1+\varepsilon}$. As in [82], we only need to focus on cases 2 and 3 of the construction, as case 1 does not replicate the points. We will consider the case 2 (case 3 is exactly similar).

Let $m_j = |S_j|$ and $m'_j = |S_{|i} \cap \cup_{s \in S_j} B(s, c+1)|$, whereas $|S| = m$. By construction, we have that $\sum m_j = m$ and $m_j/m > (m'_j/m)^{1+\varepsilon}$ for all j.

By induction, assume $L(m'_j) \leq (m'_j)^{1+\varepsilon}$ for all children. Then, we have that:

$$L(m) \leq \sum_j L(m'_j) \leq \sum_j (m'_j)^{1+\varepsilon} < m^{\varepsilon} \sum_j m_j = m^{1+\varepsilon}.$$

We now argue the total space is $O(S(n) \cdot k \log n \cdot n^{\epsilon})$. Since the depth of the tree is $O(k \log n)$, we have that the total number of points stored in the ANN data structures is $O(k \log n \cdot C(n)) = O(k \log n \cdot n^{1+\epsilon})$. Since each ANN is on at most n points, we have that, for each occurrence of a point in the ANN data structure, we have an additional factor of $S(n)/n$.[4] Hence the total space occupied by all the ANN data structures is $O(S(n)/n \cdot k \log n \cdot n^{1+\epsilon})$. Using a smaller ε (to hide the $\log n$ factor), we obtain the stated space bound of $O(S(n) \cdot k \cdot n^{\epsilon})$.

[4]Here we assume the natural condition that $S(n)$ is increasing, which is, otherwise, easy to guarantee.

4.8 Appendix: $\widetilde{O}(\log d)$-ANN for symmetric norms

We provide a simple ANN algorithm for general symmetric norms with approximation $O(\log d \log\log d)$ using near-linear space and sub-linear query time. The algorithm will leverage the results in the previous section by relating general symmetric norms to Orlicz norms. Recall the definition of level vectors in Definition 4.4.3.

Definition 4.8.1. Let $\|\cdot\|_X$ be any symmetric norm. Let $L_k > 0$ be the minimum number of coordinates needed at level k to have norm at least 1. In other words,

$$L_k = \min\{j \in [d] \mid \|\beta^{-i}\xi^{(j)}\|_X > 1\}.$$

At a high level, we will relate the norm of a vector $x \in \mathbb{R}^d$ to the norm of its level vectors $V_k(x)$. The definition above gives a way to measure the contribution of level k to the norm. For example, if $x \in \mathbb{R}^d$ has norm $\|x\|_X \geq D$, and there are only $2\log_\beta d$ non-zero levels with respect to x, then some level vector $\|V_k(x)\|_X \geq \frac{D}{2\log_\beta d}$. This implies $b_k = \Omega(\frac{DL_k}{\log_\beta d})$, since we may divide $V_k(x)$ into a sum of vectors with L_k coordinates at level k.

On the other hand, if $x \in \mathbb{R}^d$ has $\|x\|_X \leq 1$, then $b_k < L_k$ for each k. Since we consider only $2\log_\beta d$ relevant levels, for $\|x\|_S \leq 1$,

$$\sum_{k=0}^{2\log_\beta d - 1} \frac{b_k}{L_k} \leq 2\log_\beta d.$$

Additionally, $\sum_{k=0}^{2\log_\beta d-1}(b_k/L_k)$ can be decomposed as an additive contribution of coordinates. In particular, coordinate x_i contributes $1/L_k$ if $i \in B_k$. Therefore, we can hope to approximate the symmetric norm by an Orlicz norms and apply the arguments from Lemma 4.3.1.

The lemma below formalizes the ideas discussed above.

Lemma 4.8.2. *Let $\|\cdot\|_X$ be any symmetric norm. For any $D, \alpha > 1$, there exists a non-decreasing function $G\colon \mathbb{R}_+ \to \mathbb{R}_+$ with $G(0) = 0$ and $G(t) \to \infty$ as $t \to \infty$, where every vector $x \in \mathbb{R}^d$ satisfies the following:*

- *If* $\|x\|_X \leq 1$, *then* $\sum_{i=1}^{d} G(|x_i|) \leq 2\log_\beta d$.

- *If* $\|x\|_X > \alpha D \cdot 7\log_\beta d$, *then* $\sum_{i=1}^{d} G\left(\frac{|x_i|}{D}\right) \geq \alpha \cdot 2\log_\beta d$.

Proof. For $i \geq 0$, let $A_i = (\beta^{-i-1}, \beta^{-i}]$. The function $G\colon \mathbb{R}_+ \to \mathbb{R}_+$ is defined as

$$G(t) = \sum_{i=0}^{2\log_\beta d - 1} \frac{\chi_{A_i}(t)}{L_i} + \alpha \cdot 2\log_\beta d \cdot t \cdot \chi_{(1,\infty)}(t) \tag{4.2}$$

Note that $G(0) = 0$ and $G(t) \to \infty$ as $t \to \infty$.

Recall the norm satisfies, $\|\xi^{(1)}\|_X = 1$, so if $\|x\|_X \leq 1$, then $|x_i| \leq 1$ for all $i \in [d]$. This means $\chi_{(1,\infty)}(|x_i|) = 0$ so the second term of the RHS of (4.2) is zero. Therefore,

$$\sum_{i=1}^{d} G(|x_i|) = \sum_{i=1}^{d} \sum_{k=0}^{2\log_\beta d - 1} \frac{\chi_{A_k}(|x_i|)}{L_k} = \sum_{k=0}^{2\log_\beta d - 1} \frac{b_k}{L_k}$$

where b_k is defined with respect to x. Since, $b_k < L_k$ for all $0 \leq k < 2\log_\beta d$,

$$\sum_{i=1}^{d} G(|x_i|) \leq 2\log_\beta d.$$

If $x \in \mathbb{R}^d$ where $\|x\|_X > \alpha D \cdot 7\log_\beta d$, then the vector $\|\frac{x}{D}\|_X > \alpha \cdot 7\log_\beta d$. So it suffices to prove that for any vector $x \in \mathbb{R}^d$ with $\|x\|_X > \alpha \cdot 7\log_\beta d$,

$$\sum_{i=1}^{d} G(|x_i|) \geq \alpha \cdot 2\log_\beta d$$

Additionally, for any vector $x \in \mathbb{R}^d$, we may consider the vector $C(x) \in \mathbb{R}^d$ for $\tau = \frac{\beta}{d^2}$ from Definition 4.4.4. By Proposition 4.4.6, $\|C(x)\|_X \geq \|x\|_X - \frac{\beta}{d} > \alpha \cdot 6\log_\beta d$. Therefore, we may assume $x \in \mathbb{R}^d$ has $\|x\|_X > \alpha \cdot 6\log_\beta d$, and that all non-zero coordinates have absolute values greater than $\frac{\beta}{d^2}$. Equivalently, $b_k = 0$ for all $k \geq 2\log_\beta d$. If for some $i \in [d]$, $|x_i| \geq 1$, then the second term in the RHS of (4.2) is non-zero, and $G(|x_i|) \geq \alpha \cdot 2\log_\beta d$. So we may further assume all coordinates of x lie

in levels $k = 0, \ldots, 2\log_\beta d - 1$. Note that

$$\sum_{i=1}^{d} G(|x_i|) = \sum_{k=0}^{2\log_\beta d - 1} \sum_{i=1}^{d} G(|V_k(x)_i|),$$

and for each $0 \le k < 2\log_\beta d$, $\sum_{i=1}^{d} G(|V_k(x)_i|) = \dfrac{b_k}{L_k}$.

We partition the levels into two groups,

$$A = \left\{ k \mid \frac{b_k}{L_k} < 1 \right\} \qquad \text{and} \qquad B = \left\{ k \mid \frac{b_k}{L_k} \ge 1 \right\}.$$

For all $k \in B$,

$$\|V_k(x)\|_X \le \left\lceil \frac{b_k}{L_k} \right\rceil \le \frac{2b_k}{L_k}$$

since by the triangle inequality, we can break $V_k(x)$ into at most $\left\lceil \dfrac{b_k}{L_k} \right\rceil$ vectors with L_k coordinates at level k each having norm at least 1.

Suppose for the sake of contradiction that $\sum_{k\in B} \frac{b_k}{L_k} \le \alpha \cdot 2\log_\beta d$. Then

$$\alpha \cdot 4\log_\beta d \ge \sum_{k\in B} \frac{2b_k}{L_k} \ge \sum_{k\in B} \|V_k(x)\|_X.$$

Additionally, since $\|x\|_X > \alpha \cdot 6\log_\beta d$, and

$$\alpha \cdot 6\log_\beta d < \|x\|_X \le \sum_{k\in A} \|V_k(x)\|_X + \sum_{k\in B} \|V_k(x)\|_X,$$

it follows that

$$\sum_{k\in A} \|V_k(x)\|_X > \alpha \cdot 2\log_\beta d.$$

However, this is a contradiction for since $|A| \le 2\log_\beta d$ and $\|V_k(x)\|_X \le 1$. □

Lemma 4.8.3. *For any $\varepsilon \in (0, 1/2)$, there exists a data structure for ANN over any symmetric norm $\|\cdot\|_X$ which achieves approximation $O\left(\frac{\log d \log\log d}{\varepsilon^2}\right)$ using space $O(dn^{1+\varepsilon})$ and query time $O(dn^{\varepsilon})$.*

Proof. We fix $\beta = \frac{3}{2}$. The proof of this lemma follows in the same way as the proof

of Theorem 4.3.2. The one difference is that we rescale the ℓ_∞ norm by $\frac{1}{2\log_\beta d}$ after applying the embedding. □

4.8.1 The $\log^{\Omega(1)} d$-approximation is necessary

Let us remark that we cannot push the technique much further. Namely, any $G(\cdot)$ (even non-convex) requires approximation $\Omega(\sqrt{\log d})$ for the following norm. Define the norm of a vector to be

$$\|x\| = \max_{1 \le k \le d} \left(\frac{x_1^* + x_2^* + \dots x_k^*}{\sqrt{k}} \right).$$

This is the minimal norm for $a_k = \sqrt{k}$ (see Section 4.1.2 for the definition). It is not hard to check that an approximation with any $G(\cdot)$ ends up having a distortion $\Omega(\sqrt{\log d})$.

The idea is to consider the following vectors: for every $1 \le k \le d$, we consider a vector

$$\Big(\underbrace{1, 1, \dots, 1}_{k}, 0, 0, \dots, 0\Big),$$

and besides, we consider a vector

$$\Big(1, \sqrt{2}-1, \sqrt{3}-\sqrt{2}, \dots, \sqrt{d}-\sqrt{d-1}\Big).$$

The remaining calculation is a simple exercise.

4.9 Appendix: Lower bound for arbitrary metrics via expander graphs

We give an example of a *metric* that is hard for current approaches to ANN search. The lower bound is based on the notion of robust expansion, which implies all known lower bounds for ANN [146, 24]. In what follows, we will refer to $d = \log N$ as the dimension of a finite metric space of size N.

Our example of a hard metric will be the shortest path metric on any spectral expander graph. We note that a similar theorem to the one below is also known for a finite subset of the high-dimensional Earth-Mover Distance [98].

Fix M to be the metric induced by the shortest path distance on a 3-regular expander G on N nodes. In particular, assume that $1 - \lambda(G) > c$, where c is an absolute constant, and $\lambda(G) \in (0, 1)$ is the second-largest eigenvalue of the normalized adjacency matrix of G. Let d be the dimension $d = \log N$.

Theorem 4.9.1. *For any approximation $\alpha > 1$, and data set size $n \geq 1$ with $d^{\Omega(1)} \leq n \leq N^{O(1)}$, any α-ANN data structure on n points which makes t cell probes (with cells of size at most $w \leq (d \log n)^{O(1)}$), and has success probability at least $\gamma > n^{-1+o(1)}$, must use space $m = \gamma^{\Omega(1/t)} N^{\Omega(1/(\alpha t))} = \gamma^{\Omega(1/t)} 2^{\Omega(d/(\alpha t))}$.*

We proceed by introducing a few definitions from [146], and then prove lower bounds on the robust expansion.

Definition 4.9.2 ([146])**.** In the Graphical Neighbor Search problem (GNS), we are given a bipartite graph $H = (U, V, E)$ where the dataset comes from U and the queries come from V. The dataset consists of pairs $P = \{(p_i, x_i) \mid p_i \in U, x_i \in \{0, 1\}, i \in [n]\}$. On query $q \in V$, if there exists a unique p_i with $(p_i, q) \in E$, then we want to return x_i.

One can use the GNS problem to prove lower bounds on c-ANN as follows: build a GNS graph H by taking $U = V = [N]$, and connecting two points $u \in U, v \in V$ iff they are at a distance at most r (see details in [146]). We will also need to make sure that in our instances q is not closer than cr to other points except the near neighbor.

We now introduce the notion of robust expansion, used in [146] to prove lower bounds.

Definition 4.9.3 (Robust Expansion [146])**.** For a GNS graph $H = (U, V, E)$, fix a distribution e on $E \subset U \times V$, and let μ be the marginal on U and η be the marginal on V. For $\delta, \gamma \in (0, 1]$, the robust expansion $\Phi_r(\delta, \gamma)$ is defined as follows:

$$\Phi_r(\delta, \gamma) = \min_{A \subset V : \eta(A) \leq \delta} \min_{B \subset U : \frac{e(A \times B)}{e(A \times V)} \geq \gamma} \frac{\mu(B)}{\eta(A)}.$$

We now prove a lower bound on the robust expansion $\Phi_r(\delta, \gamma)$ for a GNS graph arising from the shortest path metric on the expander graph G. Fix $r = d/\alpha$. The hard distribution e is defined as follows: pick p at random from M and obtain q by running a random walk of length r starting at p. Note that for $n < N^{1/4}$ and sufficiently high constant α, the distribution satisfies the weak-independence condition required for applying the results in [146].

Fix any sets $A, B \subset M$, where $a = |A|/N$ and $b = |B|/N$. By the expander mixing lemma applied to G^r, we obtain that:

$$\left| E_{G^r}(A, B) - \tfrac{|A| \cdot |B|}{3^r N} \right| \leq \lambda^3 3^r \sqrt{|A| \cdot |B|}.$$

Considering that $\Pr[q \in B \mid p \in A] = \frac{E_{G^r}(A,B)}{aN \cdot 3^r}$, we have that:

$$\Pr[q \in B \mid p \in A] \leq b + \lambda^r \sqrt{b/a}.$$

Restricting to sets A, B such that $\Pr[q \in B \mid p \in A] \geq \gamma$, for which we must have that $\Phi_r = \Phi_r(a, \gamma) \geq b/a$ (by definition), we conclude:

$$\gamma \leq \Phi_r \cdot a + \lambda^r \sqrt{\Phi_r}.$$

Hence, either $\Phi_r = \Omega(\gamma/a)$ or $\Phi_r = \Omega(\gamma^2/\lambda^{2r})$.

Proof of Theorem 4.9.1. Applying Theorem 1.5 from [146], we have that, for $t \geq 1$ cell probes, either:

- $m^t w/n \geq \Omega(\gamma \cdot m^t)$, an impossibility;
- or $m^t w/n \geq \Omega(\gamma^2/\lambda^{2r})$, or $m^t = \Omega(\frac{n}{w}\gamma^2/\lambda^{2r})$, implying $m = \gamma^{2/t} N^{\Omega(1/(\alpha t))}$.

□

To show how bad the situation for expander metrics is, we state a lower bound on for α-ANN on the expander metric described above in the *list-of-points* model, which captures the hashing-based algorithms of [24] and in the decision tree model

of [81]. The proofs follow from a simple derivation using the robust expansion lower bounds in Section 7 of [24] and a reduction of decision trees to $O(\log m)$-cell-probe data structures similar to Appendix A in [11].

Theorem 4.9.4. *Any list-of-points data structure for (c, r)-ANN for random instances of n points in the expander metric of dimension d (described above) with query time t and space m has either $t = \Omega(n)$, or $m = \exp(\Omega(d))$.*

Theorem 4.9.5. *Let $d = \Omega(\log^{1+\varepsilon} n)$ for some $\varepsilon > 0$. Any decision tree of size m and depth t and word size w succeeding with probability γ satisfies:*

$$\frac{m^{O(\log m)} tw}{n} \geq \Phi_r \left(\frac{1}{m^{O(\log m)}}, \frac{\gamma}{O(\log m)} \right).$$

In particular, for any $\rho > 0$, if $w \leq n^{\rho}$, either

$$t \geq \widetilde{\Omega}(n^{1-\rho})$$

or

$$m = \exp\left(\Omega(d^{\varepsilon/(1+\varepsilon)})\right) \operatorname{poly}(n).$$

Chapter 5

Sketching and embedding are equivalent for norms

5.1 Introduction

One of the most exciting notions in the modern algorithm design is that of *sketching*, where an input is summarized into a small data structure. Perhaps the most prominent use of sketching is to estimate *distances between points*, one of the workhorses of similarity search. For example, some early uses of sketches have been designed for detecting duplicates and estimating resemblance between documents [47, 48, 55]. Another example is Nearest Neighbor Search, where many algorithms rely heavily on sketches, under the labels of dimension reduction (like the Johnson-Lindenstrauss Lemma [93]) or Locality-Sensitive Hashing (see e.g. [85, 108, 14]). Sketches see widespread use in streaming algorithms, for instance when the input implicitly defines a high-dimensional vector (via say frequencies of items in the stream), and a sketch is used to estimate the vector's ℓ_p norm. The situation is similar in compressive sensing, where acquisition of a signal can be viewed as sketching. Sketching—especially of distances such as ℓ_p norms—was even used to achieve improvements for *classical* computational tasks: see e.g. recent progress on numerical linear algebra algorithms [178], or dynamic graph algorithms [5, 99]. Since sketching is a crucial primitive that can lead to many algorithmic advances, it is important to understand its power and

limitations.

A primary use of sketches is for *distance estimation* between points in a metric space (X, d_X), such as the Hamming space. The basic setup here asks to design a *sketching* function $\mathbf{sk} : X \to \{0,1\}^s$, so that the distance $d_X(x, y)$ can be estimated given only the sketches $\mathbf{sk}(x), \mathbf{sk}(y)$. In the decision version of this problem, the goal is to determine whether the inputs x and y are "close" or "far", as formalized by the *Distance Threshold Estimation Problem* [157], denoted $\text{DTEP}_r(X, D)$, where, for a threshold $r > 0$ and approximation $D \geq 1$ given as parameters in advance, the goal is to decide whether $d_X(x, y) \leq r$ or $d_X(x, y) > Dr$. Throughout, it will be convenient to omit r from the subscript.[1] Efficient sketches $\mathbf{sk}$ almost always need to be randomized, and hence we allow randomization, requiring (say) 90% success probability.

The diversity of applications gives rise to a variety of natural and important metrics M for which we want to solve DTEP: Hamming space, Euclidean space, other ℓ_p norms, the Earth Mover's Distance, edit distance, and so forth. Sketches for Hamming and Euclidean distances are now classic and well-understood [85, 108]. In particular, both are "efficiently sketchable": one can achieve approximation $D = O(1)$ using sketch size $s = O(1)$ (most importantly, independent of the dimension of X). Indyk [84] extended these results to efficient sketches for every ℓ_p norm for $p \in (0, 2]$. In contrast, for ℓ_p-spaces with $p > 2$, efficient sketching (constant D and s) was proved impossible using information-theoretic arguments [157, 34]. Extensive subsequent work investigated sketching of other important metric spaces,[2] or refined bounds (like a trade-off between D and s) for "known" spaces.[3]

These efforts provided beautiful results and techniques for many specific settings. Seeking a broader perspective, a foundational question has emerged [124, Question #5]:

Question 5.1.1. *Characterize metric spaces which admit efficient sketching.*

[1] When X is a normed space it suffices to consider $r = 1$ by simply scaling the inputs x, y.

[2] Other metric spaces include edit distance [33, 37, 141, 21] and its variants [63, 131, 62, 61, 56, 16], the Earth Mover's Distance in the plane or in hypercubes [55, 86, 117, 102, 15, 12], cascaded norms of matrices [90], and the trace norm of matrices [111].

[3] These refinements include the Gap-Hamming-Distance problem [177, 89, 49, 50, 54, 163, 170], and LSH in ℓ_1 and ℓ_2 spaces [129, 140].

To focus the question, efficient sketching will mean constant D and s for us. Since its formulation circa 2006, progress on this question has been limited. The only known characterization is by [76] for distances that are decomposable by coordinates, i.e., $d_X(x, y) = \sum_{i=1}^{n} \varphi(x_i, y_i)$ for some φ. In particular, they show a number of general conditions on φ which imply an $\Omega(n)$ sketching complexity for d_X.

5.1.1 The embedding approach

To address DTEP in various metric spaces more systematically, researchers have undertaken the approach of metric embeddings. A *metric embedding* of X is a map $f : X \to Y$ into another metric space (Y, d_Y). The *distortion* of f is the smallest $D' \geq 1$ for which there exists a scaling factor $t > 0$ such that

$$\forall x, y \in X, \qquad d_Y(f(x), f(y)) \leq t \cdot d_X(x, y) \leq D' \cdot d_Y(f(x), f(y)).$$

If the target metric Y admits sketching with parameters D and s, then X admits sketching with parameters $D \cdot D'$ and s, by the simple composition $\mathbf{sk}' : x \mapsto \mathbf{sk}(f(x))$. This approach of "reducing" sketching to embedding has been very successful, including for variants of the Earth Mover's Distance [55, 86, 117, 134, 15], and for variants of edit distance [37, 141, 56, 16, 63, 131, 62, 61]. The approach is obviously most useful when Y itself is efficiently sketchable, which holds for all $Y = \ell_p$, $p \in (0, 2]$ [84] (we note that ℓ_p for $0 < p < 1$ is not a metric space, but rather a *quasimetric* space; the above definitions of embedding and distortion make sense even when Y is a quasimetric, and we will use this extended definition liberally). In fact, the embeddings mentioned above are all into ℓ_1, except for [16] which employs a more complicated target space. We remark that in many cases the distortion D' achieved in the current literature is not constant and depends on the "dimension" of X.

Extensive research on embeddability into ℓ_1 has resulted in several important distortion lower bounds. Some address the aforementioned metrics [102, 134, 106, 21], while others deal with metric spaces arising in rather different contexts such as Functional Analysis [149, 57, 58], or Approximation Algorithms [114, 30, 104, 103].

Nevertheless, obtaining (optimal) distortion bounds for ℓ_1-embeddability of several metric spaces of interest, are still well-known open questions [122].

Yet sketching is a more general notion, and one may hope to achieve better approximation by bypassing embeddings into ℓ_1. As mentioned above, some limited success in bypassing an ℓ_1-embedding has been obtained for a variant of edit distance [16], albeit with a sketch size depending mildly on the dimension of X. Our results disparage these hopes, at least for the case of *normed spaces*.

5.1.2 Our results

Our main contribution is to show that efficient sketchability of norms is *equivalent* to embeddability into $\ell_{1-\varepsilon}$ with constant distortion. Below we only assert the "sketching $\implies$ embedding" direction, as the reverse direction follows from [84], as discussed above.

Theorem 5.1.2. *Let X be a finite-dimensional normed space, and suppose that $0 < \varepsilon < 1/3$. If X admits a sketching algorithm for $\mathrm{DTEP}(X, D)$ for approximation $D > 1$ with sketch size s, then X linearly embeds into $\ell_{1-\varepsilon}$ with distortion $D' = O(sD/\varepsilon)$.*

One can ask whether it is possible to improve Theorem 5.1.2 by showing that X, in fact, embeds into ℓ_1. Since many non-embeddability theorems are proved for ℓ_1, such a statement would "upgrade" such results to lower bounds for sketches. Indeed, we show results in this direction too. First of all, the above theorem also yields the following statement.

Theorem 5.1.3. *Under the conditions of Theorem 5.1.2, X linearly embeds into ℓ_1 with distortion $O(sD \cdot \log(\dim X))$.*

Ideally, we would like an even stronger statement: efficient sketchability for norms is equivalent to embeddability into ℓ_1 with constant distortion (i.e., independent of the dimension of X as above). Such a stronger statement in fact requires the resolution of an open problem posed by Kwapien in 1969 (see [95, 41]). To be precise, Kwapien asks whether every finite-dimensional normed space X that embeds into $\ell_{1-\varepsilon}$ for $0 < \varepsilon < 1$

with distortion $D_0 \geq 1$ must also embed into ℓ_1 with distortion D_1 that depends only on D_0 and ε but not on the dimension of X (this is a reformulation of the finite-dimensional version of the original Kwapien's question). In fact, by Theorem 5.1.2, the "efficient sketching $\implies$ embedding into ℓ_1 with constant distortion" statement is *equivalent* to a positive resolution of the Kwapien's problem. Indeed, for the other direction, consider a potential counter-example to the Kwapien's problem, i.e., a normed space X that embeds into $\ell_{1-\varepsilon}$ with a constant distortion $D_0 \geq 1$, but every embedding of X into ℓ_1 incurs a distortion $D_1 = \omega(1)$, where the asymptotics is with the dimension of X (it is really a sequence of normed spaces). Hence, X admits an efficient sketch obtained by combining the embedding into $\ell_{1-\varepsilon}$ with the sketch of [84], but does not embed into ℓ_1 with constant distortion. Thus, if the answer to Kwapien's question is negative, then our desired stronger statement is false.

To bypass the resolution of the Kwapien's problem, we prove the following variant of the theorem using a result of Kalton [95]: efficient sketchability is equivalent to ℓ_1-embeddability with constant distortion for norms that are *"closed" under sum-products*. A sum-product of two normed spaces X and Y, denoted $X \oplus_{\ell_1} Y$, is the normed space $X \times Y$ endowed with $\|(x, y)\| = \|x\| + \|y\|$. It is easy to verify that ℓ_1, the Earth Mover's Distance, and the trace norm are all closed under taking sum-products (potentially with an increase in the dimension). Again, we only need to show the "sketching $\implies$ embedding" direction, as the reverse direction follows from the arguments above — if a normed space X embed into ℓ_1 with constant distortion, we can combine it with the ℓ_1 sketch of [84] and obtain an efficient sketch for X. We discuss the application of this theorem to the Earth Mover's Distance in Section 5.1.3.

Theorem 5.1.4. *Let $(X_n)_{n=1}^{\infty}$ be a sequence of finite-dimensional normed spaces. Suppose that for every $i_1, i_2 \geq 1$ there exists $m = m(i_1, i_2) \geq 1$ such that $X_{i_1} \oplus_{\ell_1} X_{i_2}$ embeds isometrically into X_m. Assume that every X_n admits a sketching algorithm for* $\mathrm{DTEP}(X_n, D)$ *for fixed approximation $D > 1$ with fixed sketch size s (both independent of n). Then, every X_n linearly embeds into ℓ_1 with bounded distortion (independent of n).*

Overall, we almost completely characterize the norms that are efficiently sketchable, thereby making a significant progress on Question 5.1.1. In particular, our results suggest that the embedding approach (embed into ℓ_p for some $p \in (0, 2]$, and use the sketch from [84]) is essentially unavoidable for norms. It is interesting to note that for general metrics (not norms) the implication "efficient sketching $\implies$ embedding into ℓ_1 with constant distortion" is false: for example the Heisenberg group embeds into ℓ_2-squared (with bounded distortion) and hence is efficiently sketchable, but it is not embeddable into ℓ_1 [110, 57, 58] (another example of this sort is provided by Khot and Vishnoi [104]). At the same time, we are not aware of any counter-example to the generalization of Theorem 5.1.2 to general metrics.

5.1.3 Applications

To demonstrate the applicability of our results to concrete questions of interest, we consider two well-known families of normed spaces, for which we obtain the first non-trivial lower bounds on the sketching complexity.

Trace norm. Let $\mathcal{T}_n$ be the vector space $\mathbb{R}^{n\times n}$ (all real square $n \times n$ matrices) equipped with the trace norm (also known as the nuclear norm and Schatten 1-norm), which is defined to be the sum of singular values. It is well-known that $\mathcal{T}_n$ embeds into ℓ_2 (and thus also into ℓ_1) with distortion $\sqrt{n}$ (observe that the trace norm is within $\sqrt{n}$ from the Frobenius norm, which embeds isometrically into ℓ_2). Pisier [149] proved a matching lower bound of $\Omega(\sqrt{n})$ for the distortion of any embedding of $\mathcal{T}_n$ into ℓ_1.

This non-embeddability result, combined with our Theorem 5.1.3, implies a sketching lower bound for the trace norm. Before, only lower bounds for specific types of sketches (linear and bilinear) were known [111].

Corollary 5.1.5. *For any sketching algorithm for* $\mathrm{DTEP}(\mathcal{T}_n, D)$ *with sketch size* s *the following bound must hold:*

$$sD = \Omega\left(\frac{\sqrt{n}}{\log n}\right).$$

Earth Mover's Distance. The (planar) Earth Mover's Distance (also known as the transportation distance, Wasserstein-1 distance, and Monge-Kantorovich distance) is the vector space $\mathrm{EMD}_n = \{p \in \mathbb{R}^{[n]^2} : \sum_i p_i = 0\}$ endowed with the norm $\|p\|_{\mathrm{EMD}}$ defined as the minimum cost needed to transport the "positive part" of p to the "negative part" of p, where the transportation cost per unit between two points in the grid $[n]^2$ is their ℓ_1-distance (for a formal definition see [134]). It is known that this norm embeds into ℓ_1 with distortion $O(\log n)$ [86, 55, 134], and that any ℓ_1-embedding requires distortion $\Omega(\sqrt{\log n})$ [134].

We obtain the first sketching lower bound for EMD_n, which in particular addresses a well-known open question [124, Question #7]. Its proof is a direct application of Theorem 5.1.4 (which we *can* apply, since EMD_n is obviously closed under taking sum-products), to essentially "upgrade" the known non-embeddability into ℓ_1 [134] to non-sketchability.

Corollary 5.1.6. *No sketching algorithm for* $\mathrm{DTEP}(\mathrm{EMD}_n, D)$ *can achieve approximation* D *and sketch size* s *that are constant (independent of* n*).*

The reason we can not apply Theorem 5.1.3 and get a clean quantitative lower bound for sketches of EMD_n is the factor $\log(\dim X)$ in Theorem 5.1.3. Indeed, the lower bound on the distortion of an embedding of EMD_n into ℓ_1 proved in [134] is $\Omega(\sqrt{\log n})$, which is smaller than $\log(\dim X) = \Theta(\log n)$.

We note that EMD_n is a (slight) generalization of the EMD metric version commonly used in computer science applications. In the latter, given two weighted sets $A, B \subset [n]^2$ of the same total weight, one has to solve, using only their sketches $\mathbf{sk}(A), \mathbf{sk}(B)$, the $\mathrm{DTEP}(\mathrm{EMD}, D)$ problem where the EMD distance is the min-cost matching between A and B. Observe that the weights used in the sets $A, B \subset [n]^2$ are all positive. The slight difference is that in $DTEP(\mathrm{EMD}_n, D)$, which asks analogously to estimate $\|p - q\|_{\mathrm{EMD}}$, each of $p, q \in \mathbb{R}^{[n]^2}$ has both "positive" and "negative" parts. Nevertheless, we show in Section 5.7 that efficient sketching of EMD on weighted sets implies efficient sketching of the EMD_n norm. Hence, the non-sketchability of EMD_n norm applies to EMD on weighted sets as well.

5.1.4 Other related work

Another direction for "characterizing tractable metrics" is in the context of streaming algorithms, where the input is an implicit vector $x \in \mathbb{R}^n$ given in the form of updates (i, δ), with the semantics that coordinate i has to be increased by $\delta \in \mathbb{R}$.

There are two known results in this vein. First, [45] characterized the streaming complexity of computing the sum $\sum_i \varphi(x_i)$, for some fixed φ (e.g., $\varphi(x) = x^2$ for ℓ_2 norm), when the updates are positive. They gave a precise property of φ that determines whether the complexity of the problem is small. Second, [112] showed that, in certain settings, streaming algorithms may as well be *linear*, i.e., maintain a sketch $f(x) = Ax$ for a matrix A, and the size of the sketch is increased by a factor logarithmic in the dimension of x.

Furthermore, after the appearance of the conference version of the current article, there has been another characterization result that significantly generalizes and extends [45]. Specifically, for every *symmetric* norm $\|\cdot\|_X$, it is proved in [42] that the sketching (and streaming) complexity of computing $\|x\|_X$ is characterized by the norm's (maximum) modulus of concentration, up to polylogarithmic factors in the dimension of X. Finally, we mention a related work [25], where an efficient data structure for the Approximate Nearest Neighbor search (ANN) is constructed for every *symmetric* norm. It is known [85, 108] that efficient sketches imply good data structures for ANN, however, the result of [25] shows that having efficient ANN data structure is a way more general property of an underlying norm.

5.1.5 Proof overview

Following common practice, we think of sketching as a communication protocol. In fact, our results hold for protocols with an *arbitrary* number of rounds (and access to public randomness).

Our proof of Theorem 5.1.2 can be divided into two parts: *information-theoretic* and *analytic*. First, we use information-theoretic tools to convert an efficient *protocol* for $\mathrm{DTEP}(X, D)$ into a so-called *threshold map* from X to a Hilbert space. Our notion

of a threshold map can be viewed as a very weak definition of embeddability (see Definition 5.4.5 for details). Second, we use techniques from nonlinear functional analysis to convert a threshold map to a *linear map* into $\ell_{1-\varepsilon}$.

Information-theoretic part. To get a good threshold map from a protocol for $\mathrm{DTEP}(X, D)$, we proceed in three steps. First, using the fact that X is a *normed space*, we are able to give a good protocol for $\mathrm{DTEP}(\ell_\infty^k(X), Dk)$ (Lemma 5.4.3). The space $\ell_\infty^k(X)$ is a product of k copies of X equipped with the norm $\|(x_1, \ldots, x_k)\| = \max_i \|x_i\|$. Then, invoking the main result from [20], we conclude non-existence of certain Poincaré-type inequalities for X (Theorem 5.4.4, in the contrapositive).

Finally, we use convex duality together with a compactness argument to conclude the existence of a desired threshold map from X to a Hilbert space (Lemma 5.4.6, again in the contrapositive).

Analytic part. We proceed from a threshold map by upgrading it to a *uniform embedding* (see Definition 5.2.1) of X into a Hilbert space (Theorem 5.4.12). For this we adapt arguments from [94, 151]. We use two tools from nonlinear functional analysis: an extension theorem for 1/2-Hölder maps from a (general) metric space to a Hilbert space [128] (Theorem 5.4.16), and a symmetrization lemma for maps from metric abelian groups to Hilbert spaces [3] (Lemma 5.4.14).

Then we convert a uniform embedding of X into a Hilbert space to a *linear* embedding into $\ell_{1-\varepsilon}$ by applying the result of Aharoni, Maurey and Mityagin [3] together with the result of Nikishin [138]. A similar argument has been used in [134].

To prove a quantitative version of this step, we examine the proofs from [3] and [138], and obtain explicit bounds on the distortion of the resulting map. We accomplish this in Section 5.5.

Embeddings into ℓ_1. To prove Theorem 5.1.3 (which has dependence on the dimension of X), we note that it is a simple corollary of Theorem 5.1.2 and a result of Zvavitch [181], which gives a dimension reduction procedure for subspaces of $\ell_{1-\varepsilon}$.

Norms closed under sum-product. Finally, we prove Theorem 5.1.4 — embeddability into ℓ_1 for norms closed under sum-product — by proving and using a finitary

version of the theorem of Kalton [95] (Lemma 5.6.1), instead of invoking Nikishin's theorem as above. We prove the finitary version by reducing it to the original statement of Kalton's theorem via a compactness argument.

Let us point out that Naor and Schechtman [134] showed how to use (the original) Kalton's theorem to upgrade a uniform embedding of EMD_n into a Hilbert space to a linear embedding into ℓ_1 (they used this reduction to exclude uniform embeddability of EMD_n). Their proof used certain specifics of EMD. In contrast, to get Theorem 5.1.4 for general norms, we seem to need a finitary version of Kalton's theorem.

We also note that in Theorems 5.1.2, 5.1.3 and 5.1.4, we can conclude embeddability into $\ell_{1-\varepsilon}^d$ and ℓ_1^d respectively, where d is *near-linear* in the dimension of the original space. This conclusion uses the known dimension reduction theorems for subspaces from [167, 181].

5.2 Preliminaries on functional analysis

We remind a few definitions and standard facts from functional analysis that will be useful for our proofs. A central notion in our proofs is the notion of *uniform embeddings*, which is a weaker version of embeddability.

Definition 5.2.1. For two *metric spaces* X and Y, we say that a map $f\colon X \to Y$ is a *uniform embedding*, if there exist two non-decreasing functions $L, U\colon \mathbb{R}_+ \to \mathbb{R}_+$ such that for every $x_1, x_2 \in X$ one has $L(d_X(x_1, x_2)) \le d_Y(f(x_1), f(x_2)) \le U(d_X(x_1, x_2))$, $U(t) \to 0$ as $t \to 0$ and $L(t) > 0$ for every $t > 0$. The functions $L(\cdot)$ and $U(\cdot)$ are called *moduli* of the embedding.

Definition 5.2.2. An *inner-product space* is a real vector space X together with an *inner product* $\langle \cdot, \cdot \rangle\colon X \times X \to \mathbb{R}$, which is a symmetric positive-definite bilinear form. A *Hilbert space* is an inner-product space X that is *complete* as a metric space.

Every inner-product space is a normed space: we can set $\|x\| = \sqrt{\langle x, x \rangle}$. For a normed space X we denote by B_X its closed unit ball. The main example of a Hilbert space is ℓ_2, the space of all real sequences $\{x_n\}$ with $\sum_i x_i^2 < \infty$, where the inner

product is defined as

$$\langle x, y \rangle = \sum_i x_i y_i.$$

Definition 5.2.3. For a set S, a function $K\colon S \times S \to \mathbb{R}$ is called a *kernel* if $K(s_1, s_2) = K(s_2, s_1)$ for every $s_1, s_2 \in S$. We say that the kernel K is *positive-definite* if for every $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{R}$ and $s_1, s_2, \ldots, s_n \in S$, one has

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j K(s_i, s_j) \geq 0.$$

We say that K is *negative-definite* if for every $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$ *with* $\alpha_1 + \alpha_2 + \ldots + \alpha_n = 0$ and $s_1, s_2, \ldots, s_n \in S$, one has

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j K(s_i, s_j) \leq 0.$$

The following are standard facts about positive- and negative-definite kernels.

Fact 5.2.4 ([160])**.** *For a kernel $K\colon S \times S \to \mathbb{R}$, there exists an embedding $f\colon S \to H$, where H is a Hilbert space, such that $K(s_1, s_2) = \langle f(s_1), f(s_2) \rangle_H$ for every $s_1, s_2 \in S$, iff K is positive-definite.*

Fact 5.2.5 ([161])**.** *For a kernel $K\colon S \times S \to \mathbb{R}$, there exists an embedding $f\colon S \to H$, where H is a Hilbert space, such that $K(s_1, s_2) = \|f(s_1) - f(s_2)\|_H^2$ for every $s_1, s_2 \in S$, iff $K(s, s) = 0$ for every $s \in S$ and K is negative-definite.*

Definition 5.2.6. For an abelian group G, we say that a function $f\colon G \to \mathbb{R}$ is *positive-definite* if a kernel $K(g_1, g_2) = f(g_1 - g_2)$ is positive-definite. Similarly, f is said to be *negative-definite* if $K(g_1, g_2) = f(g_1 - g_2)$ is negative-definite.

The following lemma essentially says that an embedding of an abelian group G into a Hilbert space can be made translation-invariant.

Lemma 5.2.7 (see the proof of Lemma 3.5 in [3])**.** *Suppose that f is a map from an abelian group G to a Hilbert space such that for every $g \in G$ we have*

$$\sup_{g_1 - g_2 = g} \langle f(g_1), f(g_2) \rangle < +\infty.$$

Then, there exists a map f' from G to a Hilbert space such that $\langle f'(g_1), f'(g_2)\rangle$ depends only on $g_1 - g_2$ and for every $g_1, g_2 \in G$ we have

$$\inf_{g_1'-g_2'=g_1-g_2} \langle f(g_1'), f(g_2')\rangle \leq \langle f'(g_1), f'(g_2)\rangle \leq \sup_{g_1'-g_2'=g_1-g_2} \langle f(g_1'), f(g_2')\rangle.$$

Finally, let $\dim X$ denote the dimension of a finite-dimensional vector space X.

5.3 Preliminaries on communication complexity

Let X be a metric space, on which we would like to solve $\mathrm{DTEP}_r(X, D)$ defined as follows for some $r > 0$ and $D \geq 1$. Alice has a point $x \in X$, Bob has a point $y \in X$, and they would like to decide between the two cases: $d_X(x, y) \leq r$ and $d_X(x, y) > Dr$. To accomplish this goal, Alice and Bob exchange at most s bits of communication.

There are several types of communication protocols that we consider, depending on the randomness used, which we present below in the order of their power. Our main result applies to the most powerful type. We will later show some connections between the protocols of different types.

- **Deterministic protocols.** This is a simple two-way communication protocol with no randomness. First, Alice sends a bit to Bob that depends only on x. Then, Bob sends a bit to Alice that depends on Alice's first communication bit and on y. Then, Alice sends a bit to Bob that depends on x and the two previous communication bits, etc. Finally, whoever sends the s-th bit must decide the answer to the DTEP problem. We define a *transcript* $\Pi_{x,y}$ to be the sequence of s bits sent by the two parties for a given pair of inputs x and y.

- **Private-coin protocols with bounded number of coins.** This is a randomized version of the previous definition. Alice and Bob each have access to an *independent* random string, denoted $a \in \{0,1\}^R$ and $b \in \{0,1\}^R$, respectively. Communication bits sent by Alice are allowed to depend on a, and those sent by Bob may depend on b. We require that for every pair of inputs, the probability

(over the random coins a, b) of the answer being correct is at least, say, $2/3$. Whenever we allow randomness, the transcript $\Pi_{x,y}$ becomes a random variable (depending on x and y). For fixed $a, b \in \{0,1\}^R$, we denote by $\Pi_{x,y}(a,b)$ the (deterministic) transcript for inputs x and y when the random strings are set to be a and b, respectively.

- **Public-coin protocols with finitely many coins.** This is a variant of the previous definition, where Alice and Bob have access to a *common* random string sampled uniformly from $\{0,1\}^R$, and the bits sent by both Alice and Bob can depend on this random string. Again, we require the probability (over the public coins) of the answer being correct to be at least, say, $2/3$ for every pair of inputs. Also, we denote by $\Pi_{x,y}(u)$ the deterministic transcript for fixed inputs and public coins u.

  Clearly, a public-coin protocol with $2R$ public coins can emulate a private-coin protocol with R random bits for each of Alice and Bob.

- **Public-coin protocols with countably many coins.** The protocols defined above are standard in the communication complexity literature. However, we need a definition that is *stronger:* we allow *countably many public coins.* The reason to consider the stronger notion is that the known protocols for DTEP based on [84] fall into this category. Since we allow infinitely many coins, we need to be careful when defining a class of allowed protocols. A sequence of coin tosses u can be identified with a point in the Cantor space $\Omega = \{0,1\}^\omega$ equipped with the standard Lebesgue measure. We require that for every pair of inputs $x, y \in X$, the function $u \mapsto \Pi_{x,y}(u)$ is measurable. This restriction allows us to consider probabilities of the form $\Pr[\Pi_{x,y} \in A]$, where $A \subseteq \{0,1\}^s$ is an arbitrary set of possible transcripts. In particular, the probability of success is well-defined, and we require it, as before, to be at least $2/3$.

The results in this paper apply to the most general protocols: public-coin protocols with countably many coins.

We now show some connections between these notions. A crucial tool in our result is a theorem of [20], which is itself based on the tools from [34]. The latter shows a lower bound for private-coin protocols with finitely many coins. We show next how the lower bounds from [20, 34] extend to the most general type, public-coin protocols with countably many coins.

5.3.1 Information complexity: private-coins vs. public-coins

In general, a lower bound for private-coins protocols does not imply a lower bound for public-coins protocols (without a loss in the parameters). However, such an implication does hold for the particular lower bound technique that we are employing. In particular, we use and exploit the notion of information complexity from [34], defined as follows. Let (x, y, λ) be distributed according to a distribution $\mathcal{D}$ over $X \times X \times \Lambda$, where Λ is an auxiliary set. We will assume that the support of $\mathcal{D}$ is *finite*. Then, we can define the information complexity with respect to $\mathcal{D}$, denoted $\mathrm{IC}_{\mathcal{D}}(\mathrm{DTEP}_r(X, D))$, to be the infimum of $I(x, y : \Pi_{x,y} \mid \lambda)$ over all private-coin protocols for $\mathrm{DTEP}_r(X, D)$, which succeed on every valid input with probability at least $2/3$, where $I(\cdot : \cdot \mid \cdot)$ is the (conditional) mutual information.

It is a standard fact that $\mathrm{IC}_{\mathcal{D}}(\mathrm{DTEP}_r(X, D))$ is a lower bound on the communication complexity of $\mathrm{DTEP}_r(X, D)$ with *private-coin protocols* since

$$I(x, y : \Pi_{x,y} \mid \lambda) \leq \sup_{x,y,a,b} |\Pi_{x,y}(a, b)|.$$

However, we are interested in using the information complexity (as defined above) to lower bound the communication complexity of $\mathrm{DTEP}_r(X, D)$ for *public-coin protocols with finite number of coins*. It turns out that $\mathrm{IC}_{\mathcal{D}}(\mathrm{DTEP}_r(X, D))$ is a valid lower bound for this case as well, as argued in the claim below.

Lemma 5.3.1. *The communication complexity of* $\mathrm{DTEP}_r(X, D)$ *for public-coin protocols with finite number of coins is at least* $\mathrm{IC}_{\mathcal{D}}(\mathrm{DTEP}_r(X, D))$.

Proof. Consider any protocol with public randomness, denoted $\Pi_{x,y}(u)$, where x, y are

the two inputs and u is the public random string. Then

$$\sup_{x,y,u}|\Pi_{x,y}(u))| \geq H(\Pi_{x,y}(u) \mid \lambda, u) \geq I(x, y : \Pi_{x,y}(u) \mid \lambda, u).$$

Now consider the following private-coins protocol $\Pi'_{x,y}(a, b)$, where a, b are the two private random strings of Alice and Bob, respectively. In the first round, Alice sends a to Bob to be used as public randomness $u = a$. Then they run $\Pi_{x,y}(u)$. In other words, the transcript of $\Pi'_{x,y}(a, b)$ is $\langle a, \Pi_{x,y}(a)\rangle$. We claim that

$$I(x, y : \Pi'_{x,y}(a, b) \mid \lambda) = I(x, y : \Pi_{x,y}(u) \mid \lambda, u).$$

Indeed, by definition of Π',

$$I(x, y : \Pi'_{x,y}(a, b) \mid \lambda) = I(x, y : a, \Pi_{x,y}(a) \mid \lambda),$$

and using the chain rule for mutual information,

$$I(x, y : a, \Pi_{x,y}(a) \mid \lambda) = I(x, y : a \mid \lambda) + I(x, y : \Pi_{x,y}(a) \mid \lambda, a).$$

The first term is exactly zero since x, y and a are independent (conditioned on λ). The remaining term gives the equality we are looking for, and proves the lemma. In particular, we see that the length of a public-coin protocol is at least the information complexity $I(x, y : \Pi'_{x,y}(a, b) \mid \lambda)$ of any private-coin protocol Π'. □

5.3.2 From countable to finite number of coins

We now observe that if we focus only on a *finite* number of possible inputs to our DTEP problem, then the existence of a protocol with countably-many coins implies the existence of a protocol with bounded number of coins. This claim will be sufficient to generalize our theorem to the most general type of protocols—public-coin protocols with countably many coins: see the remark after Theorem 5.4.4.

Claim 5.3.2. *Fix a public-coin protocol with* countably many coins *and s bits of*

communication. Let $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})$ *be* N *fixed pairs of inputs for the* DTEP *problem, and let* $\varepsilon > 0$ *be a positive parameter. Then there exists a public-coin protocol* with $R = R(s, N, \varepsilon) < \infty$ coins *and* s *bits of communication, such that for every* $1 \le i \le N$*, the success probabilities of the original and the new protocols on* $(x^{(i)}, y^{(i)})$ *differ by at most* ε*.*

Proof. Since we care about correctness of the protocol only on the inputs $(x^{(i)}, y^{(i)})$, we can think of the protocol as a distribution over a *bounded* (as a function of s and N) number of deterministic protocols (there is only a finite number of distinct protocol transcripts). Then, we can approximate this distribution within a statistical distance ε using a bounded number of public coins. □

5.4 From sketches to uniform embeddings

Our main technical result shows that, for a finite-dimensional normed space X, good sketches for $\mathrm{DTEP}(X, D)$ imply a good uniform embedding of X into a Hilbert space (Definition 5.2.1). Below is the formal statement.

Theorem 5.4.1. *Suppose a finite-dimensional normed space* X *admits a public-coin randomized communication protocol for* $\mathrm{DTEP}(X, D)$ *of size* s *for approximation* $D > 1$*. Then, there exists a map* $f\colon X \to H$ *to a Hilbert space such that for all* $x_1, x_2 \in X$*,*

$$\min\left\{1, \frac{\|x_1 - x_2\|_X}{s \cdot D}\right\} \le \|f(x_1) - f(x_2)\|_H \le K \cdot \|x_1 - x_2\|_X^{1/2},$$

where $K > 1$ *is an absolute constant.*

Theorem 5.4.1 implies a *qualitative* version of Theorem 5.1.2 using the results of Aharoni, Maurey, and Mityagin [3] and Nikishin [138] (see Theorem 5.4.2).

Theorem 5.4.2 ([3, 138])**.** *For every fixed* $0 < \varepsilon < 1$*, any finite-dimensional normed space* X *that is uniformly embeddable into a Hilbert space is linearly embeddable into* $\ell_{1-\varepsilon}$ *with a distortion that depends only on* ε *and the moduli of the assumed uniform embedding.*

To prove the full (quantitative) versions of Theorems 5.1.2 and 5.1.3, we adapt the proofs from [3] and [138] in Section 5.5 to get an explicit bound on the distortion.

In the rest of this section, we prove Theorem 5.4.1 according to the outline in Section 5.1.5, putting the pieces together in Section 5.4.4.

5.4.1 Sketching implies the absence of Poincaré inequalities

Sketching is often viewed from the perspective of a two-party communication complexity. Alice receives input x, Bob receives y, and they need to communicate to solve the DTEP problem. In particular, a sketch of size s implies a communication protocol that transmits s bits: Alice just sends her sketch $\mathbf{sk}(x)$ to Bob, who computes the output of DTEP (based on that message and his sketch $\mathbf{sk}(y)$). We assume here a public-coins model, i.e., Alice and Bob have access to a common (public) random string that determines the sketch function $\mathbf{sk}$.

To characterize sketching protocols, we build on results of Andoni, Jayram and Pătraşcu [20, Sections 3 and 4]. This works in two steps: first, we show that a protocol for $\mathrm{DTEP}(X, D)$ implies a sketching algorithm for $\mathrm{DTEP}(\ell_\infty^k(X), kD)$, with a loss of factor k in approximation (Lemma 5.4.3, see the proof in the end of the section). As usual, $\ell_\infty^k(X)$ is a normed space derived from X by taking the vector space X^k and letting the norm of a vector $(x_1, \dots x_k) \in X^k$ be the maximum of the norms of its k components. The second step is to apply a result from [20] (Theorem 5.4.4), which asserts that sketching for $\ell_\infty^k(X)$ precludes certain Poincaré inequalities for the space X.

Lemma 5.4.3. *Let X be a finite-dimensional normed space that for some $D \geq 1$ admits a communication protocol for $\mathrm{DTEP}(X, D)$ of size s. Then for every integer k, the space $\ell_\infty^k(X)$ admits sketching with approximation kD and sketch size $s' = O(s)$.*

Proof. Fix a threshold $t > 0$, and recall that we defined the success probability of sketching to be 0.9. By our assumption, there is a sketching function $\mathbf{sk}$ for X that achieves approximation D and sketch size s for threshold kt. Now define a "sketching" function $\mathbf{sk}'$ for $\ell_\infty^k(X)$ by choosing random signs $\varepsilon_1, \dots, \varepsilon_k \in \{\pm 1\}$, letting

$\mathbf{sk}' : x \mapsto \mathbf{sk}(\sum_{i=1}^{k} \varepsilon_i x_i)$, and using the same decision procedure used by $\mathbf{sk}$ (for X).

Now to examine the performance of $\mathbf{sk}'$, consider $x, y \in \ell_\infty^k(X)$. If their distance is at most t, then we always have that $\|\sum_{i=1}^{k} \varepsilon_i x_i - \sum_{i=1}^{k} \varepsilon_i y_i\| \leq \sum_{i=1}^{k} \|x_i - y_i\| \leq kt$ (i.e., for every realization of the random signs). Thus with probability at least 0.9 the sketch will declare that x, y are "close".

If the distance between x, y is greater than $kD \cdot t$, then for some coordinate, say $i = 1$, we have $\|x_1 - y_1\| > kD \cdot t$. Letting $z = \sum_{i \geq 2} \varepsilon_i (x_i - y_i)$, we can write $\|\sum_{i=1}^{k} \varepsilon_i x_i - \sum_{i=1}^{k} \varepsilon_i y_i\| = \|\varepsilon_1 (x_1 - y_1) + z\| = \|(x_1 - y_1) + \varepsilon_1 z\|$. The last term must be at least $\|x_1 - y_1\|$ under at least one of the two possible realizations of ε_1, because by the triangle inequality $2\|x_1 - y_1\| \leq \|(x_1 - y_1) + z\| + \|(x_1 - y_1) - z\|$. We see that with probability $1/2$ we have $\|\sum_{i=1}^{k} \varepsilon_i x_i - \sum_{i=1}^{k} \varepsilon_i y_i\| \geq \|x_1 - y_1\| > D \cdot kt$, and thus with probability at least $1/2 \cdot 0.9 = 0.45$ the sketch will declare that x, y are "far". This last guarantee is not sufficient for $\mathbf{sk}'$ to be called a sketch, but it can easily be amplified.

The final sketch $\mathbf{sk}''$ for $\ell_\infty^k(X)$ is obtained by $O(1)$ independent repetitions of $\mathbf{sk}'$, and returning "far" if at least 0.3-fraction of the repetitions come up with this decision. These repetitions amplify the success probability to 0.9, while increasing the sketch size to $O(s)$. □

We now state a slight modification of the theorem from [20]. We will actually use its contrapositive, to conclude the absence of Poincaré inequalities.

Theorem 5.4.4 (modification of [20])**.** *Let X be a metric space, and fix $r > 0$, $D \geq 1$. Suppose there are $\alpha > 0$, $\beta \geq 0$, and two symmetric probability measures μ_1, μ_2 on $X \times X$ such that*

- *The support of μ_1 is finite and is only on pairs with distance at most r;*
- *The support of μ_2 is finite and is only on pairs with distance greater than Dr; and*
- *For every $f : X \to B_{\ell_2}$ (where B_{ℓ_2} is the unit ball of ℓ_2),*

$$\mathop{\mathbb{E}}_{(x,y)\sim\mu_1} \|f(x) - f(y)\|^2 \geq \alpha \cdot \mathop{\mathbb{E}}_{(x,y)\sim\mu_2} \|f(x) - f(y)\|^2 \ - \beta.$$

Then for every integer k, the communication complexity of $\mathrm{DTEP}(\ell_\infty^k(X), D)$ *for protocols with* countably many public coins *(see Section 5.3 for precise definitions) and with probability of error $\delta_0 > 0$ is at least $\Omega(k) \cdot \left(\alpha(1 - 2\sqrt{\delta_0}) - \beta\right)$.*

In [20], almost the same theorem is proved with only one difference: the protocols for $\mathrm{DTEP}(\ell_\infty^k(X), D)$ are only allowed to use finitely many private coins. Here we use Claims 5.3.1 and 5.3.2 to generalize their theorem to Theorem 5.4.4.

Indeed, because the "hard distributions" μ_1 and μ_2 are finitely-supported, an inspection of the proofs from [20] shows that there is a finite set of inputs $\mathcal{I}$ such that any private-coin protocol for $\mathrm{DTEP}(\ell_\infty^k(X), D)$ that is correct on $\mathcal{I}$ with probability at least $1 - \delta_0$ must have information complexity at least $\Omega(k) \cdot \left(\alpha(1 - 2\sqrt{\delta_0}) - \beta\right)$. But by Claim 5.3.1, we get that any protocol with bounded number of public coins correct on $\mathcal{I}$ must have communication complexity at least $\Omega(k) \cdot \left(\alpha(1 - 2\sqrt{\delta_0}) - \beta\right)$. Finally, Claim 5.3.2 implies that the same is true for protocols with countably many public coins that are correct on all valid inputs with probability at least $1 - \delta_0$.

5.4.2 The absence of Poincaré inequalities implies threshold maps

We proceed to prove that non-existence of Poincaré inequalities implies the existence a "threshold map", as formalized in Lemma 5.4.6 below. The proof is similar to duality arguments that one often encounters in embedding theory: for instance, see Proposition 15.5.2 in [121]. First we define the notion of threshold maps.

Definition 5.4.5. A map $f\colon X \to Y$ between metric spaces (X, d_X) and (Y, d_Y) is said to be $(s_1, s_2, \tau_1, \tau_2, \tau_3)$-*threshold* for $0 < s_1 < s_2$, $0 < \tau_1 < \tau_2 < \tau_3$, if for all $x_1, x_2 \in X$:

- if $d_X(x_1, x_2) \leq s_1$, then $d_Y(f(x_1), f(x_2)) \leq \tau_1$;
- if $d_X(x_1, x_2) \geq s_2$, then $d_Y(f(x_1), f(x_2)) \geq \tau_2$; and
- $d_Y(f(x_1), f(x_2)) \leq \tau_3$.

We now provide the main lemma of this section, stated in the contrapositive: the non-existence of threshold maps implies a Poincaré inequality.

Lemma 5.4.6. *Suppose X is a metric space that has no $(s_1, s_2, \tau_1, \tau_2, +\infty)$-threshold map to a Hilbert space. Then, for every $\delta > 0$ there exist two symmetric probability measures μ_1, μ_2 on $X \times X$ such that*

- *The support of μ_1 is finite and is only on pairs with distance at most s_1;*
- *The support of μ_2 is finite and is only on pairs with distance at least s_2; and*
- *For every $f \colon X \to B_{\ell_2}$,*

$$\mathop{\mathbb{E}}_{(x,y)\sim\mu_1} \|f(x) - f(y)\|^2 \geq \left(\frac{\tau_1}{\tau_2}\right)^2 \cdot \mathop{\mathbb{E}}_{(x,y)\sim\mu_2} \|f(x) - f(y)\|^2 \ - \delta. \tag{5.1}$$

We prove Lemma 5.4.6 via the following three claims. The first one uses standard arguments about embeddability of finite subsets (see, e.g., Proposition 8.12 in [41], or Lemma 1.1 from [31]). We note that this claim requires a finite value for τ_3, as opposed to $\tau_3 = +\infty$, which is the only reason the definition of a threshold embedding (Definition 5.4.5) needs the parameter τ_3. In the following claims, we denote by $\binom{X}{2}$ the set of all *unordered* pairs $\{x, y\}$ with $x, y \in X$, $x \neq y$.

Claim 5.4.7. *For every metric space X and every $0 < s_1 < s_2$, $0 < \tau_1 < \tau_2 < \tau_3$ there exists an $(s_1, s_2, \tau_1, \tau_2, \tau_3)$-threshold map of X to a Hilbert space iff the same is true for every finite subset of X.*

The proof of Claim 5.4.7 uses standard definitions and facts from general topology: product topology, Tychonoff's theorem, as well as convergence and accumulation points along nets. These definitions can be found in a general topology textbook (see, e.g., [130]).

Proof. The "only if" direction is obvious, so let us turn to the "if" part. Consider the topological space

$$U = \prod_{\{x,y\}\in\binom{X}{2}} [-\tau_3^2, \tau_3^2].$$

By Tychonoff's theorem U is compact. For every finite $X' \subset X$ there exists an $(s_1, s_2, \tau_1, \tau_2, \tau_3)$-threshold map $f_{X'}$ from X' to a Hilbert space. It gives rise to a point $u_{X'} \in U$ whose coordinates are given by

$$(u_{X'})_{x,y} = \begin{cases} \|f_{X'}(x) - f_{X'}(y)\|^2, & \text{if } x, y \in X'; \\ 0, & \text{otherwise.} \end{cases}$$

Since U is compact, $u_{X'}$ has an accumulation point $u^* \in U$ along the net of finite subsets of X. Let us reformulate what it means.

Claim 5.4.8. *For every $\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_k, y_k\} \in \binom{X}{2}$ and every $\varepsilon > 0$, there exists a finite set $A \subset X$ such that for all $1 \le i \le k$, both $x_i, y_i \in A$ and $\left|(u^*)_{x_i,y_i} - \|f_A(x_i) - f_A(y_i)\|^2\right| < \varepsilon$.*

Now we define a kernel $K\colon X \times X \to \mathbb{R}$, given by (recall Definition 5.2.3):

$$K(x, y) = \begin{cases} 0, & \text{if } x = y; \\ (u^*)_{x,y}, & \text{otherwise.} \end{cases}$$

Claim 5.4.9. *The kernel $K(\cdot, \cdot)$ is negative-definite.*

Proof. Suppose that K is *not* negative-definite. It means that there exist real numbers $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{R}$ with $\sum_i \alpha_i = 0$, and $t_1, t_2, \ldots, t_n \in X$ such that

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j K(t_i, t_j) = \gamma > 0.$$

There exists $\varepsilon > 0$ such that for every $(a_{ij})_{i,j=1}^n$ with $|a_{ij} - K(t_i, t_j)| < \varepsilon$ one has

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j a_{ij} \ge \gamma/2 > 0. \tag{5.2}$$

Now apply Claim 5.4.8 to get a finite set $A \subset X$ that contains all s_i's such that

$\|f_A(t_i) - f_A(t_j)\|^2$ is within ε from $K(t_i, t_j)$ for every i, j. But by (5.2), it means that

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j \|f_A(t_i) - f_A(t_j)\|^2 \geq \gamma/2 > 0,$$

which contradicts Fact 5.2.5. This proves Claim 5.4.9. □

Thus, by Fact 5.2.5, there exists a map $f\colon X \to H$ to a Hilbert space H such that for every $x, y \in X$ one has $\|f(x) - f(y)\|^2 = K(x, y)$. The final step is to verify that f is indeed a required $(s_1, s_2, \tau_1, \tau_2, \tau_3)$-map (according to Definition 5.4.5). This can be done exactly the same way as in the proof of Claim 5.4.9. This completes the proof of Claim 5.4.7. □

Claim 5.4.10. *Suppose that (X, d_X) is a* finite *metric space and $0 < s_1 < s_2$, $0 < \tau_1 < \tau_2 < \tau_3$. Assume that there is* no *$(s_1, s_2, \tau_1, \tau_2, \tau_3)$-threshold map of X to ℓ_2. Then, there exist two symmetric probability measures μ_1, μ_2 on $X \times X$ such that*

- *μ_1 is supported only on pairs with distance at most s_1, while μ_2 is supported only on pairs with distance at least s_2; and*
- *for every $f\colon X \to \ell_2$,*

$$\mathop{\mathbb{E}}_{(x,y)\sim\mu_1} \|f(x) - f(y)\|^2 \geq \left(\frac{\tau_1}{\tau_2}\right)^2 \cdot \mathop{\mathbb{E}}_{(x,y)\sim\mu_2} \|f(x) - f(y)\|^2 - \left(\frac{2\tau_1}{\tau_3}\right)^2 \cdot \sup_{x\in X} \|f(x)\|^2. \quad (5.3)$$

Proof. Let $\mathcal{L}_2 \subset \mathbb{R}^{\binom{X}{2}}$ be the cone of squared Euclidean metrics (also known as negative-type distances) on X. Let $\mathcal{K} \subset \mathbb{R}^{\binom{X}{2}}$ be the polytope of *non-negative* functions $l\colon \binom{X}{2} \to \mathbb{R}_+$ such that for every $x, y \in X$ we have

- $l(\{x, y\}) \leq \tau_3^2$;
- if $d_X(x, y) \leq s_1$, then $l(\{x, y\}) \leq \tau_1^2$;
- if $d_X(x, y) \geq s_2$, then $l(\{x, y\}) \geq \tau_2^2$.

Notice that $\mathcal{L}_2 \cap \mathcal{K} = \emptyset$, as otherwise X allows an $(s_1, s_2, \tau_1, \tau_2, \tau_3)$-threshold map to ℓ_2. We will need the following claim, which is just a variant of the Hyperplane Separation Theorem.

Claim 5.4.11. *There exists $a \in \mathbb{R}^{\binom{X}{2}}$ such that*

$$\forall l \in \mathcal{L}_2, \qquad \langle a, l \rangle \leq 0; \tag{5.4}$$

$$\forall l \in \mathcal{K}, \qquad \langle a, l \rangle > 0. \tag{5.5}$$

Proof. Since both $\mathcal{L}_2$ and $\mathcal{K}$ are convex and closed, and, in addition, $\mathcal{K}$ is compact, there exists a separating (affine) hyperplane between $\mathcal{L}_2$ and $\mathcal{K}$. Specifically, there is a non-zero a such that for every $l \in \mathcal{L}_2$ one has $\langle a, l \rangle \leq \eta$, and for every $l \in \mathcal{K}$ one has $\langle a, l \rangle > \eta$. Since $\mathcal{L}_2$ is a cone, one can assume without loss of generality that $\eta = 0$. Indeed, the case $\eta < 0$ is impossible because $0 \in \mathcal{L}_2$, so suppose that $\eta > 0$. If for all $l \in \mathcal{L}_2$ we have $\langle a, l \rangle \leq 0$, then we are done. Otherwise, take any $l \in \mathcal{L}_2$ such that $\langle a, l \rangle > 0$, and scale it by sufficiently large $C > 0$ to get a point $Cl \in \mathcal{L}_2$ so that $\langle a, Cl \rangle = C\langle a, l \rangle > \eta$, arriving to a contradiction. □

We now continue the proof of Claim 5.4.10. We may assume without loss of generality that

$$\forall \{x, y\} \in \binom{X}{2}, \qquad \text{if } d_X(x, y) < s_2 \text{ then } a_{\{x,y\}} \leq 0. \tag{5.6}$$

To see this, let us zero every such $a_{\{x,y\}} > 0$, and denote the resulting point $\hat{a}$. Then for every $l \in \mathcal{L}_2$ (which clearly has non-negative coordinates), $\langle \hat{a}, l \rangle \leq \langle a, l \rangle \leq 0$. And for every $l \in \mathcal{K}$, let $\hat{l}$ be equal to l except that we zero the same coordinates where we zero a (which in particular satisfy $d_X(x, y) < s_2$); observe that also $\hat{l} \in \mathcal{K}$, and thus $\langle \hat{a}, l \rangle = \langle a, \hat{l} \rangle > 0$. We get that $\hat{a}$ separates $\mathcal{K}$ and $\mathcal{L}_2$ and also satisfies (5.6).

Now we define non-negative functions $\widetilde{\mu}_1, \widetilde{\mu}_2, \widetilde{\mu}_3 \colon \binom{X}{2} \to \mathbb{R}_+$ as follows:

$$\begin{aligned}
\widetilde{\mu}_1(\{x,y\}) &= -a(\{x,y\})\ \mathbf{1}_{\{d_X(x,y) \le s_1\}};\\
\widetilde{\mu}_2(\{x,y\}) &= a(\{x,y\})\ \mathbf{1}_{\{d_X(x,y) \ge s_2 \text{ and } a_{x,y} \ge 0\}};\\
\widetilde{\mu}_3(\{x,y\}) &= -a(\{x,y\})\ \mathbf{1}_{\{d_X(x,y) > s_1 \text{ and } a_{x,y} < 0\}}.
\end{aligned}$$

By (5.6), these $\widetilde{\mu}_i$ "cover" all cases, i.e.,

$$\forall\, \{x,y\} \in \tbinom{X}{2}, \quad a(\{x,y\}) = -\widetilde{\mu}_1(\{x,y\}) + \widetilde{\mu}_2(\{x,y\}) - \widetilde{\mu}_3(\{x,y\}).$$

For $i \in \{1,2,3\}$ define $\lambda_i = \sum_{\{x,y\}} \widetilde{\mu}_i(\{x,y\})$ and $\mu_i(\{x,y\}) = \widetilde{\mu}_i(\{x,y\})/\lambda_i$. We argue that μ_1 and μ_2 are as required by Claim 5.4.10, and indeed the only non-trivial property to check is the second item. From the condition that $\langle a, l\rangle \le 0$ for every $l \in \mathcal{L}_2$ we get that for every map $f \colon X \to \ell_2$,

$$\begin{aligned}
0 &\ge \sum_{\{x,y\}} a(\{x,y\}) \cdot \|f(x) - f(y)\|^2\\
&= \sum_{\{x,y\}} \Big[-\widetilde{\mu}_1(\{x,y\}) + \widetilde{\mu}_2(\{x,y\}) - \widetilde{\mu}_3(\{x,y\}) \Big] \cdot \|f(x) - f(y)\|^2,
\end{aligned}$$

which, in turn, implies

$$\lambda_1 \cdot \mathop{\mathbb{E}}_{(x,y)\sim\mu_1} \|f(x) - f(y)\|^2 \ge \lambda_2 \cdot \mathop{\mathbb{E}}_{(x,y)\sim\mu_2} \|f(x) - f(y)\|^2 - 4\lambda_3 \cdot \sup_x \|f(x)\|^2. \quad (5.7)$$

Consider the point $l \in \mathcal{K}$ with value τ_1^2 on $\mathrm{supp}(\mu_1)$, value τ_2^2 on $\mathrm{supp}(\mu_2)$, value τ_3^2 on $\mathrm{supp}(\mu_3)$, and 0 otherwise; the condition $\langle a, l\rangle > 0$ gives

$$-\lambda_1\tau_1^2 + \lambda_2\tau_2^2 - \lambda_3\tau_3^2 > 0,$$

which implies $\lambda_1 < \lambda_2 \cdot \tau_2^2/\tau_1^2$ and $\lambda_3 < \lambda_2 \cdot \tau_2^2/\tau_3^2$ (in particular, $\lambda_2 > 0$). Plugging into (5.7), we get the inequality required for Claim 5.4.10. □

We are now ready to prove Lemma 5.4.6.

Proof of Lemma 5.4.6. Consider a metric space X, for which there is no threshold map with parameters $(s_1, s_2, \tau_1, \tau_2, +\infty)$. We prove that this implies the Poincaré inequality (5.1).

Indeed, X has no $(s_1, s_2, \tau_1, \tau_2, \tau_3)$-threshold map to a Hilbert space for any finite value τ_3. We set $\tau_3 > \tau_2$ be sufficiently large so that $(2\tau_1/\tau_3)^2 < \delta$. Then, by Claim 5.4.7 there exists a finite subset $X' \subset X$ that has no $(s_1, s_2, \tau_1, \tau_2, \tau_3)$-threshold map to a Hilbert space (which without loss of generality can be chosen to be ℓ_2, since X' is finite). Now, using Claim 5.4.10, we obtain finitely-supported probability measures μ_1 and μ_2, which satisfy (5.3). This concludes the proof of Lemma 5.4.6, since its statement only considers f such that the image of f is the unit ball of ℓ_2, and, thus, $\sup_{x \in X} \|f(x)\|^2 \leq 1$. Note that the measures μ_1, μ_2 depend on the value of τ_3 (and, as a result, on δ). □

5.4.3 Threshold maps imply uniform embeddings

We now prove that threshold embeddings imply uniform embeddings, formalized as follows.

Theorem 5.4.12. *Suppose that X is a finite-dimensional normed space such that there exists a map to a Hilbert space, which is $(1, D, \tau_1, \tau_2, +\infty)$-threshold for some $D > 1$ and for some $0 < \tau_1 < \tau_2$ with $\tau_2 > 8\tau_1$. Then there exists a map h of X into a Hilbert space such that for every $x_1, x_2 \in X$,*

$$(\tau_2^{1/2} - (8\tau_1)^{1/2}) \cdot \min\left\{1, \frac{\|x_1 - x_2\|}{2D+4}\right\} \leq \|h(x_1) - h(x_2)\| \leq (2\tau_1 \|x_1 - x_2\|)^{1/2}. \quad (5.8)$$

In particular, h is a uniform embedding of X into a Hilbert space with moduli that depend only on τ_1, τ_2 and D.

Let us point out that in [94, 151], Johnson and Randrianarivony prove that for a Banach space coarse embeddability into a Hilbert space is equivalent to uniform embeddability. Our definition of a threshold map is weaker than that of a coarse embedding (for the latter see [94] say), but we show that we can adapt the proof of

[94, 151] to our setting as well (at least whenever the gap between τ_1 and τ_2 is large enough). Since we only need one direction of the equivalence, we present a part of the argument from [94] with one (seemingly new) addition: Claim 5.4.19. The resulting proof is arguably simpler than the combination of [94] and [151], and yields a clean quantitative bound (5.8).

Intuition. Let us provide some very high-level intuition of the proof of Theorem 5.4.12. We start with a threshold map f from X to a Hilbert space. First, we show that f is Lipschitz on pairs of points that are sufficiently far. In particular, f, restricted on a sufficiently crude net N of X, is Lipschitz. This allows us to use a certain extension theorem to extend the restriction of f on N to a Lipschitz function on the whole X, while preserving the property that f does not contract too much distances that are sufficiently large. Then, we get a required uniform embedding by performing a certain symmetrization step.

The actual proof is different in a number of details; in particular, instead of being Lipschitz the actual property we will be trying to preserve is different.

Useful facts. To prove Theorem 5.4.12, we need the following three results.

Lemma 5.4.13 ([159])**.** *For a set S and a map f from S to a Hilbert space, there exists a map g from S to a Hilbert space such that $\|g(x_1)-g(x_2)\| = \|f(x_1)-f(x_2)\|^{1/2}$ for every $x_1, x_2 \in S$.*

Lemma 5.4.14 (essentially Lemma 3.5 from [3])**.** *Suppose that f is a map from an abelian group G to a Hilbert space such that for every $g \in G$ we have*

$$\sup_{g_1-g_2=g} \|f(g_1) - f(g_2)\| < +\infty.$$

Then, there exists a map f' from G to a Hilbert space such that $\|f'(g_1) - f'(g_2)\|$ depends only on $g_1 - g_2$ and for every $g_1, g_2 \in G$ we have

$$\inf_{g_1'-g_2'=g_1-g_2} \|f(g_1') - f(g_2')\| \le \|f'(g_1) - f'(g_2)\| \le \sup_{g_1'-g_2'=g_1-g_2} \|f(g_1') - f(g_2')\|. \quad (5.9)$$

Proof. This lemma is similar to Lemma 5.2.7 with one twist: in the statement, we

now have distances instead of dot products. The proof of Lemma 5.2.7 relies on the characterization from Fact 5.2.4. If instead we use Fact 5.2.5, we can reuse the proof of Lemma 3.5 from [3] verbatim to prove the present lemma.

Let us sketch here the symmetrization procedure. Let $B(G)$ be the vector space of bounded functions $h\colon G \to \mathbb{R}$. Then, one can show that there exists a *finitely additive invariant mean* $M\colon B(G) \to \mathbb{R}$: a *linear* functional such that

- for every $h \in B(G)$ such that $h \geq 0$ one has $Mh \geq 0$;
- for every $h \in B(G)$ and $g \in G$ one has $Mh = M(x \mapsto h(x+g))$;
- $M(x \mapsto 1) = 1$.

The existence of such M is non-trivial and requires the axiom of choice (see, e.g., Theorem 17.5 from [78]).

Let us now consider a map f from the statement of the lemma and consider the kernel $K(g_1, g_2) = \|f(g_1) - f(g_2)\|^2$. Let us define a new function $K'(g_1, g_2)$ as follows:

$$K'(g_1, g_2) = M(x \mapsto K(x + g_1 - g_2, x)).$$

Now we need to check that:

- K' is a kernel (that is, it is non-negative and symmetric) and $K'(g, g) = 0$ for every $g \in G$;
- K' is negative-definite (see Definition 5.2.3), assuming that K is negative-definite (which is true by Fact 5.2.5);
- for every $g_1, g_2 \in G$ one has

$$\inf_{g_1' - g_2' = g_1 - g_2} \|f(g_1') - f(g_2')\|^2 \leq K'(g_1, g_2) \leq \sup_{g_1' - g_2' = g_1 - g_2} \|f(g_1') - f(g_2')\|^2$$

  assuming (5.9).

This can be done exactly the same way as in the proof of Lemma 3.5 from [3]. Finally, we observe that $K'(g_1, g_2)$ depends only on $g_1 - g_2$ and via Fact 5.2.5 gives a map f' from G to a Hilbert space with the required properties. □

Definition 5.4.15. We say that a map $f\colon X \to Y$ between metric spaces is 1/2-Hölder with constant C, if for every $x_1, x_2 \in X$ one has $d_Y(f(x_1), f(x_2)) \leq C \cdot d_X(x_1, x_2)^{1/2}$.

Theorem 5.4.16 ([128], see also Theorem 19.1 in [174]).). *Let (X, d_X) be a metric space and let H be a Hilbert space. Suppose that $f\colon S \to H$, where $S \subset X$, is a 1/2-Hölder map with a constant $C > 0$. Then there exists a map $g\colon X \to H$ that coincides with f on S and is 1/2-Hölder with the constant C.*

We are now ready to prove Theorem 5.4.12.

Proof of Theorem 5.4.12. We prove the theorem via the following sequence of claims. Suppose that X is a finite-dimensional normed space. Let f be a $(1, D, \tau_1, \tau_2, +\infty)$-threshold map to a Hilbert space.

The first claim is well-known and is a variant of Proposition 1.11 from [41].

Claim 5.4.17. *For every $x_1, x_2 \in X$ we have*

$$\|f(x_1) - f(x_2)\| \leq \max\{1, 2 \cdot \|x_1 - x_2\|\} \cdot \tau_1.$$

Proof. If $\|x_1 - x_2\| \leq 1$, then $\|f(x_1) - f(x_2)\| \leq \tau_1$, and we are done. Otherwise, let us take $y_0, y_1, \ldots, y_l \in X$ such that $y_0 = x_1$, $y_l = x_2$, $\|y_i - y_{i+1}\| \leq 1$ for every i, and $l = \lceil \|x_1 - x_2\| \rceil$. In particular, we can take $y_i = x_1 + i \cdot \frac{x_1 - x_2}{\|x_1 - x_2\|}$ for $i = 0, 1, \ldots l-1$, and $y_l = x_2$. We have

$$\|f(x_1) - f(x_2)\| \leq \sum_{i=0}^{l-1} \|f(y_i) - f(y_{i+1})\| \leq l\tau_1 = \lceil \|x_1 - x_2\| \rceil \cdot \tau_1 \leq 2\|x_1 - x_2\| \cdot \tau_1,$$

where the first step is by the triangle inequality, the second step follows from $\|y_i - y_{i+1}\| \leq 1$, and the last step follows from $\|x_1 - x_2\| \geq 1$. □

The proof of the next claim essentially appears in [94].

Claim 5.4.18. *There exists a map g from X to a Hilbert space such that for every $x_1, x_2 \in X$,*

- $\|g(x_1) - g(x_2)\| \leq (2\tau_1 \cdot \|x_1 - x_2\|)^{1/2}$;
- *if* $\|x_1 - x_2\| \geq D + 2$, *then* $\|g(x_1) - g(x_2)\| \geq \tau_2^{1/2} - (8\tau_1)^{1/2}$;

Proof. From Claim 5.4.17 and Lemma 5.4.13 we can get a map g' from X to a Hilbert space such that for every $x_1, x_2 \in X$

- $\|g'(x_1) - g'(x_2)\| \leq \max\left\{1, (2\|x_1 - x_2\|)^{1/2}\right\} \cdot \tau_1^{1/2}$;
- if $\|x_1 - x_2\| \geq D$, then $\|g'(x_1) - g'(x_2)\| \geq \tau_2^{1/2}$.

Let $N \subset X$ be a 1-net of X such that all the pairwise distances between points in N are more than 1. The map g' is $1/2$-Hölder on N with a constant $(2\tau_1)^{1/2}$, so we can apply Theorem 5.4.16 and get a map g that coincides with g' on N and is $1/2$-Hölder on the whole X with a constant $(2\tau_1)^{1/2}$. That is, for every $x_1, x_2 \in X$ we have

- $\|g(x_1) - g(x_2)\| \leq (2\tau_1 \cdot \|x_1 - x_2\|)^{1/2}$;
- if $x_1 \in N$, $x_2 \in N$ and $\|x_1 - x_2\| \geq D$, then $\|g(x_1) - g(x_2)\| \geq \tau_2^{1/2}$.

To conclude that g is as required, let us lower bound $\|g(x_1) - g(x_2)\|$ whenever $\|x_1 - x_2\| \geq D + 2$. Suppose that $x_1, x_2 \in X$ are such that $\|x_1 - x_2\| \geq D + 2$. Let $u_1 \in N$ be the closest net point to x_1 and, similarly, let $u_2 \in N$ be the closest net point to x_2. Observe that

$$\|u_1 - u_2\| \geq \|x_1 - x_2\| - \|x_1 - u_1\| - \|x_2 - u_2\| \geq (D + 2) - 1 - 1 = D.$$

We have

$$\|g(x_1) - g(x_2)\| \geq \|g(u_1) - g(u_2)\| - \|g(u_1) - g(x_1)\| - \|g(u_2) - g(x_2)\| \geq \tau_2^{1/2} - 2(2\tau_1)^{1/2},$$

as required, where the second step follows from the inequality $\|g(u_1) - g(u_2)\| \geq \tau_2^{1/2}$, which is true, since $u_1, u_2 \in N$, and that g is $1/2$-Hölder with a constant $(2\tau_1)^{1/2}$. □

The following claim completes the proof of Theorem 5.4.12.

Claim 5.4.19. *There exists a map h from X to a Hilbert space such that for every $x_1, x_2 \in X$:*

- $\|h(x_1) - h(x_2)\| \leq (2\tau_1 \cdot \|x_1 - x_2\|)^{1/2}$;
- $\|h(x_1) - h(x_2)\| \geq (\tau_2^{1/2} - (8\tau_1)^{1/2}) \cdot \min\{1, \|x_1 - x_2\|/(2D+4)\}$.

Proof. We take the map g from Claim 5.4.18 and apply Lemma 5.4.14 to it. Let us call the resulting map h. The first desired condition for h follows from a similar condition for g and Lemma 5.4.14. Let us prove the second one.

If $x_1 = x_2$, then there is nothing to prove. If $\|x_1 - x_2\| \geq D+2$, then by Claim 5.4.18 and Lemma 5.4.14, $\|h(x_1) - h(x_2)\| \geq \tau_2^{1/2} - (8\tau_1)^{1/2}$, and we are done. Otherwise, let us consider points $y_0, y_1, \ldots, y_l \in X$ such that $y_0 = 0$, $y_i - y_{i-1} = x_1 - x_2$ for every i, and $l = \left\lceil \frac{D+2}{\|x_1 - x_2\|} \right\rceil$. Since $\|y_l - y_0\| = \|l(x_1 - x_2)\| = l\|x_1 - x_2\| \geq D + 2$, we have

$$\begin{aligned}\tau_2^{1/2} - (8\tau_1)^{1/2} \leq \|h(y_l) - h(y_0)\| &\leq \sum_{i=1}^{l} \|h(y_i) - h(y_{i-1})\| \\ &= l \cdot \|h(x_1) - h(x_2)\| \leq \frac{2D+4}{\|x_1 - x_2\|} \cdot \|h(x_1) - h(x_2)\|,\end{aligned}$$

where the equality follows from the conclusion of Lemma 5.4.14. □

Finally, observe that Theorem 5.4.12 is merely a reformulation of Claim 5.4.19. □

5.4.4 Putting it all together

We now show that Theorem 5.4.1 follows by applying Lemma 5.4.3, Theorem 5.4.4, Lemma 5.4.6, and Theorem 5.4.12, in this order, with an appropriate choice of parameters.

Proof of Theorem 5.4.1. Suppose $\mathrm{DTEP}(X, D)$ admits a protocol of size s. By setting $k = Cs$ in Lemma 5.4.3 (C is a large absolute constant, to be chosen later), we conclude that $\mathrm{DTEP}(\ell_\infty^{Cs}(X), CsD)$ admits a protocol of size $s' = O(s)$.

Now choosing C large enough and applying Theorem 5.4.4 (in contrapositive), we conclude that X has no Poincaré inequalities for distance scales 1 and CsD, with $\alpha = 0.01$ and $\beta = 0.001$.

Applying Lemma 5.4.6 (in contrapositive), we conclude that X allows a threshold map to a Hilbert space with parameters $(1, CsD, 1, 10, +\infty)$.

Using Theorem 5.4.12, it follows that there is a map h from X to a Hilbert space, such that for all $x_1, x_2 \in X$,

$$\min\left\{1, \frac{\|x_1 - x_2\|}{s \cdot D}\right\} \leq \|h(x_1) - h(x_2)\| \leq K \cdot \|x_1 - x_2\|^{1/2},$$

where $K > 1$ is an absolute constant, and this proves the theorem. □

Remark: Instead of applying Lemma 5.4.3 and Theorem 5.4.4, we could have attempted to apply the reduction from [21] to get a threshold map from X to a Hilbert space directly. That approach is much simpler technically, but has two fatal drawbacks. First, we end up with a threshold map with a gap between τ_1 and τ_2 being arbitrarily close to 1, and thus, we are unable to invoke Theorem 5.4.12, which requires the gap to be more than 8. Second, the parameters of the resulting threshold map are *exponential* in the number of bits in the communication protocol, which is bad for the quantitative bounds from Section 5.5.

5.5 Quantitative bounds

In this section we prove the quantitative version of our results, namely Theorem 5.1.2 and Theorem 5.1.3, for which we will reuse Theorem 5.4.1. In particular, we prove the following theorem.

Theorem 5.5.1. *For a finite-dimensional normed space X and $\Delta > 1$, assume we have a map $f\colon X \to H$ to a Hilbert space H, such that, for an absolute constant $K > 0$ and for every $x_1, x_2 \in X$:*

- $\|f(x_1) - f(x_2)\|_H \leq K \cdot \|x_1 - x_2\|_X^{1/2}$; *and*

- *if* $\|x_1 - x_2\|_X \geq \Delta$, *then* $\|f(x_1) - f(x_2)\|_H \geq 1$.

Then, for any $\varepsilon \in (0, 1/3)$, *the space* X *linearly embeds into* $\ell_{1-\varepsilon}$ *with distortion* $O(\Delta/\epsilon)$.

Note that Theorem 5.1.2 now follows from applying Theorem 5.4.1 together with Theorem 5.5.1 for $\Delta = sD$. We can further prove Theorem 5.1.3 by using the following result of Zvavitch from [181].

Lemma 5.5.2 ([181])**.** *Every* d*-dimensional subspace of* $L_{1-\varepsilon}$ *embeds linearly into* $\ell_{1-\varepsilon}^{d\cdot\mathrm{poly}(\log d)}$ *with distortion* $O(1)$.

Indeed, applying Lemma 5.5.2 together with Theorem 5.5.1, we get that for every $0 < \varepsilon < 1/3$ the space X linearly embeds into $\ell_{1-\varepsilon}^{\mathrm{poly}(\dim X)}$ with distortion $O(\Delta/\varepsilon)$. Thus, X is embeddable into ℓ_1 with distortion

$$O\big(\Delta \cdot (\dim X)^{O(\varepsilon)}/\varepsilon\big).$$

Setting $\varepsilon = \Theta(1/\log(\dim X))$, we obtain Theorem 5.1.3.

It remains to prove Theorem 5.5.1. Its proof proceeds by adjusting the arguments from [3] and [138].

Proof of Theorem 5.5.1. Fix X, $\Delta > 0$, and the corresponding map $f\colon X \to H$. We first prove the following lemma.

Lemma 5.5.3. *There exists a probability measure* μ *on* $\mathbb{R}^{\dim X}$ *symmetric around the origin such that its (real-valued) characteristic function* $\varphi : X \to \mathbb{R}$ *has the following properties for every* $x \in X$*:*

- $\varphi(x) \geq e^{-\widetilde{K}\cdot\|x\|_X}$*; and*
- *if* $\|x\|_X \geq \Delta$, *then* $\varphi(x) \leq 1/e$.

Here $\widetilde{K} > 0$ *is an absolute constant.*

Proof. It is known from [161] that for a Hilbert space H the function $g\colon h \mapsto e^{-\|h\|_H^2}$ is positive-definite. Thus, there exists a function $\widetilde{g}\colon H \to \widetilde{H}$ to a Hilbert space $\widetilde{H}$ such

that for every $h_1, h_2 \in H$ one has $\left\langle \widetilde{g}(h_1), \widetilde{g}(h_2) \right\rangle_{\widetilde{H}} = e^{-\|h_1 - h_2\|_H^2}$. Setting $\widetilde{f} = \widetilde{g} \circ f$, we get a function $\widetilde{f} \colon X \to \widetilde{H}$ to a Hilbert space such that for an absolute constant $\widetilde{K} > 0$ for every $x_1, x_2 \in X$, we have:

- $\left\| \widetilde{f}(x_1) \right\|_{\widetilde{H}} = 1$;
- $\left\langle \widetilde{f}(x_1), \widetilde{f}(x_2) \right\rangle_{\widetilde{H}} \geq e^{-\widetilde{K} \cdot \|x_1 - x_2\|_X}$; and
- if $\|x_1 - x_2\|_X \geq \Delta$, then $\left\langle \widetilde{f}(x_1), \widetilde{f}(x_2) \right\rangle_{\widetilde{H}} \leq 1/e$.

Applying Lemma 5.2.7 and Lemma 5.2.4, we obtain a positive-definite function $\varphi \colon X \to \mathbb{R}$ such that:

- $\varphi(0) = 1$;
- for every $x \in X$ one has $\varphi(x) \geq e^{-\widetilde{K} \cdot \|x\|_X}$; and
- if $\|x\|_X \geq \Delta$, then $\varphi(x) \leq 1/e$.

We can now use Bochner's theorem, which is the following characterization of continuous positive-definite functions, via the Fourier transform.

Theorem 5.5.4 (Bochner's theorem, see, e.g., [74])**.** *If a map $f \colon \mathbb{R}^d \to \mathbb{R}$ is positive-definite, continuous in zero, and $f(0) = 1$, then there exists a probability measure μ on $\mathbb{R}^d$ such that f is the μ's characteristic function. That is, for every $x \in \mathbb{R}^d$,*

$$f(x) = \int_{\mathbb{R}^d} e^{i\langle x, v \rangle} \, \mu(dv).$$

In particular, note that we have that $\varphi(0) = 1$, φ is positive-definite and is continuous at zero. Hence, by Bochner's theorem, we get a probability measure μ over $\mathbb{R}^{\dim X}$ whose characteristic function equals to φ. That is, for every $x \in X$ we get

$$\varphi(x) = \int_{\mathbb{R}^{\dim X}} e^{i\langle x, v \rangle} \, \mu(dv),$$

where $\langle \cdot, \cdot \rangle$ is the standard dot product in $\mathbb{R}^{\dim X}$. Clearly, μ is symmetric around the origin, since φ is real-valued. This completes the proof of Lemma 5.5.3. □

Our next goal is to show that μ gives rise to a one-measurement *linear* sketch for X with approximation $O(\Delta)$ and a certain additional property that will be useful to us. The following lemma contains two standard facts about *one-dimensional* characteristic functions (see, e.g., [143]). We include the proof for completeness.

Lemma 5.5.5. *Let ν be a symmetric probability measure over the real line, and let*

$$\psi(t) = \int_{\mathbb{R}} e^{ivt}\,\nu(dv)$$

be its characteristic function (which is real-valued due to the symmetry of ν). Then,

- *if for some $R > 0$ and $0 < \varepsilon < 1$ we have $|\psi(R)| \le 1 - \varepsilon$, then*

$$\nu\Big(\big\{v \in \mathbb{R}\colon |v| \ge \Omega_\varepsilon(1/R)\big\}\Big) \ge \Omega_\varepsilon(1); \tag{5.10}$$

- *for every $\delta > 0$ one has*

$$\nu\Big(\big\{v \in \mathbb{R}\colon |v| \ge 1/\delta\big\}\Big) \le O(1/\delta) \cdot \int_{-\delta}^{\delta} \big(1 - \psi(t)\big)\,dt. \tag{5.11}$$

Proof. Let us start with proving the first claim. We have for every $\alpha > 0$

$$\begin{aligned} 1 - \varepsilon \ge |\psi(R)| &\ge \int_{\mathbb{R}} \cos(vR)\,\nu(dv) \\ &\ge \cos\alpha \cdot \nu\Big(\big\{v \in \mathbb{R}\colon |vR| \le \alpha\big\}\Big) - \nu\Big(\big\{v \in \mathbb{R}\colon |vR| > \alpha\big\}\Big) \\ &= (1 + \cos\alpha) \cdot \nu\Big(\big\{v \in \mathbb{R}\colon |vR| \le \alpha\big\}\Big) - 1, \end{aligned}$$

where the second step uses the fact that ψ is real-valued. Thus, we have

$$\nu\Big(\big\{v \in \mathbb{R}\colon |vR| \le \alpha\big\}\Big) \le \frac{2 - \varepsilon}{1 + \cos\alpha}.$$

Setting $\alpha = \Theta\big(\sqrt{\varepsilon}\big)$, we get the desired bound.

Now let us prove the second claim. We have, for every $\delta > 0$,

$$\int_{-\delta}^{\delta} \big(1 - \psi(t)\big)\, dt = \int_{-\delta}^{\delta} \int_{\mathbb{R}} \big(1 - e^{ivt}\big)\, \nu(dv)\, dt = 2\delta \cdot \int_{\mathbb{R}} \left(1 - \frac{\sin(\delta v)}{\delta v}\right)\, \nu(dv)$$
$$\geq 2(1 - \sin 1) \cdot \delta \cdot \nu\Big(\{v \in \mathbb{R} \colon |\delta v| \geq 1\}\Big),$$

where we use that $(1 - \sin y/y) > (1 - \sin 1)$ for every y such that $|y| > 1$. □

Now we will show that the probability measure μ from Lemma 5.5.3 gives a good linear sketch for X. To see this we use the conditioning on the characteristic function of μ, namely, we exploit them using the above Lemma 5.5.5. In order to do this, we look at the one-dimensional projections of μ as follows. Let $x \in X$ be a fixed vector. For a measurable subset $A \subseteq \mathbb{R}$ we define

$$\nu(A) = \mu\Big(\big\{v \in \mathbb{R}^{\dim X} \colon \langle x, v\rangle \in A\big\}\Big).$$

It is immediate to check that the characteristic function ψ of ν is as follows: $\psi(t) = \varphi(t \cdot x)$ (recall that φ is the characteristic function of μ). Next we apply Lemma 5.5.5 to ψ and use the properties of φ from the conclusion of Lemma 5.5.3. Namely, we get for every $x \in X$:

$$\mu\Big(\big\{v \in \mathbb{R}^{\dim X} \colon |\langle x, v\rangle| \geq \Omega(\|x\|_X/\Delta)\big\}\Big) = \Omega(1); \tag{5.12}$$

and for every $t > 0$,

$$\mu\Big(\big\{v \in \mathbb{R}^{\dim X} \colon |\langle x, v\rangle| \geq t \cdot \|x\|_X\big\}\Big) \leq O(1/t). \tag{5.13}$$

Indeed, (5.12) follows from the bound $\varphi(x) \leq 1/e$ whenever $\|x\|_X \geq \Delta$ and (5.10). The inequality (5.13) follows from the estimate $\varphi(x) \geq e^{-\widetilde{K} \cdot \|x\|_X}$ and (5.11) (for

$1/\delta = t\|x\|_X$) that together give

$$\mu\Big(\big\{v \in \mathbb{R}^{\dim X} \colon |\langle x, v\rangle| \geq t \cdot \|x\|_X\big\}\Big) = \nu(\{r \in \mathbb{R} \colon |r| \geq t\|x\|_X\}) \\ \leq O(t\|x\|_X) \int_{-1/t\|x\|_X}^{1/t\|x\|_X} (1 - e^{-\tilde{K}s\|x\|_X})\, ds,$$

as well as from the inequality

$$t \int_{-1/t}^{1/t} \big(1 - e^{-Cs}\big)\, ds \leq t \int_{-1/t}^{1/t} \big(1 - (1 - Cs)\big)\, ds = C/t.$$

Hence, μ gives rise to a one-measurement linear sketch of X with approximation $O(\Delta)$, whose "upper tail" is not too heavy.

Finally, we are ready to describe a desired linear embedding of X into $L_{1-\varepsilon}$; we map X into $L_{1-\varepsilon}(\mu)$ as follows: $x \mapsto (v \mapsto \langle x, v\rangle)$. The following Lemma states that the distortion of this embedding is $O(\Delta/\varepsilon)$, as required.

Lemma 5.5.6. *For $0 < \varepsilon < 1/3$ and every $x \in X$,*

$$\Omega\big(\|x\|_X/\Delta\big) \leq \|v \mapsto \langle x, v\rangle\|_{L_{1-\varepsilon}(\mu)} \leq O\big(\|x\|_X/\varepsilon\big).$$

Proof. The lower bound is straightforward:

$$\|v \mapsto \langle x, v\rangle\|^{1-\varepsilon}_{L_{1-\varepsilon}(\mu)} = \int_{\mathbb{R}^{\dim X}} \big|\langle x, v\rangle\big|^{1-\varepsilon}\, \mu(dv) \geq \Omega(1) \cdot \Omega(\|x\|_X/\Delta)^{1-\varepsilon},$$

where the last step follows from (5.12).

For the upper bound, we have for every $\alpha > 0$,

$$\|v \mapsto \langle x, v\rangle\|^{1-\varepsilon}_{L_{1-\varepsilon}(\mu)} = \int_{\mathbb{R}^{\dim X}} \big|\langle x, v\rangle\big|^{1-\varepsilon}\, \mu(dv) \\ = \int_0^\infty \mu\Big(\big\{v \in \mathbb{R}^{\dim X} \colon |\langle x, v\rangle|^{1-\varepsilon} \geq s\big\}\Big)\, ds \\ \leq \alpha + O(1) \cdot \|x\|_X \cdot \int_\alpha^\infty s^{-\frac{1}{1-\varepsilon}}\, ds \leq \alpha + O(1) \cdot \frac{\|x\|_X}{\varepsilon} \cdot \alpha^{-\frac{\varepsilon}{1-\varepsilon}},$$

where the third step follows from (5.13). Choosing $\alpha = \|x\|_X^{1-\varepsilon}/\varepsilon^{1-\varepsilon}$, we get

$$\|v \mapsto \langle x, v\rangle\|_{L_{1-\varepsilon}(\mu)} \leq O(1) \cdot 2^{\frac{1}{1-\varepsilon}} \cdot \frac{\|x\|_X}{\varepsilon}.$$

□

This concludes the proof of Theorem 5.5.1. □

5.6 Embedding into ℓ_1 via sum-products

Finally, we prove Theorem 5.1.4: good sketches for norms closed under the sum-product imply embeddings into ℓ_1 with constant distortion. First we invoke Theorem 5.4.1 and get a sequence of good uniform embeddings into a Hilbert space, whose moduli depend only on the sketch size and the approximation. Then, we use the main result of this section: Lemma 5.6.1. Before stating the lemma, let us remind a few notions. For a metric space X, recall that the metric space $\ell_1^k(X) = \bigoplus_{\ell_1}^k X_n$ is the direct sum of k copies of X, with the associated distance defined as a sum-product (ℓ_1-product) over the k copies. We define $\ell_1(X)$ similarly. We also denote $X \oplus_{\ell_1} Y$ the sum-product of X and Y.

Lemma 5.6.1. *Let $(X_n)_{n=1}^\infty$ be a sequence of finite-dimensional normed spaces. Suppose that for every $i_1, i_2 \geq 1$ there exists $m = m(i_1, i_2) \geq 1$ such that $X_{i_1} \oplus_{\ell_1} X_{i_2}$ is isometrically embeddable into X_m. If every X_n admits a uniform embedding into a Hilbert space with moduli independent of n, then every X_n is linearly embeddable into ℓ_1 with distortion independent of n.*

Note that Theorem 5.1.4 just follows from combining Lemma 5.6.1 with Theorem 5.4.1.

Before proving Lemma 5.6.1, we state the following two useful theorems. The first one (Theorem 5.6.2) follows from the fact that uniform embeddability into a Hilbert space is determined by embeddability of finite subsets [41]. The second one (Theorem 5.6.3) follows by composing results of Aharoni, Maurey, and Mityagin [3] and Kalton [95].

Theorem 5.6.2 (Proposition 8.12 from [41]). *Let $A_1 \subset A_2 \subset \ldots$ be metric spaces and let $A = \bigcup_i A_i$. If every A_n is uniformly embeddable into a Hilbert space with moduli independent of n, then the whole A is uniformly embeddable into a Hilbert space.*

Theorem 5.6.3 ([3, 95]). *A Banach space X is linearly embeddable into L_1 iff $\ell_1(X)$ is uniformly embeddable into a Hilbert space.*

We are now ready to proceed with the proof of Lemma 5.6.1.

Proof of Lemma 5.6.1. Let $X = X_1 \oplus_{\ell_1} X_2 \oplus_{\ell_1} \ldots$. More formally,

$$X = \Big\{(x_1, x_2, \ldots) : x_i \in X_i, \sum_i \|x_i\| < \infty\Big\},$$

where the norm is defined as follows:

$$\big\|(x_1, x_2, \ldots)\big\| = \sum_i \|x_i\|.$$

We claim that the space $\ell_1(X)$ embeds uniformly into a Hilbert space. To see this, consider $U_p = \ell_1^p(X_1 \oplus_{\ell_1} X_2 \oplus_{\ell_1} \ldots \oplus_{\ell_1} X_p)$, which can be naturally seen as a subspace of $\ell_1(X)$. Then, $U_1 \subset U_2 \subset \ldots \subset U_p \subset \ldots \subset \ell_1(X)$ and $\bigcup_p U_p$ is dense in $\ell_1(X)$. By the assumption of the lemma, U_p is isometrically embeddable into X_m for some m, thus, U_p is uniformly embeddable into a Hilbert space with moduli independent of p. Now, by Theorem 5.6.2, $\bigcup_p U_p$ is uniformly embeddable into a Hilbert space. Since $\bigcup_p U_p$ is dense in $\ell_1(X)$, the same holds also for the whole $\ell_1(X)$, as claimed.

Finally, since $\ell_1(X)$ embeds uniformly into a Hilbert space, we can apply Theorem 5.6.3 and conclude that X is linearly embeddable into L_1. The lemma follows since X contains every X_i as a subspace. □

5.7 Appendix: EMD reduction

Recall that EMD_n is a normed space on all signed measures on $[n]^2$ (that sum up to zero). We also take the view that a weighted set in $[n]^2$ is in fact a measure on $[n]^2$.

Lemma 5.7.1. *Suppose the EMD metric between non-negative measures (of the same total measure) admits a sketching algorithm* $\mathbf{sk}$ *with approximation* $D > 1$ *and sketch size* s*. Then the normed space* EMD_n *admits a sketching algorithm* $\mathbf{sk}'$ *with approximation* D *and sketch size* $O(s)$*.*

Proof. The main idea is that if x, y are signed measures and we add a sufficiently large term $M > 0$ to all of their coordinates, then the resulting vectors $x' = x + M \cdot \vec{1}$ and $y' = y + M \cdot \vec{1}$ are measures (all their coordinates are non-negative) of the same total mass, and $\|x - y\|_{\mathrm{EMD}}$ is equal to the EMD distance between measures x', y'. The trouble is in identifying a large enough M. We use the values of x and y themselves to agree on M. Details follow.

Without loss of generality we can fix the DTEP threshold to be $r = 1$.

We design the sketch $\mathbf{sk}'$ as follows. First choose a hash function $h : \mathbb{N} \to \{0,1\}^9$ (using public randomness). Fix an input $x \in \mathbb{R}^{n^2}$ of total measure zero, i.e., $\sum_i x_i = 0$. Let $m(x) = \min_i x_i$, and let $b(x)$ be the largest multiple of 2 that is smaller than $m(x)$. Since x has total measure zero, $b(x) < m(x) \leq 0$. Now let $b^{(1)}(x) = b(x)$ and $b^{(2)}(x) = b(x) - 2$, and then $x^{(q)} = x - b^{(q)}(x) \cdot \vec{1}$ for $q = 1, 2$. Notice that in both cases $x^{(q)} > x \geq 0$ (component-wise). Now let the sketch $\mathbf{sk}'(x)$ be the concatenation of $\mathbf{sk}(x^{(q)}), h(b^{(q)}(x))$ for $q = 1, 2$.

The distinguisher works as follows, given two sketches

$$\mathbf{sk}'(x) = (\mathbf{sk}(x^{(q)}), h(b^{(q)}(x)))_{q=1,2}$$

and

$$\mathbf{sk}'(y) = (\mathbf{sk}(y^{(q)}), h(b^{(q)}(x)))_{q=1,2}.$$

If there are $q_x, q_y \in \{1, 2\}$ whose hashes agree $h(b^{(q)}(x)) = h(b^{(q)}(y))$ (breaking ties arbitrarily if there are multiple possible agreements), then output whatever the EMD metric distinguisher would output on $\mathbf{sk}(x^{(q_x)}), \mathbf{sk}(y^{(q_y)})$. Otherwise output "far" (i.e., that $\|x - y\|_{\mathrm{EMD}} > D$).

To analyze correctness, consider the case when $\|x - y\|_{\mathrm{EMD}} \leq 1$. Without loss of generality, suppose $m(x) \geq m(y)$. Then $m(x) - m(y) \leq 1$ (otherwise x, y are further

away in EMD norm than 1). Hence either $b(x) = b(y)$ or $b(x) = b(y) + 2$. Then there exists a corresponding $q \in \{1, 2\}$ for which the hashes agree $h(b^{(1)}(x)) = h(b^{(q)}(y))$. By properties of the hash function, with sufficiently large constant probability the hashes match only when the b's match, in which case the values q_x, q_y used by the distinguisher satisfy $b^{(q_x)}(x) = b^{(q_y)}(y)$. In this case, $\|x - y\|_{\text{EMD}} = d_{\text{EMD}}(x^{(q_x)}, y^{(q_y)})$, and the correctness now depends on **sk**, and the distinguisher for the EMD metric.

Otherwise, if $\|x - y\|_{\text{EMD}} > D$, either the b-values coincide for some q_x, q_y and then the above argument applies again, or with sufficiently large constant probability the hashes will not agree and the distinguisher outputs (correctly) "far".

There is a small loss in success probability due to use of the hash function, but that can be amplified back by independent repetitions. □

Notice that the above lemma assumes a sketching algorithm for the EMD metric between any non-negative measures of the same total measure, and not only in the case where the total measure is 1. The proof can be easily modified so that any non-negative measure being used always has a fixed total measure (say 1, by simply scaling the inputs), which translates to scaling the threshold r of the DTEP problem. This is acceptable because, under standard definitions, a metric space is called sketchable if it admits a sketching scheme for every threshold $r > 0$.

Chapter 6

Hardness results for the ANN problem

6.1 Introduction

Lower bounds for NNS and ANN have received considerable attention. Such lower bounds are ideally obtained in the *cell-probe* model [127, 126], where one measures the *number of memory cells* the query algorithm accesses. Despite a number of success stories, high cell-probe lower bounds are notoriously hard to prove. In fact, there are few techniques for proving high cell-probe lower bounds, for any (static) data structure problem. For ANN in particular, we have no viable techniques to prove $\omega(\log n)$ query time lower bounds. Due to this state of affairs, one may rely on *restricted* models of computation, which nevertheless capture existing algorithmic approaches.

Early lower bounds for NNS were obtained for data structures in *exact* or *deterministic* settings [44, 52, 35, 115, 88, 53, 150, 180]. [53, 116] obtained an almost tight cell-probe lower bound for the randomized Approximate *Nearest* Neighbor Search under the ℓ_1 distance. In that problem, there is no distance threshold r, and instead the goal is to find a data point that is not much further than the *closest* data point. This twist is the main source of hardness, so the result is not applicable to the ANN problem as introduced above.

There are few results that show lower bounds for *randomized* data structures for

ANN. The first such result [19] shows that any data structure that solves $(1+\varepsilon, r)$-ANN for ℓ_1 or ℓ_2 using t cell probes requires space $n^{\Omega(1/t\varepsilon^2)}$.[1] This result shows that the algorithms of [85, 108] are tight up to constants in the exponent for $t = O(1)$.

In [146] (following up on [145]), the authors introduce a general framework for proving lower bounds for ANN under any metric. They show that lower bounds for ANN are implied by the *robust expansion* of the underlying metric space. Using this framework, [146] show that (c, r)-ANN using t cell probes requires space $n^{1+\Omega(1/tc)}$ for the Manhattan distance and $n^{1+\Omega(1/tc^2)}$ for the Euclidean distance (for every $c > 1$).

Lower bounds have also been obtained for other metrics. For the ℓ_∞ distance, [11] show a lower bound for deterministic ANN data structures. This lower bound was later generalized to randomized data structures [146, 98]. A recent result [2] adapts the framework of [146] to Bregman divergences.

To prove higher lower bounds, researchers resorted to lower bounds for restricted models. These examples include: decision trees [11] (the corresponding upper bound from [81] is in the same model), LSH [129, 140, 17] and data-dependent LSH [27].

6.1.1 Our results

We show new *cell-probe* and *restricted* lower bounds for (c, r)-ANN matching the upper bounds from Chapter 2. All our lower bounds rely on a certain canonical hard distribution for the Hamming space (defined later in Section 6.2.1). Via a standard reduction [114], we obtain similar hardness results for ℓ_p with $1 < p \leq 2$ (with c being replaced by c^p).

One cell probe

First, we show a tight lower bound on the space needed to solve ANN for a random instance, for query algorithms that use a *single* cell probe. More formally, we prove the following theorem:

Theorem 6.1.1 (see Section 6.3.2)**.** *Any data structure that:*

[1] The correct dependence on $1/\varepsilon$ requires the stronger Lopsided Set Disjointness lower bound from [147].

- *solves* (c, r)*-ANN for the random instance defined in Section 6.2.1 with probability at least* $2/3$*,*
- *operates on memory cells of size* $n^{o(1)}$*,*
- *for each query, looks up a* single *cell,*

must use at least $n^{\left(\frac{c}{c-1}\right)^2 - o(1)}$ *words of memory.*

The space lower bound matches:

- The upper bound from Chapter 2 for *random instances* that can be made single-probe;
- The same upper bound for worst-case instances with query time $n^{o(1)}$.

The previous best lower bound from [146] for a single probe are weaker by a polynomial factor.

We prove Theorem 6.1.1 by computing tight bounds on the robust expansion of a hypercube $\{-1, 1\}^d$ as defined in [146]. Then, we invoke a result from [146], which yields the desired cell probe lower bound. We obtain estimates on the robust expansion via a combination of the hypercontractivity inequality and Hölder's inequality [139]. Equivalently, one could obtain the same bounds by an application of the Generalized Small-Set Expansion Theorem for $\{-1, 1\}^d$ from [139].

Two cell probes

To state our results for two cell probes, we first define the *decision* version of ANN (first introduced in [146]). Suppose that with every data point $p \in P$ we associate a bit $x_p \in \{0, 1\}$. A new goal is: given a query $q \in \{-1, 1\}^d$ which is within distance r from a data point $p \in P$, if $P \setminus \{p\}$ is at distance at least cr from q, return x_p with probability at least $2/3$. It is easy to see that any algorithm for (c, r)-ANN would solve this decision version.

We prove the following lower bound for data structures making only two cell probes per query.

Theorem 6.1.2 (see Section 6.5)**.** *Any data structure that:*

- *solves the decision ANN for the random instance (Section 6.2.1) with probability* $2/3$*,*
- *operates on memory cells of size* $o(\log n)$*,*
- *accesses at most two cells for each query,*

must use at least $n^{\left(\frac{c}{c-1}\right)^2 - o(1)}$ *words of memory.*

Informally speaking, Theorem 6.1.2 shows that the second cell probe cannot improve the space bound by more than a subpolynomial factor. To the best of our knowledge, this is the first lower bound on the space of *any* static data structure problem without a polynomial gap between $t = 1$ and $t \geq 2$ cell-probes. Previously, the highest ANN lower bound for two queries was weaker by a polynomial factor [146]. This remains the case even if we plug the tight bound on the robust expansion of a hypercube into the framework of [146]. Thus, in order to obtain a higher lower bound for $t = 2$, we must depart from the framework of [146].

Our proof establishes a connection between two-query data structures (for the decision version of ANN), and two-query locally-decodable codes (LDC) [179]. A possibility of such a connection was suggested in [145]. In particular, we show that any data structure violating the lower bound from Theorem 6.1.2 implies a too-good-to-be-true two-query LDC, which contradicts known LDC lower bounds from [101, 39].

The first lower bound for unrestricted two-query LDCs was proved in [101] via a *quantum* argument. Later, the argument was simplified and made *classical* in [39]. It turns out that, for our lower bound, we need to resort to the original quantum argument of [101] since it has a better dependence on the noise rate a code is able to tolerate. During the course of our proof, we do not obtain a full-fledged LDC, but rather an object which can be called an *LDC on average*. For this reason, we are unable to use [101] as a black box but rather adjust their proof to the average case.

Finally, we point out an important difference with Theorem 6.1.1: in Theorem 6.1.2 we allow words to be merely of size $o(\log n)$ (as opposed to $n^{o(1)}$). Nevertheless, for

the *decision version* of ANN for random instances our upper bounds hold even for such "tiny" words. In fact, our techniques do not allow us to handle words of size $\Omega(\log n)$ due to the weakness of known lower bounds for two-query LDC for *large alphabets*. In particular, our argument can not be pushed beyond word size $2^{\widetilde{\Theta}(\sqrt{\log n})}$ *in principle*, since this would contradict known constructions of two-query LDCs over large alphabets [71]!

The general time–space trade-off

Finally, we prove *conditional* lower bound on the entire time–space trade-off from Chapter 2 matching our upper bounds that up to $n^{o(1)}$ factors. Note that—since we show polynomial query time lower bounds—proving similar lower bounds *unconditionally* is far beyond the current reach of techniques. Any such statement would constitute a major breakthrough in cell probe lower bounds.

Our lower bounds are proved in the following model, which can be loosely thought of comprising all hashing-based frameworks we are aware of:

Definition 6.1.3. A *list-of-points data structure* for the ANN problem is defined as follows:

- We fix (possibly random) sets $A_i \subseteq \{-1,1\}^d$, for $1 \leq i \leq m$; also, with each possible query point $q \in \{-1,1\}^d$, we associate a (random) set of indices $I(q) \subseteq [m]$;
- For a given dataset P, we maintain m lists of points $L_1, L_2, \ldots, L_m$, where $L_i = P \cap A_i$;
- On query q, we scan through each list L_i for $i \in I(q)$ and check whether there exists some $p \in L_i$ with $\|p - q\|_1 \leq cr$. If it exists, return p.

The total space is defined as $s = m + \sum_{i=1}^{m} |L_i|$ and the query time is $t = |I(q)| + \sum_{i \in I(q)} |L_i|$.

For this model, we prove the following theorem.

Theorem 6.1.4 (see Section 6.4)**.** *Consider any list-of-points data structure for the (c, r)-ANN problem for random instances of n points in the d-dimensional Hamming space with $d = \omega(\log n)$, which achieves a total space of $n^{1+\rho_u}$, and has query time $n^{\rho_q - o(1)}$, for $2/3$ success probability. Then it must hold that:*

$$c\sqrt{\rho_q} + (c-1)\sqrt{\rho_u} \geq \sqrt{2c-1}. \tag{6.1}$$

6.2 Preliminaries

We introduce a few techniques and concepts to be used primarily for our lower bounds. We start by defining the approximate nearest neighbor search problem.

Definition 6.2.1. The goal of the (c, r)-approximate nearest neighbor problem with failure probability δ is to construct a data structure over a set of points $P \subset \{-1, 1\}^d$ supporting the following query: given any point q such that there exists some $p \in P$ with $\|q - p\|_1 \leq r$, report some $p' \in P$ where $\|q - p'\|_1 \leq cr$ with probability at least $1 - \delta$.

6.2.1 Random Hamming instances

Here we define a random instance that will be the hard distribution for all the further lower bounds. It corresponds to the (c, r)-ANN problem over the Hamming space $\{0, 1\}^d$ and is similar to Euclidean random instances from Section 2.6 that were crucial for the algorithms from Chapter 2.

- A dataset $P \subset \{0, 1\}^d$ is given by n points, where each vector is drawn independently and uniformly at random from $\{0, 1\}^d$. We assume that $d = \omega(\log n)$.
- A query $q \in \{0, 1\}^d$ is drawn by first choosing a dataset point $p \in P$ uniformly at random, and then choosing q by flipping each bit of p independently with probability $\frac{1}{2c}$.
- The goal of the data structure is to preprocess P in order to recover the data point p from the query point q quickly.

Any $\left(c - o(1), (1 + o(1)) \cdot \frac{d}{2c}\right)$-ANN data structure for the unit sphere must be able to find p given q with high probability. Indeed, the distance between p and q is at most $(1 + o(1)) \cdot \frac{d}{2c}$ with high probability, while all the distances from q to other data points are at least $(1 - o(1)) \cdot \frac{d}{2}$ with high probability, where in both cases we use that $d = \omega(\log n)$.

6.2.2 Graphical neighbor search and robust expansion

We introduce a few definitions from [146] to setup the lower bounds for the ANN.

Definition 6.2.2 ([146])**.** In the *Graphical Neighbor Search problem* (GNS), we are given a bipartite graph $G = (U, V, E)$ where the dataset comes from U and the queries come from V. The dataset consists of pairs $P = \{(p_i, x_i) \mid p_i \in U, x_i \in \{0, 1\}, i \in [n]\}$. On query $q \in V$, if there exists a unique p_i with $(p_i, q) \in E$, then we want to return x_i.

We will sometimes use the GNS problem to prove lower bounds on (c, r)-ANN as follows: we build a GNS graph G by taking $U = V = \{-1, 1\}^d$, and connecting two points $u \in U, v \in V$ iff their Hamming distance most r (see details in [146]). We will also ensure q is not closer than cr to other points apart from the near neighbor.

[146] showed lower bounds for ANN are intimately tied to the following quantity of a metric space.

Definition 6.2.3 (Robust Expansion [146])**.** Consider a GNS graph $G = (U, V, E)$, and fix a distribution e on $E \subset U \times V$, where μ is the marginal distribution on U and η is the marginal distribution on V. For $\delta, \gamma \in (0, 1]$, the *robust expansion* $\Phi_r(\delta, \gamma)$ is:

$$\Phi_r(\delta, \gamma) = \min_{A \subset V : \eta(A) \leq \delta} \min_{B \subset U : \frac{e(A \times B)}{e(A \times V)} \geq \gamma} \frac{\mu(B)}{\eta(A)}.$$

6.2.3 Locally-decodable codes (LDC)

Our 2-probe lower bounds uses results on Locally-Decodable Codes (LDCs). We present the standard definitions and results on LDCs below, although in Section 6.5, we will use a weaker definition of LDCs for our 2-query lower bound.

Definition 6.2.4. A (t, δ, ε) locally-decodable code (LDC) encodes n-bit strings $x \in \{0,1\}^n$ into m-bit codewords $C(x) \in \{0,1\}^m$ such that, for each $i \in [n]$, the bit x_i can be recovered with probability $\frac{1}{2} + \varepsilon$ while making only t queries into $C(x)$, even if the codeword is arbitrarily modified (corrupted) in δm bits.

We will use the following lower bound on the size of the LDCs.

Theorem 6.2.5 (Theorem 4 from [101])**.** *If $C : \{0,1\}^n \to \{0,1\}^m$ is a $(2, \delta, \varepsilon)$-LDC, then*

$$m \geq 2^{\Omega(\delta \varepsilon^2 n)}.$$

6.3 One-probe data structures

6.3.1 Robust expansion of the Hamming space

The goal of this section is to compute tight bounds for the robust expansion $\Phi_r(\delta, \gamma)$ in the Hamming space of dimension d, as defined in the preliminaries. We use these bounds for all of our lower bounds in the subsequent sections.

We use the following model for generating dataset points and queries corresponding to the random instance of Section 6.2.1.

Definition 6.3.1. For any $x \in \{-1,1\}^n$, $N_\sigma(x)$ is a probability distribution over $\{-1,1\}^n$ representing the neighborhood of x. We sample $y \sim N_\sigma(x)$ by choosing $y_i \in \{-1,1\}$ for each coordinate $i \in [d]$ independently; with probability σ, we set $y_i = x_i$, and with probability $1 - \sigma$, y_i is set uniformly at random.

Given any Boolean function $f : \{-1,1\}^n \to \mathbb{R}$, the function $T_\sigma f : \{-1,1\}^n \to \mathbb{R}$ is

$$T_\sigma f(x) = \mathop{\mathbb{E}}_{y \sim N_\sigma(x)}[f(y)] \tag{6.2}$$

In the remainder of this section, will work solely on the Hamming space $V = \{-1,1\}^d$. We let

$$\sigma = 1 - \frac{1}{c} \qquad\qquad d = \omega(\log n)$$

and μ will refer to the uniform distribution over V.

A query is generated as follows: we sample a dataset point x uniformly at random and then generate the query y by sampling $y \sim N_\sigma(x)$. From the choice of σ and d, $d(x,y) \leq \frac{d}{2c}(1+o(1))$ with high probability. In addition, for every other point in the dataset $x' \neq x$, the pair (x',y) is distributed as two uniformly random points (even though $y \sim N_\sigma(x)$, because x is randomly distributed). Therefore, by taking a union-bound over all dataset points, we can conclude that with high probability, $d(x',y) \geq \frac{d}{2}(1-o(1))$ for each $x' \neq x$.

Given a query y generated as described above, we know there exists a dataset point x whose distance to the query is $d(x,y) \leq \frac{d}{2c}(1+o(1))$. Every other dataset point lies at a distance $d(x',y) \geq \frac{d}{2}(1-o(1))$. Therefore, the two distances are a factor of $c - o(1)$ away.

The following lemma is the main result of this section, and we will reference this lemma in subsequent sections.

Lemma 6.3.2 (Robust expansion)**.** *Consider the Hamming space equipped with the Hamming norm. For any $p, q \in [1,\infty)$ where $(q-1)(p-1) = \sigma^2$, any $\gamma \in [0,1]$, and $m \geq 1$,*

$$\Phi_r\left(\frac{1}{m}, \gamma\right) \geq \gamma^q m^{1+\frac{q}{p}-q}.$$

The robust expansion comes from a straight forward application from small-set expansion. In fact, one can easily prove tight bounds on robust expansion via the following lemma:

Theorem 6.3.3 (Generalized Small-Set Expansion Theorem, [139])**.** *Let $0 \leq \sigma \leq 1$. Let $A, B \subset \{-1,1\}^n$ have volumes $\exp(-\frac{a^2}{2})$ and $\exp(-\frac{b^2}{2})$ and assume $0 \leq \sigma a \leq b \leq a$. Then*

$$\Pr_{\substack{(x,y) \\ \sigma-correlated}}[x \in A, y \in B] \leq \exp\left(-\frac{1}{2}\frac{a^2 - 2\sigma ab + b^2}{1-\sigma^2}\right).$$

We compute the robust expansion via an application of the Bonami-Beckner Inequality and Hölder's inequality. This computation gives us more flexibility with

respect to parameters which will become useful in subsequent sections. We now recall the necessary tools.

Theorem 6.3.4 (Bonami-Beckner Inequality [139]). *Fix $1 \leq p \leq q$ and $0 \leq \sigma \leq \sqrt{(p-1)/(q-1)}$. Any Boolean function $f : \{-1,1\}^n \to \mathbb{R}$ satisfies*

$$\|T_\sigma f\|_q \leq \|f\|_p.$$

Theorem 6.3.5 (Hölder's Inequality). *Let $f : \{-1,1\}^n \to \mathbb{R}$ and $g : \{-1,1\}^n \to \mathbb{R}$ be arbitrary Boolean functions. Fix $s, t \in [1,\infty)$ where $\frac{1}{s} + \frac{1}{t} = 1$. Then*

$$\langle f, g \rangle \leq \|f\|_s \|g\|_t.$$

We will let f and g be indicator functions for two sets A and B, and use a combination of the Bonami-Beckner Inequality and Hölder's Inequality to lower bound the robust expansion. The operator T_σ applied to f will measure the neighborhood of set A. We compute an upper bound on the correlation of the neighborhood of A and B (referred to as γ) with respect to the volumes of A and B, and the expression will give a lower bound on robust expansion.

We also need the following lemma.

Lemma 6.3.6. *Let $p, q \in [1,\infty)$, where $(p-1)(q-1) = \sigma^2$ and $f, g : \{-1,1\}^d \to \mathbb{R}$ be two Boolean functions. Then*

$$\langle T_\sigma f, g \rangle \leq \|f\|_p \|g\|_q.$$

Proof. We first apply Hölder's Inequality to split the inner-product into two parts, apply the Bonami-Beckner Inequality to each part.

$$\langle T_\sigma f, f \rangle = \langle T_{\sqrt{\sigma}} f, T_{\sqrt{\sigma}} g \rangle \leq \|T_{\sqrt{\sigma}} f\|_s \|T_{\sqrt{\sigma}} g\|_t.$$

We pick the parameters $s = \dfrac{p-1}{\sigma} + 1$ and $t = \dfrac{s}{s-1}$, so $\frac{1}{s} + \frac{1}{t} = 1$. Note that $p \leq s$

because $\sigma < 1$ and $p \geq 1$ because $(p-1)(q-1) = \sigma^2 \leq \sigma$. We have

$$q \leq \frac{\sigma}{p-1} + 1 = t.$$

In addition,

$$\sqrt{\frac{p-1}{s-1}} = \sqrt{\sigma} \qquad \sqrt{\frac{q-1}{t-1}} = \sqrt{(q-1)(s-1)} = \sqrt{\frac{(q-1)(p-1)}{\sigma}} = \sqrt{\sigma}.$$

We finally apply the Bonami-Beckner Inequality to both norms to obtain

$$\|T_{\sqrt{\sigma}} f\|_s \|T_{\sqrt{\sigma}} g\|_t \leq \|f\|_p \|g\|_q.$$

□

We are now ready to prove Lemma 6.3.2.

Proof of Lemma 6.3.2. We use Lemma 6.3.6 and the definition of robust expansion. For any two sets $A, B \subset V$, let $a = \frac{1}{2^d}|A|$ and $b = \frac{1}{2^d}|B|$ be the measure of set A and B with respect to the uniform distribution. We refer to $\chi_A : \{-1,1\}^d \to \{0,1\}$ and $\chi_B : \{-1,1\}^d \to \{0,1\}$ as the indicator functions for A and B. Then,

$$\gamma = \Pr_{x \sim \mu, y \sim N_\sigma(x)}[x \in B \mid y \in A] = \frac{1}{a}\langle T_\sigma \chi_A, \chi_B \rangle \leq a^{\frac{1}{p}-1} b^{\frac{1}{q}}. \tag{6.3}$$

Therefore, $\gamma^q a^{q-\frac{q}{p}} \leq b$. Let A and B be the minimizers of $\frac{b}{a}$ satisfying (6.3) and $a \leq \frac{1}{m}$.

$$\Phi_r\left(\frac{1}{m}, \gamma\right) = \frac{b}{a} \geq \gamma^q a^{q-\frac{q}{p}-1} \geq \gamma^q m^{1+\frac{q}{p}-q}.$$

□

6.3.2 One-probe data structures

In this section, we prove Theorem 6.1.1. Our proof relies on the main result of [146] for the GNS problem:

Theorem 6.3.7 (Theorem 1.5 [146]). *There exists an absolute constant γ such that the following holds. Any randomized cell-probe data structure making t probes and using m cells of w bits for a weakly independent instance of GNS which is correct with probability greater than $\frac{1}{2}$ must satisfy*

$$\frac{m^t w}{n} \geq \Phi_r\left(\frac{1}{m^t}, \frac{\gamma}{t}\right).$$

Proof of Theorem 6.1.1. The lower bound follows from applying Lemma 6.3.2 directly to Theorem 6.3.7. Setting $t = 1$ in Theorem 6.3.7, we obtain

$$mw \geq n \cdot \Phi_r\left(\frac{1}{m}, \gamma\right) \geq n\gamma^q m^{1+\frac{q}{p}-q}$$

for some $p, q \in [1, \infty)$ and $(p-1)(q-1) = \sigma^2$. Rearranging the inequality and letting $p = 1 + \frac{\log\log n}{\log n}$, and $q = 1 + \sigma^2 \frac{\log n}{\log\log n}$, we obtain

$$m \geq \frac{\gamma^{\frac{p}{p-1}} n^{\frac{p}{pq-q}}}{w^{\frac{p}{pq-q}}} \geq n^{\frac{1}{\sigma^2}-o(1)}.$$

Since $\sigma = 1 - \frac{1}{c}$ and $w = n^{o(1)}$, we obtain the desired result. □

Corollary 6.3.8. *Any 1 cell probe data structure with cell size $n^{o(1)}$ for c-approximate nearest neighbors on the sphere in ℓ_2 needs $n^{1+\frac{2c^2-1}{(c^2-1)^2}-o(1)}$ many cells.*

Proof. Each point in the Hamming space $\{-1, 1\}^d$ (after scaling by $\frac{1}{\sqrt{d}}$) can be thought of as lying on the unit sphere. If two points are a distance r apart in the Hamming space, then they are $2\sqrt{r}$ apart on the sphere with ℓ_2 norm. Therefore a data structure for a c^2-approximation on the sphere gives a data structure for a c-approximation in the Hamming space. □

6.4 List-of-points data structures

In this section, we prove Theorem 6.1.4, i.e., a tight lower bound against data structure that fall inside the "list-of-points" model, as defined in Def. 6.1.3.

Theorem 6.4.1 (Restatement of Theorem 6.1.4)**.** *Let D be a list-of-points data structure which solves (c, r)-ANN for n points in the d-dimensional Hamming space with $d = \omega(\log n)$. Suppose D is specified by a sequence of m sets $\{A_i\}_{i=1}^m$ and a procedure for outputting a subset $I(q) \subset [m]$ using expected space $s = n^{1+\rho_u}$, and expected query time $n^{\rho_q - o(1)}$ with success probability $\frac{2}{3}$. Then*

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

We will prove the lower bound by giving a lower bound on list-of-points data structures which solve the random instance for the Hamming space defined in Section 6.2.1. The dataset consists of n points $\{u_i\}_{i=1}^n$ where each $u_i \sim V$ drawn uniformly at random, and a query v is drawn from the neighborhood of a random dataset point. Thus, we may assume D is a deterministic data structure.

Fix a data structure D, where $A_i \subset V$ specifies which dataset points are placed in L_i. Additionally, we may define $B_i \subset V$ which specifies which query points scan L_i, i.e., $B_i = \{v \in V \mid i \in I(v)\}$. Suppose we sample a random dataset point $u \sim V$ and then a random query point v from the neighborhood of u. Let

$$\gamma_i = \Pr[v \in B_i \mid u \in A_i]$$

represent the probability that query v scans the list L_i, conditioned on u being in L_i. Additionally, we write $s_i = \mu(A_i)$ as the normalized size of A. The query time for D is given by the following expression:

$$\begin{aligned} T &= \sum_{i=1}^m \chi_{B_i}(v) \left(1 + \sum_{j=1}^n \chi_{A_i}(u_j)\right) \\ \mathbb{E}[T] &= \sum_{i=1}^m \mu(B_i) + \sum_{i=1}^m \gamma_i \mu(A_i) + (n-1)\sum_{i=1}^m \mu(B_i)\mu(A_i) \\ &\geq \sum_{i=1}^m \Phi_r(s_i, \gamma_i) s_i + \sum_{i=1}^m s_i \gamma_i + (n-1)\sum_{i=1}^m \Phi_r(s_i, \gamma_i) s_i^2. \end{aligned} \tag{6.4}$$

Since the data structure succeeds with probability γ,

$$\sum_{i=1}^{m} s_i \gamma_i \geq \gamma = \Pr_{j \sim [n], v \sim N(u_j)} [\exists i \in [m] : v \in B_i, u_j \in A_i]. \tag{6.5}$$

Additionally, since D uses at most s space,

$$n \sum_{i=1}^{m} s_i \leq O(s). \tag{6.6}$$

Using the two constraints in (6.5) and (6.6), we will use the estimates of robust expansion in order to find a lower bound for (6.4). From Lemma 6.3.2, for any $p, q \in [1, \infty)$ where $(p-1)(q-1) = \sigma^2$ where $\sigma = 1 - \frac{1}{c}$,

$$\begin{aligned} \mathbb{E}[T] &\geq \sum_{i=1}^{m} s_i^{q - \frac{q}{p}} \gamma_i^q + (n-1) \sum_{i=1}^{m} s_i^{q - \frac{q}{p} + 1} \gamma_i^q + \gamma \\ \gamma &\leq \sum_{i=1}^{m} s_i \gamma_i \\ O\left(\frac{s}{n}\right) &\geq \sum_{i=1}^{m} s_i. \end{aligned}$$

We set $S = \{i \in [m] : s_i \neq 0\}$ and for $i \in S$, we write $v_i = s_i \gamma_i$. Then

$$\mathbb{E}[T] \geq \sum_{i \in S} v_i^q \left(s_i^{-\frac{q}{p}} + (n-1) s_i^{-\frac{q}{p}+1} \right) \geq \sum_{i \in S} \left(\frac{\gamma}{|S|} \right)^q \left(s_i^{-\frac{q}{p}} + (n-1) s_i^{-\frac{q}{p}+1} \right) \tag{6.7}$$

where we used the fact $q \geq 1$. Consider

$$F = \sum_{i \in S} \left(s_i^{-\frac{q}{p}} + (n-1) s_i^{-\frac{q}{p}+1} \right). \tag{6.8}$$

We analyze three cases separately:

- $0 < \rho_u \leq \frac{1}{2c-1}$
- $\frac{1}{2c-1} < \rho_u \leq \dfrac{2c-1}{(c-1)^2}$
- $\rho_u = 0$.

For the first two cases, we let

$$q = 1 - \sigma^2 + \sigma\beta \qquad p = \frac{\beta}{\beta - \sigma} \qquad \beta = \sqrt{\frac{1-\sigma^2}{\rho_u}} \tag{6.9}$$

Since $0 < \rho_u \leq \dfrac{2c-1}{(c-1)^2}$, one can verify $\beta > \sigma$ and both p and q are at least 1.

Lemma 6.4.2. *When $\rho_u \leq \frac{1}{2c-1}$, and $s = n^{1+\rho_u}$,*

$$\mathbb{E}[T] \geq \Omega(n^{\rho_q})$$

where ρ_q and ρ_u satisfy Equation 6.1.

Proof. In this setting, p and q are constants, and $q \geq p$. Therefore, $\frac{q}{p} \geq 1$. F can be viewed as consisting of the contributions of each s_i's in Equation 6.8, constrained by (6.6). One can easily verify that F minimized when $s_i = O(\frac{s}{n|S|})$, so substituting in (6.7),

$$\mathbb{E}[T] \geq \Omega\left(\frac{\gamma^q s^{-q/p+1} n^{q/p}}{|S|^{q-q/p}}\right) \geq \Omega(\gamma^q s^{1-q} n^{q/p})$$

since $q - q/p > 0$ and $|S| \leq s$. In addition, p, q and γ are constants, and note the fact $s = n^{1+\rho_u}$, and (6.9), we let n^{ρ_q} be the best query time we can achieve. Combining these facts, along with the lower bound for ρ_q in (6.4), we obtain the following relationship between ρ_q and ρ_u:

$$\begin{aligned}
\rho_q &= (1+\rho_u)(1-q) + \frac{q}{p} \\
&= (1+\rho_u)(\sigma^2 - \sigma\beta) + \frac{(1-\sigma^2+\sigma\beta)(\beta-\sigma)}{\beta} \\
&= \left(\sqrt{1-\sigma^2} - \sqrt{\rho_u}\sigma\right)^2 \\
&= \left(\frac{\sqrt{2c-1}}{c} - \sqrt{\rho_u} \cdot \frac{(c-1)}{c}\right)^2.
\end{aligned}$$

□

Lemma 6.4.3. *When* $\rho_u > \frac{1}{2c-1}$,

$$\mathbb{E}[T] \geq \Omega(n^{\rho_q})$$

where ρ_q *and* ρ_u *satisfy Equation 6.1.*

Proof. We follow a similar pattern to Lemma 6.4.2.

$$\frac{\partial F}{\partial s_i} = \left(-\frac{q}{p}\right) s_i^{-\frac{q}{p}-1} + \left(-\frac{q}{p}+1\right)(n-1)s_i^{-\frac{q}{p}}.$$

Consider the case when each $\frac{\partial F}{\partial s_i}(s_i) = 0$, by setting $s_i = \dfrac{q}{(p-q)(n-1)}$. Since $q < p$, this value is positive and $\sum_{i \in S} s_i \leq O\left(\frac{m}{n}\right)$ for large enough n. Thus, F is minimized at this point, and $\mathbb{E}[T] \geq \left(\frac{\gamma}{|S|}\right)^q |S| \left(\frac{q}{(p-q)(n-1)}\right)^{-\frac{q}{p}}$. Since $q \geq 1$ and $|S| \leq s$,

$$\mathbb{E}[T] \geq \left(\frac{\gamma}{s}\right)^q s \left(\frac{q}{(p-q)(n-1)}\right)^{-\frac{q}{p}}.$$

Since p, q and γ are constants, $\mathbb{E}[T] \geq \Omega(n^{\rho_q})$,

$$\rho_q = (1+\rho_u)(1-q) + \frac{q}{p}$$

which is the same expression for ρ_q as in Lemma 6.4.2. □

Lemma 6.4.4. *When* $\rho_u = 0$ *(so* $s = O(n)$*),*

$$\mathbb{E}[T] \geq n^{\rho_q - o(1)}$$

where $\rho_q = \dfrac{2c-1}{c^2} = 1 - \sigma^2$.

Proof. In this case, we let

$$q = 1 + \sigma^2 \cdot \frac{\log n}{\log\log n} \qquad p = 1 + \frac{\log\log n}{\log n}.$$

Since $q > p$, we have

$$\mathbb{E}[T] = \Omega(\gamma^q s^{1-q} n^{\frac{q}{p}}) = n^{1-\sigma^2-o(1)},$$

which is the desired expression. □

6.5 Two-probe data structures

In this section, we prove the cell probe lower bound for $t = 2$ cell probes stated in Theorem 6.1.2.

We follow the framework in [146] and prove lower bounds for GNS when $U = V$ with measure μ (see Def. 6.2.2). We assume there is an underlying graph G with vertex set V. For any point $p \in V$, we write p's *neighborhood*, $N(p)$, as the set of points with an edge incident on p in G.

In the 2-probe GNS problem, we are given a dataset $P = \{p_i\}_{i=1}^n \subset V$ of n points as well as a bit-string $x \in \{0,1\}^n$. The goal is to build a data structure supporting the following types of queries: given a point $q \in V$, if there exists a unique neighbor $p_i \in N(q) \cap P$, return x_i with probability at least $\frac{2}{3}$ after making two cell-probes.

We let D denote a data structure with m cells of w bits each. D will depend on the dataset P as well as the bit-string x. We will prove the following theorem.

Theorem 6.5.1. *There exists a constant $\gamma > 0$ such that any non-adaptive GNS data structure holding a dataset of $n \geq 1$ points which succeeds with probability $\frac{2}{3}$ using two cell probes and m cells of w bits satisfies*

$$\frac{m \log m \cdot 2^{O(w)}}{n} \geq \Omega\left(\Phi_r\left(\frac{1}{m}, \gamma\right)\right).$$

Theorem 6.1.2 will follow from Theorem 6.5.1 together with the robust expansion bound from Lemma 6.3.2 for the special case of *non-adaptive* probes. We will later show how to reduce adaptive algorithms losing a sub-polynomial factor in the space for $w = o(\log n)$ in Section 6.5.6. We now proceed to proving Theorem 6.5.1.

At a high-level, we show that a "too-good-to-be-true", 2-probe data structure implies a weaker notion of 2-query locally-decodable code (LDC) with small noise rate using the same amount of space[2]. Even though our notion of LDC is weaker

[2]A 2-query LDC corresponds to LDCs which make two probes to their memory contents. Even

than Def. 6.2.4, we adapt the tools for showing 2-query LDC lower bounds from [101]. These arguments, using quantum information theory, are very robust and work well with the weaker 2-query LDC we construct.

We note that [145] was the first to suggest the connection between ANN and LDCs. This work represents the first concrete connection which gives rise to better lower bounds.

Proof structure. The proof of Theorem 6.5.1 proceeds in six steps.

1. First we use Yao's principle to focus on deterministic non-adaptive data structures for GNS with two cell-probes. We provide distributions over n-point datasets P, as well as bit-strings x and a query q, and assume the existence of a deterministic data structure succeeding with probability at least $\frac{2}{3}$.

2. We simplify the deterministic data structure in order to get "low-contention" data structures. These are data structures which do not rely on any single cell too much (similar to Def. 6.1 in [146]).

3. We use ideas from [146] to understand how queries neighboring particular dataset points probe various cells of the data structure. We fix an n-point dataset P with a constant fraction of the points satisfying the following condition: many possible queries in the neighborhood of these points probe disjoint pairs of cells.

4. For the fixed dataset P, we show that we can recover a constant fraction of bits of x with significant probability even if we corrupt the contents of some cells.

5. We reduce to data structures with 1-bit words in order to apply the LDC arguments from [101].

6. Finally, we design an LDC with weaker guarantees and use the arguments in [101] to prove lower bounds on the space of the weak LDC.

though there is a slight ambiguity with the data structure notion of query, we say "2-query LDCs" in order to be consistent with the LDC literature.

6.5.1 Deterministic data structures

Definition 6.5.2. A non-adaptive randomized algorithm R for the GNS problem making two cell-probes is an algorithm specified by the following two components:

1. A procedure which preprocess a dataset $P = \{p_i\}_{i=1}^n$ of n points, as well as a bit-string $x \in \{0,1\}^n$ in order to output a data structure $D \in (\{0,1\}^w)^m$.

2. An algorithm R that given a query q, chooses two indices $(i,j) \in [m]^2$ and specifies a function $f_q \colon \{0,1\}^w \times \{0,1\}^w \to \{0,1\}$.

We require the data structure D and the algorithm R satisfy

$$\Pr_{R,D}[f_q(D_j, D_k) = x_i] \geq \frac{2}{3}$$

whenever $q \in N(p_i)$ and p_i is the unique such neighbor.

Note that the procedure which outputs the data structure does not depend on the query q, and that the algorithm R does not depend on the dataset P or bit-string x.

Definition 6.5.3. We define the following distributions:

- Let $\mathcal{P}$ be the uniform distribution supported on n-point datasets from V.
- Let $\mathcal{X}$ be the uniform distribution over $\{0,1\}^n$.
- Let $\mathcal{Q}(P)$ be the distribution over queries given by first drawing a dataset point $p \in P$ uniformly at random and then drawing $q \in N(p)$ uniformly at random.

Lemma 6.5.4. *Assume R is a non-adaptive randomized algorithm for GNS using two cell-probes. Then, there exists a non-adaptive deterministic algorithm A for GNS using two cell-probes succeeding with probability at least $\frac{2}{3}$ when the dataset $P \sim \mathcal{P}$, the bit-string $x \sim \mathcal{X}$, and $q \sim \mathcal{Q}(P)$.*

Proof. We apply Yao's principle to the success probability of the algorithm. By assumption, there exists a distribution over algorithms which can achieve probability of success at least $\frac{2}{3}$ for any single query. Therefore, for the fixed distributions $\mathcal{P}, \mathcal{X}$,

and $\mathcal{Q}$, there exists a deterministic algorithm achieving at least the same success probability. $\square$

In order to simplify notation, we let $A^D(q)$ denote output of the algorithm A. We assume that $A(q)$ outputs a pair of indices (j,k) as well as the function $f_q\colon \{0,1\}^w \times \{0,1\}^w \to \{0,1\}$, and thus, we use $A^D(q)$ as the output of $f_q(D_j, D_k)$. For any fixed dataset $P = \{p_i\}_{i=1}^n$ and bit-string $x \in \{0,1\}^n$,

$$\Pr_{q\sim N(p_i)}[A^D(q) = x_i] = \Pr_{q\sim N(p_i)}[f_q(D_j, D_k) = x_i].$$

This notation allows us to succinctly state the probability of correctness when the query is a neighbor of p_i.

For the remainder of the section, we let A denote a non-adaptive deterministic algorithm succeeding with probability at least $\frac{2}{3}$ using m cells of width w. The success probability is taken over the random choice of the dataset $P \sim \mathcal{P}$, $x \sim \mathcal{X}$ and $q \sim \mathcal{Q}(P)$.

6.5.2 Making low-contention data structures

For any $t \in \{1,2\}$ and $j \in [m]$, let $A_{t,j}$ be the set of queries which probe cell j at the t-th probe of algorithm A. Since A is deterministic, the indices $(i,j) \in [m]^2$ which A outputs are completely determined by two collections $\mathcal{A}_1 = \{A_{1,j}\}_{j\in[m]}$ and $\mathcal{A}_2 = \{A_{2,j}\}_{j\in[m]}$ which independently partition the query space V. On query q, if $q \in A_{1,i}$ and $q \in A_{2,j}$, algorithm A outputs the indices (i,j).

We now define the notion of low-contention data structures, which requires the data structure to not rely on any one particular cell too much by ensuring no $A_{t,j}$ is too large.

Definition 6.5.5. A deterministic non-adaptive algorithm A using m cells has *low contention* if every set $\mu(A_{t,j}) \le \frac{1}{m}$ for $t \in \{1,2\}$ and $j \in [m]$.

We now use the following lemma to argue that up to a small increase in space, a data structure can be made low-contention.

Lemma 6.5.6. *Let A be a deterministic non-adaptive algorithm for GNS making two cell-probes using m cells. There exists a deterministic non-adaptive algorithm A' for GNS making two cell-probes using $3m$ cells which has low contention and succeeds with the same probability.*

Proof. Suppose $\mu(A_{t,j}) \geq \frac{1}{m}$ for some $j \in [m]$. We partition $A_{t,j}$ into enough sets $\{A_{t,k}^{(j)}\}_k$ of measure $\frac{1}{m}$ and at most one set with measure between 0 and $\frac{1}{m}$. For each of set $A_{t,k}^{(j)}$, we make a new cell j_k with the same contents as cell j. When a query lies inside $A_{t,k}^{(j)}$ the t-th probe is made to the new cell j_k instead of cell j.

We apply the above transformation on all sets with $\mu(A_{t,j}) \geq \frac{1}{m}$. In the resulting data structure, in each partition $\mathcal{A}_1$ and $\mathcal{A}_2$, there can be at most m cells of measure $\frac{1}{m}$ and at most m sets with measure less than $\frac{1}{m}$. Therefore, the transformed data structure has at most $3m$ cells. Since the contents remain the same, the data structure succeeds with the same probability. □

Given Lemma 6.5.6, we assume that A is a deterministic non-adaptive algorithm for GNS with two cell-probes using m cells which has low contention. The extra factor of 3 in the number of cells is absorbed in the asymptotic notation.

6.5.3 Datasets which shatter

We fix some $\gamma > 0$ to be a sufficiently small constant.

Definition 6.5.7 (Weak-shattering [146])**.** We say a partition $A_1, \ldots, A_m$ of V (K, γ)-*weakly shatters a point p* if

$$\sum_{i \in [m]} \left(\mu(A_i \cap N(p)) - \frac{1}{K} \right)^+ \leq \gamma,$$

where the operator $(\cdot)^+ \colon \mathbb{R} \to \mathbb{R}^+$ is the identity on positive real numbers and zero otherwise.

Lemma 6.5.8 (Shattering [146])**.** *Let $A_1, \ldots, A_k$ be a collection of disjoint subsets of*

measure at most $\frac{1}{m}$. *Then*

$$\Pr_{p\sim\mu}[p \text{ is } (K,\gamma)\text{-weakly shattered}] \geq 1-\gamma$$

for $K = \Phi_r\left(\frac{1}{m}, \frac{\gamma^2}{4}\right) \cdot \frac{\gamma^3}{16}$.

For the remainder of the section, we let

$$K = \Phi_r\left(\frac{1}{m}, \frac{\gamma^2}{4}\right) \cdot \frac{\gamma^3}{16}.$$

We are interested in dataset points which are shattered with respect to the collections $\mathcal{A}_1$ and $\mathcal{A}_2$. Intuitively, queries which are near-neighbors of these dataset points probe various disjoint cells in the data structure, so their corresponding bit is stored in many cells.

Definition 6.5.9. Let $p \in V$ be a dataset point which is (K,γ)-weakly shattered by $\mathcal{A}_1$ and $\mathcal{A}_2$. Let $\beta_1, \beta_2 \subset N(p)$ be arbitrary subsets where each $j \in [m]$ satisfies

$$\mu(A_{1,j} \cap N(p) \setminus \beta_1) \leq \frac{1}{K}$$

and

$$\mu(A_{2,j} \cap N(p) \setminus \beta_2) \leq \frac{1}{K}$$

Since p is (K,γ)-weakly shattered, we can pick β_1 and β_2 with measure at most γ each. We will refer to $\beta(p) = \beta_1 \cup \beta_2$.

For a fixed dataset point $p \in P$, we refer to $\beta(p)$ as the set holding the *slack* in the shattering of measure at most 2γ. For a given collection $\mathcal{A}$, let $S(\mathcal{A}, p)$ be the event that the collection $\mathcal{A}$ (K,γ)-weakly shatters p. Note that Lemma 6.5.8 implies that $\Pr_{p\sim\mu}[S(\mathcal{A}, p)] \geq 1-\gamma$.

Lemma 6.5.10. *With high probability over the choice of* n*-point dataset, at most* $4\gamma n$ *points do not satisfy* $S(\mathcal{A}_1, p)$ *and* $S(\mathcal{A}_2, p)$.

Proof. The expected number of points p which do not satisfy $S(\mathcal{A}_1, p)$ and $S(\mathcal{A}_2, p)$ is at most $2\gamma n$. Therefore via a Chernoff bound, the probability that more than $4\gamma n$ points do not satisfy $S(\mathcal{A}_1, p)$ and $S(\mathcal{A}_2, p)$ is at most $\exp\left(-\frac{2\gamma n}{3}\right)$. □

We call a dataset *good* if there are at most $4\gamma n$ dataset points which are not (K, γ)-weakly shattered by $\mathcal{A}_1$ and $\mathcal{A}_2$.

Lemma 6.5.11. *There exists a good dataset $P = \{p_i\}_{i=1}^n$ where*

$$\Pr_{x\sim\mathcal{X}, q\sim\mathcal{Q}(P)}[A^D(q) = x_i] \geq \frac{2}{3} - o(1)$$

Proof. For any fixed dataset $P = \{p_i\}_{i=1}^n$, let

$$\mathbf{P} = \Pr_{x\sim\mathcal{X}, q\sim Q(p)}[A^D(q) = x_i].$$

Then,

$$\begin{aligned}
\frac{2}{3} &\leq \mathop{\mathbb{E}}_{P\sim\mathcal{P}}[\mathbf{P}] \\
&= (1 - o(1)) \cdot \mathop{\mathbb{E}}_{P\sim\mathcal{P}}[\mathbf{P} \mid \ P \text{ is good}] + o(1) \cdot \mathop{\mathbb{E}}_{P\sim\mathcal{P}}[\mathbf{P} \mid \ P \text{ is not good}] \\
\frac{2}{3} - o(1) &\leq (1 - o(1)) \cdot \mathop{\mathbb{E}}_{P\sim\mathcal{P}}[\mathbf{P} \mid P \text{ is good}].
\end{aligned}$$

Therefore, there exists a dataset which is not shattered by at most $4\gamma n$ and

$$\Pr_{x\sim\mathcal{X}, q\sim\mathcal{Q}(P)}[A^D(y) = x_i] \geq \frac{2}{3} - o(1).$$

□

6.5.4 Corrupting some cell contents of shattered points

In the rest of the proof, we fix the dataset $P = \{p_i\}_{i=1}^n$ satisfying the conditions of Lemma 6.5.11, i.e., P satisfies

$$\Pr_{x\sim\mathcal{X}, q\sim\mathcal{Q}(P)}[A^D(q) = x_i] \geq \frac{2}{3} - o(1). \tag{6.10}$$

We now introduce the notion of *corruption* of the data structure cells D, which parallels the notion of noise in locally-decodable codes. Remember that, after fixing some bit-string x, the algorithm A produces some data structure $D \in (\{0,1\}^w)^m$.

Definition 6.5.12. We call $D' \in (\{0,1\}^w)^m$ a *corrupted* version of D at k cells if D and D' differ on at most k cells, i.e., if $|\{i \in [m] : D_i \neq D'_i\}| \leq k$.

Definition 6.5.13. For a fixed $x \in \{0,1\}^n$, let

$$c_x(i) = \Pr_{q\sim N(p_i)}[A^D(q) = x_i] \tag{6.11}$$

denote the *recovery probability of bit i*. Note that from the definitions of $\mathcal{Q}(P)$, $\mathbb{E}[c_x(i)] \geq \frac{2}{3} - o(1)$, where the expectation is taken over $x \sim \mathcal{X}$ and $i \in [n]$.

In this section, we show there exist a subset $S \subset [n]$ of size $\Omega(n)$ where each $i \in S$ has constant recovery probability averaged over $x \sim \mathcal{X}$, even if the algorithm probes a corrupted version of data structure. We let $\varepsilon > 0$ be a sufficiently small constant.

Lemma 6.5.14. *Fix a vector $x \in \{0,1\}^n$, and let $D \in (\{0,1\}^w)^m$ be the data structure that algorithm A produces on dataset P and bit-string x. Let D' be a corruption of D at εK cells. For every $i \in [n]$ where events $S(\mathcal{A}_1, p_i)$ and $S(\mathcal{A}_2, p_i)$ occur,*

$$\Pr_{q\sim N(p_i)}[A^{D'}(q) = x_i] \geq c_x(i) - 2\gamma - 2\varepsilon.$$

Proof. Note that $c_x(i)$ represents the probability that algorithm A run on a uniformly chosen query from the neighborhood of p_i returns the correct answer, i.e. x_i. We denote the subset $C_1 \subset N(p)$ of queries that when run on A return x_i; so, $\mu(C_1) = c_x(i)$ by definition.

By assumption, p_i is (K, γ)-weakly shattered by $\mathcal{A}_1$ and $\mathcal{A}_2$, so by Def. 6.5.9, we specify some $\beta(p) \subset N(p)$ where $\mu(C_1 \cap \beta(p)) \leq \mu(\beta(p)) \leq 2\gamma$. Let $C_2 = C_1 \setminus \beta(p)$, where $\mu(C_2) \geq c_i(x) - 2\gamma$. Again, by assumption that p_i is (K, γ)-weakly shattered, each $j \in [m]$ and $t \in \{1, 2\}$ satisfy $\mu(C_2 \cap A_{t,j}) \leq \frac{1}{K}$. Let $\Delta \subset [m]$ be the set of εK cells where D and D' differ, and let $C_3 \subset C_2$ be given by

$$C_3 = C_2 \setminus \left(\bigcup_{j \in \Delta} (A_{1,j} \cup A_{2,j}) \right).$$

Thus,

$$\mu(C_3) \geq \mu(C_2) - \sum_{j \in \Delta} (\mu(C_2 \cap A_{1,j}) + \mu(C_2 \cap A_{2,j})) \geq c_i(x) - 2\gamma - 2\varepsilon.$$

If $q \in C_3$, then on query q, algorithm A probes cells outside of Δ, so $A^{D'}(q) = A^D(q) = x_i$. □

Lemma 6.5.15. *There exists a set $S \subset [n]$ of size $\Omega(n)$ with the following property. If $i \in S$, then events $S(\mathcal{A}_1, p_i)$ and $S(\mathcal{A}_2, p_i)$ occur, and*

$$\mathop{\mathbb{E}}_{x \sim \mathcal{X}}[c_x(i)] \geq \frac{1}{2} + \nu,$$

where ν is a constant. [3]

Proof. For $i \in [n]$, let E_i be the event that $S(\mathcal{A}_1, p_i)$ and $S(\mathcal{A}_2, p_i)$ occur and $\mathbb{E}_{x \sim \mathcal{X}}[c_x(i)] \geq \frac{1}{2} + \nu$. Additionally, let

$$\mathbf{P} = \Pr_{i \in [n]}[E_i].$$

[3] One can think of ν as around $\frac{1}{10}$.

We set $S = \{i \in [n] \mid E_i\}$, so it remains to show that $\mathbf{P} = \Omega(1)$. To this end,

$$\begin{aligned}
\frac{2}{3} - o(1) &\leq \mathop{\mathbb{E}}_{x\sim\mathcal{X}, i\in[n]}[c_x(i)] && \text{(by Equations 6.10 and 6.11)} \\
&\leq 4\gamma + \mathbf{P} + \left(\frac{1}{2} + \nu\right) \cdot (1 - \mathbf{P}) && \text{(since } P \text{ is good)} \\
\frac{1}{6} - o(1) - 4\gamma - \nu &\leq \mathbf{P} \cdot \left(\frac{1}{2} - \nu\right).
\end{aligned}$$

□

Fix the set $S \subset [n]$ satisfying the conditions of Lemma 6.5.15. We combine Lemma 6.5.14 and Lemma 6.5.15 to obtain the following condition on the dataset.

Lemma 6.5.16. *Whenever $i \in S$,*

$$\mathop{\mathbb{E}}_{x\sim\mathcal{X}}\left[\Pr_{q\sim N(p_i)}[A^{D'}(q) = x_i]\right] \geq \frac{1}{2} + \eta$$

where $\eta = \nu - 2\gamma - 2\varepsilon$ and D' differs from D in εK cells.

Proof. Whenever $i \in S$, p_i is (K, γ)-weakly shattered. By Lemma 6.5.15, A outputs x_i with probability $\frac{1}{2} + \nu$ on average when probing the data structure D on input $q \sim N(p_i)$, i.e

$$\mathop{\mathbb{E}}_{x\sim\mathcal{X}}\left[\Pr_{q\sim N(p_i)}[A^{D}(q) = x_i]\right] \geq \frac{1}{2} + \nu.$$

Therefore, from Lemma 6.5.14, if A probes D' which is a corruption of D in any εK cells, A will recover x_i with probability at least $\frac{1}{2} + \nu - 2\gamma - 2\varepsilon$ averaged over all $x \sim \mathcal{X}$ where $q \sim N(p_i)$. In other words,

$$\mathop{\mathbb{E}}_{x\sim\mathcal{X}}\left[\Pr_{q\sim N(p_i)}[A^{D'}(q) = x_i]\right] \geq \frac{1}{2} + \nu - 2\gamma - 2\varepsilon.$$

□

Summarizing the results of the section, we conclude with the following theorem.

Theorem 6.5.17. *There exists a two-probe algorithm and a subset $S \subseteq [n]$ of size $\Omega(n)$, satisfying the following property. When $i \in S$, we can recover x_i with probability*

at least $\frac{1}{2} + \eta$ *over a random choice of* $x \sim \mathcal{X}$*, even if we probe a corrupted version of the data structure at* εK *cells.*

Proof. We describe how one can recover bit x_i from a data structure generated by algorithm A. In order to recover x_i, we generate a random query $q \sim N(p_i)$ and probe the data structure at the cells specified by A. From Lemma 6.5.16, there exists a set $S \subset [n]$ of size $\Omega(n)$ for which the described algorithm recovers x_i with probability at least $\frac{1}{2} + \eta$, where the probability is taken on average over all possible $x \in \{0,1\}^n$. □

Since we fixed the dataset $P = \{p_i\}_{i=1}^n$ satisfying the conditions of Lemma 6.5.11, we will abuse a bit of notation, and refer to algorithm A as the algorithm which recovers bits of x described in Theorem 6.5.17. We say that $x \in \{0,1\}^n$ is an input to algorithm A in order to initialize the data structure with dataset $P = \{p_i\}_{i=1}^n$ and x_i is the bit associated with p_i.

6.5.5 Decreasing the word size

In order to apply the lower bounds of 2-query locally-decodable codes, we reduce to the case when the word size w is one bit.

Lemma 6.5.18. *There exists a deterministic non-adaptive algorithm* A' *which on input* $x \in \{0,1\}^n$ *builds a data structure* D' *using* $m \cdot 2^w$ *cells of* 1 *bit. For any* $i \in S$ *as well as any corruption* C *which differs from* D' *in at most* εK *cells satisfies*

$$\mathop{\mathbb{E}}_{x \in \{0,1\}^n} \left[\Pr_{q \sim N(p_i)} [A'^C(q) = x_i] \right] \geq \frac{1}{2} + \frac{\eta}{2^{2w}}.$$

Proof. Given algorithm A which constructs the data structure $D \in (\{0,1\}^w)^m$ on input $x \in \{0,1\}^n$, construct the following data structure $D' \in (\{0,1\})^{m \cdot 2^w}$. For each cell $D_j \in \{0,1\}^w$, make 2^w cells containing all parities of the w bits in D_j. This procedure increases the size of the data structure by a factor of 2^w.

Fix $i \in S$ and $q \in N(p_i)$ be a query. If the algorithm A produces a function $f_q : \{0,1\}^w \times \{0,1\}^w \to \{0,1\}$ which succeeds with probability at least $\frac{1}{2} + \zeta$ over $x \in \{0,1\}^n$, then there exists a signed parity on some input bits which equals f_q in at

least $\frac{1}{2} + \frac{\zeta}{2^{2w}}$ inputs $x \in \{0,1\}^n$. Let S_j be the parity of the bits of cell j and S_k be the parity of the bits of cell k. Let $f'_q : \{0,1\} \times \{0,1\} \to \{0,1\}$ denote the parity or the negation of the parity which equals f_q on $\frac{1}{2} + \frac{\zeta}{2^{2w}}$ possible input strings $x \in \{0,1\}^n$.

Algorithm A' will evaluate f'_q at the cell containing the parity of the S_j bits in cell j and the parity of S_k bits in cell k. Let $I_{S_j}, I_{S_k} \in [m \cdot 2^w]$ be the indices of these cells. If C' is a sequence of $m \cdot 2^w$ cells which differ in εK many cells from D', then

$$\mathop{\mathbb{E}}_{x \in \{0,1\}^n} \left[\Pr_{q \sim N(p_i)} [f'_q(C_{I_{S_j}}, C_{I_{S_k}}) = x_i] \right] \geq \frac{1}{2} + \frac{\eta}{2^{2w}}$$

whenever $i \in S$. □

For the remainder of the section, we will prove a version of Theorem 6.5.1 for algorithms with 1-bit words. Given Lemma 6.5.18, we will modify the space to $m \cdot 2^w$ and the probability to $\frac{1}{2} + \frac{\eta}{2^{2w}}$ to obtain the answer. So for the remainder of the section, assume algorithm A has 1 bit words.

6.5.6 Connection to locally-decodable codes

To complete the proof of Theorem 6.5.1, it remains to prove the following lemma.

Lemma 6.5.19. *Let A be a non-adaptive deterministic algorithm which makes 2 cell probes to a data structure D of m cells of 1 bit and recover x_i with probability $\frac{1}{2} + \eta$ on random input $x \in \{0,1\}^n$ even after εK cells are corrupted whenever $i \in S$ for some fixed S of size $\Omega(n)$. Then the following must hold:*

$$\frac{m \log m}{n} \geq \Omega\left(\varepsilon K \eta^2\right).$$

The proof of the lemma uses [101] and relies heavily on notions from quantum computing. In particular, quantum information theory applied to LDC lower bounds.

Crash course in quantum computing

We introduce a few concepts from quantum computing that are necessary in our subsequent arguments. The quantum state of a *qubit* is described by a unit-length

vector in $\mathbb{C}^2$. We write the quantum state as a linear combination of the basis states $\binom{1}{0} = |0\rangle$ and $\binom{0}{1} = |1\rangle$. The quantum state $\alpha = \binom{\alpha_1}{\alpha_2}$ can be written

$$|\alpha\rangle = \alpha_1 |0\rangle + \alpha_2 |1\rangle$$

where we refer to α_1 and α_2 as *amplitudes* and $|\alpha_1|^2 + |\alpha_2|^2 = 1$. The quantum state of an *m-qubit system* is a unit vector in the tensor product $\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$ of dimension 2^m. The basis states correspond to all 2^m bit-strings of length m. For $j \in [2^m]$, we write $|j\rangle$ as the basis state $|j_1\rangle \otimes |j_2\rangle \otimes \cdots \otimes |j_m\rangle$ where $j = j_1 j_2 \dots j_m$ is the binary representation of j. We will write the m-qubit *quantum state* $|\phi\rangle$ as unit-vector given by linear combination over all 2^m basis states. So $|\phi\rangle = \sum_{j \in [2^m]} \phi_j |j\rangle$. As a shorthand, $\langle\phi|$ corresponds to the conjugate transpose of a quantum state.

A *mixed state* $\{p_i, |\phi_i\rangle\}$ is a probability distribution over quantum states. In this case, we the quantum system is in state $|\phi_i\rangle$ with probability p_i. We represent mixed states by a density matrix $\sum p_i |\phi_i\rangle \langle\phi_i|$.

A measurement is given by a family of Hermitian positive semi-definite operators which sum to the identity operator. Given a quantum state $|\phi\rangle$ and a measurement corresponding to the family of operators $\{M_i^* M_i\}_i$, the measurement yields outcome i with probability $\| M_i |\phi\rangle \|^2$ and results in state $\frac{M_i|\phi\rangle}{\|M_i|\phi\rangle\|}$, where the norm $\| \cdot \|$ is the ℓ_2 norm. We say the measurement makes the *observation* M_i.

Finally, a quantum algorithm makes a query to some bit-string $y \in \{0,1\}^m$ by starting with the state $|c\rangle |j\rangle$ and returning $(-1)^{c \cdot y_j} |c\rangle |j\rangle$. One can think of c as the control qubit taking values 0 or 1; if $c = 0$, the state remains unchanged by the query, and if $c = 1$ the state receives a $(-1)^{y_j}$ in its amplitude. The queries may be made in superposition to a state, so the state $\sum_{c \in \{0,1\}, j \in [m]} \alpha_{cj} |c\rangle |j\rangle$ becomes $\sum_{c \in \{0,1\}, j \in [m]} (-1)^{c \cdot y_j} \alpha_{cj} |c\rangle |j\rangle$.

Weak quantum random access codes from GNS algorithms

Definition 6.5.20. $C : \{0,1\}^n \to \{0,1\}^m$ is a $(2, \delta, \eta)$-LDC if there exists a randomized decoding algorithm making at most 2 queries to an m-bit string y non-adaptively,

and for all $x \in \{0,1\}^n$, $i \in [n]$, and $y \in \{0,1\}^m$ where $d(y, C(x)) \leq \delta m$, the algorithm can recover x_i from the two queries to y with probability at least $\frac{1}{2} + \eta$.

In their paper, [101] prove the following result about 2-query LDCs.

Theorem 6.5.21 (Theorem 4 in [101])**.** *If $C : \{0,1\}^n \to \{0,1\}^m$ is a $(2, \delta, \eta)$-LDC, then $m \geq 2^{\Omega(\delta\eta^2 n)}$.*

The proof of Theorem 6.5.21 proceeds as follows. They show how to construct a 1-query quantum-LDC from a classical 2-query LDC. From a 1-query quantum-LDC, [101] constructs a quantum random access code which encodes n-bit strings in $O(\log m)$ qubits. Then they apply a quantum information theory lower bound due to Nayak [135]:

Theorem 6.5.22 (Theorem 2 stated in [101] from Nayak [135])**.** *For any encoding*

$$x \mapsto \rho_x$$

of n-bit strings into m-qubit states, such that a quantum algorithm, given query access to ρ_x, can decode any fixed x_i with probability at least $1/2 + \eta$, it must hold that $m \geq (1 - H(1/2 + \eta))n$.

Our proof will follow a pattern similar to the proof of Theorem 6.5.21. We assume the existence of a GNS algorithm A which builds a data structure $D \in \{0,1\}^m$.

Our algorithm A from Theorem 6.5.17 does not satisfy the strong properties of an LDC, preventing us from applying 6.5.21 directly. However, it does have some LDC-*ish* guarantees. In particular, we can recover bits in the presence of εK corruptions to D. In the LDC language, this means that we can tolerate a noise rate of $\delta = \frac{\varepsilon K}{m}$. Additionally, we cannot necessarily recover *every* coordinate x_i, but we can recover x_i for $i \in S$, where $|S| = \Omega(n)$. Also, our success probability is $\frac{1}{2} + \eta$ over the random choice of $i \in S$ and the random choice of the bit-string $x \in \{0,1\}^n$. Our proof follows by adapting the arguments of [101] to this weaker setting.

Lemma 6.5.23. *Let $r = \frac{2}{\delta a^2}$ where $\delta = \dfrac{\varepsilon K}{m}$ and $a \leq 1$ is a constant. Let D be the data structure from above (i.e., satisfying the hypothesis of Lemma 6.5.19). Then there exists a quantum algorithm that, starting from the $r(\log m + 1)$-qubit state with r copies of $|U(x)\rangle$, where*

$$|U(x)\rangle = \frac{1}{\sqrt{2m}} \sum_{c\in\{0,1\}, j\in[m]} (-1)^{c\cdot D_j} |c\rangle |j\rangle$$

can recover x_i for any $i \in S$ with probability $\frac{1}{2} + \Omega(\eta)$ (over a random choice of x).

Assuming Lemma 6.5.23, we can complete the proof of Lemma 6.5.19.

Proof of Lemma 6.5.19. The proof is similar to the proof of Theorem 2 of [101]. Let ρ_x represent the s-qubit system consisting of the r copies of the state $|U(x)\rangle$, where $s = r(\log m + 1)$; ρ_x is an encoding of x. Using Lemma 6.5.23, we can assume we have a quantum algorithm that, given ρ_x, can recover x_i for any $i \in S$ with probability $\alpha = \frac{1}{2} + \Omega(\eta)$ over the random choice of $x \in \{0,1\}^n$.

We will let $H(A)$ be the Von Neumann entropy of A, and $H(A|B)$ be the conditional entropy and $H(A : B)$ the mutual information.

Let XM be the $(n + s)$-qubit system

$$\frac{1}{2^n} \sum_{x\in\{0,1\}^n} |x\rangle \langle x| \otimes \rho_x.$$

The system corresponds to the uniform superposition of all 2^n strings concatenated with their encoding ρ_x. Let X be the first subsystem corresponding to the first n qubits and M be the second subsystem corresponding to the s qubits. We have

$$H(XM) = n + \frac{1}{2^n} \sum_{x\in\{0,1\}^n} H(\rho_x) \geq n = H(X)$$

$$H(M) \leq s,$$

since M has s qubits. Therefore, the mutual information $H(X : M) = H(X) + H(M) - H(XM) \leq s$. Note that $H(X|M) \leq \sum_{i=1}^n H(X_i|M)$. By Fano's inequality,

if $i \in S$,

$$H(X_i|M) \leq H(\alpha)$$

where we are using the fact that Fano's inequality works even if we can recover x_i with probability α averaged over all x's. Additionally, if $i \notin S$, $H(X_i|M) \leq 1$. Therefore,

$$\begin{aligned} s \geq H(X:M) &= H(X) - H(X|M) \\ &\geq H(X) - \sum_{i=1}^{n} H(X_i|M) \\ &\geq n - |S|H(\alpha) - (n - |S|) \\ &= |S|(1 - H(\alpha)). \end{aligned}$$

Furthermore, $1 - H(\alpha) \geq \Omega(\eta^2)$ since, and $|S| = \Omega(n)$, we have

$$\begin{aligned} \frac{2m}{a^2 \varepsilon K}(\log m + 1) &\geq \Omega\left(n\eta^2\right) \\ \frac{m \log m}{n} &\geq \Omega\left(\varepsilon K \eta^2\right). \end{aligned}$$

□

It remains to prove Lemma 6.5.23, which we proceed to do in the rest of the section. We first show that we can simulate our GNS algorithm with a 1-query quantum algorithm.

Lemma 6.5.24. *Fix an $x \in \{0,1\}^n$ and $i \in [n]$. Let $D \in \{0,1\}^m$ be the data structure produced by algorithm A on input x. Suppose $\Pr_{q \sim N(p_i)}[A^D(q) = x_i] = \frac{1}{2} + b$ for $b > 0$. Then there exists a quantum algorithm which makes one quantum query (to D) and succeeds with probability $\frac{1}{2} + \frac{4b}{7}$ to output x_i.*

Proof. We use the procedure in Lemma 1 of [101] to determine the output algorithm A on input x at index i. The procedure simulates two classical queries with one quantum query. □

Without loss of generality, all quantum algorithms which make 1-query to D can

be specified in the following manner: there is a quantum state $|Q_i\rangle$, where

$$|Q_i\rangle = \sum_{c\in\{0,1\},j\in[m]} \alpha_{cj} |c\rangle |j\rangle$$

which queries D. After querying D, the resulting quantum state is $|Q_i(x)\rangle$, where

$$|Q_i(x)\rangle = \sum_{c\in\{0,1\},j\in[m]} (-1)^{c\cdot D_j} \alpha_{cj} |c\rangle |j\rangle .$$

There is also a quantum measurement $\{R, I-R\}$ such that, after the algorithm obtains the state $|Q_i(x)\rangle$, it performs the measurement $\{R, I-R\}$. If the algorithm observes R, it outputs 1 and if the algorithm observes $I-R$, it outputs 0.

From Lemma 6.5.24, we know there exist a state $|Q_i\rangle$ and a measurement $\{R, I-R\}$ where if algorithm A succeeds with probability $\frac{1}{2}+\eta$ on random $x \sim \{0,1\}^n$, then the quantum algorithm succeeds with probability $\frac{1}{2}+\frac{4\eta}{7}$ on random $x \sim \{0,1\}^n$.

In order to simplify notation, we write $p(\phi)$ as the probability of making observation R from state $|\phi\rangle$. Since R is a positive semi-definite matrix, $R = M^*M$ and so $p(\phi) = \| M|\phi\rangle \|^2$.

In exactly the same way as [101], we can remove parts of the quantum state $|Q_i(x)\rangle$ where $\alpha_{cj} > \frac{1}{\sqrt{\delta m}} = \frac{1}{\sqrt{\varepsilon K}}$. If we let $L = \{(c,j) \mid \alpha_{cj} \leq \frac{1}{\sqrt{\varepsilon K}}\}$, after keeping only the amplitudes in L, we obtain the quantum state $\frac{1}{a}|A_i(x)\rangle$, where

$$|A_i(x)\rangle = \sum_{(c,j)\in L} (-1)^{c\cdot D_j} \alpha_{cj} |c\rangle |j\rangle , \qquad a = \sqrt{\sum_{(c,j)\in L} \alpha_{cj}^2}.$$

Lemma 6.5.25. *Fix $i \in S$. The quantum state $|A_i(x)\rangle$ satisfies*

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^n}\left[p\left(\frac{1}{a}A_i(x)\right) \mid x_i = 1\right] - \mathop{\mathbb{E}}_{x\in\{0,1\}^n}\left[p\left(\frac{1}{a}A_i(x)\right) \mid x_i = 0\right] \geq \frac{8\eta}{7a^2}.$$

Proof. Note that since $|Q_i(x)\rangle$ and $\{R, I-R\}$ simulate A and succeed with probability

at least $\frac{1}{2} + \frac{4\eta}{7}$ on a random $x \in \{0,1\}^n$, we have that

$$\frac{1}{2} \mathop{\mathbb{E}}_{x\in\{0,1\}^n} \left[p\left(Q_i(x)\right) \mid x_i = 1\right] + \frac{1}{2} \mathop{\mathbb{E}}_{x\in\{0,1\}^n} \left[1 - p\left(Q_i(x)\right) \mid x_i = 0\right] \geq \frac{1}{2} + \frac{4\eta}{7},$$

which we can simplify to say

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^n} \left[p\left(Q_i(x)\right) \mid x_i = 1\right] + \mathop{\mathbb{E}}_{x\in\{0,1\}^n} \left[p\left(Q_i(x)\right) \mid x_i = 0\right] \geq \frac{8\eta}{7}.$$

Since $|Q_i(x)\rangle = |A_i(x)\rangle + |B_i(x)\rangle$ and $|B_i(x)\rangle$ contains at most εK parts, if all probes to D in $|B_i(x)\rangle$ had corrupted values, the algorithm should still succeed with the same probability on random inputs x. Therefore, the following two inequalities hold:

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^n} \left[p\left(A_i(x) + B(x)\right) \mid x_i = 1\right] + \mathop{\mathbb{E}}_{x\in\{0,1\}^n} \left[p\left(A_i(x) + B(x)\right) \mid x_i = 0\right] \geq \frac{8\eta}{7} \quad (6.12)$$

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^n} \left[p\left(A_i(x) - B(x)\right) \mid x_i = 1\right] + \mathop{\mathbb{E}}_{x\in\{0,1\}^n} \left[p\left(A_i(x) - B(x)\right) \mid x_i = 0\right] \geq \frac{8\eta}{7} \quad (6.13)$$

Note that $p(\phi \pm \psi) = p(\phi) + p(\psi) \pm \left(\langle\phi| R |\psi\rangle + \langle\psi| D |\phi\rangle\right)$ and $p(\frac{1}{c}\phi) = \frac{p(\phi)}{c^2}$. One can verify by averaging the two inequalities (6.12) and (6.13) that we get the desired expression. □

Lemma 6.5.26. *Fix $i \in S$. There exists a quantum algorithm that starting from the quantum state $\frac{1}{a}|A_i(x)\rangle$, can recover the value of x_i with probability $\frac{1}{2} + \frac{2\eta}{7a^2}$ over random $x \in \{0,1\}^n$.*

Proof. The algorithm and argument are almost identical to Theorem 3 in [101], we just check that it works under the weaker assumptions. Let

$$q_1 = \mathop{\mathbb{E}}_{x\in\{0,1\}^n} \left[p\left(\frac{1}{a}A_i(x)\right) \mid x_i = 1\right] \qquad q_0 = \mathop{\mathbb{E}}_{x\in\{0,1\}^n} \left[p\left(\frac{1}{a}A_i(x)\right) \mid x_i = 0\right].$$

From Lemma 6.5.25, we know $q_1 - q_0 \geq \frac{8\eta}{7a^2}$. In order to simplify notation, let $b = \frac{4\eta}{7a^2}$. So we want a quantum algorithm which starting from state $\frac{1}{a}|A_i(x)\rangle$ can recover x_i with probability $\frac{1}{2} + \frac{b}{2}$ on random $x \in \{0,1\}^n$. Assume $q_1 \geq \frac{1}{2} + b$, since otherwise

$q_0 \leq \frac{1}{2} - b$ and the same argument will work for 0 and 1 flipped. Also, assume $q_1 + q_0 \geq 1$, since otherwise simply outputting 1 on observation R and 0 on observation $I - R$ will work.

The algorithm works in the following way: it outputs 0 with probability $1 - \frac{1}{q_1+q_0}$ and otherwise makes the measurement $\{R, I-R\}$ on state $\frac{1}{a}|A_i(x)\rangle$. If the observation made is R, then the algorithm outputs 1, otherwise, it outputs 0. The probability of success over random input $x \in \{0,1\}^n$ is

$$\begin{aligned}&\mathop{\mathbb{E}}_{x\in\{0,1\}^n}[\Pr[\text{returns correctly}]] \\ &\quad = \frac{1}{2}\mathop{\mathbb{E}}_{x\in\{0,1\}^n}[\Pr[\text{returns 1}] \mid x_i = 1] + \frac{1}{2}\mathop{\mathbb{E}}_{x\in\{0,1\}^n}[\Pr[\text{returns 0}] \mid x_i = 0]. \quad (6.14)\end{aligned}$$

When $x_i = 1$, the probability the algorithm returns correctly is $(1-q)p\left(\frac{1}{a}A_i(x)\right)$ and when $x_i = 0$, the probability the algorithm returns correctly is $q+(1-q)(1-p(\frac{1}{a}A_i(x)))$. So simplifying (6.14),

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^n}[\Pr[\text{returns correctly}]] = \frac{1}{2}(1-q)q_1 + \frac{1}{2}(q + (1-q)(1-q_0)) \geq \frac{1}{2} + \frac{b}{2}.$$

□

Now we can finally complete the proof of Lemma 6.5.23.

Proof of Lemma 6.5.23. Again, the proof is exactly the same as the finishing arguments of Theorem 3 in [101], and we simply check the weaker conditions give the desired outcome. On input $i \in [n]$ and access to r copies of the state $|U(x)\rangle$, the algorithm applies the measurement $\{M_i^* M_i, I - M_i^* M_i\}$ where

$$M_i = \sqrt{\varepsilon K} \sum_{(c,j)\in L} \alpha_{cj} |c,j\rangle \langle c,j|.$$

This measurement is designed in order to yield the state $\frac{1}{a}|A_i(x)\rangle$ on $|U(x)\rangle$ if the measurement makes the observation $M_i^* M_i$. The fact that the amplitudes of $|A_i(x)\rangle$ are not too large makes $\{M_i^* M_i, I - M_i^* M_i\}$ a valid measurement.

The probability of observing $M_i^* M_i$ is $\langle U(x)| M_i^* M_i |U(x)\rangle = \frac{\delta a^2}{2}$, where we used that $\delta = \frac{\varepsilon K}{m}$. So the algorithm repeatedly applies the measurement until observing outcome $M_i^* M_i$. If it never makes the observation, the algorithm outputs 0 or 1 uniformly at random. If the algorithm does observe $M_i^* M_i$, it runs the output of the algorithm of Lemma 6.5.26. The following simple calculation (done in [101]) gives the desired probability of success on random input,

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x\in\{0,1\}^n} [\Pr[\text{returns correctly}]] \\
&\geq \left(1-(1-\delta a^2/2)^r\right)\left(\frac{1}{2}+\frac{2\eta}{7a^2}\right) + (1-\delta a^2/2)^r \cdot \frac{1}{2} \geq \frac{1}{2}+\frac{\eta}{7a^2}.
\end{aligned}$$

□

On adaptivity

We can extend our lower bounds from the non-adaptive to the adaptive setting.

Lemma 6.5.27. *If there exists a deterministic data structure which queries two memory cells adaptively and succeeds with probability at least $\frac{1}{2}+\eta$, there exists a deterministic data structure which makes the two queries non-adaptively and succeeds with probability at least $\frac{1}{2}+\frac{\eta}{2^w}$.*

Proof. The algorithm guesses the outcome of the first cell probe and simulates the adaptive algorithm with the guess. After knowing which two probes to make, we probe the data structure non-adaptively. If the algorithm guessed the contents of the first cell-probe correctly, then we output the value of the non-adaptive algorithm. Otherwise, we output a random value. This algorithm is non-adaptive and succeeds with probability at least $\left(1-\frac{1}{2^w}\right)\cdot\frac{1}{2}+\frac{1}{2^w}\left(\frac{1}{2}+\eta\right)=\frac{1}{2}+\frac{\eta}{2^w}$. □

Applying Lemma 6.5.27, from an adaptive algorithm succeeding with probability $\frac{2}{3}$, we obtain a non-adaptive algorithm succeeding with probability $\frac{1}{2}+\Omega(2^{-w})$. This value is lower than the intended $\frac{2}{3}$, but we may still reduce to a weak LDC, where we require $\gamma=\Theta(2^{-w})$, $\varepsilon=\Theta(2^{-w})$, and $|S|=\Omega(2^{-w}n)$. With these minor changes to

the parameters in Subsections 6.5.1 through 6.5.6, one can easily verify

$$\frac{m \log m \cdot 2^{\Theta(w)}}{n} \geq \Omega\left(\Phi_r\left(\frac{1}{m}, \gamma\right)\right).$$

This inequality yields tight lower bounds (up to sub-polynomial factors) for the Hamming space when $w = o(\log n)$.

In the case of the Hamming space, we can compute robust expansion in a similar fashion to Theorem 6.1.1. In particular, for any $p, q \in [1, \infty)$ where $(p-1)(q-1) = \sigma^2$, we have

$$\begin{aligned}
\frac{m \log m \cdot 2^{O(w)}}{n} &\geq \Omega(\gamma^q m^{1+q/p-q}) \\
m^{q-q/p+o(1)} &\geq n^{1-o(1)} \gamma^q \\
m &\geq n^{\frac{1-o(1)}{q-q/p+o(1)}} \gamma^{\frac{q}{q-q/p+o(1)}} = n^{\frac{p}{pq-q}-o(1)} \gamma^{\frac{p}{p-1}-o(1)}.
\end{aligned}$$

Let $p = 1 + \frac{wf(n)}{\log n}$ and $q = 1 + \sigma^2 \frac{\log n}{wf(n)}$ where we require that $wf(n) = o(\log n)$ and $f(n) \to \infty$ as $n \to \infty$. Then,

$$m \geq n^{\frac{1}{\sigma^2}-o(1)} 2^{\frac{\log n}{wf(n)}} \geq n^{\frac{1}{\sigma^2}-o(1)}.$$

Bibliography

[1] Twitter. Available as `https://twitter.com/`.

[2] Amirali Abdullah and Suresh Venkatasubramanian. A Directed Isoperimetric Inequality with Application to Bregman Near Neighbor Lower Bounds. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC '2015)*, pages 509–518, 2015.

[3] I. Aharoni, B. Maurey, and B. S. Mityagin. Uniform Embeddings of Metric Spaces and of Banach Spaces into Hilbert Spaces. *Israel Journal of Mathematics*, 52(3):251–265, 1985.

[4] Thomas D. Ahle, Rasmus Pagh, Ilya Razenshteyn, and Francesco Silvestri. On the Complexity of Inner Product Similarity Join. In *Proceedings of the 35th ACM Symposium on Principles of Database Systems (PODS '2016)*, pages 151–164, 2016. Available as arXiv:1510.02824.

[5] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing Graph Structure via Linear Measurements. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA '2012)*, pages 459–467, 2012.

[6] Nir Ailon and Bernard Chazelle. The Fast Johnson–Lindenstrauss Transform and Approximate Nearest Neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009.

[7] Nir Ailon and Holger Rauhut. Fast and RIP-Optimal Transforms. *Discrete and Computational Geometry*, 52(4):780–798, 2014.

[8] Zeyuan Allen-Zhu, Yang Yuan, and Karthik Sridharan. Exploiting the Structure: Stochastic Gradient Methods Using Raw Clusters. In *Proceedings of Advances in Neural Information Processing Systems 29 (NIPS '2016)*, pages 1642–1650, 2016. Available as arXiv:1602.02151.

[9] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

[10] Alexandr Andoni. *Nearest Neighbor Search: the Old, the New, and the Impossible.* PhD thesis, MIT, 2009.

[11] Alexandr Andoni, Dorian Croitoru, and Mihai Patrascu. Hardness of Nearest Neighbor under L-infinity. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2008)*, pages 424–433, 2008.

[12] Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David P. Woodruff. Efficient Sketches for Earth-Mover Distance, with Applications. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2009)*, pages 324–330, 2009.

[13] Alexandr Andoni and Piotr Indyk. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2006)*, pages 459–468, 2006.

[14] Alexandr Andoni and Piotr Indyk. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[15] Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Earth Mover Distance over High-Dimensional Spaces. In *Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms (SODA '2009)*, pages 343–352, 2009.

[16] Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Overcoming the ℓ_1 Non-Embeddability Barrier: Algorithms for Product Metrics. In *Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms (SODA '2009)*, pages 865–874, 2009.

[17] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and Optimal LSH for Angular Distance. In *Proceedings of Advances in Neural Information Processing Systems 28 (NIPS '2015)*, pages 1225–1233, 2015. Available as arXiv:1509.02897.

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

[19] Alexandr Andoni, Piotr Indyk, and Mihai Patrascu. On the Optimality of the Dimensionality Reduction Method. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2006)*, pages 449–458, 2006.

[20] Alexandr Andoni, T. S. Jayram, and Mihai Patrascu. Lower Bounds for Edit Distance and Product Metrics via Poincaré-Type Inequalities. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA '2010)*, pages 184–192, 2010.

[21] Alexandr Andoni and Robert Krauthgamer. The Computational Hardness of Estimating Edit Distance. *SIAM Journal on Computing*, 39(6):2398–2429, 2010.

[22] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS '2010)*, pages 377–386, 2010.

[23] Alexandr Andoni, Robert Krauthgamer, and Ilya Razenshteyn. Sketching and Embedding are Equivalent for Norms. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC '2015)*, pages 479–488, 2015. Available as arXiv:1411.2577.

[24] Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Optimal Hashing-based Time–Space Trade-offs for Approximate Near Neighbors. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA '2017)*, 2017. Available as arXiv:1608.03580.

[25] Alexandr Andoni, Huy L. Nguyen, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Approximate Near Neighbors for General Symmetric Norms. In *Proceedings of the 49th ACM Symposium on the Theory of Computing (STOC '2017)*, 2017. Available as arXiv:1611.06222.

[26] Alexandr Andoni and Ilya Razenshteyn. Optimal Data-Dependent Hashing for Approximate Near Neighbors. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC '2015)*, pages 793–801, 2015. Available as arXiv:1501.01062.

[27] Alexandr Andoni and Ilya Razenshteyn. Tight Lower Bounds for Data-Dependent Locality-Sensitive Hashing. In *Proceedings of the 32nd International Symposium on Computational Geometry (SoCG '2016)*, pages 9:1–9:11, 2016. Available as arXiv:1507.04299.

[28] Alexandr Andoni, Ilya Razenshteyn, and Negev Shekel Nosatzki. LSH Forest: Practical Algorithms Made Theoretical. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA '2017)*, 2017.

[29] Andreas Argyriou, Rina Foygel, and Nathan Srebro. Sparse Prediction with the k-Support Norm. In *Proceedings of Advances in Neural Information Processing Systems 25 (NIPS '2012)*, pages 944–952, 2012.

[30] Yonatan Aumann and Yuval Rabani. An $O(\log k)$ Approximate Min-Cut Max-Flow Theorem and Approximation Algorithm. *SIAM Journal on Computing*, 27(1):291–301, 1998.

[31] Keith Ball. Markov Chains, Riesz Transforms and Lipschitz Maps. *Geometric and Functional Analysis*, 2(2):137–172, 1992.

[32] Keith Ball. *An Elementary Introduction to Modern Convex Geometry*, volume 31 of *MSRI Publications*. Cambridge University Press, 1997.

[33] Ziv Bar-Yossef, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. Approximating Edit Distance Efficiently. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2004)*, pages 550–559, 2004.

[34] Ziv Bar-Yossef, T.S. Jayram, Ravi Kumar, and D. Sivakumar. An Information Statistics Approach to Data Stream and Communication Complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[35] Omer Barkol and Yuval Rabani. Tighter Lower Bounds for Nearest Neighbor Search and Related Problems in the Cell Probe Model. *Journal of Computer and System Sciences*, 64(4):873–896, 2002.

[36] Yair Bartal and Lee-Ad Gottlieb. Approximate Nearest Neighbor Search for ℓ_p-Spaces ($2 < p < \infty$) via Embeddings. Available as arXiv:1512.01775, 2015.

[37] Tugkan Batu, Funda Ergun, and Cenk Sahinalp. Oblivious String Embeddings and Edit Distance Approximation. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA '2006)*, pages 792–801, 2006.

[38] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New Directions in Nearest Neighbor Searching with Applications to Lattice Sieving. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms (SODA '2016)*, pages 10–24, 2016.

[39] Avraham Ben-Aroya, Oded Regev, and Ronald de Wolf. A Hypercontractive Inequality for Matrix-Valued Functions with Applications to Quantum Computing and LDCs. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2008)*, pages 477–486, 2008.

[40] Colin Bennett and Robert Sharpley. *Interpolation of Operators*, volume 129 of *Pure and Applied Mathematics*. Academic Press, 1988.

[41] Yoav Benyamini and Joram Lindenstrauss. *Geometric Nonlinear Functional Analysis. Vol. 1*, volume 48 of *American Mathematical Society Colloquium Publications*. American Mathematical Society, Providence, RI, 2000.

[42] Jaroslaw Blasiok, Vladimir Braverman, Stephen R. Chestnut, Robert Krauthgamer, and Lin F. Yang. Streaming Symmetric Norms via Measure Concentration. In *Proceedings of the 49th ACM Symposium on the Theory of Computing (STOC '2017)*, 2017. Available as arXiv:1511.01111.

[43] Mariusz Bojarski, Anna Choromanska, Krzysztof Choromanski, Francois Fagan, Cedric Gouy-Pailler, Anne Morvan, Nourhan Sakr, Tamas Sarlos, and Jamal Atif. Structured Adaptive and Random Spinners for Fast Machine Learning

Computations. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS '2017)*, 2017.

[44] Allan Borodin, Rafail Ostrovsky, and Yuval Rabani. Lower Bounds for High Dimensional Nearest Neighbor Search and Related Problems. In *Proceedings of the 31st ACM Symposium on the Theory of Computing (STOC '1999)*, pages 312–321, 1999.

[45] Vladimir Braverman and Rafail Ostrovsky. Zero-One Frequency Laws. In *Proceedings of the 42rd ACM Symposium on the Theory of Computing (STOC '2010)*, pages 281–290, 2010.

[46] Bo Brinkman and Moses Charikar. On the Impossibility of Dimension Reduction in ℓ_1. *Journal of the ACM*, 52(5):766–788, 2005.

[47] Andrei Z. Broder. On the Resemblance and Containment of Documents. In *Proceedings of the Compression and Complexity of Sequences (SEQUENCES '1997)*, pages 21–29, 1997.

[48] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic Clustering of the Web. *Computer Networks and ISDN Systems*, 29(8–13):1157–1166, 1997.

[49] Joshua Brody and Amit Chakrabarti. A Multi-Round Communication Lower Bound for Gap Hamming and Some Consequences. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity (CCC '2009)*, pages 358–368, 2009.

[50] Joshua Brody, Amit Chakrabarti, Oded Regev, Thomas Vidick, and Ronald de Wolf. Better Gap-Hamming Lower Bounds via Better Round Elimination. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX '2010, and 14th International Workshop, RANDOM '2010*, pages 476–489, 2010.

[51] J. Lawrence Carter and Mark N. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[52] Amit Chakrabarti, Bernard Chazelle, Benjamin Gum, and Alexey Lvov. A Lower Bound on the Complexity of Approximate Nearest-Neighbor Searching on the Hamming Cube. In *Proceedings of the 31st ACM Symposium on the Theory of Computing (STOC '1999)*, pages 305–311, 1999.

[53] Amit Chakrabarti and Oded Regev. An Optimal Randomized Cell Probe Lower Bound for Approximate Nearest Neighbor Searching. *SIAM Journal on Computing*, 39(5):1919–1940, 2010.

[54] Amit Chakrabarti and Oded Regev. An Optimal Lower Bound on the Communication Complexity of Gap-Hamming-Distance. *SIAM Journal on Computing*, 41(5):1299–1317, 2012.

[55] Moses Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC '2002)*, pages 380–388, 2002.

[56] Moses Charikar and Robert Krauthgamer. Embedding the Ulam Metric into ℓ_1. *Theory of Computing*, 2(11):207–224, 2006.

[57] Jeff Cheeger and Bruce Kleiner. Differentiating Maps into L^1, and the Geometry of BV Functions. *Annals of Mathematics*, 171(2):1347–1385, 2010.

[58] Jeff Cheeger, Bruce Kleiner, and Assaf Naor. Compression Bounds for Lipschitz Maps from the Heisenberg Group to L_1. *Acta Mathematica*, 207(2):291–373, 2011.

[59] Kenneth L. Clarkson. A Randomized Algorithm for Closest-Point Queries. *SIAM Journal on Computing*, 17(4):830–847, 1988.

[60] Kenneth L. Clarkson and David P. Woodruff. Sketching for M-Estimators: A Unified Approach to Robust Regression. In *Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms (SODA '2015)*, pages 921–939, 2015.

[61] Graham Cormode and S. Muthukrishnan. The String Edit Distance Matching Problem with Moves. *ACM Transactions on Algorithms*, 3(1):2:1–2:19, 2007.

[62] Graham Cormode, S. Muthukrishnan, and Cenk Sahinalp. Permutation Editing and Matching via Embeddings. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming (ICALP '2001)*, pages 481–492, 2001.

[63] Graham Cormode, Mike Paterson, Suleyman Cenk Sahinalp, and Uzi Vishkin. Communication Complexity of Document Exchange. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA '2000)*, pages 197–206, 2000.

[64] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. Fast Locality-Sensitive Hashing. In *Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data (KDD '2011)*, pages 1073–1081, 2011.

[65] Sanjoy Dasgupta and Anupam Gupta. An Elementary Proof of a Theorem of Johnson and Lindenstrauss. *Random Structures and Algorithms*, 22(1):60–65, 2003.

[66] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-Sensitive Hashing Scheme Based on p-Stable Distributions. In *Proceedings of the 20th ACM Symposium on Computational Geometry (SoCG '2004)*, pages 253–262, 2004.

[67] Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Nearest Neighbor Based Greedy Coordinate Descent. In *Proceedings of Advances in Neural Information Processing Systems 24 (NIPS '2011)*, pages 2160–2168, 2011.

[68] Persi Diaconis and David Freedman. A Dozen de Finetti-style Results in Search of a Theory. *Annales de l'institut Henri Poincaré*, 23(S2):397–423, 1987.

[69] S. J. Dilworth and S. J. Montgomery-Smith. The Distribution of Vector-Valued Rademacher Series. *Annals of Probability*, 21(4):2046–2052, 1993.

[70] Moshe Dubiner. Bucketing Coding and Information Theory for the Statistical High-Dimensional Nearest-Neighbor Problem. *IEEE Transactions on Information Theory*, 56(8):4166–4179, 2010.

[71] Zeev Dvir and Sivakanth Gopi. 2-Server PIR with Sub-Polynomial Communication. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC '2015)*, pages 577–584, 2015.

[72] Kave Eshghi and Shyamsundar Rajaram. Locality Sensitive Hash Functions Based on Concomitant Rank Order Statistics. In *Proceedings of the 14th ACM SIGKDD Conference on Knowledge Discovery and Data (KDD '2008)*, pages 221–229, 2008.

[73] Uriel Feige and Gideon Schechtman. On the Optimality of the Random Hyperplane Rounding Technique for MAX CUT. *Random Structures and Algorithms*, 20(3):403–440, 2002.

[74] William Feller. *An Introduction to Probability Theory and its Applications, Vol. 2.* Wiley, New York, 1971.

[75] Matteo Frigo and Steven G. Johnson. The Design and Implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[76] Sudipto Guha, Piotr Indyk, and Andrew McGregor. Sketching Information Divergences. *Journal of Machine Learning Research*, 72(1–2):5–19, 2008.

[77] 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.

[78] Edwin Hewitt and Kenneth A. Ross. *Abstract Harmonic Analysis: Volume I.* Grundlehren der mathematischen Wissenschaften. Springer New York, 1994.

[79] Thomas Hofmann, Aurélien Lucchi, Simon Lacoste-Julien, and Brian McWilliams. Variance Reduced Stochastic Gradient Descent with Neighbors. In *Proceedings of Advances in Neural Information Processing Systems 28 (NIPS '2015)*, pages 2305–2313, 2015.

[80] Piotr Indyk. *High-Dimensional Computational Geometry.* PhD thesis, Stanford University, 2001.

[81] Piotr Indyk. On Approximate Nearest Neighbors under ℓ_∞ Norm. *Journal of Computer and System Sciences*, 63(4):627–638, 2001.

[82] Piotr Indyk. Approximate Nearest Neighbor Algorithms for Fréchet Distance via Product Metrics. In *Proceedings of the 18th ACM Symposium on Computational Geometry (SoCG '2002)*, pages 102–106, 2002.

[83] Piotr Indyk. Approximate Nearest Neighbor under Edit Distance via Product Metrics. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA '2004)*, pages 646–650, 2004.

[84] Piotr Indyk. Stable Distributions, Pseudorandom Generators, Embeddings, and Data Stream Computation. *Journal of the ACM*, 53(3):307–323, 2006.

[85] Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the 30th ACM Symposium on the Theory of Computing (STOC '1998)*, pages 604–613, 1998.

[86] Piotr Indyk and Nitin Thaper. Fast Color Image Retrieval via Embeddings. Workshop on Statistical and Computational Theories of Vision (at ICCV), 2003.

[87] Piotr Indyk and David Woodruff. Optimal Approximations of the Frequency Moments of Data Streams. In *Proceedings of the 35th ACM Symposium on the Theory of Computing (STOC '2005)*, pages 202–208, 2005.

[88] T. S. Jayram, Subhash Khot, Ravi Kumar, and Yuval Rabani. Cell-Probe Lower Bounds for the Partial Match Problem. *Journal of Computer and System Sciences*, 69(3):435–447, 2004.

[89] T. S. Jayram, Ravi Kumar, and D. Sivakumar. The One-Way Communication Complexity of Hamming Distance. *Theory of Computing*, 4(1):129–135, 2008.

[90] T. S. Jayram and David P. Woodruff. The Data Stream Space Complexity of Cascaded Norms. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2009)*, pages 765–774, 2009.

[91] Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, January 2011.

[92] Fritz John. Extremum Problems with Inequalities as Subsidiary Conditions. In *Studies and Essays Presented to R. Courant on his 60th Birthday, January 8, 1948*, pages 187–204. Interscience Publishers, Inc., New York, N. Y., 1948.

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

[94] William B. Johnson and N. Lovasoa Randrianarivony. l_p ($p > 2$) does not Coarsely Embed into a Hilbert Space. *Proceedings of the American Mathematical Society*, 134(4):1045–1050, 2006.

[95] N. J. Kalton. Banach Spaces Embedding into L_0. *Israel Journal of Mathematics*, 52(4):305–319, 1985.

[96] Bala Kalyanasundaram and Georg Schintger. The Probabilistic Communication Complexity of Set Intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992.

[97] Michael Kapralov. Smooth Tradeoffs Between Insert and Query Complexity in Nearest Neighbor Search. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems (PODS '2015)*, pages 329–342, 2015.

[98] Michael Kapralov and Rina Panigrahy. NNS Lower Bounds via Metric Expansion for ℓ_∞ and EMD. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP '2012)*, pages 545–556, 2012.

[99] Bruce M. Kapron, Valerie King, and Ben Mountjoy. Dynamic Graph Connectivity in Polylogarithmic Worst Case Time. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA '2013)*, pages 1131–1142, 2013.

[100] Christopher Kennedy and Rachel Ward. Fast Cross-Polytope Locality-Sensitive Hashing. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS '2017)*, 2017. Available as arXiv:1602.06922.

[101] Iordanis Kerenidis and Ronald de Wolf. Exponential Lower Bound for 2-Query Locally Decodable Codes via a Quantum Argument. *Journal of Computer and System Sciences*, 69(3):395–420, 2004. Available as arXiv:quant-ph/0208062.

[102] Subhash Khot and Assaf Naor. Nonembeddability Theorems via Fourier Analysis. *Mathematische Annalen*, 334(4):821–852, 2006.

[103] Subhash Khot and Rishi Saket. SDP Integrality Gaps with Local ℓ_1-Embeddability. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2009)*, pages 565–574, 2009.

[104] Subhash Khot and Nisheeth Vishnoi. The Unique Games Conjecture, Integrality Gap for Cut Problems and Embeddability of Negative-Type Metrics into ℓ_1. *Journal of the ACM*, 62:1(8):8:1–8:39, 2015.

[105] Felix Krahmer and Rachel Ward. A Unified Framework for Linear Dimensionality Reduction in L1. *Results in Mathematics*, 70(1):209–231, 2016.

[106] Robert Krauthgamer and Yuval Rabani. Improved Lower Bounds for Embeddings into L_1. *SIAM Journal on Computing*, 38(6):2487–2498, 2009.

[107] Ilan Kremer, Noam Nisan, and Dana Ron. On Randomized One-Round Communication Complexity. *Computational Complexity*, pages 21–49, 1999.

[108] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.

[109] Thijs Laarhoven. *Search Problems in Cryptography: From Fingerprinting to Lattice Sieving*. PhD thesis, Eindhoven University of Technology, 2015.

[110] James R. Lee and Assaf Naor. L_p Metrics on the Heisenberg Group and the Goemans-Linial Conjecture. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2006)*, pages 99–108, 2006.

[111] Yi Li, Huy L. Nguyên, and David P. Woodruff. On Sketching Matrix Norms and the Top Singular Vector. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA '2014)*, pages 1562–1581, 2014.

[112] Yi Li, Huy L Nguyên, and David P. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In *Proceedings of the 46th ACM Symposium on the Theory of Computing (STOC '2014)*, pages 174–184, 2014.

[113] Moshe Lichman. UCI Machine Learning Repository, 2013.

[114] Nathan Linial, Eran London, and Yuri Rabinovich. The Geometry of Graphs and Some of its Algorithmic Applications. *Combinatorica*, 15(2):215–245, 1995.

[115] Ding Liu. A Strong Lower Bound for Approximate Nearest Neighbor Searching. *Information Processing Letters*, 92(1):23–29, 2004.

[116] Mingmou Liu, Xiaoyin Pan, and Yitong Yin. Randomized Approximate Nearest Neighbor Search with Limited Adaptivity. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '2016)*, pages 23–33, 2016.

[117] Qin Lv, Moses Charikar, and Kai Li. Image Similarity Search with Compact Data Structures. In *Proceedings of the 13th ACM International Conference on Information and Knowledge Management (CIKM '2004)*, pages 208–217, 2004.

[118] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB '2007)*, pages 950–961, 2007.

[119] Albert W. Marshall, Ingram Olkin, and Barry C. Arnold. *Inequalities: Theory of Majorization and its Applications*. Academic Press, 2011.

[120] Jiří Matoušek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel Journal of Mathematics*, 93:333–344, 1996.

[121] Jiří Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.

[122] Jiří Matoušek and Assaf Naor. Open Problems on Embeddings of Finite Metric Spaces. Available as `http://kam.mff.cuni.cz/~matousek/metrop.ps.gz`. Last accessed in November, 2014., 2011.

[123] Andrew M. McDonald, Massimiliano Pontil, and Dimitris Stamos. Spectral k-Support Norm Regularization. In *Proceedings of Advances in Neural Information Processing Systems 27 (NIPS '2014)*, pages 3644–3652, 2014.

[124] Andrew McGregor. Open problems in data streams and related topics. *IITK Workshop on Algorithms For Data Streams*, 2006. Available as `http://sublinear.info`.

[125] Stefan Meiser. Point Location in Arrangements of Hyperplanes. *Information and Computation*, 106(2):286–303, 1993.

[126] Peter Bro Miltersen. Cell Probe Complexity – a Survey. In *Advances in Data Structures*, 1999.

[127] Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On Data Structures and Asymmetric Communication Complexity. In *Proceedings of the 27th ACM Symposium on the Theory of Computing (STOC '1995)*, pages 103–111, 1995.

[128] George J. Minty. On the Extension of Lipschitz, Lipschitz–Hölder Continuous, and Monotone Functions. *Bulletin of the American Mathematical Society*, 76(2):334–339, 1970.

[129] Rajeev Motwani, Assaf Naor, and Rina Panigrahy. Lower Bounds on Locality Sensitive Hashing. *SIAM Journal on Discrete Mathematics*, 21(4):930–935, 2007.

[130] James R. Munkres. *Topology*. Prentice Hall, Incorporated, 2000.

[131] S. Muthukrishnan and S. C. Sahinalp. Approximate Nearest Neighbors and Sequence Comparisons with Block Operations. In *Proceedings of the 32nd ACM Symposium on the Theory of Computing (STOC '2000)*, pages 416–424, 2000.

[132] Assaf Naor. A Spectral Gap Precludes Low-Dimensional Embeddings. In *Proceedings of the 33rd International Symposium on Computational Geometry (SoCG '2017)*, 2017.

[133] Assaf Naor and Yuval Rabani. On Approximate Nearest Neighbor Search in ℓ_p, $p > 2$. Manuscript, 2006.

[134] Assaf Naor and Gideon Schechtman. Planar Earthmover is not in L_1. *SIAM Journal on Computing*, 37(3):804–826, 2007.

[135] Ashwin Nayak. Optimal lower bounds for quantum automata and random access codes. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS '1999)*, pages 369–376, 1999.

[136] Jelani Nelson. *Sketching and Streaming High-Dimensional Vectors*. PhD thesis, Massachusetts Institute of Technology, 2011.

[137] Huy L. Nguyên. *Algorithms for High Dimensional Data*. PhD thesis, Princeton University, 2014. Available as http://arks.princeton.edu/ark:/88435/dsp01b8515q61f.

[138] E. M. Nikišin. A Resonance Theorem and Series in Eigenfunctions of the Laplace Operator. *Izvestiya Rossiiskoi Akademii Nauk Seriya Matematicheskaya*, 36:795–813, 1972.

[139] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[140] Ryan O'Donnell, Yi Wu, and Yuan Zhou. Optimal Lower Bounds for Locality-Sensitive Hashing (Except When q is Tiny). *ACM Transactions on Computation Theory*, 6(1):5, 2014.

[141] Rafail Ostrovsky and Yuval Rabani. Low Distortion Embedding for Edit Distance. *Journal of the ACM*, 54(5):23:1–23:16, 2007.

[142] Mark H. Overmars and Jan van Leeuwen. Some Principles for Dynamizing Decomposable Searching Problems. *Information Processing Letters*, 12(1):49–53, 1981.

[143] Dmitry Panchenko. Lecture notes in probability theory, 2008.

[144] Rina Panigrahy. Entropy Based Nearest Neighbor Search in High Dimensions. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA '2006)*, pages 1186–1195, 2006.

[145] Rina Panigrahy, Kunal Talwar, and Udi Wieder. A Geometric Approach to Lower Bounds for Approximate Near-Neighbor Search and Partial Match. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2008)*, pages 414–423, 2008.

[146] Rina Panigrahy, Kunal Talwar, and Udi Wieder. Lower Bounds on Near Neighbor Search via Metric Expansion. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS '2010)*, pages 805–814, 2010.

[147] Mihai Patrascu. Unifying the Landscape of Cell-Probe Lower Bounds. *SIAM Journal on Computing*, 40(3):827–847, 2011.

[148] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP '2014)*, pages 1532–1543, 2014.

[149] Gilles Pisier. Some Results on Banach Spaces Without Local Unconditional Structure. *Compositio Mathematica*, 37(1):3–19, 1978.

[150] Mihai Pătraşcu and Mikkel Thorup. Higher Lower Bounds for Near-Neighbor and Further Rich Problems. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2006)*, pages 646–654, 2006.

[151] N. Lovasoa Randrianarivony. Characterization of Quasi-Banach Spaces which Coarsely Embed into a Hilbert Space. *Proceedings of the American Mathematical Society*, 134(5):1315–1317, 2006.

[152] Ilya Razenshteyn and Ludwig Schmidt. FALCONN: Similarity Search over High-Dimensional Data. Available as `https://falconn-lib.org/`, 2015.

[153] Ilya Razenshteyn and Ludwig Schmidt. FFHT: A C99 Implementation of the Fast Hadamard Transform. Available as `https://github.com/FALCONN-LIB/FFHT`, 2015.

[154] Ilya Razenshteyn, Zhao Song, and David P. Woodruff. Weighted Low Rank Approximations with Provable Guarantees. In *Proceedings of the 48th ACM Symposium on the Theory of Computing (STOC '2016)*, pages 250–263, 2016.

[155] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[156] Walter Rudin. *Functional Analysis*. McGraw-Hill, Inc., 2nd edition, 1991.

[157] Michael Saks and Xiaodong Sun. Space Lower Bounds for Distance Approximation in the Data Stream Model. In *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC '2002)*, pages 360–369, 2002.

[158] Ludwig Schmidt, Matthew Sharifi, and Ignacio Lopez Moreno. Large-Scale Speaker Identification. In *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '2014)*, 2014.

[159] Isaac J. Schoenberg. On Certain Metric Spaces Arising from Euclidean Spaces by a Change of Metric and their Imbedding in Hilbert Space. *Annals of Mathematics*, 38(4):787–793, 1937.

[160] Issac J. Schoenberg. Remarks to Maurice Fréchet's article "Sur La Définition Axiomatique D'une Classe D'espace Distanciés Vectoriellement Applicable Sur L'espace De Hilbert". *Annals of Mathematics*, 36(3):724–732, 1935.

[161] Issac J. Schoenberg. Metric Spaces and Positive Definite Functions. *Transactions of the American Mathematical Society*, 44(3):522–536, 1938.

[162] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press, 2006.

[163] Alexander A. Sherstov. The Communication Complexity of Gap Hamming Distance. *Theory of Computing*, 8(8):197–208, 2012.

[164] Malcolm Slaney, Yury Lifshits, and Junfeng He. Optimal Parameters for Locality-Sensitive Hashing. *Proceedings of the IEEE*, 100(9):2604–2623, 2012.

[165] Ryan Spring and Anshumali Shrivastava. Scalable and Sustainable Deep Learning via Randomized Hashing. In *Proceedings of the 23rd ACM SIGKDD Conference on Knowledge Discovery and Data (KDD '2017)*, 2017. Available as arXiv:1602.08194.

[166] Narayanan Sundaram, Aizana Turmukhametova, Nadathur Satish, Todd Mostak, Piotr Indyk, Samuel Madden, and Pradeep Dubey. Streaming Similarity Search over One Billion Tweets Using Parallel Locality-Sensitive Hashing. In *Proceedings of the 39th International Conference on Very Large Data Bases (VLDB '2013)*, pages 1930–1941, 2013.

[167] Michel Talagrand. Embedding Subspaces of L_1 into l_1^N. *Proceedings of the American Mathematical Society*, 108(2):363–369, 1990.

[168] Kengo Terasawa and Yuzuru Tanaka. Spherical LSH for Approximate Nearest Neighbor Search on Unit Hypersphere. In *Proceedings of the 10th International Workshop on Algorithms and Data Structures (WADS '2007)*, pages 27–38, 2007.

[169] Gregory Valiant. Finding Correlations in Subquadratic Time, with Applications to Learning Parities and the Closest Pair Problem. *Journal of the ACM*, 62(2):13, 2015.

[170] Thomas Vidick. A Concentration Inequality for the Overlap of a Vector on a Large Set, with Application to the Communication Complexity of the Gap-Hamming-Distance Problem. *Chicago Journal of Theoretical Computer Science*, 2012(1), 2012.

[171] Jingdong Wang, Heng Tao Shen, Kingkuan Song, and Jianqiu Ji. Hashing for Similarity Search: a Survey. Available as arXiv:1408.2927, 2014.

[172] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to Hash for Indexing Big Data – A Survey. Available as arXiv:1509.05472, 2015.

[173] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature Hashing for Large Scale Multitask Learning. In *Proceedings of the 26th International Conference on Machine Learning (ICML '2009)*, pages 1113–1120, 2009.

[174] J. H. Wells and L. R. Williams. *Embeddings and Extensions in Analysis*. Springer-Verlag, New York-Heidelberg, 1975.

[175] Ryan Williams. A New Algorithm for Optimal 2-Constraint Satisfaction and its Implications. *Theoretical Computer Science*, 348(2–3):357–365, 2005.

[176] Przemysław Wojtaszczyk. *Banach Spaces for Analysts*. Cambridge University Press, 1991.

[177] David P. Woodruff. Optimal Space Lower Bounds for all Frequency Moments. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA '2004)*, pages 167–175, 2004.

[178] David P. Woodruff. Sketching as a Tool for Numerical Linear Algebra. *Foundations and Trends in Theoretical Computer Science*, 10:1–157, 2014.

[179] Sergey Yekhanin. *Locally Decodable Codes*, volume 6 of *Foundations and Trends in Theoretical Computer Science*. Now Publishers, 2012.

[180] Yitong Yin. Simple Average-Case Lower Bounds for Approximate Near-Neighbor from Isoperimetric Inequalities. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP '2016)*, pages 84:1–84:13, 2016. Available as arXiv:1602.05391.

[181] A. Zvavitch. More on Embedding Subspaces of L_p into l_p^N, $0 < p < 1$. In *Geometric aspects of functional analysis*, volume 1745 of *Lecture Notes in Math.*, pages 269–280. Springer, Berlin, 2000.