Efficient Metric Representations for Big Data

by

Tal Wagner

B.Sc., Technion – Israel Institute of Technology (2011)
M.Sc., Weizmann Institute of Science (2013)

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 2020

Author . . . .
Department of Electrical Engineering and Computer Science
August 28, 2020

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

Efficient Metric Representations for Big Data

by

Tal Wagner

Submitted to the Department of Electrical Engineering and Computer Science
on August 28, 2020, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Contemporary datasets are often represented as points in a high-dimensional metric space. To deal with increasingly larger datasets, many algorithms rely on efficient or compressed representations of the induced metric. In this thesis, we study several fundamental aspects of efficient metric representations. Our results include:

- Fully determining the minimal number of bits required to represent all distances, up to a given precision, in a finite Euclidean or Manhattan metric space.
- A space-efficient data structure for Euclidean approximate nearest neighbor search in high dimensions.
- A sublinear time algorithm for low-rank approximation of distance matrices, which is optimal in the number of entries it reads of the input matrix.
- A fast algorithm for nearest neighbor search in the Optimal Transport distance.

Previous bounds on Euclidean metric compression have been restricted to discretizing a classical dimensionality reduction theorem of Johnson and Lindenstrauss (1984). Our results improve over those bounds, thereby establishing an asymptotic advantage of generic sketching over dimension reduction. All of our algorithms are both proven analytically and implemented and validated empirically.

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

Acknowledgments

It is a pleasure to thank the people who had part in making this thesis possible.

To Piotr Indyk, my advisor, first and foremost — for sharing with me his vast knowledge and brilliant insights into algorithms, and for closely guiding me through research and research life. Piotr's immense expertise is only half the story; the other three halves are his kindness as a person, unflagging readiness to help, and deep sense of commitment to his students. I cannot overstate how lucky I've been to have a truly supportive advisor who always had my best interest in mind. Thanks, Piotr!

To Regina Barzilay, for her care and support throughout these years. Regina hosted me for a summer in MIT when I was an undergrad in Israel, and that exciting visit — which on my part was largely a whim — proved to be pivotal in my life, as it seeded the idea to return as a grad student. Ever since then she has been a constant source of help, advice and inspiration, her door always open, and I am deeply grateful for the role she played in my life here. Thank you for everything, Regina.

To my mentors and hosts outside MIT, for invaluable opportunities to learn from outstanding researchers in diverse settings. Uri Feige and Robi Krauthgamer gave me their guidance early on, and supported me through the transition to MIT. Nir Ailon, Robi and Ely Porat hosted me for visits in the Technion, Weizmann and Bar-Ilan. Sudipto Guha, Nina Mishra, Ilya Razenshteyn and Udi Wieder mentored me during summer internships which were both enjoyable and productive, and considerably expanded my perspective on research in data and algorithms. Ilya in particular has been a collaborator and friend since long before, always infectious with his vivid excitement about research, and my go-to person in many aspects of academic life. When I entered the program in MIT, he was officially appointed as my "buddy" by the CSAIL Buddy Program. Several years later, I am happy to report he has taken the role very seriously.

To the many others in MIT who were generous to lend me their time and support. I especially thank Costis Daskalakis and Ronitt Rubinfeld for serving on my thesis committee, Charles Leiserson for serving with Ronitt on my RQE committee, Nancy Lynch for serving as my academic advisor, and Janet Fischer and Rebecca Yadegar for expertly taking care of all administrative matters.

To my peers in Piotr's group: Arturs Backurs, Talya Eden, Sepideh Mahabadi, Shyam Narayanan, Ilya, Ludwig Schmidt, Sandeep Silwal, Ali Vakilian and Yang Yuan. Our group has been a pleasure to belong to, and a valuable source of peer support, collaborations and friendships (and saag paneer).

To my research collaborators: I've been privileged to have many, and our joint efforts were the most intense and exciting learning experiences of my PhD. In addition to all those already mentioned above, I would like to say thanks in particular to Noga Alon, Mike Dinitz, Yihe Dong, Shiva Kasiviswanathan, Krzysztof Onak, Daniel Reichman, Baruch Schieber, Igor Shinkar, and David Woodruff. Especially to Piotr, Ilya and David, for their collaboration and guidance on key parts of this thesis.

Finally, to the people who shared this road with me, near and far, and have made my time in MIT so happy to look back on. To the WHPs — Sirma, Yunus, Lea, Chris, David, Martin, Judith, Valerio, Lukas, Tugce, Andrew, Yagiz, Thras, Alex, Viirj, Ciki, Zoe, and all the others — for all the great times we've had together, on the alleys of New York, the ski slopes of Vermont, the lakes of Maine and the beaches of Barbados, but most of all in our daily lives around MIT. To Itay and Saleet, for our local branch of Tel-Aviv and a regular dosage of coffee and Hebrew. To Gali and Dan, for moral support, perspective, and crossword puzzles beyond count. To Maya, for making my trips to Seattle so much fun, and for so much more. And to my friends and family back home, and especially my parents Herzliya and Moshe, for their unending support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

From big data to high-dimensional metric spaces. The dominant paradigm in data analysis and machine learning is to represent large collections of real-world objects — like images, videos, or text — as vectors of many numerical features, viewed as points in a high-dimensional space. This allows us to quantify distances between those objects, forming the basis for subsequent learning and processing. Examples include:

- **Similarity search:** Given a new data object, we can search for the nearest objects to it in the dataset, and use what is known of them to infer knowledge on the new object.
- **Density estimation:** Given a new data object, we can estimate whether it is typical or anomalous by the density of objects around it, i.e., how many near objects we have seen in the past.
- **Clustering:** Given a large set of objects, we can group them into clusters such that objects in the same cluster are within small distance of each other.

This decades-old approach has seen dramatic developments in the last few years, due to the discovery that neural networks are capable of producing excellent high-dimensional embeddings of complex data domains. Still, while the particular means of generating those embeddings are now substantially different, the high-dimensional embedding paradigm remains prevalent and de-facto standard.

Along with its many benefits, this paradigm presents formidable computational challenges. Contemporary datasets become increasingly larger, and their embeddings require a large number of numerical features (i.e., high dimension) in order to obtain good results. Datasets with billions of points and hundreds of features are now common, posing significant requirements on processing time and memory.

Efficient metric representations. An overarching theme in tackling these challenges is *efficient representations* of the data. This approach is used across all areas of computing, under various nomenclatures — *compression*, *sketching*, *hashing*, *quantization*, and *sparsification*, to name a few — with far-reaching applications from fundamental theory to cutting-edge systems. In essence, the approach is to represent the data in a more *compact* way, that occupies less memory, and is often also *simpler* in a structural sense. Some benefits of efficient representations are the following:

- **Specialized hardware:** Accelerated computing devices, like graphical processing units (GPUs), are key to contemporary machine learning efforts, but have limited memory.
- **Edge devices:** End user devices, like handheld devices, are required to perform increasingly complex computations, while being limited in memory.
- **Communication:** In distributed and cloud architectures, the amount of data transferred between machines often poses a bottleneck.
- **Faster algorithms:** Efficient data representations often lead to faster running times, due to either their smaller size or their simplified structural form.

Beyond dimension reduction. The theory of metric compression is deeply rooted in the phenomenon of *Euclidean dimension reduction*, discovered in the seminal work of Johnson and Lindenstrauss [JL84]. They showed that all distances in a dataset can be captured in a *logarithmic* dimension, with arbitrarily good accuracy. This discovery has had a revolutionary impact on algorithms for big data, where dimensionality reduction techniques are by now standard and widely used. A major theme of this thesis is exploring the possibility to go beyond dimension reduction, and compress the data in more general ways.

1.1 Thesis Contributions

1.1.1 Overview

Metric compression and nearest neighbor search. Perhaps the most fundamental question on metric data compression is: *What is the minimal amount of space required to represent all distances between data points, up to a given precision?* Apart from its inherent mathematical interest, this question lies at the basis of a string of empirical successes in the applied fields of machine learning.

Our treatment of this question occupies the majority of this thesis. We resolve it fully for several important classes of metrics: Euclidean distances, Manhattan distances, and general metric spaces. We extend our results to support distances to new query points, resulting in the smallest known data structure for approximate nearest neighbor search. Our results mark the first improvement in storage size over techniques based on classical dimensionality reduction. As the latter techniques are known to be sharp, our work establishes that *compression is possible beyond dimension reduction.*

We proceed to implement a practical variant of our algorithm, which retains the asymptotic improvement over dimension reduction, while empirically matching or improving over the performance of state of the art heuristics.

Low-rank approximation of distance matrices. Distances can be naturally represented in tabular or *matrix* form, where each entry contains the distance between two points corresponding to its row and column. We present an algorithm for computing a low-rank approximation of such matrix, which is both faster and simpler than previous work, and performs better empirically. We also show that the algorithm is *optimal* in terms of the number of entries it reads of the input matrix.

Scalable nearest neighbor search for Optimal Transport. Learning in complex data domains, like text or images, often requires richer notions of distance. The Optimal Transport (OT) distance is an increasingly popular tool in these cases. However, its expressiveness comes at a high computational cost, hindering scalability. To alleviate this difficulty, we present a fast approximation algorithm for nearest neighbor search in the OT distance, based on an efficient random tree representation of the metric. It leads to significant speed up over state of the art search pipelines.

1.1.2 Metric Compression and Nearest Neighbor Search

Given a large dataset of a high-dimensional points in $\mathbb{R}^d$, many algorithms are based on the distances between the points. A prototypical example is *nearest neighbor search* (NNS), where given a query point, the goal is to return the closest point to it in the dataset. NNS is a cornerstone of machine learning, owing to its simplicity, accuracy and robustness [SDI06, AI17, Efr17].

To speed up NNS, a popular approach is to compute a *compressed representation* of the data. This approach has had remarkable success and spurred a long line of research, e.g., [Bro97, IM98, KOR00, SH09, WTF09, JDS11, WLKC16, WZS+18]. In a recent striking example [JDJ17a, JDJ17b], it has been demonstrated that compressing the data to fit on a GPU can lead to accurate NNS over billions of images.

This approach is tightly related to the phenomenon of Euclidean dimension reduction, discovered in the seminal work of Johnson and Lindenstrauss [JL84]. It states that, while exact distances between n points may require up to $n-1$ dimensions to represent, approximate distances require only $O(\log n)$ dimensions. This fundamental fact has far-reaching implications in theory [IM98, Mat08, AC09] and is manifestly useful in practice [Fod02, SVM14, CG15]. In particular, it lays formal foundations for metric compression, by proving that dramatic compression is possible with almost no loss in the accuracy of distance computation.

However, the most empirically successful compression algorithms go *beyond* dimension reduction: rather than representing points as points of lower dimension, they design more general and sophisticated bit representations, tailored to the dataset. These are often called “hashes” or “codes” [WLKC16, WZS+18]. This evident practical success raises the question: *is there a rigorous advantage to metric compression beyond dimension reduction?*

Optimal (Euclidean) metric compression. In Chapter 2, we study the problem in its basic form: given a dataset of n points in $\mathbb{R}^d$, with minimal distance 1 and maximal

	Bits per point	**No. query points**	**Query type**
Prior work	$O(\log^2 n)$	any q	distances
	$\Omega(\log^2 n)$	$q \geq n$	distances
This thesis	$\Theta(\log n)$	—	none
	$\Theta(\log n \cdot \log q)$	$q \leq n$	distances
	$O(\log n + \log q)$	any q	nearest neighbor

Table 1.1: Bound on metric compression with n data points and q query points, in a typical regime with relative error $\epsilon = \Omega(1)$, ambient dimension $d = n^{O(1)}$ and diameter $\Phi = n^{O(1)}$.

distance Φ, the goal is to represent all distances up to a relative error of ϵ in as little space as possible. The model is illustrated in Figure 1-1. For Euclidean metrics, a line of work based on discretization of the dimension reduction theorem of [JL84] has shown an upper bound of $O(\epsilon^{-2}\log(n)\log\Phi)$ bits per point [Ach03, AMS99, CCFC02, KOR00]. We prove a better bound, of $\Theta(\epsilon^{-2}\log n + \log\log\Phi)$ amortized bits per point, which is tight in all parameters. Our technique also leads to tight bounds for compression of ℓ_1 (Manhattan) metric spaces and of general metric spaces.

New query points and nearest neighbor search. In Chapter 3 we extend our compression technique to handle new query points that were known during the computation of the compressed representation. This is required, in particular, for nearest neighbor search. The query model is illustrated in Figure 1-2. Here too we obtain nearly tight bounds, improving over discretized dimension reduction. Our results are summarized in Table 1.1.

Practical and provable metric compression. In Chapter 4, we bridge the gap between the theory and practice of the above metric compression problem, by presenting a practical variant of our compression algorithm. On one hand, it attains nearly tight provable compression. On the other hand, we implement it and show it either matches or improves over state-of-the-art heuristics from the applied literature mentioned above.

These chapters are based on the following papers:

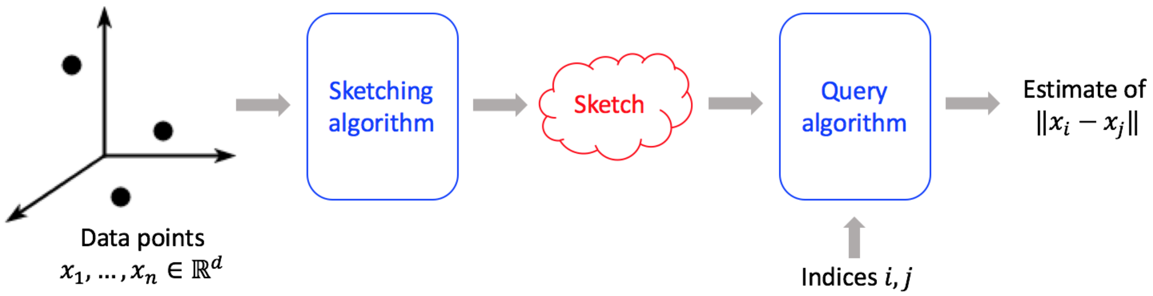


Figure 1-1: Metric compression

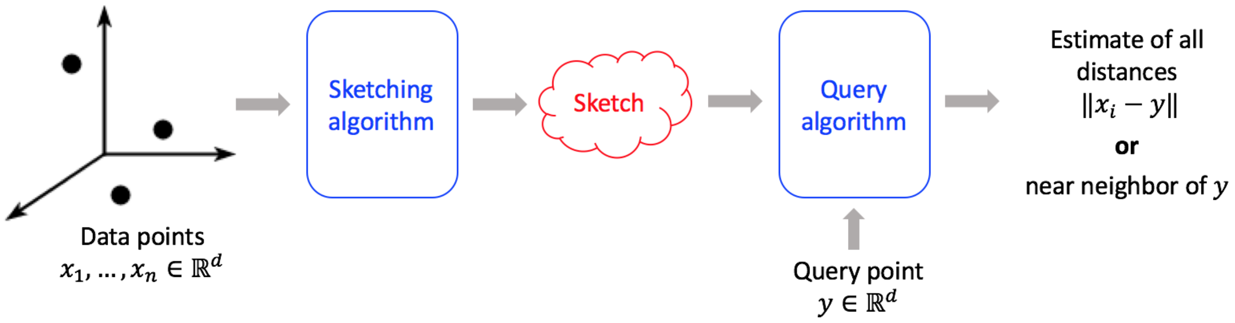


Figure 1-2: Metric compression with query points

- [IW17]: *Near-optimal (Euclidean) metric compression*, joint with Piotr Indyk. Appeared in the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017).

- [IRW17]: *Practical data-dependent metric compression with provable guarantees*, joint with Piotr Indyk and Ilya Razenshteyn. Appeared in the 31st Conference on Neural Information Processing Systems (NeurIPS 2017).

- [IW18]: *Approximate nearest neighbors in limited space*, joint with Piotr Indyk. Appeared in the 31st Annual Conference on Learning Theory (COLT 2018).

- [IW20]: *Optimal (Euclidean) metric compression*, joint with Piotr Indyk. In preparation (2020).

1.1.3 Low-Rank Approximation of Distance Matrices

Given a matrix $A \in \mathbb{R}^{m\times n}$, a low-rank approximation of A is a matrix $A' \in \mathbb{R}^{m\times n}$ of rank $k \ll \min\{n,m\}$ which is close to A. The most well-studied notion of closeness between matrices is the Frobenius norm error, $\|A - A'\|_F$. Low-rank approximations are widely used to speed up matrix-based algorithms: for example, while A takes $\Omega(mn)$ space to store and $\Omega(mn)$ time to multiply by a vector, A' requires only $\Omega(k(m+n))$ space and time. An optimal low-rank approximation can be computed in polynomial time using the Singular Value Decomposition (SVD), albeit of degree prohibitively high for large matrices. A long line of research (see the surveys [Mah11, Woo14]) has developed approximate algorithms that run in near-linear time in the matrix size. In particular, they output a matrix A' of rank k that satisfies the following guarantee:

$$\|A - A'\|_F^2 \leq \min_{A_k \text{ of rank } k} \|A - A_k\|_F^2 + \epsilon\|A\|_F^2.$$

While linear time is necessary for general matrices, sublinear time algorithms are possible for certain specialized classes of matrices. This includes *distance matrices*, where each entry i,j contains the distance between point i and point j in an associated metric space. Distance matrices are a natural notion that arises in various applications (see the survey [DPRV15]).

In Chapter 5, we give an algorithm for low-rank approximation of distance matrices that runs in time $\tilde{O}(m+n)\cdot \mathrm{poly}(k, 1/\epsilon)$ and achieves the above guarantee. The algorithm is both simpler and faster asymptotically than previous work, and is shown to perform well empirically. Furthermore, it reads $O((m+n)k/\epsilon)$ entries of the input matrix, which we prove is tight in all parameters – that is, any algorithm that achieves the above guarantee for distance matrices must read $\Omega((m+n)k/\epsilon)$ entries of the input matrix.

This chapter is based on the paper:

- [IVWW19]: *Sample-optimal low rank approximation of distance matrices*, joint with Piotr Indyk, Ali Vakilian and David Woodruff, with contributions to implementation by Ainesh Bakshi. Appeared in the 32nd Annual Conference on Learning Theory (COLT 2019).

1.1.4 Scalable Nearest Neighbor Search for Optimal Transport

The Optimal Transport (OT) distance, also known as the Earth Mover Distance (EMD) or the Wasserstein-1 distance, is a prominent notion of distance between sets of points or between distributions. In the discrete setting, the distance between two finitely supported distrbutions μ, ν over $\mathbb{R}^d$ is defined as the cost of the minimum-cost flow between their supports, where the cost of transport between every pair of points is equal to their Euclidean distance.

Machine learning applications of OT include classification of images and text documents. These applications rely on nearest neighbor search (NNS) with respect to the OT distance. Since computing even a single OT distance is a costly operation, which entails solving an optimization problem, large scale NNS has been a major challenge, and much work has focused on developing approximation algorithms. Existing algorithms largely fall into two categories: some return a rather crude approximation in linear time in the support size, while others return a very accurate approximation, albeit in quadratic time.

In Chapter 6, we give a linear time algorithm that generally achieves the same accuracy as quadratic time methods. It capitalizes on long line of research on approximating general metric spaces by random tree metrics. The latter metrics have a simplified form that renders many computational problems much more tractable. This approach had been applied to OT before, with good results. Nonetheless, we modify it in a crucial way that dramatically improves its accuracy, both in terms of its worst-case NNS approximation factor, and in terms of its empirical performance.

Specifically, the classical Quadtree algorithm finds an $O(\log(ns)\log(d\Phi))$-approximate nearest neighbor, where n is the dataset size, s is an upper bound on the support sizes, d is the dimension of the supported points, and Φ is their coordinate range. Our algorithm, Flowtree, finds an $O(\log s \cdot \log(d\Phi))$-approximate nearest neighbor — dispensing with the dependence on n, which is provably necessary for Quadtree. Furthermore, for uniform distributions, we can improve the analysis to show that Flowtree finds an $O(\log^2 s)$-approximate nearest neighbor. Our extensive experiments show that Flowtree speed-up state-of-the-art NNS systems for the OT distance by a large factor.

This chapter is based on the paper:

- [BDI+20]: *Scalable nearest neighbor search for Optimal Transport*, joint with Arturs Backurs, Yihe Dong, Piotr Indyk and Ilya Razenshteyn. Appeared in the 37th International Conference on Machine Learning (ICML 2020).

1.1.5 Other Results

This thesis encompasses a partial set of the results produced as part of the author's Ph.D. studies, in the interest of producing a concise and conherent document. Other results in the same vein include applications of efficient metric representations to streaming semi-supervised learning [WGKM18], kernel density estimation [BIW19], fair clustering [BIO+19], and nearest neighbor search assisted by neural networks [DIRW20], as well as compact representations of graph distances [DKW15, Wag20].

1.2 Preliminaries and Notation

Generally, notation and preliminaries will be introduced as we need them along the thesis. In this section we review some basic notions that will be used throughout.

General notation.

- For an integer $k > 0$, we use $[k]$ to denote the set $\{1, \ldots, k\}$.
- $\tilde{O}(f)$ denotes $O(f \cdot \mathrm{polylog}(f))$.
- All logs are in base 2.

1.2.1 Metric Spaces

A *metric space* is a pair (X, d), where X is any set and $\mathrm{d} : X \times X \to \mathbb{R}$ is a *distance function* that satisfies:

- $\mathrm{d}(x, x) = 0$ for every $x \in X$;

- $\mathrm{d}(x, y) > 0$ for every $x, y \in X$ such that $x \neq y$;
- $\mathrm{d}(x, y) = \mathrm{d}(y, x)$ for every $x, y \in X$;
- *(triangle inequality:)* $\mathrm{d}(x, y) \leq \mathrm{d}(x, z) + \mathrm{d}(z, y)$ for every $x, y, z \in X$.

The *aspect ratio* of a metric space is the ratio of largest to smallest distance:

$$\Phi = \frac{\max_{x,y \in X} \mathrm{d}(x, y)}{\min_{x,y \in X : x \neq y} \mathrm{d}(x, y)}.$$

Often it will be convenient to normalize the smallest distance to 1, and have Φ coincide with the *diameter* of the metric space.

Embedding and distortion. An *embedding* of a metric space (X, d) into a metric space (X', d') is a map $f : X \to X'$. We say that f has *distortion* $\phi \geq 1$ if there is $r > 0$ such that for every $x, y \in X$,

$$r \cdot \mathrm{d}(x, y) \leq \mathrm{d}'(f(x), f(y)) \leq \phi \cdot r \cdot \mathrm{d}(x, y).$$

If the distortion is $\phi = 1$ then (X, d) embeds *isometrically* in (X', d').

More generally, for any estimate $\widetilde{E}_{xy}$ of the distance $\mathrm{d}(x, y)$, we say it has distortion ϕ if

$$r \cdot \mathrm{d}(x, y) \leq \widetilde{E}_{xy} \leq \phi \cdot r \cdot \mathrm{d}(x, y).$$

By "distortion $1 \pm \epsilon$" we mean the special case where

$$(1 - \epsilon) \cdot \mathrm{d}(x, y) \leq \widetilde{E}_{xy} \leq (1 + \epsilon) \cdot \mathrm{d}(x, y).$$

ℓ_p-Metrics. For every $p \in [1, \infty)$, the ℓ_p-norm of $x \in \mathbb{R}^d$ is defined as

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

The ℓ_∞-norm is defined as

$$\|x\|_\infty = \max_{i \in [d]} |x_i|.$$

It is a standard fact that $\mathrm{d}(x,y) = \|x-y\|_p$ is a distance function on $\mathbb{R}^d$ for every $1 \le p \le \infty$ and dimension $d > 0$. The ℓ_2-distance is the *Euclidean* distance. The ℓ_1-distance is sometimes called the *Manhattan* distance, and in the special case where the point coordinates are in $\{0,1\}$, it is called the *Hamming* distance.

We say that (X, d) is an *ℓ_p-metric space* if it embeds isometrically into $\mathbb{R}^d$ with the ℓ_p-distance and some dimension d. That is, if there is a map $f : X \to \mathbb{R}^d$ such that

$$\mathrm{d}(x,y) = \|f(x) - f(y)\|_p \quad \text{for every} \quad x, y \in X.$$

If X is any finite set of size n, we can identify it with the set $[n]$ and denote $x_i = f(i)$. Thus, the ℓ_p-metric space is given by $x_1, \ldots, x_n \in \mathbb{R}^d$, with distances $\mathrm{d}(i,j) = \|x_i - x_j\|_p$ for every $i, j \in [n]$.

1.2.2 Johnson-Lindenstrauss Dimension Reduction

We now state the classical Euclidean dimension reduction theorem of Johnson and Lindenstrauss (often abbreviated as *JL*). In the next theorems, $N(0,1)$ denotes the standard Gaussian (normal) distribution, whose probability density function is $\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$.

Theorem 1.2.1 ([JL84]). *Let $x_1, \ldots, x_n \in \mathbb{R}^d$, $\epsilon, \delta \in (0,1)$, and $d' \ge c\epsilon^{-2} \log(n/\delta)$ for a sufficiently large constant $c > 0$. There is a distribution over matrices $M \in \mathbb{R}^{d' \times d}$ (for example, i.i.d. entries from $\frac{1}{\sqrt{d'}} \cdot N(0,1)$) such that with probability $1 - \delta$, for all $i, j \in [n]$,*

$$(1-\epsilon)\|x_i - x_j\|_2 \le \|Mx_i - Mx_j\|_2 \le (1+\epsilon)\|x_i - x_j\|_2.$$

A similar statement can be made for embedding ℓ_2 into ℓ_1:

Theorem 1.2.2 ($\ell_2 \to \ell_1$ JL). *Let $x_1, \ldots, x_n \in \mathbb{R}^d$, $\epsilon, \delta \in (0,1)$, and $d' \ge c\epsilon^{-2} \log(n/\delta)$ for a sufficiently large constant $c > 0$. There is a distribution over matrices $M \in \mathbb{R}^{d' \times d}$ (for example, i.i.d. entries from $\frac{1}{d'} \cdot N(0,1)$) such that with probability $1 - \delta$, for all $i, j \in [n]$,*

$$(1-\epsilon)\|x_i - x_j\|_2 \le \|Mx_i - Mx_j\|_1 \le (1+\epsilon)\|x_i - x_j\|_2.$$

The above theorem can be found, in various forms, in [Ind06, Mat08, AC09, MWY13].

Distributional variants. The JL theorem can also be stated in the following form, sometimes called *distributional JL*.

Theorem 1.2.3 (distributional JL)**.** *Let $\epsilon, \delta \in (0,1)$, and $d' \geq c\epsilon^{-2}\log(1/\delta)$ for a sufficiently large constant $c > 0$. There is a distribution over matrices $M \in \mathbb{R}^{d' \times d}$ (for example, i.i.d. entries from $\frac{1}{\sqrt{d'}} \cdot N(0,1)$) such that for every $x \in \mathbb{R}^d$,*

$$\Pr[(1-\epsilon)\|x\|_2 \leq \|Mx\|_2 \leq (1+\epsilon)\|x\|_2] \geq 1-\delta.$$

Theorem 1.2.4 (distributional $\ell_2 \to \ell_1$ JL)**.** *Let $\epsilon, \delta \in (0,1)$, and $d' \geq c\epsilon^{-2}\log(1/\delta)$ for a sufficiently large constant $c > 0$. There is a distribution over matrices $M \in \mathbb{R}^{d' \times d}$ (for example, i.i.d. entries from $\frac{1}{d'} \cdot N(0,1)$) such that for every $x \in \mathbb{R}^d$,*

$$\Pr[(1-\epsilon)\|x\|_2 \leq \|Mx\|_1 \leq (1+\epsilon)\|x\|_2] \geq 1-\delta.$$

Note that Theorems 1.2.1 and 1.2.2 follow from Theorems 1.2.3 and 1.2.4, respectively, by scaling δ down by $\binom{n}{2}$ and taking a union bound over all vectors $x_i - x_j$. Many known proofs of the JL theorem (e.g., [FM88, IM98, DG03, Ach03]) indeed take this approach and prove the distributional variant to derive the JL theorem.

1.2.3 Probabilistic Concentration

The following are standard concentration inequalities.

Fact 1.2.5 (Markov's inequality)**.** *Let X be a non-negative random variable. For every $c > 0$,*

$$\Pr[X \geq c \cdot \mathbb{E}[X]] \leq 1/c.$$

Lemma 1.2.6 (Hoeffding's inequality)**.** *Let $X_1, \ldots, X_n$ be random variables such that X_i is supported on the interval $[a_i, b_i]$. Let $X = \sum_{i=1}^n X_i$. For every $t > 0$,*

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

1.2.4 Universal Hashing

We review some standard notions and results on universal hashing.

Definition 1.2.7 (universal hash family)**.** *Let U be a an arbitrary set, and $m > 0$ an integer. A distribution over functions $H : U \to [m]$ is called a* universal hash family *if for all $u, u' \in U$,*

$$\Pr_H[H(u) = H(u') \leq \frac{|U|}{m}.$$

Claim 1.2.8. *Suppose $H : U \to [m]$ is sampled from a universal hash family. Let $k > 0$ be an integer, and let $\eta > 0$ be such that $m \geq 10k/\eta$. Let $S \subset U$ be of size at most k, and let $u \in S$. Then, with probability at least $1 - \eta$, u has no collisions with other elements in S, i.e.,*

$$\Pr_H[\exists_{u' \in S} \;\; s.t. \;\; H(u) = H(u')] < \eta.$$

Proof. By Definition 1.2.7, the expected number of collisions with u from S is at most k/m, which by hypothesis is at most $\eta/10$. By Markov's inequality, with probability at least $1 - \eta$ there are at most $1/10$ collisions, which means no collisions. $\square$

Theorem 1.2.9 ([CW79])**.** *There is an explicit universal family of hash functions $H : U \to [m]$ such that each function can be described by $O(\log |U|)$ bits.*

Chapter 2

Optimal (Euclidean) Metric Compression

We study the problem of representing all distances between n points in $\mathbb{R}^d$, with arbitrarily small distortion, using as few bits as possible. We settle it fully, up to constants, for Euclidean distances and for ℓ_1 (a.k.a. Manhattan) distances. Our bounds mark the first improvement over compression schemes based on the classical dimensionality reduction theorem of Johnson and Lindenstrauss [JL84], and settle an open problem from [AK16]. Since it is known that no better dimension reduction bound than [JL84] is possible, our results establish that *metric compression is possible beyond dimension reduction.*

2.1 Introduction

The metric sketching problem is defined as follows:

Definition 2.1.1 (metric sketching)**.** *Let* $1 \le p \le \infty$ *and* $0 < \epsilon < 1$. *In the* ℓ_p-metric sketching problem*, we are given a set of* n *points* $x_1, \ldots, x_n \in \mathbb{R}^d$ *with* ℓ_p*-distances in the range* $[1, \Phi]$*. We need to design a pair of algorithms:*

- *Sketching algorithm: given* $x_1, \ldots, x_n$*, it computes a bitstring called a* ***sketch****.*
- *Estimation algorithm: given the sketch, it can report for every* $i, j \in [n]$ *a distance*

estimate $\widetilde{E}_{ij}$ such that

$$(1-\epsilon)\|x_i - x_j\|_p \leq \widetilde{E}_{ij} \leq (1+\epsilon)\|x_i - x_j\|_p.$$

The goal is to minimize the bit length of the sketch.

Put simply, the goal is to represent all distances between $x_1, \ldots, x_n$, up to distortion $1 \pm \epsilon$, using as few bits as possible. The sketching algorithm can be randomized. In that case, we require that with probability $1 - 1/\text{poly}(n)$ it returns a sketch such that the requirement of the estimation algorithm is satisfied for all pairs $i, j \in [n]$ simultaneously. The estimation algorithm is generally deterministic.

We remark that the assumption on the distances being in $[1, \Phi]$ is essentially without loss of generality, by scaling. If $\min_{i \neq j} \|x_i - x_j\| = M$, we can store in the sketch a 2-approximation M' of M and scale all distances down by M'. This costs $O(\log \log M)$ additional bits globally. Then, in all bounds below, Φ becomes the *aspect ratio*, which is the ratio of largest to smallest distance in the given point set.

Euclidean metrics. The most notable case is Euclidean metrics, or $p = 2$. For this case, the Johnson-Lindenstrauss dimensionality reduction theorem [JL84] enables reducing the dimension of the input point set to $d' = O(\epsilon^{-2} \log n)$. This leads to a sketch of size $O(\epsilon^{-2} \log n)$ ***machine words*** per point. The ***bit size*** of the sketch generally depends on the numerical range of distances, encompassed by Φ (a typical setting to consider below is $\Phi = n^{O(1)}$).

For example, if the coordinates of the input points are integers in the range $[-\Phi, \Phi]$ (note that in this case the diameter is $O(\sqrt{d}\Phi)$), then the discretized variant of [JL84] due to Achlioptas [Ach03], and related algorithms like AMS sketch [AMS99] and CountSketch [CCFC02, TZ12], yield a sketch size of $O(\epsilon^{-2} \log(n) \log(d\Phi))$ bits per point. More generally, for any point set with diameter Φ (regardless of coordinate representation), the distance sketches of Kushilevitz, Ostrovsky and Rabani [KOR00] yield a sketch size of $O(\epsilon^{-2} \log(n) \log \Phi)$ bits per point. In a closely related model, Alon and Klartag [AK16] studied approximating squared Euclidean distances between points of norm at most 1, up to an *additive* error of ϵ (whereas distortion $1 \pm \epsilon$ is equivalent to *relative* error ϵ). For that problem, they proved a tight sketching bound of $O(\epsilon^{-2}\log n)$ bits per point.

Larsen and Nelson [LN17] recently showed that the dimension upper bound of [JL84] is tight. Previously, it had been known to be tight up to $\log(1/\epsilon)$ [Alo03]. However, these lower bounds do not apply to metric sketching, since a sketch does not have to be in the form of lower-dimensional points. Perhaps surprisingly, prior to our work, it had not been known whether the above "discretized JL" upper bound of $O(\epsilon^{-2}\log(n)\log\Phi)$ is tight for metric sketching, or can be improved much further. We show it is indeed not tight, by proving an improved and optimal bound of $O(\epsilon^{-2}\log n + \log\log\Phi)$ amortized bits per points. In light of the foregoing dimension reduction lower bound, our result establishes that sketching techniques can go beyond dimension reduction in compressing Euclidean metric spaces.

General metrics. The above formulation also captures sketching of *general metric spaces* — that is, the input is any metric space $([n], \mathrm{d})$ with distances between 1 and Φ — since they embed isometrically into ℓ_∞ with dimension $d = n$. Specificially, for every $i \in [n]$, one defines $x_i = (\mathrm{d}(i,1), \ldots, \mathrm{d}(i,n)) \in \mathbb{R}^n$. It is not hard to see that $\mathrm{d}(i,j) = \|x_i - x_j\|_\infty$ for every $i, j \in [n]$. General metric sketching has been studied extensively, under the name *distance oracles* [TZ05], in larger distortion regimes than $1 \pm \epsilon$, see Section 2.1.2. We provide tight bounds for distortion $1 \pm \epsilon$.

2.1.1 Our Results

We resolve the optimal sketching size with distortion $1 \pm \epsilon$ for several important classes of metrics: Euclidean metrics, ℓ_1 (a.k.a. Manhattan) metrics, and general metrics.

Theorem 2.1.2 (Euclidean metric compression). *For ℓ_2-metric sketching with n points (of arbitrary dimension) and distances in $[1, \Phi]$, $\Theta(\epsilon^{-2} n \log n + n \log\log\Phi)$ bits are both sufficient and necessary.*

The sketching algorithm is randomized and runs in time $O(n^{1+\alpha}\log\Phi + nd\log d + \epsilon^{-2} n \cdot \min\{d\log n, \log^3 n\})$, where d is the ambient dimension of the input metric, and $\alpha > 0$ is an arbitrarily small constant. The estimation algorithm runs in time $O(\epsilon^{-2}\log(n)\log(\epsilon^{-1}\Phi\log n))$.

Theorem 2.1.2 improves over the best previous bound of $O(\epsilon^{-2}n\log(n)\log\Phi)$, mentioned earlier.[1] It also strengthens the above mentioned upper bound of [AK16] for sketching with additive error, and resolves an open problem posed by them.

By known embedding results, both the upper and lower bound in Theorem 2.1.2 in fact holds for ℓ_p-metrics for every $1 \le p \le 2$, including the notable case ℓ_1. See Section 2.5.

We note that the sketching algorithm in the above theorem is randomized. This means that with probility $1/\text{poly}(n)$, it may output a sketch that distorts the distances by more than a $(1 \pm \epsilon)$ factor. However, this does not effect the sketch size nor the running time. We also remark that by "arbitrary dimension" we mean that the sketching scheme does not restrict the dimension of the input points. The lower bound generally holds for $d = \Omega(\epsilon^{-2}\log n)$.

General metrics. For general metric spaces, we give the following tight bounds.

Theorem 2.1.3 (General metric compression)**.** *For general metric sketching with n points and distances in $[1,\Phi]$, $\Theta(n^2\log(1/\epsilon) + n\log\log\Phi)$ bits are both sufficient and necessary.*

Note that storing all exact distances in a general metric takes at least $O(n^2\log\Phi)$ bits. Naïvely, one could round each distance to its nearest power of $(1+\epsilon)$, which yields a sketch of size $O(n^2\log(1/\epsilon)+n^2\log\log\Phi)$ bits. Theorem 2.1.3 improves the second term to $n\log\log\Phi$.

ℓ_p-metrics. Both of the theorems above are based on a more general upper bound, that holds for all ℓ_p-metrics.

Theorem 2.1.4 (ℓ_p-metric compression)**.** *Let $1 \le p \le \infty$. For ℓ_p-metric sketching with n points in dimension d and distances in $[1,\Phi]$, $O(n(d+\log n)\log(1/\epsilon)+n\log\log\Phi)$ bits are sufficient. The sketching algorithm is deterministic and runs in time $O(n^2\log\Phi + nd\log(1/\epsilon))$. The estimation algorithm runs in time $O(d\log(d\Phi))$ for $p<\infty$, and $O(d\log\Phi)$ for $p=\infty$.*

The upper bound of Theorem 2.1.3 follows immediately from Theorem 2.1.4, since as mentioned earlier, general metric spaces with n points embed isometrically into ℓ_∞ with

[1] We remark that naïvely rounding each coordinate of the dimension-reduced points to its nearest power of $(1+\epsilon)$ does not yield a valid sketch. For example, consider two coordinates with values t and $t+1$, where $t = (1+\epsilon)^i$ for some integer i. The squared different between them is 1, whereas after rounding it becomes 0, and the distortion is unbounded.

dimension $d = n$. Similarly, for Euclidean metrics, one can apply the Johnson-Lindenstrauss transform as a preprocessing step in order to reduce the dimension of the input points to $O(\epsilon^{-2}\log n)$, and then apply Theorem 2.1.4. This gives an upper bound looser than that of Theorem 2.1.2 by $O(\log(1/\epsilon))$. To obtain the tight bound, we will use additional properties special to Euclidean metrics.

2.1.2 Additional Related Work

Distance oracles. The distance oracle problem [TZ05] is equivalent to sketching of general metrics, and has been studied in a different distortion regime. A long line of work (including [PS89, ADD+93, Mat96, TZ05, WN12, Che15] and more) has shown that for every integer $k \geq 1$, it is possible to compute a sketch of size $\tilde{O}(n^{1+1/k})$ with distortion $2k-1$, which is tight up to logarithmic factors under the Erdős Girth Conjecture. Notably, for distortion 3 and above, the sketch size is $o(n^2)$. (However, note that in order to achieve a near-linear sketch size, the distortion must be almost logarithmic.) On the other hand, for any distortion less than 3, it is not hard to show (by considering all shortest-path metrics induced by bipartite simple graphs) that a sketch size of $\Omega(n^2)$ is necessary. For distortion $1 \pm \epsilon$, to our knowledge, the best upper bound prior to our work had been $O(n^2(\log\log\Phi + \log(1/\epsilon)))$ bits, which follows from naïve rounding as mentioned above.

Lower bounds. The papers [JW13, MWY13] prove lower bounds for metric sketching in a related but different model (that we consider in the next chapter of this thesis). In particular, they show that $\Omega(\epsilon^{-2} n \log(n/\delta)\log\Phi)$ bits are needed in order to sketch the ℓ_1 or ℓ_2 distances between two sets of n points with probability $1-\delta$. This lower bound matches the "discretized Johnson-Lindenstrauss" upper bound mentioned above [Ach03, AMS99, CCFC02, KOR00]. However, their model is different in that the sketching algorithm only sees half of the n input point set while computing the sketch, whereas the other half are only seen by the estimation algorithm. In our model, the sketching algorithm sees the whole input point set. We consider the unseen point model in Chapter 3.

2.1.3 Technical Overview

The basic strategy in the sketch is to store each point by its relative location to a nearby point which had already been (approximately) stored. More precisely, let $X = \{x_1, \dots, x_n\}$ be the point set we wish to sketch. For every point $x \in X$, we aim to define a *surrogate* $s^*(x) \in \mathbb{R}^d$, which is an approximation of x that can be efficiently stored in the sketch. To this end, we choose an *ingress* point $in(x) \in X$ near x, and define $s^*(x)$ inductively by its location relative to $s^*(in(x))$, namely $s^*(x) = s^*(in(x)) + [x - s^*(in(x))]_\gamma$, where $[y]_\gamma$ denotes rounding y to a γ-net, with an appropriate precision γ. We then hope to use the distance between the surrogates, $\|s^*(x_i) - s^*(x_j)\|$, as an estimate for the distance $\|x_i - x_j\|$ for all pairs $i, j \in [n]$. The challenge is to choose the ingresses and the precisions in a way that on one hand ensures a small relative error estimate for each pair, while on the other hand does not occupy too many storage bits.

In order to ensure a relative error approximation of every distance, we need to consider all possible distance scales. To this end we construct a hierarchical clustering tree of the metric space, and define the ingresses and surrogates for clusters (or tree nodes) instead of individual points. Here, it may seem natural to use separating decomposition trees such as [Bar96, CCG⁺98, FRT04], which provide both a separating property (far points are in different clusters) and a packing property (close points are often in the same cluster). However, such trees are bound to incur a super-constant gap between the two properties [Bar96, Nao17], which would lead to suboptimal sketch size. Instead, our tree transitively merges any two clusters within a sufficiently small distance. This yields a perfect separation property, but no packing property — the diameter of each cluster may be unbounded. We replace it by a global bound on all cluster diameters in the tree (Lemma 2.3.2).

The tree size is first reduced to linear by compressing long non-branching paths. From a distance estimation point of view, this means that if a cluster is very well separated from the rest of the metric, then we can replace it entirely with one representative point (called *center*) for the purpose of estimating the distances between internal and external points. Then, the crucial step is a careful choice of the ingresses, that ensures that if we set the precisions so as to get correct estimates between all pairs, the total sketch occupies sufficiently few bits.

This completes the description of the data structure, which we call *relative location tree.*

In order to estimate the distance $\|x_i - x_j\|$ for a given pair $i, j \in [n]$, we can identify in the tree two nodes v_i, v_j, such that (i) the center of v_i is a sufficiently good proxy for x_i from the point of view of x_j, and vice-versa, (ii) the error between the center of v_i and its surrogate is proportional to $\epsilon \cdot \|x_i - x_j\|$, and the same holds for v_j, and (iii) the surrogates of v_i and v_j can be recovered from the sketch (by following ingresses along the tree) up to a shift, which while unknown, is the same for both. Then we may return the distance between the shifted surrogates as the output distance estimate.

Euclidean metrics. The above outline describes our upper bound for sketching ℓ_p-metrics. However, for Euclidean metrics, the resulting sketch size is suboptimal in the dependence on ϵ. To achieve the optimal bound we further develop the above sketch.

The obstacle is that in order to get optimal dependence on ϵ, we cannot afford to deterministically round a point to an ϵ-near point. In the relative location tree, this comes up in two places: in rounding the displacement (i.e., the location relative to an ingress) to an ϵ-net when defining a surrogate, and in clumping a well-separated cluster, from the point of view of external points, to one representative point.

Alon and Klartag [AK16] showed that in order to achieve the optimal dependence on ϵ as an *additive* error, one can use randomized rounding to a grid that acts as a $\Omega(1)$-net rather than an ϵ-net, and rely on probabilistic concentration to achieve ϵ-closeness with high probability. To use this approach toward *relative* error, we incorporate it into our techniques described above. We build a relative location tree with $\epsilon = \Omega(1)$; this does not exceed the optimal sketch size for Euclidean metrics, but does not provide the desired approximation of distances. We then augment it with randomized rounding of displacement vectors between some nodes to their surrogates, and between each representative point of a well-separated cluster to some other points in the cluster. This allows us to recover a notion of *probabilistic surrogates* from the sketch. Specifically, in order to estimate the distance $\|x_i - x_j\|$ for a given pair $i, j \in [n]$, we sum an appropriate subset of those roundings for i and for j. This yields two random variables X_i, X_j, each supported on a hypercube with side length proportional to $\frac{1}{\sqrt{d}}\|x_i - x_j\|$, with respective expected values x_i, x_j, up to an unknown but similar shift.

For technical reasons related to probabilistic independence, we return a proxy of the distance $\|X_i - X_j\|$ rather than the distance itself, and the result is tightly concentrated at the correct value $\|x_i - x_j\|$.

2.2 Preliminaries: Grid Nets

Let $1 \leq p \leq \infty$. Let $\mathcal{B}_p^d = \{x \in \mathbb{R}^d : \|x\|_p \leq 1\}$ denote the d-dimensional ℓ_p-unit ball. Let $\gamma > 0$. A subset $N \subset \mathbb{R}^d$ is called a γ-*net* of $\mathcal{B}_p^d$ if for every $x \in \mathcal{B}_p^d$ there is $y \in N$ such that $\|x - y\|_p \leq \gamma$. It is a well-known fact that $\mathcal{B}_p^d$ has a γ-net of size $(c/\gamma)^d$ for a constant $c > 0$, and that the size bound is tight up to the constant c.

We will use a specific net, given by the intersection of the ball with an appropriately scaled grid. For $\rho > 0$, let $\mathcal{G}^d[\rho] \subset \mathbb{R}^d$ denote the uniform d-dimensional grid with cell side length ρ, which is the set of points $\mathbb{R}^d$ that each of their coordinate is an integer multiple of ρ.

The net we use is $\mathcal{N}_\gamma = 2 \cdot \mathcal{B}_p^d \cap \mathcal{G}^d[\gamma/d^{1/p}]$ (where $2 \cdot \mathcal{B}_p^d$ is the origin-centered ball of radius 2) . We drop the dependence on d and p from the notation $\mathcal{N}_\gamma$ for simplicity. Also, in the case $p = \infty$, we use $d^{1/p} = 1$ as a convention.

It is easily seen that each cell of $\mathcal{N}_\gamma$ is a hypercube of diameter γ, and it is indeed a γ-net of $\mathcal{B}_p^d$. It is also well-known that it has asymptotically optimal size, meaning that $|\mathcal{N}_\gamma| = (c'/\gamma)^d$ for a constant $c' > 0$ (see, e.g., [HPIM12] or [AK16]). Finally, given $x \in \mathcal{B}_p^d$, we can find $y \in \mathcal{N}_\gamma$ such that $\|x - y\|_p \leq \gamma$ by dividing each coordinate by $\gamma/d^{1/p}$, rounding it to the largest smaller integer, and multiplying it by $\gamma/d^{1/p}$. We call this operation *rounding x to $\mathcal{N}_\gamma$*. In summary,

Lemma 2.2.1. *For every $x \in \mathcal{B}_p^d$, we can round it to $\mathcal{N}_\gamma$ in time $O(d)$, and store the resulting point of $\mathcal{N}_\gamma$ with $O(d \log(1/\gamma))$ bits.*

We also record another variant of the above lemma.

Fact 2.2.2. *Let $x \in \mathbb{R}^d$ and $\gamma > 0$. The number of points in $\mathcal{G}^d[\gamma/d^{1/p}]$ which are at distance at most 2γ from x (in the ℓ_p-norm distance) is $O(1)^d$.*

2.3 The Relative Location Tree

In this section we prove Theorem 2.1.4, which implies the upper bound in Theorem 2.1.3, and will also serve as a stepping stone toward Theorem 2.1.2. The sketching scheme is based on a new data structure that we call *relative location tree*.

Let $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ be a given point set endowed with the ℓ_p-metric for a fixed $1 \le p \le \infty$, with minimal distance 1 and diameter Φ. We assume w.l.o.g. that Φ is an integer. To simplify notation, we drop the subscript p from ℓ_p-norms (that is, we write $\|x - y\|$ for $\|x - y\|_p$).

2.3.1 Hierarchical Tree Construction

We start by building a hierarchical clustering tree T^* over the points X, by the following bottom-up process. In the bottom level, numbered 0, every point x_i forms a singleton cluster $\{x_i\}$. Level $\ell > 0$ is generated from level $\ell - 1$ by merging any clusters at distance less than 2^ℓ, until no such pair remains. (The distance between two clusters $C, C' \subset X$ is defined as $\mathrm{dist}(C, C') = \min_{x \in C, x' \in C'} \|x - x'\|$.) By level Φ, the pointset has been merged into one cluster, which forms the root of the tree.

For every tree node v in T^*, we denote its level by $\ell(v)$, its associated cluster by $C(v) \subset X$, its cluster diameter by $\Delta(v)$, and its degree (number of children) by $\deg(v)$. For every $x_i \in X$, let $\mathrm{leaf}(x_i)$ denote the tree leaf whose associated cluster is $\{x_i\}$.

Note that the nodes at each level of T^* form a partition of X. On one hand, we have the following separation property.

Claim 2.3.1. *If x_i, x_j are at different clusters of the partition induced by level ℓ, then $\|x_i - x_j\| \ge 2^\ell$.*

On the other hand, we have the following global bound on the cluster diameters.

Lemma 2.3.2. $\sum_{v \in T^*} 2^{-\ell(v)} \Delta(v) \le 4n$.

Proof. We write uv to denote an edge from a parent u to a child v. We call it a 1-*edge* if $\deg(u) = 1$, and a *non*-1-*edge* otherwise. Note that since T^* has n leaves, it has at most $2n$ non-1-edges. We define edge weights and node weights in T^* as follows. The weight of an edge uv is $\mathrm{wt}(uv) = 0$ if uv is a 1-edge, and $\mathrm{wt}(uv) = 2^{\ell(u)}$ otherwise. The weight of a node v, denoted $\mathrm{wt}(v)$, is the sum of all edge weights in the tree under v (that is, $\mathrm{wt}(v) = \sum_{uu'} \mathrm{wt}(uu')$ where the sum is over all edges uu' such that u is reachable from v by a downward path in T^*).

We argue that $\Delta(v) \leq \mathrm{wt}(v)$ for every node v. This is seen by bottom-up induction on T^*. In the base case v is a leaf, and then $\Delta(v) = \mathrm{wt}(v) = 0$. For the induction step, fix a node u and consider two cases. In the first case, u has degree 1 and a single outgoing 1-edge uv. Then $C(u) = C(v)$ by the tree construction, and $\mathrm{wt}(u) = \mathrm{wt}(v)$ since $\mathrm{wt}(uv) = 0$, thus the claim follows by induction. In the second case u has multiple outgoing edges $\{uv_i : i = 1, \ldots, k\}$. Since $\{C(v_i) : i = 1, \ldots, k\}$ is a partition of $C(u)$, the diameter $\Delta(u)$ is upper-bounded by $\sum_{i=1}^{k}(\Delta(v_i) + \mathrm{dist}(C(v_i), C(u) \setminus C(v_i))$. By induction, $\Delta(v_i) \leq \mathrm{wt}(v_i)$ for every i. By the tree construction, $\mathrm{dist}(C(v_i), C(u) \setminus C(v_i)) \leq 2^{\ell(u)} = \mathrm{wt}(uv_i)$. Together, $\Delta(u) \leq \sum_{i=1}^{k}(\mathrm{wt}(v_i) + \mathrm{wt}(uv_i)) = \mathrm{wt}(u)$, as needed.

Consequently, it now suffices to prove the bound $\sum_{v\in T} 2^{-\ell(v)}\mathrm{wt}(v) \leq 4n$. To this end we count the contribution of each edge to the sum. A 1-edge has no contribution since its weight is 0. For a non-1-edge uv, let $u_0 = u$, and let u_i be the parent of u_{i-1} for all $i > 0$ until the root is reached. Then uv contributes its weight $2^{\ell(u)}$ to $\mathrm{wt}(u_i)$ for every $i \geq 0$, and its total contribution is $\sum_{i\geq 0} 2^{-\ell(u_i)} \cdot 2^{\ell(u)}$. Since $\ell(u_i) = \ell(u) + i$, the latter sum equals $\sum_{i\geq 0} 2^{-i} < 2$. Since T^* has at most $2n$ non-1-edges, the desired bound follows. $\square$

Path compression. Next, we compress long non-branching paths in T^*. A 1-*path* in T^* is a downward path $v_0, v_1, \ldots, v_k$ such that $v_1, \ldots, v_{k-1}$ are degree-1 nodes. It is called *maximal* if v_0 and v_k are not degree-1 nodes (v_k may be a leaf). For every such path in T^*, if $k > \log(2^{-\ell(v_k)}\Delta(v_k)/\epsilon)$, we replace the path from v_1 to v_k with a *long edge* directly connecting v_1 to v_k. We mark it as long and annotate it with the original path length, k. The rest of the edges are called *short edges*. The tree after path compression will be denoted by T. We note that $\ell(v)$ will continue to denote the original level of v in T^* (or equivalently, the level in T if the long edges are counted according to their lengths).

Lemma 2.3.3. *$\sum_{v\in T} \log\left(2^{-\ell(v)}\Delta(v)\right) \leq 4n$.*

Proof. Follow from Lemma 2.3.2 since every node v in T is also present in T^* (with the same level $\ell(v)$ and associated cluster diameter $\Delta(v)$), and since $\log(z) < z$ for all $z \in \mathbb{R}$. $\square$

Lemma 2.3.4. *T has at most $2n(2 + \log(1/\epsilon))$ nodes.*

Proof. We charge the degree-1 nodes on every maximal 1-path in T to the bottom node of the path. The total number of nodes in T can then be written as $\sum_{v:\deg(v)\neq 1} k(v)$, where $k(v)$ is the length of the maximal 1-path whose bottom node is v. Due to path compression, we have $k(v) \leq \log(2^{-\ell(v)}\Delta(v)) + \log(1/\epsilon)$. Since T has n leaves, it has at most $2n$ nodes whose degree is not 1, so the total contribution of the second term is at most $2n\log(1/\epsilon)$. For the total contribution of the first term, we need to show $\sum_{v:\deg(v)\neq 1} \log\left(2^{-\ell(v)}\Delta(v)\right) \leq 4n$. This is given by Lemma 2.3.3. $\square$

Subtrees. We partition T into *subtrees* by removing the long edges. Let $\mathcal{F}(T)$ denote the set of resulting subtrees. Furthermore let $\mathcal{L}(T)$ denote the set of nodes of T which are leaves of subtrees in $\mathcal{F}(T)$. Note that a node in $\mathcal{L}(T)$ is either a leaf in T or the top node of a long edge in T. These nodes are special in that they represent clusters whose diameter can be bounded individually.

Lemma 2.3.5. *For every $u \in \mathcal{L}(T)$, $\Delta(u) \leq 2^{\ell(u)}\epsilon$.*

Proof. If u is a leaf in T then $C(u)$ contains a single point, thus $\Delta(v) = 0$, and the lemma holds. Otherwise, u is the top node of a long edge in T. Let v be the bottom node of that edge. By path compression, the long edge represents a 1-path of length at least $\log(2^{-\ell(v)}\Delta(v)/\epsilon)$, hence $\ell(u) \geq \ell(v) + \log(2^{-\ell(v)}\Delta(v)/\epsilon)$, and hence $2^{\ell(u)} \geq 2^{\ell(v)+\log(2^{-\ell(v)}\Delta(v)/\epsilon)} = \epsilon^{-1}\Delta(v)$. Since no clusters are merged along a 1-path, we have $C(u) = C(v)$, hence $\Delta(u) = \Delta(v)$, and the lemma follows. $\square$

2.3.2 Tree Annotations: Centers, Ingresses, and Surrogates

We now augment T with annotations that would efficiently encode information on the location of its clusters.

Centers. For every node v in T we choose a *center* from the points in its cluster $C(v)$, in a bottom-up manner, as follows. For a leaf $v = \text{leaf}(x_i)$, let $c(v) = i$. For a non-leaf v with children $u_1, \ldots, u_k$, let $c(v) = \min\{c(u_i) : i \in [k]\}$. The point $x_{c(v)}$ is the center of v.

Ingresses. Next, for every node v in T we assign an *ingress* node, denoted $in(v)$. Intuitively, the ingress is a node in T such that $x_{c(in(u))}$ is close to $x_{c(u)}$, and our eventual purpose is to store the latter by its location relative to the former.

The ingresses are defined in each subtree $T' \in \mathcal{F}(T)$ separately. For the root r of T', we set $in(r) = r$ for convenience (as we will not require ingresses for subtree roots). Now we assign ingresses to all children of every node v in T', and this would take care of the rest of the nodes in T'. Let $u_1, \ldots, u_k$ be the children of v, such that w.l.o.g. $c(v) = c(u_1)$. Consider the simple graph H_v whose nodes are $u_1, \ldots, u_k$, where u_i, u_j are neighbors iff $\text{dist}(C(u_i), C(u_j)) \leq 2^{\ell(v)}$. The fact that $C(u_1), \ldots, C(u_k)$ have been merged into $C(v)$ in the tree construction means that H_v is a connected graph. Fix an arbitrary spanning tree τ_v of H_v and root it at u_1. For u_1, the ingress is $in(u_1) = v$. For u_i with $i > 1$, let u_j be its parent node in τ_v. Let $x \in C(u_j)$ be the closest point to $C(u_i)$ in $C(u_j)$ (i.e., $x = \text{argmin}_{x' \in C(u_j)} \min_{x'' \in C(u_i)} \|x' - x''\|$). Let $u_x \in \mathcal{L}(T)$ be the leaf of T' whose cluster contains x. The ingress of u_i is $\text{in}(u_i) = u_x$. See Figure 2-1 for illustration.

(Note that there is a downward path in T from u_j to leaf(x), and u_x is the bottom node on that path that belongs to T'. Equivalently, u_x is the bottom node on the path that is reachable from u without traversing a long edge.)

The following lemma bounds the distance between a node center and its ingress center.

Lemma 2.3.6. *For every node u in T, $\|x_{c(u)} - x_{c(in(u))}\| \leq 3 \cdot 2^{\ell(u)} + \Delta(u)$.*

Proof. Fix a subtree $T' \in \mathcal{F}(T)$. If u is the root of T', the claim is obvious since $u = in(u)$. Next, using the same notation as above, we prove the claim for all children $u_1, \ldots, u_k$ of a given node v in T'. For u_1 we have $c(in(u_1)) = c(v) = c(u_1)$, and the claim holds. For u_i with $i > 1$, recall that u_j denotes its ancestor in τ_v, and that x is a point in $C(u_j)$ that realizes the distance $\text{dist}(C(u_i), C(u_j))$, which is upper-bounded by $2^{\ell(v)}$. Therefore,

$$\|x_{c(u_i)} - x\| \leq \text{dist}(\{x\}, C(u_i)) + \Delta(u_i) \leq 2^{\ell(v)} + \Delta(u_i).$$

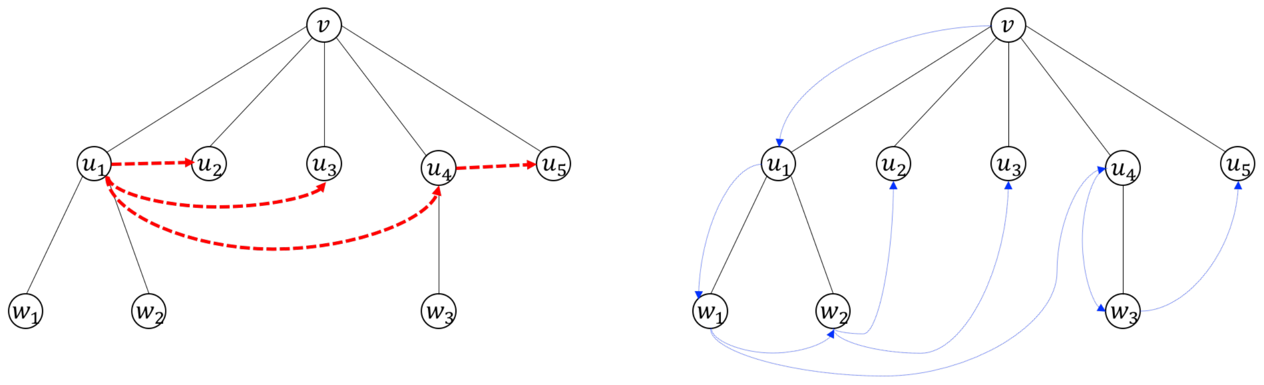


Figure 2-1: Choice of ingresses. Left: The black tree is a subtree $T' \in \mathcal{F}(T)$, rooted at v, with $c(v) = c(u_1) = c(w_1)$. The dashed red arrows form a spanning tree τ_v on the children of v. Right: The blue dotted arrows denote the ingresses in T' (each arrow source is the ingress of the arrow destination), in the case that $C(w_1)$ contains the point closest to $x_{c(u_4)}$ in $C(u_1)$ (thus w_1 is chosen as the ingress of u_4), and that $C(w_2)$ contains the point closest to $x_{c(u_2)}$ in $C(u_1)$ and the point closest to $x_{c(u_3)}$ in $C(u_1)$ (thus w_2 is chosen as the ingress of u_2 and u_3).

Noting that $\ell(v) = \ell(u_i) + 1$, we find

$$\|x_{c(u_i)} - x\| \leq 2 \cdot 2^{\ell(u_i)} + \Delta(u_i). \tag{2.1}$$

Recall that $in(u_i)$ was chosen as the leaf in T' whose cluster contains x. In particular, $x_{c(in(u_i))}$ and x are both contained in $C(in(u_i))$. By Lemma 2.3.5, $\|x_{c(in(u_i))} - x\| \leq 2^{\ell(in(u_i))}$. Since $in(u_i)$ is a descendant of a sibling of u_i, we have $\ell(in(u_i)) \leq \ell(u_i)$, hence $\|x_{c(in(u_i))} - x\| \leq 2^{\ell(u_i)}$. Combined with Equation (2.1), this implies the lemma by the triangle inequality. $\square$

We also record the following fact.

Claim 2.3.7. *For every node u in T, $\ell(in(u)) \leq \ell(u) + 1$.*

Proof. The ingress is either u itself, the parent of u in T, or a descendant of the parent. $\square$

Ingress ordering. The nodes in every subtree $T' \in \mathcal{F}(T)$ can be ordered such that every node appears after its ingress (except the root, which is its own ingress, and would be first in the ordering). Such ordering is given by a depth-first scan (DFS) on T', in which additionally, the children of every node v are traversed in a DFS order on τ_v. Since the ingress of every non-root node is either its parent in T', or a descendant of the sibling in T' which is its predecessor in τ_v, this ordering has the desired property. This will be important since the rest of the proof utilizes induction on the ingresses.

Surrogates. Now we can define the surrogates. We start by defining a *coarse surrogate* $s^*(v)$ for every node v in T. They are defined in every subtree $T' \in \mathcal{F}(T)$ separately, by induction on the ingress order in T'. For the root v of T', we let $s^*(v) = x_{c(v)}$. For a non-root v in T', we denote

$$\gamma(v) := \left(5 + \left\lceil \frac{\Delta(v)}{2^{\ell(v)}} \right\rceil \right)^{-1}, \tag{2.2}$$

and

$$\eta^*(v) = \frac{\gamma(v)}{2^{\ell(v)}} \cdot (x_{c(v)} - s^*(in(v))). \tag{2.3}$$

Let $\eta(v)$ be the rounding of $\eta^*(v)$ to the grid net $\mathcal{N}_{\gamma(v)}$ (see Section 2.2). By this we mean that $\eta(v)$ is obtained by rounding each coordinate of $\eta^*(v)$ to the largest smaller integer multiple of $\gamma(v)$. We define $s^*(v)$, by induction on $s^*(in(v))$, as

$$s^*(v) = s^*(in(v)) + \frac{2^{\ell(v)}}{\gamma(v)} \cdot \eta(v). \tag{2.4}$$

The following lemma bounds the distance between a node center and its surrogate.

Lemma 2.3.8. *For every v in T, $\|x_{c(v)} - s^*(v)\| \leq 2^{\ell(v)}$.*

Proof. By induction on the ingress ordering in the subtree $T' \in \mathcal{F}(T)$ that contains v. In the base case, v is the root and the claim holds trivially since $s^*(v) = x_{c(v)}$. For a non-root v, we have $\|x_{c(in(v))} - s^*(in(v))\| \leq 2^{\ell(in(v))} \leq 2 \cdot 2^{\ell(v)}$, where the first inequality is by induction on the ingress and the second is by Claim 2.3.7. By Lemma 2.3.6 we have $\|x_{c(v)} - x_{c(in(v))}\| \leq 3 \cdot 2^{\ell(v)} + \Delta(v)$, and together, by the triangle inequality, $\|x_{c(v)} - s^*(in(v))\| \leq 5 \cdot 2^{\ell(v)} + \Delta(v)$. By Equations (2.2) and (2.3), this implies $\|\eta^*(v)\| \leq 1$. Now, since $\mathcal{N}_{\gamma(v)}$ is a $\gamma(v)$-net for the unit ball, we have $\|\eta^*(v) - \eta(v)\| \leq \gamma(v)$. Finally,

$$\begin{aligned}
\|x_{c(v)} - s^*(v)\| &= \|x_{c(v)} - s^*(in(v)) - \tfrac{2^{\ell(v)}}{\gamma(v)} \cdot \eta(v)\| && \text{Equation (2.4)} \\
&= \|x_{c(v)} - s^*(in(v)) - \tfrac{2^{\ell(v)}}{\gamma(v)} \cdot (\eta^*(v) - \eta^*(v) + \eta(v))\| \\
&= \|\tfrac{2^{\ell(v)}}{\gamma(v)} \cdot (\eta(v) - \eta^*(v))\| && \text{Equation (2.3)} \\
&\leq 2^{\ell(v)}. && \square
\end{aligned}$$

Leaf surrogates. For every subtree leaf $v \in \mathcal{L}(T)$ we also use a finer surrogate $s^*_\epsilon(v)$, called *leaf surrogate*. To this end, let $\eta_\epsilon(v)$ be the rounding of $\eta^*(v)$ to the grid net $\mathcal{N}_{\gamma(v)\cdot\epsilon}$, where $\gamma(v)$ and $\eta^*(v)$ are the same as before. The leaf surrogate is defined as

$$s^*_\epsilon(v) = s^*(in(v)) + \frac{2^{\ell(v)}}{\gamma(v)} \cdot \eta_\epsilon(v). \tag{2.5}$$

Note that $s^*(in(v))$ is the surrogate of $in(v)$ defined earlier (the definition of $s^*_\epsilon(v)$ is not inductive.)

Lemma 2.3.9. *For every $v \in \mathcal{L}(T)$, $\|x_{c(v)} - s^*_\epsilon(v)\| \leq 2^{\ell(v)} \cdot \epsilon$.*

Proof. The proof of Lemma 2.3.8 showed that $\|\eta^*(v)\| \leq 1$. Hence, as $\mathcal{N}_{\gamma(v)\cdot\epsilon}$ is a $(\gamma(v)\cdot\epsilon)$-net for the unit ball, we have $\|\eta^*(v) - \eta_\epsilon(v)\| \leq \gamma(v) \cdot \epsilon$. Thus,

$$\begin{aligned}
\|x_{c(v)} - s^*_\epsilon(v)\| &= \|x_{c(v)} - s^*(in(v)) - \tfrac{2^{\ell(v)}}{\gamma(v)} \cdot \eta_\epsilon(v)\| && \text{Equation (2.5)} \\
&= \|x_{c(v)} - s^*(in(v)) - \tfrac{2^{\ell(v)}}{\gamma(v)} \cdot (\eta^*(v) - \eta^*(v) + \eta_\epsilon(v))\| \\
&= \|\tfrac{2^{\ell(v)}}{\gamma(v)} \cdot (\eta_\epsilon(v) - \eta^*(v))\| && \text{Equation (2.3)} \\
&\leq 2^{\ell(v)}\epsilon. && \square
\end{aligned}$$

2.3.3 Sketch Size

The sketch stores the tree T, with the following annotations. For each edge we store whether it is long or short, and for the long edges we store their original lengths. For each node v we store the center label $c(v)$, the ingress label $in(v)$, the precision $\gamma(v)$, and the element $\eta(v)$ of the $\gamma(v)$-net $\mathcal{N}_{\gamma(v)}$. For every node v in $\mathcal{L}(T)$ we also store $\eta_\epsilon(v)$, which is an element of the $(\gamma(v)\cdot\epsilon)$-net $\mathcal{N}_{\gamma(v)\cdot\epsilon}$. This completes the description of the relative location tree. We now bound the total size of the sketch.

Claim 2.3.10. *(i) T has at most $2n$ long edges. (ii) $|\mathcal{L}(T)| \le 3n$.*

Proof. For part (i), recall that the bottom node of every long edge has degree different than 1. Since T has n leaves, it may have at most $2n$ such nodes. Part (ii) follows from (i) since every leaf of a subtree in $\mathcal{F}(T)$ is either a leaf of T or the top node of a long edge. □

Lemma 2.3.11. *The total sketch size is $O(n(d+\log n)\log(1/\epsilon) + n\log\log\Phi)$ bits.*

Proof. By Lemma 2.3.4 we have $|T| = O(n\log(1/\epsilon))$. The tree structure of T can be stored with $O(|T|)$ bits by the Eulerian Tour Technique. The length of every long edge is bounded by the number of levels in T, which is $O(\log\Phi)$, and hence by Claim 2.3.10(i) their total storage cost is $O(n\log\log\Phi)$ bits. The center of each node is an integer in $[n]$, and can be encoded by $\log n$ bits. The ingress of each node is either itself, its parent, or a leaf of a subtree in $\mathcal{F}(T)$, hence by Claim 2.3.10(ii) it is one of $O(n)$ nodes, and can be encoded by $O(\log n)$ bits. Together, the total storage cost of the centers and ingresses is $O(|T|\log n)$. The total number of bits required to store the $\gamma(v)$'s is

$$\begin{aligned}\sum_{v\in T}\log\left(\frac{1}{\gamma(v)}\right) &= \sum_{v\in T}\log\left(5+\left\lceil\frac{\Delta(v)}{2^{\ell(v)}}\right\rceil\right)\\ &\le O(|T|) + \sum_{v\in T}\log\left(\frac{\Delta(v)}{2^{\ell(v)}}\right)\\ &\le O(n\log(1/\epsilon)), \end{aligned} \tag{2.6}$$

having used Lemmas 2.3.3 and 2.3.4 for the last inequality. Finally, for every node v, $\eta(v)$ is encoded as an element of $\mathcal{N}_{\gamma(v)}$, which by Lemma 2.2.1 takes $O(d\log(1/\gamma(v)))$ storage bits. Hence by eq. (2.6), their total storage size is $O(d)\cdot\sum_{v\in T}\log\left(\frac{1}{\gamma(v)}\right) = O(dn\log(1/\epsilon))$ bits. For nodes in $\mathcal{L}(T)$ we also store $\eta_\epsilon(v)$, which is an element in a $(\gamma(v)\cdot\epsilon)$-net. By Lemma 2.2.1, this adds $O(d\log(1/\epsilon))$ per node, and by Claim 2.3.10(ii) there are $O(n)$ such node, hence the total additional cost is $O(nd\log(1/\epsilon))$ bits. Adding up all of the sketch components, the total sketch size is $O(n(d+\log n)\log(1/\epsilon)+n\log\log\Phi)$ bits. □

2.3.4 Distance Estimation

We now show how to use the relative location tree to approximate the distance $\|x_i - x_j\|$ for every pair $i, j \in [n]$. This proves the sketch size bound in Theorem 2.1.4 (running times are analyzed in the next section). The key point is that within each subtree in $\mathcal{F}(T)$, we can recover the surrogates up to a fixed (unknown) shift from the sketch.

Shifted surrogates. Let $T' \in \mathcal{F}(T)$ be a subtree. For every node v in T', we define a *shifted surrogate* $s(v) \in \mathbb{R}^d$ by induction on the ingress order in T', as follows. For the root v of T', let $s(v) = \mathbf{0}$ (the origin in $\mathbb{R}^d$). For a non-root v in T', let $s(v) = s(in(v)) + \frac{2^{\ell(v)}}{\gamma(v)} \cdot \eta(v)$. For $v \in \mathcal{L}(T)$, let the *shifted leaf surrogate* be $s_\epsilon(v) = s(in(v)) + \frac{2^{\ell(v)}}{\gamma(v)} \cdot \eta_\epsilon(v)$.

Note that we can compute the shifted surrogates from the sketch, since it stores $in(v)$, $\gamma(v)$ and $\eta(v)$ for every node, and it also stores the lengths of the long edges, which allow us to recover $\ell(v)$. For $v \in \mathcal{L}(T)$ we can compute the shifted leaf surrogate, since the sketch stores $\eta_\epsilon(v)$. Furthermore, observe that the induction step that defines the shifted surrogates is identical to the one defining the surrogates (Equation (2.4)), and they differ only in the induction base. This implies,

Claim 2.3.12. *Let v be a node in a subtree $T' \in \mathcal{F}(T)$ whose root is r. Then $s(v) = s^*(v) - x_{c(r)}$. Furthermore, if v is a leaf of T', then $s_\epsilon(v) = s^*_\epsilon(v) - x_{c(r)}$.*

We remark that $x_{c(r)}$ cannot be recovered for the sketch. Indeed, there can be as many as $\Omega(n)$ subtrees in $\mathcal{F}(T)$, and thus storing all of their root centers could amount to fully (or at least approximately) storing $\Omega(n)$ points — the same problem we are trying to solve.

Estimation algorithm. Given $i, j \in [n]$, we show how to return a $(1 \pm \epsilon)$-approximation of $\|x_i - x_j\|$. Let u_{ij} be the lowest common ancestor of $\text{leaf}(x_i)$ and $\text{leaf}(x_j)$ in T. Let $T' \in \mathcal{F}(T)$ be the subtree that contains u_{ij}. Let v_i be the leaf of T' whose cluster contains x_i, and similarly define v_j for x_j. See Figure 2-2 for illustration. The estimate we return is $\|s_\epsilon(v_i) - s_\epsilon(v_j)\|$.

Lemma 2.3.13. $\|s_\epsilon(v_i) - s_\epsilon(v_j)\| = (1 \pm 4\epsilon) \cdot \|x_i - x_j\|$.

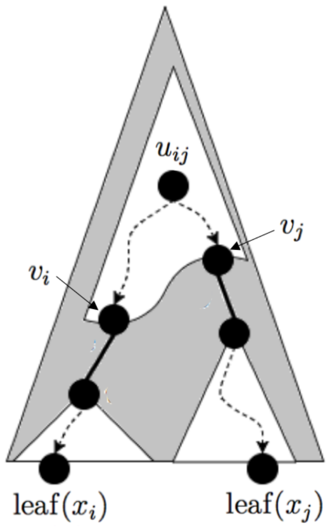


Figure 2-2: Distance estimation for $\|x_i - x_j\|$. The external shaded triangle is the tree T. The white regions are subtrees. The dashed arrows are downward paths in T. The thick arcs are long edges. The output estimate is $\|s_\epsilon(v_i) - s_\epsilon(v_j)\|$.

Proof. By Claim 2.3.12, $\|s_\epsilon(v_i) - s_\epsilon(v_j)\| = \|s^*_\epsilon(v_i) - s^*_\epsilon(v_j)\|$. By the triangle inequality,

$$\|s^*_\epsilon(v_i) - s^*_\epsilon(v_j)\| = \|x_i - x_j\| \pm \left(\|x_i - s^*_\epsilon(v_i)\| + \|x_j - s^*_\epsilon(v_j)\|\right). \tag{2.7}$$

Since $v_i \in \mathcal{L}(T)$ and $x_i, x_{c(v_i)} \in C(v_i)$, we have $\|x_i - x_{c(v_i)}\| \leq 2^{\ell(v_i)}\epsilon$ by Lemma 2.3.5. Combining this with Lemma 2.3.9 yields $\|x_i - s^*_\epsilon(v_i)\| \leq 2 \cdot 2^{\ell(v_i)}\epsilon$ by the triangle inequality. Since u_{ij} cannot be a leaf in its subtree T' (since then its degree would be either 0 or 1, contradicting its choice as the lowest common ancestor of leaf(x_i) and leaf(x_j)), we have $\ell(v_i) \leq \ell(u_{ij}) - 1$, and thus $\|x_i - s^*(v_i)\| \leq 2^{\ell(u_{ij})}\epsilon$. The same holds for x_j, and summing these together, $\|x_i - s^*_\epsilon(v_i)\| + \|x_j - s^*_\epsilon(v_j)\| \leq 2 \cdot 2^{\ell(u_{ij})}\epsilon$. By Claim 2.3.1 $2^{\ell(u_{ij})-1} \leq \|x_i - x_j\|$, and hence $\|x_i - s^*_\epsilon(v_i)\| + \|x_j - s^*_\epsilon(v_j)\| \leq \|x_i - x_j\| \cdot 4\epsilon$. Plugging this into eq. (2.7). proves the lemma. $\square$

Scaling ϵ by a constant, this concludes the proof of the sketch size bound in Theorem 2.1.4.

2.3.5 Running Times

Sketching time. We spend $O(n^2 \log \Phi)$ constructing T^*. Path compression takes linear time in the size of T^*, which is $O(n \log \Phi)$. To define the ingresses, we need to construct the graph H_v over the children of every node v, and find a spanning tree in it. This takes $O(k_v^2)$ time if v has k_v children. Since in every level ℓ there are up to n nodes, we have $\sum_{v:\ell(v)=\ell} k_v \le n$, and therefore the total time for level ℓ is $O(\sum_{v:\ell(v)=\ell} k_v^2) \le O(n^2)$. Over $O(\log \Phi)$ levels in the tree, this too takes $O(n^2 \log \Phi)$ time. Then, for every node v we need to compute $\gamma(v)$, $\eta^*(v)$ and $\eta(v)$ in order to define the surrogates. This is involves arithmetic operations on d-dimensional vectors in $O(d)$ time each, as well as rounding $\eta^*(v)$ to a grid net, which by Lemma 2.2.1 takes $O(d)$ time. Since there are $O(n \log(1/\epsilon))$ nodes (Lemma 2.3.4), this takes $O(nd \log(1/\epsilon))$ time overall.

Estimation Time. Since the height of the tree is $O(\log \Phi)$, we spend that much time finding the lowest common ancestor of $\text{leaf}(x_i), \text{leaf}(x_j)$ and finding v_i, v_j. Then we need to compute the shifted leaf surrogates $s_\epsilon(v_i), s_\epsilon(v_j)$. Due to the inductive definition of the surrogates, this might require traversing the ingress ordering on the subtree backward all the way to the root. In the worst case we might traverse all nodes in T, which could take $\Omega(n)$ time.

To avoid this, we can augment the sketch with additional information that improves the query time without asymptotically increasing the sketch size. In particular, we explicitly store the shifted surrogates for some nodes in T, called *landmark nodes*. Let $K = \lceil \log(2\Phi \cdot d^{1/p}) \rceil$. We choose landmark nodes in each subtree $T' \in \mathcal{F}(T)$ separately, as follows: Let T'_{in} be the tree that describes the ingress ordering in T' (this is a tree on the nodes in T' with the same root, where the parent of each node v is $in(v)$). Start with a lowest node $v \in T'_{in}$; climb upward K steps (or less if the root is reached), to a node $\hat{v}$; declare $\hat{v}$ a landmark node, remove it from T'_{in} with all its decendants; iterate. Since every iteration but the last removes at least K nodes from T'_{in}, we finish with at most $O(|T'_{in}|/K)$ landmark nodes. Summing over all subtrees, we have $O(|T|/K)$ landmark nodes in total.

For every landmark node, we explicitly store the shifted surrogate in the sketch. Note that choosing landmark nodes and computing their shifted surrogates require the same time as computing the (non-shifted) surrogates (both involve tracing the ingress ordering in each subtree and processing each node in $O(d)$ time), so they do not asymptotically change the sketching time. Furthermore, computing the shifted surrogate of a given non-landmark node can now be done in $O(dK)$ time. Thus, the total estimation time for $p < \infty$ is $O(d\log(d\Phi))$, and for $p = \infty$ (where $d^{1/p} = 1$) it is $O(d\log\Phi)$.

It remains to see that storing the shifted surrogates for landmark nodes does not asymptotically increase the sketch size. To this end, note that the shifted surrogates are defined recursively, starting at $\mathbf{0}$, and in each step adding a vector of the form $\gamma(v)^{-1} \cdot 2^{\ell(v)} \cdot \eta(v)$. Since $\eta(v)$ is an element in $\mathcal{N}_{\gamma(v)}$ — the grid net with cell side length $\gamma(v)/d^{1/p}$ — every coordinate of a shifted surrogate is an integer multiple of $d^{-1/p}$. On the other hand, we have the following:

Claim 2.3.14. *For every node v in T, $\|s(v)\| \leq 2\Phi$.*

Proof. Let r be the root of the subtree $T' \in \mathcal{F}(T)$ that contains v. By Claim 2.3.12, $\|s(v)\| = \|s^*(v) - x_{c(r)}\|$. By triangle inequality, $\|s^*(v) - x_{c(r)}\| \leq \|s^*(v) - x_{c(v)}\| + \|x_{c(v)} - x_{c(r)}\|$. By Lemma 2.3.8, $\|s^*(v) - x_{c(r)}\| \leq \Phi$. Since Φ is an upper bound on the diameter of the input point set X, $\|x_{c(v)} - x_{c(r)}\| \leq \Phi$. Together, $\|s(v)\| \leq 2\Phi$. $\square$

The claim implies in particular that each coordinate of a shifted surrogate is at most 2Φ. Being also an integer multiple of $d^{-1/p}$, it can be represented by $\lceil\log(2\Phi \cdot d^{1/p})\rceil = K$ bits. Thus each shifted surrogates is stored by $O(dK)$ bits, and since we store this for $O(|T|/K)$ landmark nodes, the overall additional cost is $O(d|T|) = O(nd\log(1/\epsilon))$ (Lemma 2.3.4), which does not asymptotically increase the sketch size.

2.4 Euclidean Metrics

In this section we prove the upper bound in Theorem 2.1.2. We start with Johnson-Lindenstrauss dimension reduction, Theorem 1.2.1. By applying the theorem as a preprocessing step before our sketching algorithm, we may henceforth assume that $d = O(\epsilon^{-2}\log n)$. Since we may arbitrarily increase the dimension (by adding zero coordinates), we will also assume w.l.o.g. that $d \geq 3\epsilon^{-2}\log n$.

2.4.1 Sketch Augmentations

In this section we describe the sketch. First, we compute the sketch from Section 2.3, with, say, $1/2$ instead of ϵ (the choice of constant does not matter). Next we describe some augmentations to the sketch. To this end, we choose $2d$ i.i.d. random variables uniformly over $[0,1]$, and arrange them into two vectors $\sigma', \sigma'' \in \mathbb{R}^d$ that will serve as random shifts. (They will be not stored in the sketch, so there is no concern about the precision of their representation.) We will use uniform grids as defined in Section 2.2. By the *"bottom-left"* corner of a grid cell, we mean the point in the cell (considered as a closed set of $\mathbb{R}^d$) in which each coordinate is minimized. That is, the bottom-left corner of the d-dimensional hypercube $[a_1, b_1] \times \ldots \times [a_d, b_d]$ is $(a_1, \ldots, a_d)$.

Below, let $\sigma \in \{\sigma', \sigma''\}$. Note that $\|\frac{1}{\sqrt{d}}\sigma\| \leq 1$ for all supported σ. We now describe additional information that we store in the sketch for every subtree leaf $v \in \mathcal{L}(T)$.

Augmentation I: Surrogate grid quantization. By Lemma 2.3.8 we have $\|x_{c(v)} - s^*(v)\| \leq 2^{\ell(v)}$. By the triangle inequality, $\|x_{c(v)} + \frac{1}{\sqrt{d}}2^{\ell(v)}\sigma - s^*(v)\| \leq 2 \cdot 2^{\ell(v)}$. By Fact 2.2.2, the grid with cell side $\frac{1}{\sqrt{d}}2^{\ell(v)}$ has $\exp(d)$ cells intersecting the origin-centered ball of radius $2 \cdot 2^{\ell(v)}$. Therefore, with $O(d)$ bits we can store the bottom-left corner of the grid cell containing $x_{c(v)} + \frac{1}{\sqrt{d}}2^{\ell(v)}\sigma - s^*(v)$. Since σ is random, this bottom-left corner is a d-dimensional random variable, which we denote by $A_v = (A_v^1, \ldots, A_v^d)$.

Augmentation II: Long-edge grid quantization. If the subtree $T' \in \mathcal{F}(T)$ that contains v also contains the root of T, we do not need to store additional information for v. Otherwise, the root of T' is the bottom node of a long edge. Let u be the top node of that long edge, and note that $u \in \mathcal{L}(T)$. See Figure 2-3 for illustration.

Since v is a descendant of u in T we have $x_{c(v)} \in C(u)$, and therefore by Lemma 2.3.5, $\|x_{c(v)} - x_{c(u)}\| \leq 2^{\ell(u)}$. (Recall that we use a relative location tree with $\epsilon = \Omega(1)$.) By the triangle inequality, $\|x_{c(v)} + \frac{1}{\sqrt{d}}2^{\ell(u)}\sigma - x_{c(u)}\| \leq 2 \cdot 2^{\ell(u)}$. By Fact 2.2.2, the grid with cell side $\frac{1}{\sqrt{d}}2^{\ell(u)}$ has $\exp(d)$ cells intersecting the origin-centered ball of radius $2 \cdot 2^{\ell(u)}$. Therefore, with $O(d)$ bits we can store the bottom-left corner of the grid cell containing $x_{c(v)} + \frac{1}{\sqrt{d}}2^{\ell(u)}\sigma - x_{c(u)}$. Since σ is random, this corner is a d-dimensional random variable, which we denote by

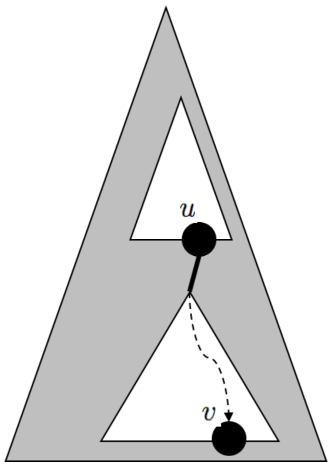


Figure 2-3: Augmentation to the Euclidean sketch. The external shaded triangle is the tree T. The white regions are subtrees. The thick arc is a long edge. For every subtree leaf v and $\sigma \in \{\sigma'\sigma''\}$, the sketch encodes $x_{c(v)} + \frac{1}{\sqrt{d}}2^{\ell(v)}\sigma - s^*(v)$ in Augmentation I. If u is defined for v, then the sketch also encodes $x_{c(v)} + \frac{1}{\sqrt{d}}2^{\ell(u)}\sigma - x_{c(u)}$ in Augmentation II.

$B_v = (B_v^1, \ldots, B_v^d)$.

Remark. Note that we store each of the above augmentations twice — once with the random shift σ' and once with σ''. To ease notation, let us not denote them separately, and simply keep in mind that we have two independent copies of each A_v and B_v.

Total sketch size. By Lemma 2.3.11, the relative location tree with $\epsilon = \Omega(1)$ is stored in $O(n(\log n + d + \log\log\Phi))$ bits. The above augmentations store $O(d)$ additional bits per node in $\mathcal{L}(T)$, of which there are $O(n)$ (Claim 2.3.10), and this does not increase the sketch size asymptotically. Since $d = O(\epsilon^{-2}\log n)$ by the preceding dimension reduction step, the total sketch size is $O(\epsilon^{-2} n \log n + n \log\log\Phi)$ bits.

2.4.2 Probabilistic Surrogates

We now show how to recover, for every point $x \in X$, a random variable that would serve as a probabilistic (shifted) surrogate. For the two next lemmas, fix a subtree $T' \in \mathcal{F}(T)$ with root r. For every $i \in [n]$ such that $x_i \in C(r)$, denote by v_i the leaf of T' whose cluster contains x. That is, v_i is the lowest node on the downward path from r to leaf(x_i) that does not traverse a long edge.

Lemma 2.4.1. *Let $i \in [n]$ be such that $x_i \in C(r)$. We can recover from the sketch a d-dimensional random variable $X_i = (X_i^1, \ldots, X_i^d) \in \mathbb{R}^d$, such that:*

- *Its coordinates are independent.*
- *Each coordinate is supported on an interval of length at most $\frac{1}{\sqrt{d}} 3 \cdot 2^{\ell(v_i)}$.*
- $\mathbb{E}[X_i] = x_i - x_{c(r)}$*, coordinate-wise.*

We prove this by proving a somewhat more general claim by induction.

Lemma 2.4.2. *Let $i \in [n]$ be such that $x_i \in C(r)$. For every subtree leaf $v \in \mathcal{L}(T)$ which is a descendant of v_i in T (note that v is not in T' unless $v = v_i$), we can recover from the sketch a d-dimensional random variable $Y_v = (Y_v^1, \ldots, Y_v^d) \in \mathbb{R}^d$, such that:*

- *Its coordinates are independent.*
- *Each coordinate is supported on an interval of length at most $\frac{1}{\sqrt{d}}(3 \cdot 2^{\ell(v_i)} - 2 \cdot 2^{\ell(v)})$.*
- $\mathbb{E}[Y_v] = x_{c(v)} - x_{c(r)}$*, coordinate-wise.*

Lemma 2.4.2 clearly implies Lemma 2.4.1 in the special case $v = \text{leaf}(x_i)$.

Proof of Lemma 2.4.2. The proof is by induction on the subtree leaves (nodes in $\mathcal{L}(T)$) that lie on the downward path from v_i to $\text{leaf}(x_i)$.

Induction base. In the base case, $v = v_i$. We take $Y_v = A_v + s(v)$. Note that A_v is stored by Augmentation I, and $s(v)$ is a shifted surrogate, so both can be recovered from the sketch. We show that Y_v satisfies the required properties.

Let us simplify some notation for convenience. Let $L = \frac{1}{\sqrt{d}} 2^{\ell(v)}$. Let $\mathcal{G}[L]$ be origin-centered uniform grid with cell side length L. Let $y = x_{c(v)} - s^*(v)$, with coordinates $y = (y_1, \ldots, y_d)$. Let $H = [a_1, a_1 + L] \times \ldots \times [a_d, a_d + L] \subset \mathbb{R}^d$ be the hypercube cell of $\mathcal{G}[L]$ that contains y. Fix $\sigma \in \{\sigma', \sigma''\}$, and let $(\sigma_1, \ldots, \sigma_d)$ denote its coordinates.

In Augmentation I, A_v is the bottom-left corner of the cell of $\mathcal{G}[L]$ that contains $y + L\sigma$, where each coordinate of σ is an i.i.d. uniformly random shift in $[0, 1]$. This means that each coordinate $j \in [d]$ of A_v is set to $A_v^j = a_j$ if $y_j + L\sigma_j < a_j + L$, and to $A_v^j = a_j + L$ otherwise.

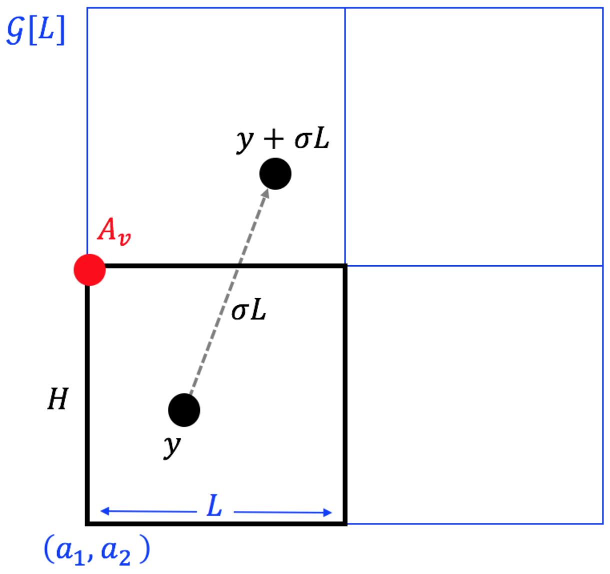


Figure 2-4: Base case of Lemma 2.4.2 (in two dimensions). H is the hypercube cell of $\mathcal{G}[L]$ that contains $y = x_{c(v)} - s^*(v)$. Note that H is not random. A_v from Augmentation I is the bottom-left corner of the grid cell that contains $y + \sigma L$, where σ is a shift with uniform i.i.d. coordinates in $[0, 1]$. Thus, A_v is supported on the corners of H, and $\mathbb{E}_\sigma[A_v] = y$.

The latter condition rearranges to $\sigma_j < 1 - \frac{1}{L}(y_j - a_j)$ (note that this value is in $[0, 1]$ since a_j is defined such that $a_j \leq y_j < a_j + L$), which occurs with probability $1 - \frac{1}{L}(y_j - a_j)$. Therefore,

$$\mathbb{E}_{\sigma_j}[A_v^j] = a_j \cdot (1 - \tfrac{1}{L}(y_j - a_j)) + (a_j + L) \cdot \tfrac{1}{L}(y_j - a_j) = y_j.$$

Furthermore, A_v is supported on the corners of the grid cell H, and hence each coordinate is supported on an interval of length $L = \frac{1}{\sqrt{d}} 2^{\ell(v)}$. Finally, since the coordinates of σ are independent, then so are the coordinates of A_v. (This is the same randomized rounding scheme from [AK16]; see Figure 2-4 for illustration.) By taking $Y_v = A_v + s(v)$, the support length of each coordinate and the independence between the coordinates are preserved, while the expectation changes to

$$\mathbb{E}_\sigma[Y_v] = \mathbb{E}_\sigma[A_v] + s(v) = y + s(v) = x_{c(v)} - s^*(v) + s(v) = x_{c(v)} - x_{c(r)},$$

coordinate-wise, where we have used Claim 2.3.12 for the rightmost equality. This proves the base case.

Induction step. Let v be a descendant of v_i, which is different than v_i. Let u be the next node in $\mathcal{L}(T)$ on the upward path from v to v_i. By induction, the statement of Lemma 2.4.2 holds for u. Therefore we have a random variable Y_u with independent coordinates, each supported on an interval of length $\frac{1}{\sqrt{d}}(3 \cdot 2^{\ell(v_i)} - 2 \cdot 2^{\ell(u)})$, such that $\mathbb{E}[Y_u] = x_{c(u)} - x_{c(r)}$ coordinate-wise.

Augmentation II stores B_v, defined as the bottom-left corner of the cell of the origin-centered grid $\mathcal{G}[\frac{1}{\sqrt{d}}2^{\ell(u)}]$ that contains $x_{c(v)} + \frac{1}{\sqrt{d}}2^{\ell(u)}\sigma - x_{c(u)}$. Similarly to what was shown in the base step for A_v, this implies that $\mathbb{E}_\sigma[B_v] = x_{c(v)} - x_{c(u)}$, that B_v has independent coordinates, and that it is supported on the corners of the grid cell that contains $x_{c(v)} - x_{c(u)}$, which means that each coordinate is supported on an interval of length $\frac{1}{\sqrt{d}}2^{\ell(u)}$.

We let $Y_v = Y_u + B_v$. It is easily seen that Y_v has the correct expectation ($\mathbb{E}[Y_v] = x_{c(v)} - x_{c(r)}$ coordinate-wise), that its coordinates are independent, and that each is supported on an interval of length $\frac{1}{\sqrt{d}}(3 \cdot 2^{\ell(v_x)} - 2^{\ell(u)})$. The proof is complete by noticing that $\ell(v) < \ell(u)$, hence $\ell(v) \le \ell(u) - 1$, hence the length is at most $\frac{1}{\sqrt{d}}(3 \cdot 2^{\ell(v_x)} - 2 \cdot 2^{\ell(v)})$. $\square$

2.4.3 Distance Estimation

Let $i, j \in [n]$. We show how to estimate $\|x_i - x_j\|$ from the sketch. Let r be the lowest node in T which is the root of a subtree in $T' \in \mathrm{F}(T)$ and such that $x_i, x_j \in C(r)$ (i.e., r is a common ancestor of leaf(x_i) and leaf(x_j)). Let v_i be the leaf of T' whose cluster contains x_i, and similarly define v_j for x_j. Let $\ell_{ij} := \max\{\ell(v_i), \ell(v_j)\}$. Note that by Claim 2.3.1 we have $\|x - y\| \ge 2^{\ell_{ij}}$.

Using Lemma 2.4.1, we can read off the sketch random variables $X'_i, X'_j, X''_i, X''_j \in \mathbb{R}^d$, such that each has independent coordinates supported on an interval of length $\frac{3}{\sqrt{d}}2^{\ell_{ij}}$, such that $\mathbb{E}[X'_i] = \mathbb{E}[X''_i] = x_i - x_{c(r)}$ and $\mathbb{E}[X'_j] = \mathbb{E}[X''_j] = x_j - x_{c(r)}$ coordinate-wise, and such that (X'_i, X'_j) are independent of (X''_i, X''_j). The latter property is achieved by using the random shift σ' for (X'_i, X'_j) and the random shift σ'' for (X''_i, X''_j). The estimate we return is $\sqrt{(X'_i - X'_j)^T(X''_i - X''_j)}$. (Note that if we had $X'_i = X''_i$ and $X'_j = X''_j$ then the estimate would just be $\|X'_i - X'_j\|$; however, we will make use of the independence between (X'_i, X'_j) and (X''_i, X''_j).) We now show it is a sufficiently accurate estimate.

To this end, let $Z_1 = X'_i - X'_j$ and $Z_2 = X''_i - X''_j$. The returned estimate is $\sqrt{Z_1^T Z_2}$. Note that Z_1, Z_2 are independent, each has independent coordinates supported on an interval of length $\frac{6}{\sqrt{d}}2^{\ell_{ij}}$, and $\mathbb{E}[Z_1] = \mathbb{E}[Z_2] = x - y$ coordinate-wise.

The following lemma is adapted from Alon and Klartag [AK16].

Lemma 2.4.3. *Suppose $d \geq 3\epsilon^{-2}\log n$. Let $S > 0$. Let $z_1, z_2 \in \mathbb{R}^d$. Let Z_1, Z_2 be independent random variables with independent coordinates, with each coordinate supported of an interval of length $\frac{1}{\sqrt{d}}S$, and such that $\mathbb{E}[Z_1] = z_1$ and $\mathbb{E}[Z_2] = z_2$ coordinate-wise. Then,*

$$\Pr[|Z_1^T Z_2 - z_1^T z_2| \leq \epsilon \cdot S(\|z_1\| + \|z_2\| + S)] \geq 1 - \frac{4}{n^3}.$$

Proof. We will denote the coordinates of Z_1 by $(Z_1^1, \ldots, Z_1^d)$, and similarly for z_1, Z_2, and z_2. By the triangle inequality,

$$\begin{aligned}|Z_1^T Z_2 - z_1^T z_2| &= |Z_1^T Z_2 - Z_1^T z_2 + Z_1^T z_2 + z_1^T z_2| \\ &\leq |(Z_1 - z_1)^T z_2| + |Z_1^T (Z_2 - z_2)|.\end{aligned}$$

We start with the term $(Z_1 - z_1)^T z_2 = \sum_{i=1}^d z_2^i (Z_1^i - z_1^i)$. By hypothesis, for every coordinate $i \in [d]$ we have $\mathbb{E}[Z_1^i - z_1^i] = 0$ and $|Z_1^i - z_1^i| \leq \frac{1}{\sqrt{d}}S$. Therefore the sum of squares of the summands is upper-bounded by $\frac{1}{d}S^2\|z_2\|^2$. Now by Hoeffding's inequality (Lemma 1.2.6),

$$\Pr\left[\left|(Z_1 - z_1)^T z_2\right| > \epsilon S\|z_2\|\right] \leq 2e^{-2\epsilon^2 d} \leq \frac{2}{n^3},$$

where we have used $d \geq 3\epsilon^{-2}\log n$.

We proceed to the term $Z_1^T(Z_2 - z_2) = \sum_{i=1}^d Z_1^i (Z_2^i - z_2^i)$. Again by hypothesis $\mathbb{E}[Z_2^i - z_2^i] = 0$, and since Z_1, Z_2 are independent, $\mathbb{E}[Z_1^i(Z_2^i - z_2^i)] = 0$. The sum of squares is upper-bounded by $\frac{1}{d}S^2\|Z_1\|^2$ as above, and by the triangle inequality, $\|Z_1\| \leq \|z_1\| + \|Z_1 - z_1\| \leq \|z_1\| + S$. Altogether, $\sum_{i=1}^d (Z_1^i(Z_2^i - z_2^i))^2 \leq \frac{1}{d}S^2(\|z_1\| + S)^2$, and by Hoeffding's inequality,

$$\Pr\left[\left|Z_1^T(Z_2 - z_2)\right| > \epsilon S(\|z_1\| + S)\right] \leq 2e^{-2\epsilon^2 d} \leq \frac{2}{n^3}.$$

The lemma follows by a union bound over the two terms. □

Applying the lemma to Z_1, Z_2 defined above (with $z_1 = z_2 = x - y$ and $S = 6 \cdot 2^{\ell_{ij}}$),

$$\Pr[|Z_1^T Z_2 - \|x-y\|^2| \leq \epsilon \cdot 6 \cdot 2^{\ell_{ij}}(2\|x-y\| + 6 \cdot 2^{\ell_{ij}})] \geq 1 - \frac{4}{n^3}.$$

Since $\|x - y\| \geq 2^{\ell_{ij}}$, this implies

$$\Pr\left[\left|Z_1^T Z_2 - \|x-y\|^2\right| \leq \epsilon \cdot 48\|x-y\|^2\right] \geq 1 - \frac{4}{n^3},$$

and thus $Z_1^T Z_2 = (1 \pm O(\epsilon)) \cdot \|x-y\|^2$, which renders our estimate $\sqrt{Z_1^T Z_2}$ correct (up to scaling ϵ by a constant) with that probability. The total success probability is $1 - O(1/n)$, by a union bound over all pairs $i, j \in [n]$, and over the application of the Johnson-Lindenstrauss theorem that was used as a preprocessing step.

2.4.4 Running Times

We start with the sketching time. The Johnson-Lindenstrauss theorem can be performed either naïvely in time $O(\epsilon^{-2} nd \log n)$, or in time $O(nd \log d + \epsilon^{-2} n \cdot \min\{d \log n, \log^3 n\})$ by the Fast Johnson-Lindenstrauss Transform of Ailon and Chazelle [AC09]. Note that here, d is the ambient dimension of the input metric (before dimension reduction).

Next we compute the sketch from Section 2.3. To avoid confusion in notation, let us denote its error and dimension parameters by ϵ' and d' respectively. We construct that sketch with $\epsilon' = \Omega(1)$ and $d' = \Theta(\epsilon^{-2} \log n)$, which as per Section 2.3.5 takes time $O(n^2 \log \Phi + nd')$. The $n^2 \log \Phi$ term can be reduced to $O(n^{1+\alpha} \log \Phi)$ for any constant $0 < \alpha < 1$, at the cost of increasing the sketch size by an additive factor of $O(nd' \log(1/\alpha))$. This does not asymptotically increase its size $O(nd' + n \log\log \Phi)$ as long as $\alpha = \Omega(1)$.

To this end, let $c = \alpha^{-1/2}$. In constructing the relative location tree, we use the algorithm of [HPIM12] to compute c-approximate connected components in each level. Their algorithm is based on Locality-Sensitive Hashing (LSH), which in Euclidean spaces can be implemented in time $O(n^{1+1/c^2})$ [AI06]. Using c-approximate connected components means that clusters in level ℓ of the relative location tree may be merged if the distance between them is up to $c \cdot 2^\ell$ (rather than just 2^ℓ), and to account for this constant loss, we need to scale ϵ' down to ϵ'/c. Since the dependence of the sketch size on ϵ' is $O(nd' \log(1/\epsilon'))$, it increases by an additive factor of $O(nd' \log(1/\alpha))$.

Finally, the sketch augmentations in Section 2.4.1 take time d per node in $\mathcal{L}(T)$ to compute, so in total, $O(nd)$ time. The overall sketching time is as stated in Theorem 2.1.2.

The estimation time is the same as Theorem 2.1.4, with dimension $\Theta(\epsilon^{-2} \log n)$.

2.5 ℓ_p-Metrics with $1 \leq p < 2$

We point out that by known embedding results, both the upper and lower bounds for Euclidean metric compression in Theorem 2.1.2 apply more generally to ℓ_p-metrics for every $1 \leq p \leq 2$.

Theorem 2.5.1. *Let $1 \leq p \leq 2$ and $\epsilon > 0$. For ℓ_p-metric sketching with n points and diameter Φ (of arbitrary dimension), $\Theta(\epsilon^{-2} n \log n + n \log\log \Phi)$ bits are both sufficient and necessary.*

The upper bound relies on the well-known fact that every such metric embeds isometrically into a negative-type metric, i.e., into a squared Euclidean metric. We use the following constructive version of this fact, from [LN14, Theorem 116], based on [MN04].

Theorem 2.5.2 ([LN14][2])**.** *Let $1 \leq p < 2$. Let $X \subset \mathbb{R}^d$ be a point set with ℓ_p-aspect ratio Φ. There is a mapping $f : X \to \mathbb{R}^{d \cdot \mathrm{poly}(\log \Phi, \log d, 1/\epsilon)}$ such that for every $x, y \in X$,*

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

Proof of Theorem 2.5.1. Both the upper and lower bound follow from Theorem 2.1.2. For the upper bound, by Theorem 2.5.2 we have a map f such that it suffices to report $\|f(x_i) - f(x_j)\|_2^{2/p}$ for every i, j. Then it suffices to sketch the Euclidean metric on $f(x_1), \ldots, f(x_n)$. The lower bound follows from the standard fact that Euclidean metrics embed isometrically into ℓ_p-metrics for every $1 \leq p < 2$ (see, e.g., [Mat13]). □

[2]The statement in [LN14] is for $\Phi = d^{O(1)}$, but applies to any $\Phi > 0$. The statement given here is by setting $R = d^{-1/q}$ in [LN14, Theorem 116] and scaling the minimal distance in the given metric to 1.

2.6 Lower Bounds

In this section we prove tight compression lower bounds for Euclidean metric spaces and for general metric spaces, matching the upper bounds in Theorems 2.1.2 and 2.1.3 respectively. This finishes the proofs of those two theorems.

Theorem 2.6.1 (Euclidean metrics)**.** *The ℓ_2-metric sketching problem with n points, aspect ratio Φ and dimension $d = \Omega(\epsilon^{-2}\log n)$ requires $\Omega(\epsilon^{-2} n \log n + n \log\log\Phi)$ bits.*

Proof. We start by proving the first term of the lower bound, $\Omega(\epsilon^{-2} n \log n)$. Let $0 < \gamma < 0.5$ be a constant and let $\epsilon \geq \Omega(1/n^{0.5-\gamma})$ be smaller than a sufficiently small constant. Let $k = 1/\epsilon^2$, and suppose w.l.o.g. k is an integer by scaling ϵ down by an appropriate constant. Note that $k = O(n^{1-2\gamma}) \ll n$ since $\epsilon \geq \Omega(1/n^{0.5-\gamma})$.

Let B be the set of standard basis vectors in $\mathbb{R}^n$. Let $a_1, \ldots, a_n$ be an arbitrary distinct vectors in $\{0,1\}^n$, each having exactly k coordinates set to 1 (and the rest to 0). Let $A = \{\frac{1}{\sqrt{k}} a_i : i \in [n]\}$. Note that $A \cup B$ is a set of $2n$ points in $\mathbb{R}^n$, each with unit norm.

Suppose we have a sketch for the Euclidean distances in $A \cup B$ up to distortion $1 \pm \frac{1}{8}\epsilon$. This means it can report the squared Euclidean distances up to distortion $1 \pm \frac{1}{2}\epsilon$ (by simply squaring its output). For every $i, j \in [n]$, denote by $a_i(j)$ the j-th coordinate of a_i, or equivalently, $a_i(j) = a_i^T e_j$. Then,

$$\|\tfrac{1}{\sqrt{k}} a_i - e_j\|_2^2 = \|\tfrac{1}{\sqrt{k}} a_i\|_2^2 - \tfrac{2}{\sqrt{k}} a_i^T e_j + \|e_j\|_2^2 = 2 - 2\epsilon a_i(j).$$

Thus, if $a_i(j) = 0$ then $\|\frac{1}{\sqrt{k}} a_i - e_j\|_2^2 = 2$, and the sketch is guaranteed to return at least $2 - \epsilon$. Conversely, if $a_i(j) = 0$ then $\|\frac{1}{\sqrt{k}} a_i - e_j\|_2^2 = 2 - 2\epsilon$, and the sketch is guaranteed to return at most $(1 + \frac{1}{2}\epsilon)(2 - 2\epsilon) = 2 - \epsilon - \epsilon^2$. Consequently, we can recover every $a_j(i)$ from the sketch, and thus recover A. The number of possible choices for A is $\binom{\binom{n}{k}}{n}$, which by a known estimate ($\binom{m}{\ell} \geq (\frac{m}{\ell})^\ell$ for all integers m, ℓ) is at least $((\frac{n}{k})^k/n)^n$. Therefore, the resulting bit lower bound on the sketch size is

$$\log\left(\left(\frac{(\frac{n}{k})^k}{n}\right)^n\right) = nk\log\left(\frac{n}{k}\right) - n\log n = \frac{n}{\epsilon^2}\cdot\log(n\epsilon^2) = \Omega(\gamma\cdot\epsilon^{-2} n\log n),$$

where the final bound is since $\log(n\epsilon^2) \geq \log(n^{2\gamma}) = 2\gamma\log n$, and since we can make ϵ small enough such that $\epsilon^2 < \gamma$. Note that the dimension of the point sets constructed above can be reduced to $O(\epsilon^{-2}\log n)$ by the Johnson-Lindenstrauss theorem [JL84]. This proves the first term of the lower bound in the theorem statement.

Next we prove the second term of the lower bound, $\Omega(n \log\log\Phi)$. Suppose w.l.o.g. that $\log\Phi$ is an integer. Consider the point set $X = \{1, \dots, n\}$. Define a map $g : X \to \mathbb{R}$ by setting $g(1) = 0$, and for every $x \in X \setminus \{1\}$ setting $g(x) = 2^{\phi(x)}$ with an arbitrary $\phi(x) \in \{1, \dots, \log\Phi\}$. The number of choices for g is $(\log\Phi)^{n-1}$, and every choice of g is a Euclidean embeddings of X with one-dimension and aspect ratio at most Φ. We can fully recover g given a Euclidean distance sketch for X with distortion better than 2, since $\phi(x) = |g(x) - g(1)| =$ for every $x \in X$, and every two possible values of $\phi(x)$ are separated by at least a factor of 2. This yields a sketching bound of $\log\left((\log\Phi)^{n-1}\right) = \Omega(n\log\log\Phi)$ bits.

To get the final lower bound $\Omega(\epsilon^{-2} n \log n + n\log\log\Phi)$, we augment the two metric families constructed above into one. We constructed a family $\mathcal{F}_1$ of metrics embedded in $\mathbb{R}^n$, of size $|\mathcal{F}_1| \geq 2^{\Omega(\gamma\cdot\epsilon^{-2} n\log n)}$, and a family $\mathcal{F}_2$ of metrics embedded in $\mathbb{R}^1$, of size $|\mathcal{F}_2| \geq 2^{\Omega(n\log\log\Phi)}$. For every $D' \in \mathcal{F}_1$ and $D" \in \mathcal{F}_2$, we can naturally define a metric $D' \oplus D"$ embedded in $\mathbb{R}^{n+1}$ by embedding D' in the first n dimensions and $D"$ in the remaining dimension. This defines a family $\mathcal{F} = \{D' \oplus D" : D' \in \mathcal{F}_1, D" \in \mathcal{F}_2\}$ of Euclidean metrics over n points in $\mathbb{R}^{n+1}$ with aspect ratio $\Phi + O(1)$, of size $|\mathcal{F}_1| \cdot |\mathcal{F}_2|$, such given a Euclidean distance sketch with distortion $1 \pm \epsilon$ of a metric in $\mathcal{F}$, the metric can be fully recovered. The lower bound $b = \Omega(\epsilon^{-2} n\log n + n\log\log\Phi)$ follows. ☐

Theorem 2.6.2 (general metrics)**.** *The general metric sketching problem with n points and aspect ratio Φ requires $\Omega(n^2\log(1/\epsilon) + n\log\log\Phi)$ bits.*

Proof. Let $\epsilon > 0$ be smaller than a sufficiently small constant. Suppose w.l.o.g. that ϵ^{-1} is an integer. We construct a metric space (X, d) with $X = \{1, \dots, n\}$. For every $x, y \in X$ such that $x < y$, set $\mathrm{d}(x, y) = 1 + k(x, y)\cdot\epsilon$, with an arbitrary integer $k(x, y) \in \{0, 1, \dots, \epsilon^{-1} - 1\}$. Note that $1 \leq \mathrm{d}(x, y) < 2$ for all x, y. This defines a metric space regardless of the choice of the $k(x, y)$'s. Indeed, we only need to verify the triangle inequality, and it holds trivially since all pairwise distances are lower-bounded by 1 and upper-bounded by 2. Hence we have defined a family of $(1/\epsilon)^{\binom{n}{2}}$ metrics.

Next, observe that a sketch with distortion $(1 \pm \frac{1}{4}\epsilon)$ is sufficient to fully recover a metric from this family. Indeed, for every $x, y \in X$, the sketch is guaranteed to report $\mathrm{d}(x, y)$ up to an additive error of $\frac{1}{4}\epsilon \cdot \mathrm{d}(x, y)$, which is less than $\frac{1}{2}\epsilon$, while the minimum difference between every pair of possible distances is ϵ by construction. By scaling ϵ by a constant, this proves a lower bound of $\log\left((1/\epsilon)^{\binom{n}{2}}\right) = \Omega(n^2 \log(1/\epsilon))$ on the sketch size in bits. The second lower bound term $\Omega(n \log\log \Phi)$, and the combination of the two terms into one lower bound, is by the same proof as Theorem 2.6.1. $\square$

Chapter 3

New Query Points and Nearest Neighbor Search

The previous chapter considered the problem of optimally sketching all distances within a given dataset of points in $\mathbb{R}^d$. However, in many applications to machine learning and data analysis, it is also necessary to handle new query points that were not part of the dataset. In this chapter, we extend the techniques of Chapter 2 to this setting. We consider two problems: reporting an approximate nearest neighbor of a query point in the dataset, and reporting approximate distances from a query point to all points in the dataset. For both problems we obtain nearly optimal compression bounds, improving over bounds arising from classical dimension reduction. Our results also show that reporting approximate nearest neighbors requires asymptotically less space than reporting approximate distances.

3.1 Introduction

A typical setting in information retrieval or machine learning is the following:

1. In the *preprocessing* or *training* stage, we get a dataset of objects (often embedded as point in $\mathbb{R}^d$), that we need to process in order to facilitate subsequent queries.

2. In the *query* or *inference* stage, we get a query object (say, a point in $\mathbb{R}^d$) that did not appear in the dataset, and we need to report measurements based on the dataset.

Reference	Bits per point	No. queries	Query type
[JL84, Ach03, KOR00, ...]	$O(\log^2 n)$	$q = n^{O(1)}$	distances
[NN19]	$O(\log^2 n)$	any q	distances
[MWY13]	$\Omega(\log^2 n)$	$q \geq n$	distances
Theorem 2.1.2	$O(\log n)$	—	none
Theorems 3.1.4 and 3.1.5	$\Theta(\log n \cdot \log q)$	$q \leq n$	distances
Theorem 3.1.3	$O(\log n + \log q)$	any q	nearest neighbors

Table 3.1: Our results and related work on distance sketches with n data points and q query points, in a typical regime where ϵ, δ are small constants and $d, \Phi = n^{O(1)}$.

A prototypical example of this setting is the nearest neighbor search problem, where the goal is to return the closest point in the dataset to the query point. More generally one can ask to estimate all distances from a query point to each point in the dataset.

The metric sketching problem from Definition 2.1.1 is closely related to this setting, but does not quite fit: it requires all points to be known already during the preprocessing (sketching) stage, meaning that they must all appear in the dataset. The input to the query stage is not a new point, but a pair of indices of dataset points. On the other hand, the dimension reduction theorem of [JL84], and the subsequent work on sketching [Ach03, KOR00] discussed in Chapter 2, are capable of handling new query points — as many as $n^{O(1)}$, where n is the number of points in the dataset. Very recently, Nelson and Narayanan [NN19] (following [MMMR18]) showed that (non-linear) dimension reduction can handle an *unbounded* number of query points.

Is dimension reduction optimal for distance sketches with query points? Ostensibly, a positive answer was given by Molinaro, Woodruff and Yaroslavtzev [MWY13], who proved a sketching lower bound that matches the discretized dimension reduction upper bound. However, their theorem relied on two key assumptions: One, that the number of queries is *at least as large* as the dataset size. This leaves open the small query set regime, which is arguably common in applications. For example, in many standard benchmark datasets (e.g., MNIST, SIFT, GloVe, 20newsgroups, etc.) the query set size is smaller than the dataset size by at least an order of magnitude. The other key assumption in [MWY13] is that the sketch needs to report the approximate distance between each query and each data point. This leaves open the possibility of better sketching bounds for *nearest neighbor search*, which does not necessarily entail reporting all distances.

The question arises whether our results from Chapter 2 can be extended to support query points, while retaining the improvement over the discretized dimension reduction upper bound of $O(\epsilon^{-2}\log(n)\log\Phi)$ bits per point, which is mutual to all of the previous approaches listed above. In this chapter we show that the answer is yes, by further developing our tools from the previous chapter. If we remove either one of the key assumptions in [MWY13], then the lower bound breaks, and better sketches are possible. Our results, in the context of the foregoing discussion, are summarized in Table 3.1.

3.1.1 Formal Problem Statements

We formalize the sketching problems considered in this chapter in terms of one-way communication complexity (with private randomness). The setting is as follows:

- Alice has n data points, $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, with integer coordinates in $[-\Phi, \Phi]$.
- Bob has q query points, $Y = \{y_1, \ldots, y_q\} \subset \mathbb{R}^d$, with integer coordinates in $[-\Phi, \Phi]$.
- Alice computes a compact representation (sketch) of her data points and sends it to Bob, who then needs to report the output.

We define two problems in this model, each parameterized by $n, q, d, \Phi, \epsilon, \delta$:

Definition 3.1.1. *In the* ***all-nearest-neighbors*** *problem, Bob needs to report a* $(1+\epsilon)$*-approximate nearest neighbor in* X *for all his points simultaneously, with probability* $1-\delta$*. That is, for every* $j \in [q]$*, Bob reports an index* $i_j \in [n]$ *such that*

$$\Pr\left[\forall j \in [q],\quad \|y_j - x_{i_j}\| \le (1+\epsilon)\min_{i\in[n]}\|y_j - x_i\|\right] \ge 1-\delta.$$

Definition 3.1.2. *In the* ***all-cross-distances*** *problem, Bob needs to estimate all distances* $\|x_i - y_j\|$ *up to distortion* $(1\pm\epsilon)$ *simultaneously, with probability* $1-\delta$*. That is, for every*

$i \in [n]$ and $j \in [q]$, Bob reports an estimate $\tilde{E}_{ij}$ such that

$$\Pr\left[\forall i \in [n], j \in [q], \;\; (1-\epsilon)\|x_i - y_j\| \le \tilde{E}_{ij} \le (1+\epsilon)\|x_i - y_j\|\right] \ge 1-\delta.$$

Distances throughout this chapter are Euclidean. The assumption of integer coordinates is w.l.o.g. as they can be scaled to any precision (similarly to Chapter 2). As per the above discussion in Section 3.1, we are mostly interested in the regime $q \le n$, i.e., where the number of query points is small compared to the number of data points.

3.1.2 Our Results

The next theorem states our sketching upper bound for approximate nearest neighbor search.

Theorem 3.1.3. *For the all-nearest-neighbors problem, there is a sketch of size*

$$O\left(n\left(\frac{\log n + \log q + \log(1/\delta)}{\epsilon^2}\cdot\log\left(\frac{1}{\epsilon}\right) + \log\log(d\Phi)\right)\right) + \frac{\mathrm{polylog}(n,q,d,\Phi,\delta^{-1},\epsilon^{-1})}{\epsilon^2} \;\; \textit{bits.}$$

The important term is the multiplier of n, which is the amortized number of bits per point. The additive "global" polylogarithmic term is due to storing private random bits in the sketch. The precise global term is $O((\epsilon^{-2}\log^2(d\Phi) + \log\log(q/\delta))\cdot\log(nq/\delta))$. If one allows either shared randomness, or non-constructive sketches with private randomness, then it can be eliminated (see Remark 3.2.5). However, we state our bounds for explicit constructive sketches without shared randomness.

Theorem 3.1.3 improves over the discretized dimension reduction upper bound whenever $\epsilon \gg 1/\mathrm{poly}(n)$. For example, for a constant number of queries, the dependence on n per point is $\log n$ bits, rather than $\log^2 n$. In Theorem 3.5.1 we give a lower bound of $\Omega(\epsilon^{-2} n \log n)$ in the regime $\delta = 1/n^{O(1)}$, showing that the first term in Theorem 3.1.3 is almost tight.

We turn to our results for distance estimation sketches with queries.

Theorem 3.1.4. *For the all-cross-distances problem, there is a sketch of size*

$$O\left(\frac{n}{\epsilon^2}\left(\log n\cdot\log(1/\epsilon) + \log(d\Phi)\log\left(\frac{q}{\delta}\right)\right)\right) + \frac{\mathrm{polylog}(n,q,d,\Phi,\delta^{-1},\epsilon^{-1})}{\epsilon^5} \;\; \textit{bits.}$$

Note that the dependence per point on Φ is logarithmic, as opposed to doubly logarithmic in Theorem 3.1.3. This dependence is necessary, as our next theorem shows.

Theorem 3.1.5. *Suppose that $q \leq n$, $d^{1-\rho} \geq \epsilon^{-2}\log(n/\delta)$, $\Phi \geq 1/\epsilon$, and $1/n^{0.5-\rho'} \leq \epsilon \leq \epsilon_0$ for some constants $\rho, \rho' > 0$ and a sufficiently small constant ϵ_0. Then, for the all-cross-distances problem, any sketch must use at least*

$$\Omega\left(\frac{n}{\epsilon^2}\left(\log n + \log(d\Phi)\log\left(\frac{q}{\delta}\right)\right)\right) \quad \textit{bits.}$$

Note that the upper and lower bounds on the storage bits per point in Theorems 3.1.4 and 3.1.4 are matching up to a factor $O(\log(1/\epsilon))$. Also note that Theorem 3.1.5 shows, in light of Theorem 3.1.3, that distance estimation with queries requires asymptotically larger sketches than approximate nearest neighbor search.

Table 3.1 summarizes our results and related work in a typical demonstrative setting, where ϵ, δ are small constants, and the ambient dimension d and the coordinates range Φ are both polynomial in n (this essentially means they fit in a machine word).

3.1.3 Technical Overview

The basis for our sketches in this chapter is the relative location tree from Section 2.3. As discussed above, the data structure given there does not support new query points, and we will need to modify and add to it in order to support them.

Let us review the relative location tree in order to highlight where the difficulties lie. The construction in Section 2.3 identifies a hierarchy of clusters that are very well separated from the rest of the points. Technically, they are identified by long non-branching paths in a hierarchical decomposition tree of the metric space, which are then compressed into "long edges". This compression step perturbs the location of the clusters relative to each other, but only to a tolerable error, owing to their separation (see Figure 3-1 for illustration). The removal of long edges from the tree induces a partition into *subtrees*, where each subtree corresponds to a well-separated cluster.

Distances between points in different clusters can be estimated efficiently by comparing appropriate representatives of their respective clusters, based on the separation property.

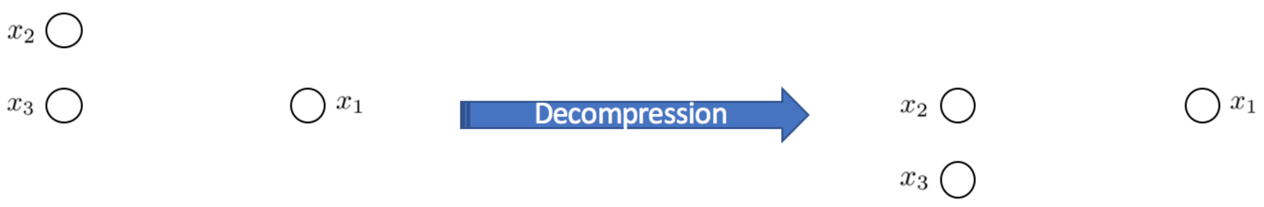


Figure 3-1: Compression and decompression of a two-dimensional dataset. The location of the well-separated cluster $\{x_2, x_3\}$ can be perturbed by the lossy compression algorithm, without significantly changing the distances to x_1.

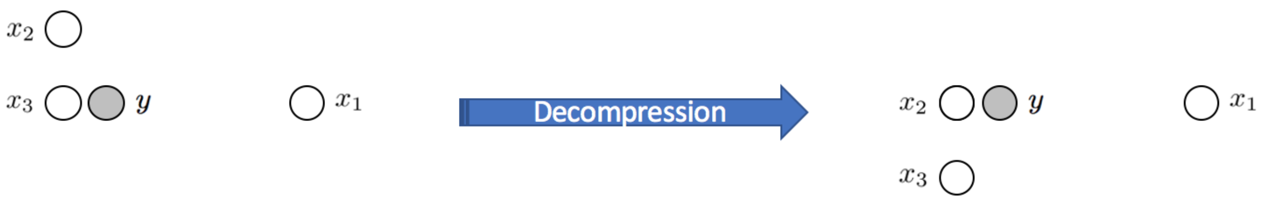


Figure 3-2: Compression and decompression of a dataset x_1, x_2, x_3 in the presence of a new query point y, which is unknown during compression. The same small perturbation in the location of $\{x_2, x_3\}$ as in Figure 3-1 fails to preserve x_3 as the nearest neighbor of y.

Within each cluster, we defined a *surrogate* for each point, which is a quantized representative close to the original point, up to an unknown shift. Even though the shifts are too expensive to store in the sketch, all surrogates in a cluster are guaranteed to share the same shift. Therefore, the distance between points in the same cluster can be estimated by the distance between their surrogates.

This reasoning breaks down if we need to handle new query points, for two reasons. First, without knowing the query points in advance, we cannot identify clusters which are well-separated from them. Thus, the path compression step may introduce arbitrarily large error. This is illustrated in Figure 3-2. Second, if we store the surrogates up to an unknown shift, we cannot estimate their distance to a new query point which is not similarly shifted. Therefore, we must include in the sketch some information about the absolute location of the clusters.

In this chapter we modify and augment the relative location tree in two ways, corresponding to the two difficulties described above. First, instead of replacing each non-branching path completely by a “long edge”, the sketch also stores a *suffix* of the path. Intuitively, this preserves some *local* information about the vicinity of each well-separated cluster, allowing us to handle query points that fall very close to it and avoid cases as depicted in Figure 3-2. Second, we store hashed quantizations of the unknown shifts of the clusters. While storing all shifts leads to a prohibitive sketch size, we use universal hashing to store “just enough” of them for each query, without knowing the queries in advance. We note that this modification makes the data structure inherently randomized; in contrast, the data structure of Section 2.3 was deterministic.

Nearest neighbor search. Given a query y, the search procedure is performed top down on the tree. Within each subtree, we attempt to reverse the hashed surrogate shift by enumerating over candidate shifts near y. If y is sufficiently near the cluster, then we are likely (up to hash collisions) to recover the correct shift. Once the unshifted surrogates in the current subtree are recovered, we find the one nearest to y. If it represents a singleton point (i.e., a leaf in the overall tree) then we report it as the nearest neighbor. Otherwise, it represents a well-separated cluster, and we proceed to explore it by descending into the corresponding subtree.

Now, one of two cases occurs. If y is sufficiently near the new cluster, then as above, we recover the shift correctly and iterate. However, if y is not near the cluster (meaning y lies in the “dead zone” between the well-separated cluster and the rest of the data points), then the search procedure recovers an incorrect shift (without knowing of it), and all of its future decisions would be based on wrong distance estimates. Nonetheless, note that in this case, y is far from a well-separated cluster that has been identified as closest to it in an upper level. This means that *all* points in that cluster are valid approximate nearest neighbors of y, and the algorithm can return anything.

Distance estimation. While the above query procedure computes distances from y to the surrogates, those distances are not valid estimates for the true distances between y and the data points represented by those surrogates. (In fact, by Theorem 3.1.5, this task is not possible within the space bound of Theorem 3.1.3.) Technically, the reason is that while the surrogate distance gives us an upper bound on the true distance, we have no lower bound. Indeed, as described in the previous paragraph, the search procedure does not know when it reaches the true distance from y to its nearest neighbors, and continues to try to move closer, thereby beginning to produce wrong distance estimates.

To remedy this, we augment the data structure with additional information that will help us tell between the above two cases above — i.e., whether y is near or far from its nearest well-separated cluster in every level. In particular, we use a distance sketch of [KOR00] in each well-separated cluster (subtree). Once we have identified the cluster of approximate nearest neighbors of y (which is when the above second case occurs for the first time), we estimate the distance to one representative of that cluster using the distance sketch of [Ach03]. These are intuitively the most difficult distances to estimate. Distances to farther data points can be estimated using the surrogates, as done in Chapter 2.

3.2 The Sketch

In this section we describe the basic data structure (generated by Alice) used for all both Theorem 3.1.3 and Theorem 3.1.4. For the latter result, we will augment it further in Section 3.4.

Recall that in the problems we consider now (Section 3.1.1), our data points and query points reside in the hypercube $[-\Phi,\Phi]^d$. Therefore, the maximal possible distance between any pair of points (even if one is Alice's and one is Bob's) is

$$\Delta^* = 2\sqrt{d}\Phi.$$

Since we consider Euclidean distances, we can begin with a preliminary application of a Johnson-Lindenstrauss transform (Theorem 1.2.1) in order to reduce the dimension to $O(\epsilon^{-2}\log(nq/\delta))$. This is guaranteed to preserve all distances between Alice's n points to Bob's q points, with probability at least $1-\delta$, up to distortion $1\pm\epsilon$. The coordinate range becomes $O(d\Phi)$ instead of Φ (where d is the original ambient dimension), which would not effect any of the bounds under discussion. We note, however, that the dimension reduction transform itself (i.e., the matrix M from Theorem 1.2.1) needs to be stored in Alice's sketch, in order for Bob to be able to apply it to his points as well. We will account for this storage when we compute the sketch size in Section 3.2.2.

3.2.1 Modified Relative Location Tree

Our starting point for the sketch is the relative location tree as constructed in Section 2.3 in the previous chapter. We will use the same construction and notation introduced there. We now describe how we modify and augment it.

Additional levels. The tree in Section 2.3 was built bottom-up by iteratively merging clusters, until all of the data points are merged into one cluster. Here, even after that happens, we continue adding levels to the top of the tree, until level $\lceil \log(\Delta^*) \rceil$. This means that the root may be the head of a 1-path.

Top-out compression. In Section 2.3, we replaced every maximal 1-path of the tree of length at least $\Lambda(v) = \log(\Delta(v)/(2^{\ell(v)}\epsilon))$, where v is the bottom node of the path, with a *long edge*. Here, we replace the path except for its bottom $\Lambda(v)$ nodes. More precisely, let $u_0, u_1, \ldots, u_k = v$ be a maximal 1-path in T^*, with $k > \Lambda(v)$. We connect u_0 directly to $u_{k-\Lambda(v)}$ by the long edge. The nodes $u_1, \ldots, u_{k-\Lambda(v)-1}$ are removed from the tree, and the long edge is annotated with length $k - \Lambda(v)$.

This modified path compression step endows the tree with the following property.

Lemma 3.2.1. *Let v be the bottom node of a long edge (i.e., v is the root of a subtree in $\mathcal{F}(T)$), and $x, x' \in C(v)$. Then $\|x - x'\| \leq 2^{\ell(v)}\epsilon$.*

Proof of Lemma 3.2.1. By top-out compression, v is the top of a downward 1-path of length $\Lambda(v')$, where v' is its bottom node. Since no clusters are joined along a 1-path, we have $C(v') = C(v)$, hence $x, x' \in C(v')$ and hence $\|x - x'\| \leq \Delta(v')$. Noting that

$$\ell(v) = \ell(v') + \Lambda(v') = \ell(v') + \log\left(\frac{\Delta(v')}{2^{\ell(v')}\epsilon}\right) = \log\left(\frac{\Delta(v')}{\epsilon}\right),$$

and rearranging, we find $\Delta(v') = 2^{\ell(v)}\epsilon$, which implies the claim. □

Importantly, top-out compression does not asymptotically change the size of the tree compared to the path compression step from Section 2.3. In particular, Lemma 2.3.4 still holds. To see this, note that its proof only requires that for every node v in the compressed tree T, the length of the maximal 1-path whose bottom node is v (denoted there $k(v)$) is at most $\Lambda(v)$. This property is preserved by top-out compression.

Grid quantization. Our modification to the surrogates, defined next, will use quantization onto grids. We now introduce appropriate notation. Let $\rho > 0$. In Section 2.2, we denoted by $\mathcal{G}^d[\rho/\sqrt{d}]$ the d-dimensional uniform grid with side length $\rho/\sqrt{d}$. Let us now denote,

$$\mathcal{Q}_\rho = \mathcal{G}^d[\rho/\sqrt{d}] \cap [-\Phi, \Phi]^d,$$

i.e., $\mathcal{Q}_\rho$ is the restriction of the grid to $[-\Phi, \Phi]^d$. Observe that $|\mathcal{Q}_\rho| = O(\rho^{-1}\sqrt{d}\Phi)^d = O(\rho^{-1}\Delta^*)^d$, and that each cell of $\mathcal{Q}_\rho$ has diameter at most ρ. For every $x \in \mathbb{R}^d$, we denote by $\mathcal{Q}_\rho(x)$ the closest point to x in $\mathcal{Q}_\rho$.

We restate Fact 2.2.2 in the above notation.

Fact 3.2.2. *There is a universal constant $\Gamma > 0$ such that the following holds. Let $x \in \mathbb{R}^d$ and $\rho > 0$. The number of points in $\mathcal{Q}_\rho$ which are at Euclidean distance at most 2ρ from x is at most Γ^d.*

Surrogates. We modify the definition of the surrogates in each subtree by replacing the base case in their inductive definition; the inductive step remains the same. In particular, fix a subtree $T' \in \mathcal{F}(T)$ rooted at r. In Section 2.3, we set the surrogate root to its true center, $s^*(r) = x_{c(r)}$, which could not be recovered from the sketch. Here, instead, we set it to its quantization onto a suitable grid, in order to facilitate recovery under certain conditions. To differentiate the surrogates in this chapter from those in the previous chapter, we denote them by $\hat{s}(v)$. Thus, the root surrogate is defined as:

$$\hat{s}(r) = \mathcal{Q}_{2^{\ell(r)}}(x_{c(r)}). \tag{3.1}$$

The remaining surrogates are defined by the same induction rule as Equation (2.4), i.e.,

$$\hat{s}(v) = \hat{s}(in(v)) + \frac{2^{\ell(v)}}{\gamma(v)} \cdot \eta(v).$$

We also define *leaf surrogates* $\hat{s}_\epsilon(v)$ for all $v \in \mathcal{L}(T)$ as in Equation (2.5), i.e.,

$$\hat{s}_\epsilon(v) = \hat{s}(in(v)) + \frac{2^{\ell(v)}}{\gamma(v)} \cdot \eta_\epsilon(v).$$

Lemma 3.2.3. *For every node v, $\|x_{c(v)} - \hat{s}(v)\| \leq 2^{\ell(v)}$. Furthermore, if $v \in \mathcal{L}(T)$, then $\|x_{c(v)} - \hat{s}_\epsilon(v)\| \leq 2^{\ell(v)}\epsilon$.*

Proof. We start with the (non-leaf) surrogates $\hat{s}(v)$. The proof is by induction on the ingress ordering in each subtree. Let $T' \in \mathcal{F}(T)$ be a subtree rooted at r. In the base case, $v = r$. By construction of the grids above, for every $\rho > 0$ and $x \in [-\Phi, \Phi]^d$ we have $\|x - \mathcal{Q}_\rho(x)\| \leq \rho$. With $\rho = 2^{\ell(r)}$ and $x = x_{c(r)}$, the claim for the base case follows from Equation (3.1). The induction step is the same as in Lemma 2.3.8.

The proof of the "furthermore" part (for leaf surrogates) is the same as Lemma 2.3.9. □

Root surrogate hashing. Ideally, we would have liked to store the surrogates of all subtree roots, which would allow us to recover all of the surrogates from the sketch. However, this is prohibitive: there can be as many as $\Omega(n)$ subtree roots in T, and each of their surrogates is quantized into about $d\log(\Delta^*)$ bits. This leads to same sketch size as discretized dimension reduction, which we are trying to improve.

Instead, we store hashes of the root surrogates. We will use an explicit construction of universal hash families. Such constructions are well-known, e.g., the classical Carter-Wegman construction [CW79] (see Section 1.2.4).

For every level ℓ in the tree, we pick a hash function H_ℓ from a universal family,

$$H_\ell : \mathcal{Q}_{2^\ell} \to [m] \ \text{ with } \ m = \left\lceil 10 \cdot \Gamma^d \cdot \log(\Delta^*) \cdot \frac{q}{\delta} \right\rceil,$$

where Γ is the constant from Fact 3.2.2. The hash function in each level is chosen independently at random from the other levels. For every subtree root r, we store its hashed surrogate $H_{\ell(r)}(\mathcal{Q}_{2^{\ell(r)}}(x_{c(r)}))$. We also store the description of each hash function H_ℓ for every level ℓ.

3.2.2 Sketch Size

Let us review the sketch. It contains the relative location tree T from Section 2.3, with the above modified path compression step (top-out compression). In addition, for every subtree root r, we store its hash quantized surrogate $H_{\ell(r)}(\mathcal{Q}_{2^{\ell(r)}}(x_{c(r)}))$, and the hash function description H_ℓ for every level ℓ.

Lemma 3.2.4. *The total sketch size in bits is*

$$O\left(n\left(\frac{\log(nq/\delta)\cdot\log(1/\epsilon)}{\epsilon^2}+\log\log(d\Phi)\right)+\left(\frac{\log^2(d\Phi)}{\epsilon^2}+\log\log\left(\frac{q}{\delta}\right)\right)\log\left(\frac{nq}{\delta}\right)\right).$$

Proof. The sketching algorithm starts with a preliminary dimension reduction step (Theorem 1.2.1). Alice needs to store the dimension reduction map in the sketch in order for Bob to apply it to his points as well. In order to preserve all nq distances between her points to his points, the target dimension needs to be $O(\epsilon^{-2}\log(nq/\delta))$. By [KMN11], the map can be stored in $O(\log d+\log(nq/\delta)\cdot\log\log((nq/\delta)/\epsilon))$ bits, where d is the ambient dimension. The effect of applying this map is that for the rest of the sketch, the dimension is reduced to $O(\epsilon^{-2}\log(qn/\delta))$, and the coordinate range increases from Φ to $O(d\Phi)$.

The cost of storing the relative location tree is $O(\epsilon^{-2}n\log(qn/\delta)\log(1/\epsilon)+n\log\log(\Delta^*))$ bits, by Lemma 2.3.11 (with dimension $O(\epsilon^{-2}\log(qn/\delta))$ and diameter Δ^*). Note that as explained above, top-out compression does not change this storage size.

By Claim 2.3.10, there are $O(n)$ subtree roots in the tree. For each one we store its surrogate hashed into a range of size m, where $m=\lceil O(1)^{\epsilon^{-2}\log(nq/\delta)}\cdot\log(\Delta^*)\cdot q/\delta\rceil$. Therefore their total storage size is $O(n(\epsilon^{-2}\log(nq/\delta)+\log\log(\Delta^*)+\log(q/\delta)))$ bits. This quantity is dominated by the storage cost of the relative location tree.

Finally, we store the description of the hash function H_ℓ in every level $\ell=1,\ldots,\lceil\log(\Delta^*)\rceil$. By Theorem 1.2.9, a hash function with domain U can be represented by $O(\log|U|)$ bits. In our case, H_ℓ has domain $\mathcal{Q}_{2^\ell}$, so the total storage size is $O(\sum_{\ell=1}^{\lceil\log(\Delta^*)\rceil}|\mathcal{Q}_{2^\ell}|)$ bits. Since

$$|\mathcal{Q}_{2^\ell}| = O(2^{-\ell}(\Delta^*)^{\epsilon^{-2}\log(nq/\delta)}) \le O((\Delta^*)^{\epsilon^{-2}\log(nq/\delta)}),$$

for $\ell \ge 1$, the total size is $O(\epsilon^{-2}\log(nq/\delta)\cdot\log^2(\Delta^*))$ bits.

The sketch size stated in the lemma follows by adding all of the above terms, recalling that $\Delta^* = O(d\Phi)$ (where d here is the ambient dimension), and simplifying by eliminating dominated terms. $\square$

Remark 3.2.5. Our model in Theorems 3.1.3 and 3.1.4 is constructive sketching without shared randomness. Therefore our sketch explicitly stores the random bits that generate the hash functions and of the projection map. If public randomness is allowed, these do not need to be stored. Alternatively, if one is only interested in the communication complexity, one can use the general reduction from public to private randomness due to [New91], which replaces the public coins by augmenting $O(\log(nd\Phi))$ bits to the sketch (since Alice's input has size $O(nd\Phi)$ bits). The bound in Theorem 3.1.3 then improves to

$$O\left(n\left(\epsilon^{-2}\log n\cdot\log(1/\epsilon)+\log\log(d\Phi)+\log\left(\frac{q}{\delta}\right)\right)+\log\Phi\right)\quad\text{bits,}$$

and the bound in Theorem 3.1.4 improves to

$$O\left(\epsilon^{-2}n\cdot\left(\log n\cdot\log(1/\epsilon)+\log(d\Phi)\log\left(\frac{q}{\delta}\right)\right)\right)\quad\text{bits.}$$

However, that reduction is non-constructive; we state our bounds so as to describe explicit sketches.

3.3 Approximate Nearest Neighbor Search

In this section we describe our approximate nearest neighbor search query procedure, and prove Theorem 3.1.3.

3.3.1 Query Algorithm

Suppose Bob wants to report a $(1+\epsilon)$-approximate nearest neighbor in X for a point $y \in Y$.

Algorithm Report Nearest Neighbor:

1. Start at the subtree $T' \in \mathcal{F}(T)$ that contains the root of T.
2. Recover all surrogates $\{\hat{s}(v) : v \in T'\}$, by the subroutine below.
3. Let v be the leaf of T' that minimizes $\|y - \hat{s}(v)\|$.
4. If v is the head of a long edge, recurse on the subtree under that long edge. Otherwise v is a leaf in T, and in that case return $c(v)$.

Subroutine Recover Surrogates: This is a subroutine that attempts to recover all surrogates $\{\hat{s}(v) : v \in T'\}$ in a given subtree $T' \in \mathcal{F}(T)$, using both Alice's sketch and Bob's point y.

Observe that to this end, the only information missing from the sketch is the root surrogate $\hat{s}(r)$, which served as the induction base for defining the rest of the surrogates. The induction steps are fully defined by $\ell(v)$, $\mathrm{in}(v)$, $\gamma(v)$, and $\eta(v)$, which are stored in the sketch for every node $v \neq r$ in the subtree. The missing root surrogate was defined as $\hat{s}(r) = \mathcal{Q}_{2^{\ell(r)}}(x_{c(r)})$. Instead, the sketch stores its hashed value $H_{\ell(r)}(\mathcal{Q}_{2^{\ell(r)}}(x_{c(r)}))$ and the hash function $H_{\ell(r)}$.

The subroutine attempts to reverse the hash. It enumerates over all points $p \in \mathcal{Q}_{2^{\ell(r)}}$ such that $\|p - y\| \leq 2 \cdot 2^{\ell(r)}$. For each p it computes $H_{\ell(r)}(p)$. If $H_{\ell(r)}(x_{c(r)}) = H_{\ell(r)}(p)$ then it sets $\hat{s}(r) = p$ and recovers all surrogates accordingly. If either no p, or more than one p, satisfy $H_{\ell(r)}(x_{c(r)}) = H_{\ell(r)}(p)$, then it proceeds with $\hat{s}(r)$ set to an arbitrary point (say, the origin in $\mathbb{R}^d$).

3.3.2 Analysis

Let $r_0, r_1, \ldots$ be the roots of the subtrees traversed by the algorithm.

Claim 3.3.1. $\|x_{c(r_0)} - y\| \leq 2^{\ell(r_0)}$.

Proof. Recall from Section 3.2 that $\|x_{c(r_0)} - y\| \leq \Delta^*$. Also, by the additional levels added to T in Section 3.2.1, $\ell(r_0) = \lceil \log(\Delta^*) \rceil$. The claim follows. $\square$

Let t be the smallest such that r_t satisfies $\|x_{c(r_t)} - y\| > 2^{\ell(r_t)}$. (The algorithm does not identify t, but we will use it for the analysis.)

Lemma 3.3.2. *With probability $1 - \delta/q$, for every $i = 0, \dots, t-1$ simultaneously, the subroutine recovers $\hat{s}(r_i)$ correctly as $\mathcal{Q}_{2^{\ell(r)}}(x_{c(r)})$. (Consequently, all surrogates in the subtree rooted by r_i are also recovered correctly.)*

Proof. Fix a subtree $T' \in \mathcal{F}(T)$ rooted in r, that satisfies $\|y - x_{c(r)}\| \leq 2^{\ell(r)}$. Since $\|x_{c(r)} - \hat{s}(r)\| \leq 2^{\ell(r)}$ by Lemma 3.2.3, we have $\|y - \hat{s}(r)\| \leq 2 \cdot 2^{\ell(r)}$. Hence the surrogate recovery subroutine tries $\hat{s}(r)$ as one of the hash pre-image candidates, and will identify that $H_{\ell(r)}(\hat{s}(r))$ matches the hash stored in the sketch. Furthermore, by Fact 3.2.2, the number of candidates is at most Γ^d (where $\Gamma = O(1)$). Since the range of $H_{\ell(r)}$ has size $m = \lceil 10 \cdot \Gamma^d \cdot \log(\Delta^*) \cdot q/\delta \rceil$, then by Claim 1.2.8 (with $k = \Gamma^d$ and $\eta = \delta/\lceil q \log(\Delta^*) \rceil$), with probability $1 - \delta/\lceil q \log(\Delta^*) \rceil$ there are no collisions, and $\hat{s}(r)$ is recovered correctly. The lemma follows by taking a union bound over the first t subtrees traversed by the algorithm, i.e. those rooted by r_i for $i = 0, 1, \dots, t-1$. Noting that t is upper-bounded by the number of levels in the tree, which is $\lceil \log(\Delta^*) \rceil$, we get that all the $\hat{s}(r_i)$'s are recovered correctly simultaneously with probability $1 - \delta/q$. $\square$

From now on we assume that the event in Lemma 3.3.2 succeeds, meaning in steps $0, 1, \dots, t-1$, the algorithm recovers all surrogates correctly. We henceforth prove that under this event, the algorithm returns a $(1+\epsilon)$-approximate nearest neighbor of y. In what follows, let $x^* \in X$ be a fixed true nearest neighbor of y in X.

Lemma 3.3.3. *Let $T' \in \mathcal{F}(T)$ be a subtree rooted in r, such that $x^* \in C(r)$. Let v a leaf of T' that minimizes $\|y - \hat{s}(v)\|$. Then either $x^* \in C(v)$, or every $z \in C(v)$ is a $(1 + O(\epsilon))$-approximate nearest neighbor of y.*

Proof. Suppose w.l.o.g. by scaling that $\epsilon < 1/6$. If $x^* \in C(v)$ then we are done. Assume now that $x^* \in C(u)$ for a leaf $u \neq v$ of T'. Let $\ell = \max\{\ell(v), \ell(u)\}$.

We start by showing that $\|y-x^*\| > \frac{1}{4}\cdot 2^\ell$. Assume by contradiction this is not the case. Since u is a subtree leaf and $x^* \in C(u)$, we have $\|x^* - x_{c(u)}\| \le 2^\ell\epsilon$ by Lemma 2.3.5. We also have $\|x_{c(u)} - \hat{s}(u)\| \le 2^\ell\epsilon$ by Lemma 3.2.3. Together,

$$\|y - \hat{s}(u)\| \le (\tfrac{1}{4} + 2\epsilon)2^\ell.$$

On the other hand, by the triangle inequality,

$$\|y - \hat{s}(v)\| \ge \|x^* - x_{c(v)}\| - \|y - x^*\| - \|x_{c(v)} - \hat{s}(v)\|.$$

Noting that

- $\|x^* - x_{c(v)}\| \ge 2^\ell$, by Lemma 2.3.1, since x^* and $x_{c(v)}$ are separated at level ℓ,
- $\|y - x^*\| \le \frac{1}{4}\cdot 2^\ell$, by the contradiction hypothesis,
- $\|x_{c(v)} - \hat{s}(v)\| \le 2^\ell\epsilon$, by Lemma 3.2.3,

we get

$$\|y - \hat{s}(v)\| \ge (\tfrac{3}{4} - \epsilon)2^\ell > (\tfrac{1}{4} + 2\epsilon)2^\ell \ge \|y - \hat{s}(u)\|.$$

This contradicts the choice of v. Thus, $\|y - x^*\| > \frac{1}{4}\cdot 2^\ell$.

The lemma now follows because for every $z \in C(v)$,

$$\begin{aligned}
\|y - z\| &\le \|y - \hat{s}(v)\| + \|\hat{s}(v) - x_{c(v)}\| + \|x_{c(v)} - z\| && (3.2)\\
&\le \|y - \hat{s}(u)\| + \|\hat{s}(v) - x_{c(v)}\| + \|x_{c(v)} - z\| && (3.3)\\
&\le \|y - x^*\| + \|x^* - x_{c(u)}\| + \|x_{c(u)} - \hat{s}(u)\| + \|\hat{s}(v) - x_{c(v)}\| + \|x_{c(v)} - z\| && (3.4)\\
&\le \|y - x^*\| + 4\cdot 2^\ell\epsilon && (3.5)\\
&\le (1+16\epsilon)\|y - x^*\|, && (3.6)
\end{aligned}$$

where (3.2) and (3.4) are by the triangle inequality, (3.3) is since $\|y - \hat{s}(v)\| \le \|y - \hat{s}(u)\|$ by choice of v, (3.5) is by Lemmas 2.3.5 and 3.2.3, and (3.6) is since we have shown that $\|y - x^*\| > \frac{1}{4}\cdot 2^\ell$. Therefore z is a $(1+16\epsilon)$-approximate nearest neighbor of y. □

Proof of Theorem 3.1.3. We may assume w.l.o.g. that ϵ is smaller than a sufficiently small constant. Suppose that the event in Lemma 3.3.2 holds, hence all surrogates in the subtrees rooted by $r_0, r_1, \ldots, r_{t-1}$ are recovered correctly. We consider two cases.

In the first case, $x^* \notin C(r_t)$. Let $i \in \{1, \ldots, t\}$ be the smallest such that $x^* \notin C(r_i)$. By applying Lemma 3.3.3 on r_{i-1}, we have that every point in $C(r_i)$ is a $(1+O(\epsilon))$-approximate nearest neighbor of y. After reaching r_i, the algorithm would return the center of some leaf reachable from r_i, and it would be a correct output.

In the second case, $x^* \in C(r_t)$. We will show that every point in $C(r_t)$ is a $(1+O(\epsilon))$-approximate nearest neighbor of y, so once again, once the algorithm arrives at r_t it can return anything. By Lemma 3.2.1, every $x \in C(r_t)$ satisfies

$$\|x - x^*\| \leq 2^{\ell(r_t)}\epsilon. \tag{3.7}$$

In particular, $\|x_{c(r_t)} - x^*\| \leq 2^{\ell(r_t)}\epsilon$. By definition of t we have $\|x_{c(r_t)} - y\| > 2^{\ell(r_t)}$. Applying the triangle inequality and then the two latter bounds, we get

$$\|y - x^*\| \geq \|y - x_{c(r_t)}\| - \|x_{c(r_t)} - x^*\| > (1-\epsilon)2^{\ell(r_t)}.$$

Combining this with eq. (3.7), we find that every $x \in C(r_t)$ satisfies $\|x - x^*\| \leq \frac{\epsilon}{1-\epsilon}\|y - x^*\|$, and hence (for $\epsilon \leq 1/2$),

$$\|y - x\| \leq \|y - x^*\| + \|x^* - x\| \leq (1+2\epsilon)\|y - x^*\|.$$

Thus x is a $(1+2\epsilon)$-nearest neighbor of y.

The proof assumes the occurrence of the event in Lemma 3.3.2, which occurs with probability $1 - \delta/q$. By a union bound, the simultaneous success probability of the q query points of Bob is $1-\delta$ as required. Finally, ϵ can be scaled down by a constant. □

3.4 Distance Estimation

In this section we prove Theorem 3.1.4.

3.4.1 Sketch Augmentations

We augment the sketch from Section 3.2 with additional information, relying on the following distance sketches due to Achlioptas [Ach03] (following [JL84]) and Kushilevitz, Ostrovsky and Rabani [KOR00].

Lemma 3.4.1 (discretized JL [Ach03])**.** *Let $\epsilon, \delta' > 0$. Let $d' = c\epsilon^{-2}\log(1/\delta')$ for a sufficiently large constant $c > 0$. Let M be a random $d' \times d$ matrix in which every entry is chosen independently uniformly at random from $\{-1/\sqrt{d'}, 1/\sqrt{d'}\}$. Then, for every $x, y \in \mathbb{R}^d$, with probability $1-\delta'$,*

$$(1-\epsilon)\|x-y\| \le \|Mx - My\| \le (1+\epsilon)\|x-y\|.$$

Lemma 3.4.2 ([KOR00])**.** *Let $R > 0$ be fixed and let $\epsilon, \delta'' > 0$. There is a randomized map sk_R of vectors in $\mathbb{R}^d$ into $O(\epsilon^{-2}\log(1/\delta''))$ bits, with the following guarantee. For every $x, y \in \mathbb{R}^d$, given $\mathrm{sk}_R(x)$ and $\mathrm{sk}_R(y)$, one can output the following with probability $1-\delta''$:*

- *If $\|x-y\| \le (1-\epsilon)R$, output "Small".*
- *If $\|x-y\| \ge R$, output "Large".*

We augment the sketch from Section 3.2 as follows. We sample a matrix M from Lemma 3.4.1, with $\delta' = \delta/q$. In addition, for every level ℓ in the tree T, we sample a map sk_{2^ℓ} from Lemma 3.4.2, with $\delta'' = \delta/(q\lceil\log(\Delta^*)\rceil)$. The maps of the different levels are sampled independently. For every subtree root r in T, we store $Mx_{c(r)}$ and $\mathrm{sk}_{2^{\ell(r)}}(x_{c(r)})$ in the sketch. We also store the matrix M itself, and the description of each map sk_{2^ℓ} for every ℓ.

Total sketch size. Let us calculate the number of bits we have added to the sketch.

- Let r be a subtree root. Recall that its center $x_{c(r)}$ has d integer coordinates in $[-\Phi, \Phi]$. By Lemma 3.4.1, each of the d' coordinates of $Mx_{c(r)}$ is the sum of the coordinates of $x_{c(r)}$ (with random signs and times a constant $1/\sqrt{d'}$), so each can be represented by $O(d\log\Phi)$ bits. There are at most $2n+1$ subtree roots by Lemma 2.3.10(i), since every subtree root is either the root of T or the bottom node of a long edge. Therefore, storing $Mx_{c(r)}$ for every r adds $O(nd' \log(d\Phi))$ bits to the sketch. Since $d' = O(\epsilon^{-2}\log(1/\delta'))$ and $\delta' = \delta/q$, this adds $O(\epsilon^{-2}n \cdot \log(q/\delta) \cdot \log(d\Phi))$ bits to the sketch.

- The matrix M itself can be stored, by [KMN11], using $O(\log d + \log(q/\delta) + \log(\log(q/\delta)\epsilon))$ bits.

- By Lemma 3.4.2, storing $\mathrm{sk}_{2^{\ell(r)}}(x_{c(r)})$ for a subtree root r takes $O(\epsilon^{-2}\log(1/\delta''))$ bits. As stated above there are at most $O(n)$ subtree roots, and $\delta'' = \delta/(q\lceil\log(\Delta^*)\rceil)$, where we recall that $\Delta^* = O(d\Phi)$. Altogether, this adds $O(\epsilon^{-2}n(\log(q/\delta) + \log\log(d\Phi)))$ bits to the sketch.

- Storing the description of the maps can be made to take $\epsilon^{-5} \cdot \mathrm{polylog}(q, d, \Phi, \delta^{-1}, \epsilon^{-1})$ bits. This follows from known reductions, and we prove it in Claim 3.4.5 below for completeness.

Note that we can assume w.l.o.g. that $q \le n$, since otherwise, the known discretized dimension reduction sketching bound already yields Theorem 3.1.4. (This only serves to remove the term $\epsilon^{-2}n\log(q)\log(1/\epsilon)$ from the bound.) In total, we get the sketch size stated in Theorem 3.1.4.

3.4.2 Query Algorithm

Given the sketch, an index $k \in [n]$ of a point in X, and a new query point y, the algorithm needs to estimate $\|y - x_k\|$ up to $1 \pm O(\epsilon)$ distortion. It proceeds as follows.

1. Perform the approximate nearest neighbor query algorithm from Section 3.3. Let $r_0, r_1, \ldots$ be the downward sequence of subtree roots traversed by it.

2. For each r_j, estimate from the sketch whether $\|y - x_{c(r_j)}\| \le 2^{\ell(r_j)}$. This can be done by Lemma 3.4.2, since the sketch stores $\mathrm{sk}_{2^{\ell(r_j)}}(x_{c(r_j)})$ and also the map $\mathrm{sk}_{2^{\ell(r_j)}}$, with which we can compute $\mathrm{sk}_{2^{\ell(r_j)}}(y)$.

3. Let t be the smallest j that satisfies $\|y - x_{c(r_j)}\| > 2^{\ell(r_j)}$ according the estimates of Lemma 3.4.2. (This attempts to recover from the sketch the same t as defined in the

analysis in Section 3.3.)

4. Let $t_k \in \{0, \ldots, t\}$ be the maximal such that $x_k \in C(r_{t_k})$.

   (In words, r_{t_k} is the root of the subtree in which x_k and y "part ways".)

5. If $t_k = t$, return $\|My - Mx_{c(r_t)}\|$. Note that M and $Mx_{c(r_t)}$ are stored in the sketch.

6. If $t_k < t$, let v_k be the bottom node on the downward path from r_{t_k} to leaf(x_k) that does not traverse a long edge. Return $\|y - \hat{s}(v_k)\|$.

3.4.3 Analysis

Fix a query point y. Define the "good event" $\mathcal{A}(y)$ as the intersection of the following:

1. For every subtree root r_j traversed by the query algorithm above, the invocation of Lemma 3.4.2 on $\mathrm{sk}_{2^{\ell(r_j)}}(x_{c(r_j)})$ and $\mathrm{sk}_{2^{\ell(r_j)}}(y)$ succeeds in deciding whether $\|y - x_{c(r_j)}\| \leq 2^{\ell(r_j)}$. Specifically, this ensures that that choice of t by the query algorithm satisfies both

$$\|y - x_{c(r_j)}\| \leq 2^{\ell(r_j)} \text{ for every } j < t, \tag{3.8}$$

   and

$$\|y - x_{c(r_t)}\| \geq (1 - \epsilon)2^{\ell(r_t)}. \tag{3.9}$$

   Recalling that we invoked the lemma with $\delta' = \delta/(q\lceil\log(\Delta^*)\rceil)$, we can take a union bound and succeed in all levels simultaneously with probability $1 - \delta/q$.

2. $\|My - Mx_{c(r_t)}\| = (1 \pm \epsilon)\|y - x_{c(r_t)}\|$. By Lemma 3.4.1 this holds with probability $1 - \delta/q$.

Therefore,

Claim 3.4.3. *$\mathcal{A}(y)$ occurs with probability $1 - O(\delta/q)$.*

Furthermore,

Lemma 3.4.4. *Conditioned on the occurrence of $\mathcal{A}(y)$, Lemma 3.3.2 holds with probability $1 - \delta/q$. Namely, the query algorithm correctly recovers all surrogates in the subtrees rooted by r_j for $j = 0, 1, \ldots, t - 1$.*

Proof. The proof of Lemma 3.3.2 in Section 3.3 relied on having $\|y - x_{c(r_j)}\| \le 2^{\ell(r_j)}$ for every $j < t$. Conditioning on $\mathcal{A}(y)$ (Equation (3.8)) ensures this holds. $\square$

Proof of Theorem 3.1.4. Let $\mathcal{A}^*(y)$ denote the event in which both $\mathcal{A}(y)$ occurs and the conclusion of Lemma 3.3.2 occurs. By Claim 3.4.3 and Lemma 3.4.4, $\mathcal{A}^*(y)$ happens with probability $1 - O(\delta/q)$. From now on we will assume that $\mathcal{A}^*(y)$ occurs, and conditioned on this, we will show that the distance from y to any data point can be deterministically estimated correctly.

To this end, fix $k \in [n]$ and suppose our goal is to estimate $\|y - x_k\|$. Let t_k and v_k be as defined by the distance query algorithm above. We handle the two cases of the algorithm separately.

Case I: $t_k = t$. This means $x_k \in C(r_t)$. By Lemma 3.2.1 we have $\|x_k - x_{c(r_t)}\| \le 2^{\ell(r_t)}\epsilon$. By the occurence of $\mathcal{A}^*(y)$ (Equation (3.9)), we have $\|y - x_{c(r_t)}\| > (1-\epsilon)2^{\ell(r_t)}$. Together, by the triangle inequality,

$$\|y - x_k\| \le \|y - x_{c(r_t)}\| + \|x_k - x_{c(r_t)}\| \le \|y - x_{c(r_t)}\| + 2^{\ell(r_t)}\epsilon \le (1 + \tfrac{\epsilon}{1-\epsilon})\|y - x_{c(r_t)}\|,$$

and similarly

$$\|y - x_k\| \ge \|y - x_{c(r_t)}\| - \|x_k - x_{c(r_t)}\| \ge \|y - x_{c(r_t)}\| - 2^{\ell(r_t)}\epsilon \ge (1 - \tfrac{\epsilon}{1-\epsilon})\|y - x_{c(r_t)}\|.$$

These mean that $\|y - x_{c(r_t)}\|$ is a $(1 \pm O(\epsilon))$ estimate (if, say, $\epsilon < 1/2$) for $\|y - x_k\|$. Since $\mathcal{A}^*(y)$ occurs, it holds that $\|My - Mx_{c(r_t)}\| = (1 \pm \epsilon)\|y - x_{c(r_t)}\|$, hence $\|My - Mx_{c(r_t)}\|$ is also a good estimate for $\|y - x_k\|$, and this is what the algorithm returns.

Case II: $t_k < t$. Let T'_{t_k} be the subtree rooted by r_{t_k}. By the occurence of $\mathcal{A}^*(y)$, all surrogates in T'_{t_k} are recovered correctly, and in particular $\hat{s}(v_k)$ is recovered correctly. By Lemma 3.2.3 we have $\|x_{c(v_k)} - \hat{s}(v_k)\| \le 2^{\ell(v_k)}\epsilon$, and by Lemma 2.3.5 (noting that $x_k \in C(v_k)$ by choice of v_k) we have $\|x_k - x_{c(v_k)}\| \le 2^{\ell(v_k)}\epsilon$. Together,

$$\|x_k - \hat{s}(v_k)\| \le 2 \cdot 2^{\ell(v_k)}\epsilon. \tag{3.10}$$

Let v be the leaf in T'_{t_k} that minimizes $\|y-\hat{s}(v)\|$ (over all leaves of T'_{t_k}). Equivalently, v is the top node of the long edge whose bottom node is r_{t_k+1}. Let $\ell = \max\{\ell(v), \ell(v_k)\}$. By choice of t_k we have $v \neq v_k$, hence the centers of these two leaves are separated already at level ℓ, hence $\|x_{c(v_k)} - x_{c(v)}\| \geq 2^\ell$ by Lemma 2.3.1. By two applications of Lemma 3.2.3 we have $\|x_{c(v_k)} - \hat{s}(v_k)\| \leq 2^\ell\epsilon$ and $\|x_{c(v)} - \hat{s}(v)\| \leq 2^\ell\epsilon$. Putting the last three bounds together, we have by the triangle inequality,

$$\|\hat{s}(v_k) - \hat{s}(v)\| \geq \|x_{c(v_k)} - x_{c(v)}\| - \|x_{c(v_k)} - \hat{s}(v_k)\| - \|x_{c(v)} - \hat{s}(v)\| \geq (1-2\epsilon)\cdot 2^\ell.$$

Since y is closer to $\hat{s}(v)$ than to $\hat{s}(v_k)$ (by choice of v), we have

$$\|y - \hat{s}(v_k)\| \geq \frac{1}{2}\cdot\|\hat{s}(v_k) - \hat{s}(v)\| \geq \left(\frac{1}{2} - \epsilon\right)\cdot 2^\ell.$$

Combining this with Equation (3.10) yields

$$\|x_k - \hat{s}(v_k)\| \leq 2\epsilon\cdot\frac{1}{1/2-\epsilon}\cdot\|y-\hat{s}(v_k)\| = O(\epsilon)\cdot\|y-\hat{s}(v_k)\|.$$

Therefore,

$$\|y - x_k\| = \|y-\hat{s}(v_k)\| \pm \|x_k - \hat{s}(v_k)\| = (1\pm O(\epsilon))\cdot\|y-\hat{s}(v_k)\|,$$

which means that $\|y-\hat{s}(v_k)\|$ is a good estimate for $\|y-x_k\|$, and this is what the algorithm returns.

Conclusion. Combining both cases, we have shown that for any query point y, all distances from y to X can be estimated correctly with probability $1-O(\delta/q)$. Taking a union bound over q queries, and scaling δ and ϵ appropriately by a constant, yields the theorem. □

3.4.4 Appendix: Explicit Storage of KOR Maps

In this section we argue that the description of a sketching map from Lemma 3.4.2 can be stored explicitly with a small number of bits, in order to obtain explicit sketches without shared randomness (see Remark 3.2.5). This entails storing an explicit embedding transform into ℓ_1, and then applying a unary embedding with small target dimension (whereas a naïve unary embedding would lead to dimension linear in Φ, which is exponential in the input description size). Both follow from known techniques: the former from [Ind06], and the latter from [HPIM12, Lemmas 3.1 and 3.2]. For completeness, we include an explicit account.

Claim 3.4.5. *A sketching map from Lemma 3.4.2 can be stored in $\epsilon^{-5}\cdot\mathrm{polylog}(d,\Phi,1/\delta'',1/\epsilon)$ bits (where d is the ambient dimension of the input points, Φ is their coordinates range, and ϵ,δ'' are the relative error and success probability, respectively, from Lemma 3.4.2).*

Proof. Consider a map sk_R from Lemma 3.4.2. The distance sketch of [KOR00] goes by embedding Euclidean distances into ℓ_1 and then into Hamming space. We review the steps and bound their storage size.

1. Apply a Johnson-Lindenstrauss transform (Theorem 1.2.3) to reduce the Euclidean dimension to $O(\epsilon^{-2}\log(1/\delta''))$. By [KMN11], the transform can be represented by polylogarithmically many bits.

2. Apply Theorem 1.2.4, again with dimension

$$d'' = O(\epsilon^{-2}\log(1/\delta'')),$$

   to embed the distances into ℓ_1. The transform matrix has $O(\epsilon^{-4}\log^2(1/\delta''))$ entries, and by [Ind06], it suffices to represent each one with logarithmically many bits.

3. Round each point coordinate to an integer multiple of $\epsilon R/d''$. This changes each distance by at most an additive $\pm O(\epsilon R)$, which we can allow.

4. Impose on the points a uniform grid with side length R/δ'', shifted by an i.i.d. uniformly random shift in each dimension. There are

$$M = \frac{R/\delta''}{\epsilon R/d''} = \frac{d''}{\epsilon\delta''}$$

   possible shift values in each dimension, so the total storage size of the shifts is $d''\log M$ bits.

5. Within each grid cell, by shifting and scaling, we may assume w.l.o.g. that the coordinates are integers between 0 to M. Embed the points in each cell into a $(d''M)$-dimensional Hamming space $\{0,1\}^{d''M}$, by the unary embedding, which transforms a coordinate with value k into a string of k ones followed by $M-k$ zeros. Observe that the embedding is isometric (w.r.t. ℓ_1-distances in the source space).

6. The [KOR00] sketch in the Hamming case is defined by $d''M$ bits per bit in the sketch, so in total, $O(d''M \cdot \epsilon^{-2}\log(1/\delta'')) = (\delta'')^{-1}\epsilon^{-5}\log^3(1/\delta'')$ bits. The inverse-linear dependence on the success probability δ'' can be decreased to $O(\log(1/\delta''))$ by a standard technique (reporting the majority output of $O(\log(1/\delta''))$ independent repetitions, each run with constant success probability).

Thus the bit size of representing the map sk_R is as stated in the lemma. We need to argue that the above transformations preserve (i) the sketch size per point and (ii) the success probability.

For (i), given an input point $x \in \mathbb{R}^d$, we augment its sketch $\mathrm{sk}_R(x)$ with the ID of the grid cell that contains it, hashed into one of $O(1/\delta'')$ buckets by universal hashing (Section 1.2.4). This only adds $O(\log(1/\delta''))$ bits to the sketch and does not change its asymptotic size. We also need to add the description of the universal hash function to the description of sk_R. If x has integer coordinates in $[-\Phi,\Phi]$, then there are at most $O(d\Phi)^{d''}$ grid cells, and by Theorem 1.2.9, a universal hash function costs $O(d''\log(d\Phi)) = O(\epsilon^{-2}\log(1/\delta'')\log(d\Phi))$ bits to store.

For (ii), when we compare $\mathrm{sk}_R(x)$ and $\mathrm{sk}_R(y)$ for a pair x, y, if their hashed cell IDs are different we report "Large", and otherwise we report the output of the [KOR00] sketch. If $\|x-y\| \le R$, then, since the grid side is R/δ'', it is not hard to see they fall in different cells of the uniformly shifted grid with probability at most $O(\delta'')$. On the other hand, if $\|x-y\| \ge (1+\epsilon)R$ and x, y fall in different cells, the probability that their hashed cell IDs are the same is at most δ'' by the universal hashing property. Thus the total failure probability added to Lemma 3.4.2 by the above transformations is $O(\delta'')$. $\square$

3.5 Lower Bound for Nearest Neighbor Search

In this section we provide a lower bound for the all-nearest-neighbors problem, matching the upper bound in Theorem 3.1.3 up to a factor of $O(\log(1/\epsilon))$, in the regime $\delta < 1/n^2$. We recall that δ is the success probability of the sketch in reporting an approximate nearest neighbor for a single unknown query, so this assumption essentially means that the sketch is required to succeed on n^2 queries simultaneously.

Theorem 3.5.1. *Suppose that $d \geq \Omega(\epsilon^{-2}\log n)$, $\Phi \geq 1/\epsilon$, and $1/n^{0.5-\beta} \leq \epsilon \leq \epsilon_0$ for a constant $\beta > 0$ and a sufficiently small constant ϵ_0. Suppose also that $\delta < 1/n^2$. Then, for the all-nearest-neighbors problem, Alice must use a sketch of at least $\Omega(\beta\epsilon^{-2}n\log n)$ bits.*

Proof. We start with dimension $d = n + 1 + \log n$; it can then be reduced by standard dimension reduction. Fix $k = 1/\epsilon^2$ and assume w.l.o.g. that k is a square integer (by taking ϵ to be appropriately small). Note that since $\epsilon > 1/\sqrt{n}$ we have $k \leq n$, and that since $\Phi \geq 1/\epsilon$ we have $\sqrt{k} \leq \Phi$.

The data set will consist of $2n$ points, $x_1, \ldots, x_n$ and $z_1, \ldots, z_n$. Let $i \in [n]$. We choose the first n coordinates of x_i to be an arbitrary k-sparse vector, in which each nonzero coordinate equals $1/\sqrt{k}$. Note that the norm of this part is 1. The $(n+1)$th coordinate of x_i is set to 0. The remaining $\log n$ coordinates encode the binary encoding of i, with each coordinate multiplied by 10.

Next we define z_i. The first n coordinates are 0. The $(n+1)$th coordinate equals $\sqrt{1-\epsilon}$. The remaining $\log n$ coordinates encode i similarly to x_i.

The number of different choices for $\{x_1, \ldots, x_n\}$ is $\binom{n}{k}^n$. Therefore if we show that one can fully recover $x_1, \ldots, x_n$ from a given all-nearest-neighbor sketch of the dataset, we would get the desired lower bound

$$\log\left(\binom{n}{k}^n\right) \geq nk\log(n/k) = \epsilon^{-2}n\log(\epsilon^2 n) = \epsilon^{-2}n\log(n^{2\beta}) = 2\beta\epsilon^{-2}n\log n.$$

Suppose we have such a sketch. For given $i, j \in [n]$ we now show how to recover the jth coordinate of x_i, denoted $x_i(j)$, with a single approximate nearest neighbor query. Let y_{ij} be the following vector in $\mathbb{R}^d$: The first $n+1$ coordinates are all zeros, except for the jth coordinate which is set to 1. The last $\log n$ coordinates encode i similarly to x_i and z_i.

Consider the distances from y_{ij} to all data points. We start with x_i. It is identical to y_i in the last $\log n + 1$ coordinates, so we will restrict both to the first n coordinates and denote the restricted vectors by $x_i^{:n}$ and $y_{ij}^{:n}$. $x_i^{:n}$ is a k-sparse vector with nonzero entries equal to $1/\sqrt{k}$, hence $\|x_i^{:n}\| = 1$. $y_{ij}^{:n}$ is just the standard basis vector e_j in $\mathbb{R}^n$. Hence,

$$\|x_i - y_{ij}\|^2 = \|x_i^{:n} - y_{ij}^{:n}\|^2 = \|x_i^{:n}\|^2 + \|y_{ij}^{:n}\|^2 - 2(x_i^{:n})^T y_{ij}^{:n} = 2 - 2x_i(j).$$

This equals 2 if $x_i(j) = 0$ and $2 - 2/\sqrt{k} = 2 - 2\epsilon$ if $x_i(j) = 1/\sqrt{k}$.

Next consider z_i. It is identical to y_{ij} in all except the jth coordinate, which is 0 in z_i and 1 in y_{ij}, and the $(n+1)$th coordinate, which is 0 for y_{ij} and $\sqrt{1-\epsilon}$ for z_i. Therefore, $\|z_i - y_{ij}\|^2 = 2 - \epsilon$.

Finally, for every $i' \neq i$, both $x_{i'}$ and $z_{i'}$ are at distance at least 10 from y_{ij} due to the encoding of i (as binary multiplied by 10) in the last $\log n$ coordinates.

In summation we have established the following:

- If $x_i(j) \neq 0$, then the closest point to y_{ij} in the dataset is x_i at distance $\sqrt{2-2\epsilon}$, and the next closest point is z_i at distance $\sqrt{2-\epsilon}$.

- If $x_i(j) = 0$, then the closest point to y_{ij} in the dataset is z_i at distance $\sqrt{2-\epsilon}$, and the next closest point is x_i at distance 2.

Therefore, if the sketch supports $(1+\frac{1}{8}\epsilon)$-approximate nearest neighbors, we can recover the true nearest neighbor of y_{ij} and thus recover $x_i(j)$. By hypothesis, the query succeeds with probability $\delta < 1/n^2$. By a union bound over all $i, j \in [n]$ we can recover all of $x_1, \ldots, x_n$ simultaneously, and the theorem follows. $\square$

3.6 Lower Bound for Distance Estimation

In this section we prove Theorem 3.1.5. The proof is along the lines of [MWY13], who proved (among other results) this statement for the case $q = n$. We describe how to adapt their framework to our problem, and refer to [MWY13] for missing details that remain similar.

3.6.1 Preliminaries: The Augmented Indexing Problem

The proof follows the approach of [JW13], of proving one-way communication lower bounds by reduction to variants of the augmented indexing problem, defined next.

Definition 3.6.1 (Augmented Indexing)**.** *In the Augmented Indexing problem $AugInd(k, \delta)$, Alice gets a vector A with k entries, whose elements are entries of a universe of size $20/\delta$. Bob gets an index $i \in [k]$, an element e, and the elements $A(i')$ for every $i' < i$. Bob needs to decide whether $e = A(i)$, and succeed with probability $1 - \delta$.*

Jayram and Woodruff [JW13] give a one-way communication lower bound of $\Omega(k \log(1/\delta))$ for this problem. The main component in [MWY13] is a modified one-way communication model, in which the protocol is allowed to abort with a substantially larger (constant) probability than it is allowed to err. We will call it simply *the abortion model* and refer to [MWY13] for the exact definition (which we will not require). They prove the same lower bound for Augmented Indexing in abortion model as in the usual one-way communication model.

Lemma 3.6.2 ([MWY13])**.** *In the abortion model, the one-way communication complexity of $AugInd(k, \delta)$ is $\Omega(k \log(1/\delta))$.*

3.6.2 Variants of Augmented Indexing

We start by defining a variant of augmented indexing that will be suitable for our purpose.

Definition 3.6.3. *In the Matrix Augmented Indexing problem $MatAugInd(k, m, \delta)$, Alice gets a matrix A of order $k \times m$, whose entries are elements of a universe of size $1/\delta$. Bob gets indices $i \in [k]$ and $j \in [m]$, an element e, and the elements $A(i, j')$ for every $j < j'$. Bob needs to decide whether $e = A(i, j)$, and succeed with probability $1 - \delta$.*

This problem is clearly at least as difficult as $AugInd(km, \delta)$ from Definition 3.6.1, since in the latter Bob gets more information (namely, if we arrange the vector A in $AugInd(km, \delta)$ as a $k \times m$ matrix, then Bob gets all entries of A which lexicographically precede $A(i, j)$). We get the following immediate corollary from Lemma 3.6.2.

Corollary 3.6.4. *In the abortion model, the one-way communication complexity of the $MatAugInd(k, m, \delta)$ problem is $\Omega(km \log(1/\delta))$.*

[MWY13] reformulate Augmented Indexing so that Alice's input is a set instead of vector. Similarly, we reformulate Matrix Augmented Indexing as follows.

Definition 3.6.5. *Let $m > 0$ and $k > 0$ be integers, and $\delta \in (0, 1)$. Partition the interval $[m/\delta]$ into m intervals $I_1, \ldots, I_m$ of size $1/\delta$ each.*

In the Augmented Set List problem AugSetList(k, m, δ), Alice gets a list of subsets $S_1, \ldots, S_k \subset [m/\delta]$, such that each S_i has size exactly m and contains exactly one element from each interval $I_1, \ldots, I_m$. Bob gets an index $i \in [k]$, an element $e \in [m/\delta]$ and a subset T of S_i that contains exactly the elements of S_i that are smaller than e. Bob needs to decide whether $e \in S_i$, and succeed with probability at least $1 - \delta$.

The equivalence to Matrix Augmented Indexing is not hard to show; the details are similar to [MWY13] and we omit them here. By the equivalence, we get the following corollary from Corollary 3.6.4.

Corollary 3.6.6. *In the abortion model, the one-way communication complexity of the AugSetList(k, m, δ) problem is $\Omega(km \log(1/\delta))$.*

Next we define the q-fold version of the same problem.

Definition 3.6.7. *In the problem q-AugSetList(k, m, δ), Alice and Bob get q instances of AugSetList$(k, m, \delta/q)$, and Bob needs to answer correctly on all of them simultaneously with probability at least $1 - \delta$.*

The main technical result of [MWY13] is, loosely speaking, a direct-sum theorem which lifts a lower bound in the abortion model to a q-fold lower bound in the usual model. Applying their theorem to Corollary 3.6.6, we obtain the following.

Corollary 3.6.8. *The one-way communication complexity of the q-AugSetList(k, m, δ) problem is $\Omega(qkm \log(q/\delta))$.*

Finally, we construct a "generalized augmented indexing" problem over r copies of the above problem.

Definition 3.6.9. *In the problem r-Ind(q-AugSetList(k, m, δ)), Alice gets r instances, $A_1, \ldots, A_r$, of q-AugSetList(k, m, δ). Bob gets an index $j \in [r]$, his part B_j of instance*

j, and Alice's instances $A_1, \ldots, A_{j-1}$. Bob needs to solve instance j with success probability at least $1-\delta$.

By standard direct sum results in communication complexity (see [MWY13]) we obtain from Corollary 3.6.8 the final lower bound we need.

Lemma 3.6.10. *The one-way communication complexity of the r-Ind(q-AugSetList(k, m, δ)) problem is $\Omega(rqkm \log(q/\delta))$.*

3.6.3 Lower bound by Number of Queries

We now prove the following term of the lower bound in Theorem 3.1.5

Lemma 3.6.11. *Suppose that $d^{1-\rho} \geq \epsilon^{-2} \log(q/\delta)$ for a constant $\rho > 0$, and ϵ is at most a sufficiently small constant. Then, for the all-cross-distances problem, Alice must use a sketch of at least $\Omega(\epsilon^{-2} n \log(d\Phi) \log(q/\delta))$ bits.*

The proof of Lemma 3.6.11 is by reducing r-*Ind*(q-*AugSetList*(k, m, δ)) to the all-cross-distances problem. We will use two reductions, to get a lower bound once in terms of d and once in terms of Φ. Specifically, in the first reduction we will set $m = 1/\epsilon^2$, $k = n/q$ and $r = \rho \log d$ (where ρ is the constant from the statement of Lemma 3.6.11). Then the lower bound we would get by Proposition 3.6.10 is $\Omega\left(\epsilon^{-2} n \log d \log(q/\delta)\right)$. In the second reduction we will set $r = \rho \log \Phi$, yielding the lower bound $\Omega\left(\epsilon^{-2} n \log \Phi \log(q/\delta)\right)$. Together they lead to Lemma 3.6.11.

In both settings, recall we are reducing to the following problem: For dimension $d = \Omega(\epsilon^{-2} \log(q/\delta))$ and aspect ratio Φ, Alice gets n points, Bob gets q points, and Bob needs to estimate all cross-distances up to distortion $1 \pm \epsilon$.

Consider an instance of r-*Ind*(q-*AugSetList*(k, m, δ)). It can be visualized as follows: Alice gets a matrix S with $n = qk$ rows and r columns, where each entry contains a set of size m. Bob gets an index $j \in [r]$, indices $i_1, \ldots, i_q \in [k]$, elements $e_1, \ldots, e_q$, subsets $T_1 \subset S(i_1, j), \ldots, T_q \subset S(i_q, j)$, and the first $j-1$ columns of the matrix S.

We now use the encoding scheme of [JW13], in the set formulation which was given in [MWY13]. We restate the result.

Lemma 3.6.12 ([JW13]). *Let $m = 1/\epsilon^2$ and $0 < \eta < 1$. Suppose we have the following setting:*

- *Alice has subsets $S_1, \dots, S_r$ of $[m/\eta]$.*
- *Bob has an index $j \in [r]$, an element $e \in [m/\eta]$, the subset $T \subset S_j$ of elements smaller than e, and the sets $S_1, \dots, S_{j-1}$.*

There is a shared-randomness mapping of their inputs into points v_A, v_B and a scale $\Psi > 0$ (the scale is known to both), such that

1. *$v_A, v_B \in \{0,1\}^D$ for $D = O(\epsilon^{-2} \log(\frac{1}{\eta}) \exp(r))$.*
2. *If $e \in S_j$ (YES instance) then w.p. $1-\eta$, $\|v_A - v_B\|^2 \le (1-2\epsilon)\Psi$.*
3. *If $e \notin S_j$ (NO instance) then w.p. $1-\eta$, $\|v_A - v_B\|^2 \ge (1-\epsilon)\Psi$*

Lower bound by dimension. We start with the first reduction that yields a lower bound in terms of d.

Lemma 3.6.13. *Under the assumptions of Lemma 3.6.11, for the all-cross-distances problem, Alice must use a sketch of at least $\Omega(\epsilon^{-2} n \log(d) \log(q/\delta))$ bits.*

Proof. We invoke Lemma 3.6.12 with $r = \rho \log d$ and $\eta = \delta/q$. Note that the latter is the desired success probability in each instance of *q-AugSetList*(k, m, δ) (cf. Definition 3.6.7). Alice encodes each row of the matrix, $(S(i,1), \dots, S(i,r))$, into a point x_i, thus n points $x_1, \dots, x_n$. Bob encodes $(S(i,1), \dots, S(i_z, j-1), T_i, j, e_z)$ for each $z \in [q]$ into a point y_z, thus q points $y_1, \dots, y_z$. For every $z \in [q]$, the problem represented by row i_z in the matrix S is reduced by Lemma 3.6.12 to estimating the distance $\|x_{i_z} - y_z\|$. By Item 1 of Lemma 3.6.12, the points $\{x_i\}_{i \in [n]}$, $\{y_z\}_{z \in [q]}$ have binary coordinates and dimension $D = O(\epsilon^{-2} \log(q/\delta) d^\rho)$. By the hypothesis $d^{1-\rho} \ge \epsilon^{-2} \log(q/\delta)$ of Lemma 3.6.11, $D = O(d)$. Therefore Alice and Bob can now feed them into a given black-box solution of the all-cross-distances problem, which estimates all the required distances and solves *r-Ind*(*q-AugSetList*(k, m, δ)).

Let us establish the success probability of the reduction. Since we set $\eta = \delta/q$ in Lemma 3.6.12, it preserves each distance $\|x_{i_z} - y_z\|$ for $z \in [q]$ with probability $1 - \delta/q$. By a union bound, it preserves all of them simultaneously with probability $1-\delta$. The success probability of the all-cross-distances problem, simultaneously on all query points $\{\tilde{y}_z : z \in [q]\}$, is again $1-\delta$. Altogether, the reduction succeeds with probability $1-O(\delta)$. As a result, the all-cross-distances problem solves the given instance of r-*Ind*(q-*AugSetList*(k, m, δ)), and Lemma 3.6.13 follows. □

Lower bound by coordinate range. We proceed to the second reduction that would yield a lower bound in terms of Φ.

Lemma 3.6.14. *Under the assumptions of Lemma 3.6.11, for the all-cross-distances problem, Alice must use a sketch of at least* $\Omega(\epsilon^{-2} n \log(\Phi) \log(q/\delta))$ *bits.*

Proof. We may assume that $\Phi \geq d$ since otherwise Lemma 3.6.14 already follows from Lemma 3.6.13. Therefore $\Phi^{1-\rho} \geq \epsilon^{-2}\log(q/\delta)$.

The reduction is very similar to the one in Lemma 3.6.13. Again we evoke Lemma 3.6.12 with $\eta = \delta/q$, but this time we set $r = \rho \log \Phi$. Again we denote Alice's encoded points by $x_1, \ldots, x_n$, and Bob's by $y_1, \ldots, y_q$. By Item 1 of Lemma 3.6.12, the points have binary coordinates and dimension $D = O(\epsilon^{-2}\log(q/\delta)\Phi^{\rho})$. The difference from Lemma 3.6.13 is that since it is possible that $\Phi \gg d$, the dimension D is too large for the given black-box solution of the all-cross-distances problem (which is limited to dimension $O(d)$).

To solve this, Alice and Bob project their points into dimension $D' = O(\epsilon^{-2}\log(q/\delta))$ by a Johnson-Lindenstrauss transform, using shared randomness. Let $\tilde{x}_1, \ldots, \tilde{x}_n$ and $\tilde{y}_1, \ldots, \tilde{y}_z$ denote the projected points. After the projection each coordinate has magnitude at most $O(\epsilon^{-2}\log(q/\delta)\Phi^{\rho})$. By our assumption $\Phi^{1-\rho} \geq \Omega(\epsilon^{-2}\log(q/\delta))$, this is at most $O(\Phi)$. Since the dimension D' is $O(d)$, Alice and Bob can now feed $\tilde{x}_1, \ldots, \tilde{x}_n$ and $\tilde{y}_1, \ldots, \tilde{y}_z$ into a given black-box solution of the all-cross-distances problem with dimension $O(d)$ and aspect ratio $O(\Phi)$.

Let us establish the success probability of the reduction. As before, Lemma 3.6.12 preserves all the required distances, $\|x_{i_z} - y_z\|$ for $z \in [q]$, with probability $1-\delta$. The Johnson-Lindenstrauss transform into dimension D' preserves each distance as $\|\tilde{x}_{i_z} - \tilde{y}_z\|$ with probability at least $1-\delta$, since we picked the dimension to be $D' = O(\epsilon^{-2}\log(\frac{q}{\delta}))$. The success probability of the all-cross-distances problem simultaneously is again $1-\delta$. Altogether, the reduction succeeds with probability $1 - O(\delta)$. As a result, the all-cross-distances problem solves the given instance of r-*Ind*(q-*AugSetList*(k, m, δ)), and Lemma 3.6.14 follows. $\square$

3.6.4 Lower Bound by Number of Data Points

Finally we prove the remaining term of the lower bound in Theorem 3.1.5, by reduction to the metric sketching problem from Chapter 2.

Lemma 3.6.15. *Suppose that $d \geq \Omega(\epsilon^{-2} \log n)$; $\Phi \geq 1/\epsilon$; ϵ is at most a sufficiently small constant; and $\epsilon \geq 1/n^{0.5-\rho'}$ for a constant $\rho' > 0$. Then, for the all-cross-distances problem, Alice must use a sketch of at least $\Omega(\rho' \epsilon^{-2} n \log n)$ bits.*

Proof. We reduce Euclidean metric sketching without new query points (Definition 2.1.1) to all-cross-distances (with query points). For the latter problem, we have already proved the desired lower bound in Theorem 2.6.1.

Suppose we have a given sketching procedure for the all-cross-distances problem that uses $s = s(n, d, \Phi, 1, \epsilon, \delta)$ amortized bits per point. (Note that s is the sketch size with $q = 1$ query point.) We invoke it on the input point set X to the metric sketching problem, and denote the resulting sketch by S_0. For every point $y \in \{-\Phi \ldots \Phi\}^d$, with probability $1-\delta$, all distances $\{\|x-y\| : x \in X\}$ can be recovered from S_0. In particular, this holds in expectation for $(1-\delta)n$ of the points in X. By Markov's inequality, this holds for $\frac{1}{2}(1-\delta)n > \frac{1}{4}n$ of the points in X with probability at least $1/2$. We proceed by recursion on the remaining $\frac{3}{4}n$ points in X, and so on, until for every $x, y \in X$ we have produced an all-cross-distances that correctly estimates $\|x - y\|$. The sketch produced in the ith step of the recursion is denoted by S_i and has total size $(\frac{3}{4})^i ns$ bits. After $t = O(\log n)$ steps, with nonzero probability $1/n^{O(1)}$, we have produced a sequence of sketches $S_0, \ldots, S_t$ from which every distance in $\{\|x - x'\| : x, x' \in X\}$ can be recovered, with a total size of $O(\sum_{i=0}^{t} (\frac{3}{4})^i ns) = O(ns)$ bits. (Note that the lower bound in Theorem 2.6.1 was information theoretic and did not depend on the success probability.) This completes the reduction to Definition 2.1.1 and the lower bound follows. $\square$

3.6.5 Concluding the Lower Bound Proof

To conclude the proof of Theorem 3.1.5, note that whenever we prove two lower bounds B and B', they can be combined into one lower bound $B+B'$ by taking the direct product of the respective families of hard metrics, as in the proof of Theorem 2.6.1. Thus, Lemmas 3.6.13 and 3.6.14 imply Lemma 3.6.11, which together with Lemma 3.6.15 implies Theorem 3.1.5.

Chapter 4

Practical and Provable Metric Compression

In this chapter we revisit the metric sketching problem from a practical point of view. Chapters 2 and 3 were focused on achieving optimal theoretical bounds, which necessitated using rather involved techniques that do not seem amenable to implementation. Nonetheless, we show that the underlying algorithmic ideas can yield a practical variant of the algorithm. Our resulting algorithm, QuadSketch, provably attains nearly optimal sketching bound — close to our bounds from Chapters 2 and 3, and asymptotically better than discretized dimension reduction — and at the same time, it empirically matches of improves over the performance of state-of-the-art heuristics on standard benchmark datasets.

4.1 Introduction

Our motivation in studying metric compression, as discussed in Chapter 1, has been twofold: aside from being a natural question of inherent theoretical interest, it also arises in practical applications that involve searching through large datasets. So far we have focused on the theoretical aspects. Now we turn to the practical ones.

Can our techniques from the previous chapters lead to practical algorithms? Ostensibly, our Relative Location Tree technique from Chapters 2 and 3 — while provably achieving optimal sketching bounds — may seem too complicated to be beneficial in practice.

Reference	Bits per point	Construction runtime
Discretized [JL84]	$O(\epsilon^{-2}\log^2 n)$	$\tilde{O}(\epsilon^{-2}n)$
Theorem 2.1.2	$O(\epsilon^{-2}\log n)$, tight	$\tilde{O}(n^{1+\alpha}+\epsilon^{-2}n)$, $\alpha\in(0,1]$
QuadSketch (Theorem 4.4.8)	$O(\epsilon^{-2}\log n\cdot\log(\epsilon^{-1}\log n))$	$\tilde{O}(\epsilon^{-2}n)$

Table 4.1: QuadSketch guarantees for Euclidean metric compression with distortion $(1\pm\epsilon)$, compared to previous bounds. The bounds are listed with $\Phi = n^{O(1)}$ for simplicity.

However, we will show that its key algorithmic ideas can be adjusted to benefit both in theory and in practice. We call the resulting algorithm ***QuadSketch***. On one hand, theoretically, its compression bound is looser than the optimal bound by a factor of $O(\log\log n)$, whereas the discretized dimension reduction bound is looser by $O(\log n)$. See Table 4.1 for a more precise comparison. On the other hand, empirically, we implement QuadSketch and show that it attains state-of-the-art performance in practical metric sketching.

4.1.1 Our Results

We focus on Euclidean distances. Let us start by a recap of the metric sketching setting from Definition 2.1.1. We are given a set of points $X = \{x_1,\dots,x_n\}\subset\mathbb{R}^d$, with pairwise distances in the range $[1,\Phi]$. Our goal is to construct a compressed bit representation of X, called a *sketch*, that has the following property: given any $i,j\in[n]$, we can recover from the sketch an estimate of the distance $\|x_i-x_j\|$ up to distortion $(1\pm\epsilon)$.

For this chapter it will be convenient to explicitly consider *point decompression*. This means that given $i\in[n]$, we can recover from the sketch a point $\tilde{x}_i\in\mathbb{R}^d$ that serves as a proxy for the original point x_i. In particular, every distance $\|x_i-x_j\|$ is approximated by $\|\tilde{x}_i-\tilde{x}_j\|$. We will also assume for simplicity that the input point dimension is already reduced to $O(\epsilon^{-2}\log n)$, by a standard [JL84]-type transform.

QuadSketch. The main contribution of this chapter is a *simple* and data-adaptive sketching algorithm, which is both provable and practical. Being quite intuitive, it is based on a randomized variant of a classical *quadtree* (see, e.g., [Sam84, HP11]), followed by pruning some nodes and edges.

In its simplest variant, the trade-off between the sketch size and approximation quality is governed by a single parameter Λ. Specifically, Λ controls the pruning step, in which the algorithm identifies "non-important" bits among those stored in the quadtree (i.e., bits whose omission would have little effect on the approximation quality), and removes them from the sketch. Higher values of Λ result in sketches that are longer but produce more accurate distance estimates.

Provable guarantees. The basic distance preservation guarantee of QuadSketch is summarized in the following theorem.

Theorem 4.1.1. *Given $\epsilon, \delta > 0$, let $\Lambda = O(\log(d \log \Phi / \epsilon\delta))$ and $L = \log \Phi + \Lambda$. QuadSketch runs in time $\tilde{O}(ndL)$, and produces a sketch of size $O(nd\Lambda + n \log n)$ bits, with the following guarantee: for every $i \in [n]$,*

$$\Pr\left[\forall_{j\in[n]} \|\tilde{x}_i - \tilde{x}_j\| = (1 \pm \epsilon)\|x_i - x_j\|\right] \geq 1 - \delta.$$

In particular, with probability $1 - \delta$, if $\tilde{x}_{i^*}$ is the nearest neighbor of $\tilde{x}_i$ in $\tilde{X}$, then x_{i^*} is a $(1 + \epsilon)$-approximate nearest neighbor of x_i in X.

Note that Λ in the theorem represents the number of *bits per coordinate*. For example, in the typical setting where $d = O(\epsilon^{-2} \log n)$ and $\Phi = n^{O(1)}$, QuadSketch uses $\Lambda = O(\log \log n + \log(1/\epsilon))$ bits per coordinate.

We remark that setting of the above theorem is somewhat simplified compared to those considered in the previous chapters: it does not estimate all distances simultaneously (as required in Chapter 2), but only the distances from every given point; and does not apply to query points outside the dataset X (as required in Chapter 3). Nonetheless, QuadSketch achieves the same sketching bound from Theorem 4.1.1 for both of these problems formally. We show this later in this chapter, in Theorems 4.4.8 and 4.4.9, respectively.

Empirical performance. We evaluate QuadSketch experimentally on both real and synthetic data sets: a SIFT feature data set from [JDS11], MNIST [LC98], time series data reflecting taxi ridership in New York City [GMRS16], and a synthetic data set (Diagonal) containing random points from a one-dimensional subspace (i.e., a line) embedded in a high-dimensional space. The data sets are quite diverse: SIFT and MNIST data sets are de-facto "standard" test cases for nearest neighbor search and distance preserving sketches, NYC taxi data was designed to contain anomalies and "irrelevant" dimensions, while Diagonal has extremely low intrinsic dimension.

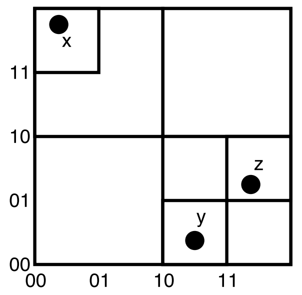
We compare QuadSketch to Product Quantization (PQ) [JDS11], a state of the art empirical method for computing distance preserving sketches, as well as a naïve baseline of simple uniform quantization. The sketch length/accuracy tradeoffs for QuadSketch and PQ are comparable on SIFT and MNIST data, with PQ having better accuracy for shorter sketches while QuadSketch having better accuracy for longer sketches. On NYC taxi data, the accuracy of QuadSketch is higher over the whole range of sketch lengths. Finally, Diagonal exemplifies a situation where the low dimensionality of the data set hinders the performance of PQ, while QuadSketch naturally adapts to this data set. Overall, QuadSketch performs well on "typical" data sets, while its provable guarantees ensure robust performance in a wide range of scenarios. Both algorithms improve over the baseline quantization method.

4.2 Algorithm: QuadSketch

Generally speaking, QuadSketch is an analog of the relative location tree from Chapters 2 and 3, except that the hierarchical clustering tree is replaced with a randomly shifted quadtree. The path compression step is similar to Section 2.3, and here it takes a natural interpretation of eliminating non-significant bits from the binary expansion of the point coordinates. Finally, we can dispense with the different definitions of centers, ingresses and surrogates from Section 2.3, as they all naturally coincide with grid corners in the quadtree.

We now describe the algorithm in a self-contained manner. It takes as input the point set X, and two parameters L and Λ that control the amount of compression.

Step 1: Randomly shifted grid. The algorithm starts by imposing a randomly shifted axis-parallel grid on the points. We first enclose the whole point set in an axis-parallel hypercube H. Let $\Delta = 2^{\lceil \log \Phi \rceil}$. Set up H to be centered at x_1 with side length 4Δ. Now



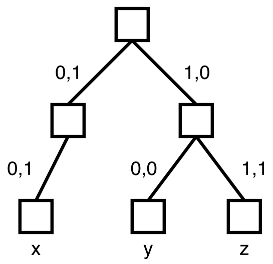


Figure 4-1: Quadtree construction for three points x, y, z. The coordinates on the horizontal and vertical axes are written in binary representation, which corresponds to the edge labels in the quadtree.

choose $\sigma_1, \ldots, \sigma_d \in [-\Delta, \Delta]$ independently and uniformly at random, and shift H in each coordinate j by σ_j. By the choice of side length 4Δ, one can see that H after the shift still contains the whole point set. For every integer ℓ such that $-\infty < \ell \leq \log(4\Delta)$, let G_ℓ denote the axis-parallel grid with cell side 2^ℓ which is aligned with H.

Step 2: Quadtree construction. The 2^d-ary quadtree on the nested grids G_ℓ is naturally defined by associating every grid cell c in G_ℓ with the tree node at level ℓ, such that its children are the 2^d grid cells in $G_{\ell-1}$ which are contained in c. The edge connecting a node v to a child v' is labeled with a bitstring of length d defined as follows: the j^{th} bit is 0 if v' coincides with the bottom half of v along coordinate j, and 1 if v' coincides with the upper half along that coordinate (see Figure 4-1 for example).

In order to construct the tree, we start with H as the root, and bucket the points contained in it into the 2^d children cells. We only add child nodes for cells that contain at least one point of X. Then we continue by recursion on the child nodes. The quadtree construction is finished after L levels. We denote the resulting edge-labeled tree by T^*. A construction with $L = 2$ is illustrated in Figure 4-1.

We define the *level* of a tree node with side length 2^ℓ to be ℓ (note that ℓ can be negative). The *degree* of a node in T^* is its number of children. Since all leaves are located at the bottom level, each point $x_i \in X$ is contained in exactly one leaf, which we henceforth denote by v_i.

Step 3: Pruning. Consider a downward path $u_0, u_1, \ldots, u_k$ in T^*, such that $u_1, \ldots, u_{k-1}$ are nodes with degree 1, and u_0, u_k are nodes with degree other than 1 (u_k may be a leaf). For every such path in T^*, if $k > \Lambda + 1$, we remove the nodes $u_{\Lambda+1}, \ldots, u_{k-1}$ from T^* with all their adjacent edges (and edge labels). Instead we connect u_k directly to u_Λ as its child. We refer to that edge as the *long edge*, and label it with the length of the path it replaces, $k - \Lambda$. The original edges from T^* are called *short edges*. At the end of the pruning step, we denote the resulting tree by T.

The sketch. For each point $x_i \in X$ the sketch stores the index of the leaf v_i that contains it. In addition it stores the structure of the tree T, encoded using the Eulerian Tour Technique[1]. Specifically, starting at the root, we traverse T in the Depth First Search (DFS) order. In each step, DFS either explores the child of the current node (downward step), or returns to the parent node (upward step). We encode a downward step by 0 and an upward step by 1. With each downward step we also store the label of the traversed edge (a length-d bitstring for a short edge or the edge length for a long edge, and an additional bit marking if the edge is short or long).

Decompression. Recovering $\tilde{x}_i$ from the sketch is done simply by following the downward path from the root of T to the associated leaf v_i, collecting the edge labels of the short edges, and placing zeros instead of the missing bits of the long edges. The collected bits then correspond to the binary expansion of the coordinates of $\tilde{x}_i$.

More formally, for every node u (not necessarily a leaf) we define $c(u) \in \mathbb{R}^d$ as follows: For $j \in \{1, \ldots, d\}$, concatenate the j^{th} bit of every short edge label traversed along the downward path from the root to u. When traversing a long edge labeled with length k, concatenate k zeros.[2] Then, place a binary floating point in the resulting bitstring, after the bit corresponding to level 0. (Recall that the levels in T are defined by the grid cell side lengths, and T might not have any nodes in level 0; in this case we need to pad with 0's either on the right or on the left until we have a 0 bit in the location corresponding to level

[1]See e.g., https://en.wikipedia.org/wiki/Euler_tour_technique.

[2]This is the primary lossy step in our sketching method: the original bits could be arbitrary, but they are replaced with zeros.

0.) The resulting binary string is the binary expansion of the j^{th} coordinate of $c(u)$. Now $\tilde{x}_i$ is defined to be $c(v_i)$.

Block QuadSketch. The successful Product Quantization algorithm for metric sketching [JDS11] is based on partitioning the coordinates into blocks, and sketching (or quantizing) block separately. We can further modify QuadSketch in a similar manner. Specifically, we partition the d dimensions into m blocks $B_1 \dots B_m$ of size d/m each, and apply QuadSketch separately to each block. More formally, for each B_i, we apply QuadSketch to the point set $(x_1)_{B_i} \dots (x_n)_{B_i}$, where x_B denotes the (d/m)-dimensional vector obtained by projecting x on the dimensions in B. The following theorem is an immediate corollary of Theorem 4.1.1.

Theorem 4.2.1. *Let $\epsilon, \delta > 0$. Let m be an integer divisor of d. Let $\Lambda = O(\log(d \log \Phi/\epsilon\delta))$ and $L = log\Phi + \Lambda$. The m-block variant of QuadSketch runs in time $\tilde{O}(ndL)$, and produces a sketch of size $O(nd\Lambda + nm \log n)$ bits, with the following guarantee: For every $i \in [n]$,*

$$\Pr\left[\forall_{j\in[n]} \|\tilde{x}_i - \tilde{x}_j\| = (1 \pm \epsilon)\|x_i - x_j\|\right] \geq 1 - m\delta.$$

It can be seen that increasing the number of blocks m up to a certain threshold, namely of $d\Lambda/\log n$, does not affect the asymptotic bound on the sketch size. Although we cannot prove that varying m can *improve* the accuracy of the sketch, this seems to be the case empirically, as demonstrated in the experimental section.

4.3 Experiments

Algorithms. We evaluate QuadSketch experimentally and compare its performance to Product Quantization (PQ) [JDS11], a state-of-the-art compression scheme for approximate nearest neighbors, and to a baseline of uniform scalar quantization, which we refer to as Grid. For each dimension of the dataset, Grid places k equally spaced landmark scalars on the interval between the minimum and the maximum values along that dimension, and rounds each coordinate to the nearest landmark.

All three algorithms work by partitioning the data dimensions into blocks, and performing a quantization step in each block independently of the other ones. QuadSketch and PQ take the number of blocks as a parameter, whereas Grid uses blocks of size 1. The quantization step is the basic algorithm described in Section 4.2 for QuadSketch, k-means for PQ, and uniform scalar quantization for Grid.

Datasets. We test the algorithms on four datasets: The SIFT data used in [JDS11], MNIST [LC98] (with all vectors normalized to 1), NYC Taxi ridership data [GMRS16], and a synthetic dataset called Diagonal, consisting of random points on a line embedded in a high-dimensional space. The properties of the datasets are summarized in Table 4.2. Note that we were not able to compute the exact diameters for MNIST and SIFT, hence we only report estimates for Φ for these data sets, obtained via random sampling.

The Diagonal dataset consists of $10,000$ points of the form $(x, x, \ldots, x)$, where x is chosen independently and uniformly at random from the interval $[0, 40000]$. This yields a dataset with very large aspect ratio Φ, with the intention of demonstrating the effect of this parameter on the performance of metric sketching algorithms. Furthermore, since Diagonal is constructed such that all coordinates are maximally correlated, the heuristic partitioning into blocks — the driving force behind PQ — is not expected to be beneficial. This is meant to demonstrate that QuadSketch can successfully adapt to "worst-case data", thanks to its worst-case guarantees, whereas heuristic algorithms do not perform well.

Queries. For SIFT and MNIST we use the standard query set provided with each dataset. For Taxi and Diagonal we use 500 queries chosen at random from each dataset. For the sake of consistency, for all data sets, we apply the same quantization process jointly to both the point set and the query set, for both PQ and QuadSketch. We note, however, that both algorithms can be run on "out of sample" queries (for QuadSketch, we show this in detail in Section 4.4.4).

Parameter setting. For each dataset, we enumerate the number of blocks over all divisors of the dimension d. For QuadSketch, L ranges in $2, \ldots, 20$, and Λ ranges in $1, \ldots, L-1$. For PQ, the number of k-means landmarks per block ranges in $2^5, 2^6, \ldots, 2^{12}$. For both algorithms we include the results for all combinations of the parameters, and plot the envelope of the

Dataset	Points	Dimension	Aspect ratio (Φ)	Data type
SIFT	$1,000,000$	128	≥ 83.2	Image descriptors
MNIST	$60,000$	784	≥ 9.2	Hand-written digits
NYC Taxi	$8,874$	48	49.5	Time-series
Diagonal (synthetic)	$10,000$	128	$20,478,740.2$	Synthetic

Table 4.2: Datasets used in our empirical evaluation. The aspect ratio of SIFT and MNIST is estimated on a random sample.

best performing combinations.

Performance measures. We report two measures of performance for each dataset: (a) the *accuracy*, defined as the fraction of queries for which the sketch returns the true nearest neighbor, and (b) the *average distortion*, defined as the ratio between the (true) distances from the query to the reported near neighbor and to the true nearest neighbor. The sketch size is measured in bits per coordinate. The results appear in Figures 4-2 to 4-5. Note that the vertical axis in the distortion plots corresponds to the value of ϵ, not $1+\epsilon$.

4.3.1 Effect of Parameter Setting

We plot how the different parameters of QuadSketch effect its performance. Recall that L determines the number of levels in the quadtree prior to the pruning step, and Λ controls the amount of pruning. By construction, the higher we set these parameters, the larger the sketch will be and with better accuracy. The empirical tradeoff for the SIFT dataset is plotted in Figure 4-6.

The optimal setting for the number of blocks is not monotone, and generally depends on the specific dataset. It is noted in [JDS11] that on SIFT data, an intermediate number of blocks gives the best results, and this is confirmed by our experiments. Table 4.3 lists the performance on the SIFT dataset for a varying number of blocks, for a fixed setting of $L = 6$ and $\Lambda = 5$. It shows that the sketch quality remains essentially the same, while the size varies significantly, with the optimal size attained at 16 blocks.

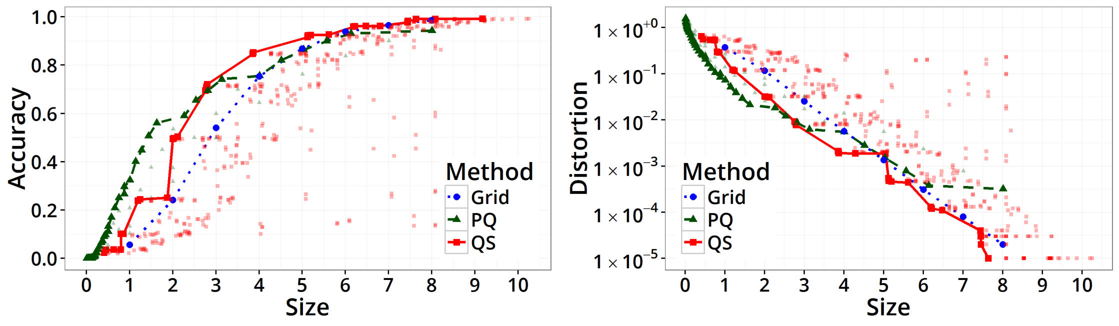


Figure 4-2: Results for the SIFT dataset

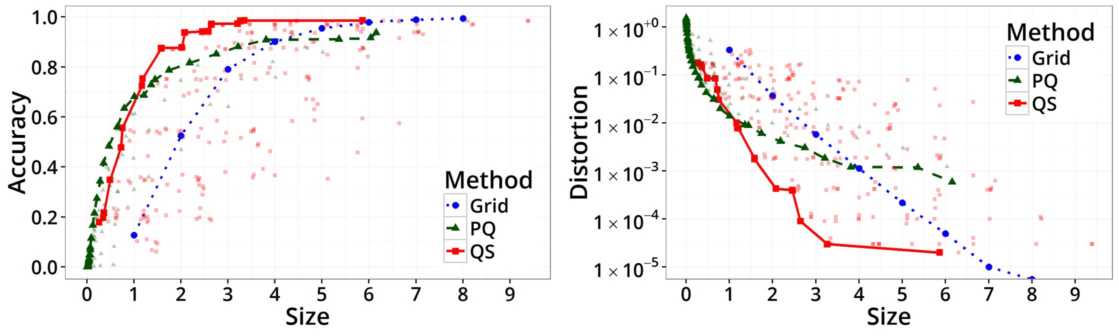


Figure 4-3: Results for the MNIST dataset

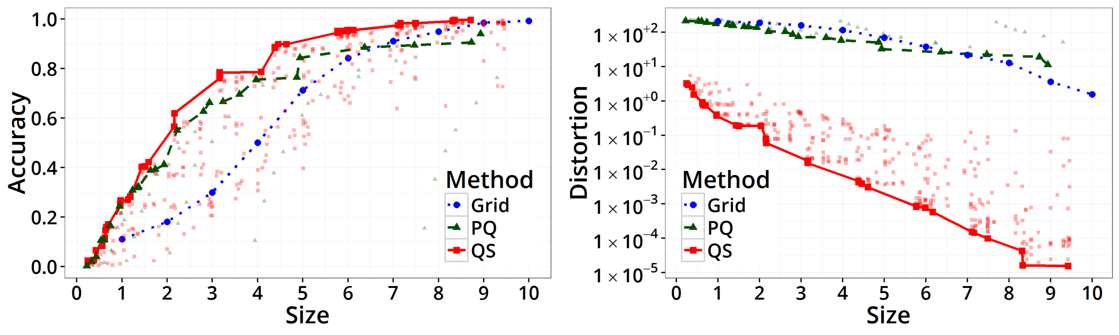


Figure 4-4: Results for the Taxi dataset

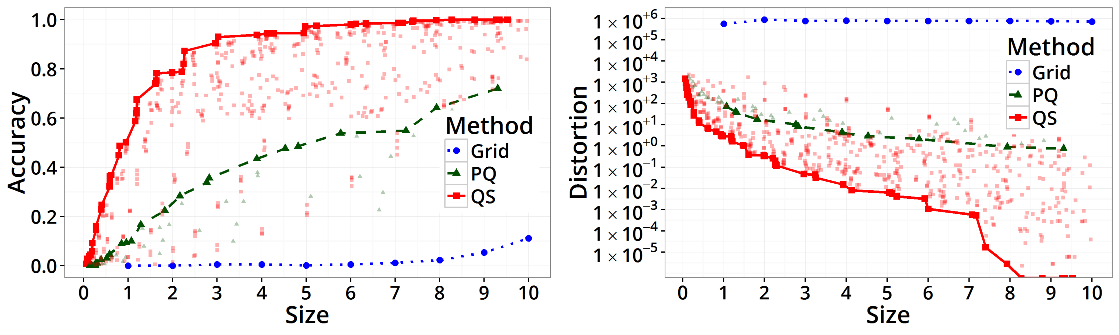
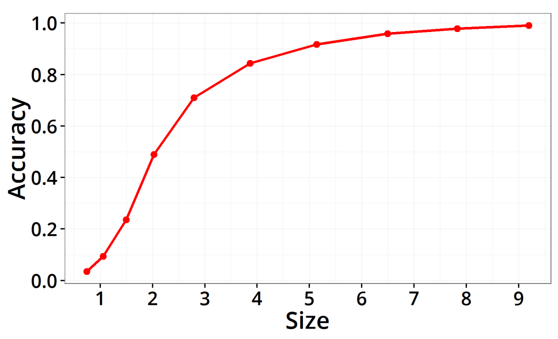


Figure 4-5: Results for the Diagonal dataset



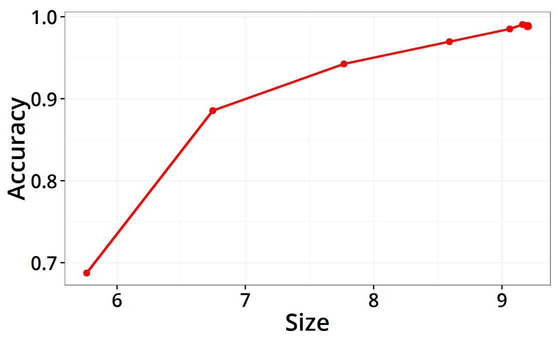


Figure 4-6: On the left, L varies from 2 to 11 for a fixed setting of 16 blocks and $\Lambda = L - 1$ (no pruning). On the right, Λ varies from 1 to 9 for a fixed setting of 16 blocks and $L = 10$. Increasing Λ beyond 6 does not have further effect on the resulting sketch.

# Blocks	Bits per coordinate	Accuracy	Average distortion
1	5.17	0.719	1.0077
2	4.523	0.717	1.0076
4	4.02	0.722	1.0079
8	3.272	0.712	1.0079
16	**2.795**	0.712	1.008
32	3.474	0.712	1.0082
64	4.032	0.713	1.0081
128	4.079	0.72	1.0078

Table 4.3: QuadSketch accuracy on SIFT data by number of blocks, with $L = 6$ and $\Lambda = 5$.

4.4 Proofs

In this section we prove the theoretical guarantees of QuadSketch.

4.4.1 Basic QuadSketch Guarantee

We start by proving Theorem 4.1.1.

Recall that we have a point set $x_1, \ldots, x_n \in \mathbb{R}^d$ with aspect ratio Φ, and given error parameters $\epsilon, \delta > 0$. For the remainder of the section we fix the setting

$$\Lambda = \log\left(\frac{16 \cdot d^{1.5} \cdot \log \Phi}{\epsilon\delta}\right), \tag{4.1}$$

and assume w.l.o.g. it is an integer.

The analysis relies on the *padded decomposition* property of a randomly shifted quadtree, as defined in the seminal work of Bartal [Bar96]. We now define the notion in our setting. To this end, recall that in Section 4.2, we denoted by G_ℓ the grid with side length 2^ℓ for every integer ℓ.

Definition 4.4.1 (padded point in grid)**.** *We say that a point x_i is* $(\epsilon, \Lambda, \ell)$-padded*, if the grid cell in G_ℓ that contains x_i also contains the ball of radius $\rho(\ell)$ centered at x_i, where*

$$\rho(\ell) = \frac{8 \cdot 2^{\ell-\Lambda} \cdot \sqrt{d}}{\epsilon}.$$

Definition 4.4.2 (padded point in tree)**.** *We say that x_i is* (ϵ, Λ)-padded *in the quadtree T, if it is $(\epsilon, \Lambda, \ell)$-padded for every level ℓ of T.*

The fact that a randomly shifted quadtree has the padded decomposition property was observed in [Ind01]. The next lemma reproduces the argument.

Lemma 4.4.3. *If the grids are randomly shifted, as in the QuadSketch construction (Section 4.2), then every point x_i is (ϵ, Λ)-padded in T with probability $1 - \delta$.*

Proof. Fix a point x_i, a coordinate $k \in \{1, \ldots, d\}$ and a level ℓ. Let $x_i(k)$ denote the value of x_i in coordinate k. Along this coordinate, we are randomly shifting a 1-dimensional grid partitioned into intervals of length 2^ℓ. Since the shift is uniformly random, the probability for $x_i(k)$ to be at distance at most $\rho(\ell)$ from an endpoint of the interval that contains it equals $2\rho(\ell)/2^\ell$. By plugging our setting of $\rho(\ell)$ and Λ, this probability equals $\delta/(d \log \Phi)$. Taking a union bound over the d coordinates, we have probability at most $\delta / \log \Phi$ for x_i to be at distance at most $\rho(\ell)$ from the boundary of the cell of G_ℓ that contains it. In the complement event x_i is $(\epsilon, \Lambda, \ell)$-padded in G_ℓ. Taking another union bound over the $L = \log \Phi + \Lambda$ levels in the quadtree, x_i is (ϵ, Λ)-padded with probability at least $1 - \delta$. □

We now state and prove the main lemma. It states that if a point is padded, as defined above, then QuadSketch accurately estimates the distance from that point to all other points, and therefore satisfies the guarantee of Theorem 4.1.1.

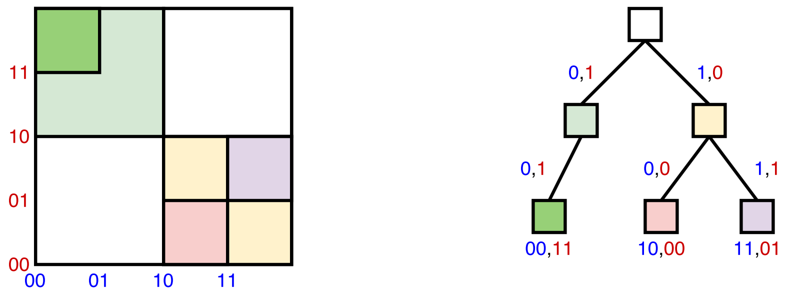


Figure 4-7: By collecting the edge label bits along every dimension from the root to a node, and padding with zeros as necessary, we obtain the binary expansion of the bottom-left corner of the associated grid cell.

Lemma 4.4.4. *If a point x_i is (ϵ, Λ)-padded in T, then for every $j \in [n]$,*

$$(1-\epsilon)\|\tilde{x}_i - \tilde{x}_j\| \leq \|x_i - x_j\| \leq (1+\epsilon)\|\tilde{x}_i - \tilde{x}_j\|,$$

where $\{\tilde{x}_i\}$ are as defined in Section 4.2.

Proof. We recall that T is a pruned quadtree in which every node v is associated with a grid cell of an axis-parallel grid G_ℓ with side length 2^ℓ, which is aligned with and contained in H. We call ℓ the *level* of v, and denote it henceforth by $\ell(v)$. We will use the term "bottom-left corner" of a grid cell for the corner that minimizes all coordinate values (i.e., the high-dimensional analog of a bottom-left corner in the plane).

Let r be the root of T. We may assume w.l.o.g. that the bottom-left corner of H is the origin in $\mathbb{R}^d$, since translating H together with the entire point set does not change pairwise distances. Under this assumption, we make the following observation, illustrated in Figure 4-7.

Claim 4.4.5. *Let v be a node in T. If the path from r to v contains only short edges, then $c(v)$ (defined by the decompression algorithm in Section 4.2) is the bottom-left corner of the grid cell associated with v.*

Let x_i be a padded point, and x_j be any point. Recall that we denote by v_i and v_j the leaves corresponding to x_i and x_j respectively (see Section 4.2). Let w be the lowest common ancestor of v_i and v_j in T. Since x_i and x_j are in separate grid cells of $G_{\ell(w)-1}$, and by padding the cell containing x_i also contains the ball of radius $\rho(\ell(w)-1)$ around x_i, we have

$$\|x_i - x_j\| \geq \rho(\ell(w)-1) = \frac{8 \cdot 2^{\ell(w)-1-\Lambda}\sqrt{d}}{\epsilon}. \tag{4.2}$$

Let u_i be the lowest node on the downward path from w to v_i, that can be reached without traversing a long edge. Similarly define u_j for v_j. See Figure 4-8 for illustration.

Note that u_i must be either the leaf v_i, or an internal node whose only outgoing edge is a long edge. In both cases, u_i is the bottom of a path of degree-1 nodes of length Λ:

- If u_i is a leaf: Since the point set has aspect ratio Φ, then after $\log\Phi$ levels the grid becomes sufficiently fine such that each grid cell contains at most one point x_i. Since we generate the quadtree with $L = \log\Phi + \Lambda$ levels, then each point x_i is in its own grid cell for at least the bottom Λ levels of the quadtree.

- If u_i is an internal node which is the head of a long edge: Since the pruning step only places long edges at the bottom of degree-1 paths of length Λ, then u_i must be the bottom node of such path.

On the other hand, w is an ancestor of u_i, and it must have degree at least 2, since it is also an ancestor of u_j. Hence, w is at least Λ levels above u_i, implying

$$\ell(v_i) \leq \ell(w) - \Lambda. \tag{4.3}$$

Applying the same arguments to u_j we get also $\ell(v_j) \leq \ell(w) - \Lambda$.

Let $c^*(u_i), c^*(u_j) \in \mathbb{R}^d$ be the bottom-left corners of the grid cells associated with u_i and u_j. If all edges on the downward paths from the root of T to u_i and u_j were short, then Claim 4.4.5 would yield that $c^*(u_i) = c(u_i)$ and $c^*(u_j) = c(u_j)$. In general, there might be some long edges on those paths, but they all must lie on the subpath from the root of T down to w, which is the same for both paths. This is because by the choice of u_i and u_j, all downward edges from w to either of them are short. Therefore $c(u_i)$ and $c(u_j)$ are shifted from the true bottom-left corners by the same shift, which we denote by

$$\eta = c^*(u_i) - c(u_i) = c^*(u_j) - c(u_j). \tag{4.4}$$

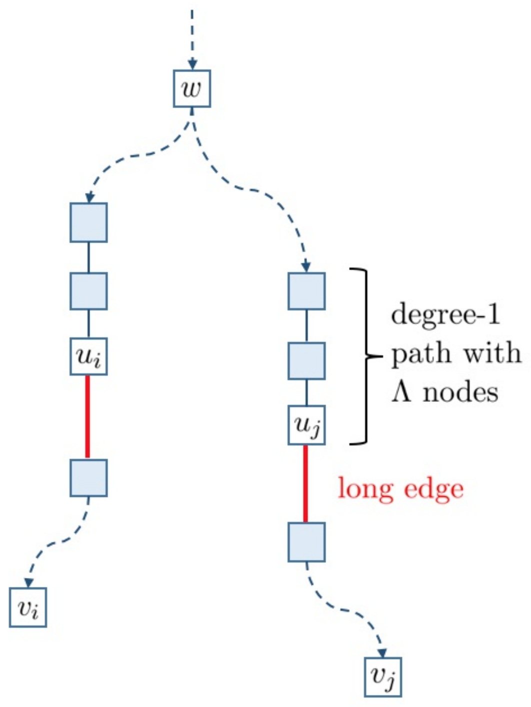


Figure 4-8: In the proof of Lemma 4.4.4, w is the lowest common ancestor of v_i, v_j, the leaves corresponding to x_i, x_j. The node u_i is the lowest node on the downward path from w to v_i which is achievable without traversing any long edges (marked in red). The node u_j is defined similarly for v_j.

Next, observe that the grid cell associated with u_i has side $2^{\ell(u_i)}$ and it contains both $c^*(u_i)$ and x_j. Therefore,

$$\|x_i - c^*(u_i)\| \leq 2^{\ell(u_i)}\sqrt{d}. \tag{4.5}$$

Furthermore, since u_i is an ancestor of v_i, then by the definition of $c(u_i)$ and $c(v_i)$, in each coordinate, the binary expansions of these two vertices are equal from the location $\ell(u_i)$ and up. In the less significant locations, $c(u_i)$ is zeroed while $c(v_i)$ may have arbitrary bits. This means that the difference between $c(u_i)$ and $c(v_i)$ in each coordinate can be at most $2^{\ell(u_i)}$ in absolute value, and consequently $\|c(v_i) - c(u_i)\| \leq 2^{\ell(u_i)}\sqrt{d}$. Recalling that the decompression algorithm defines $\tilde{x}_i = c(v_i)$, we get

$$\|\tilde{x}_i - c(u_i)\| \leq 2^{\ell(u_i)}\sqrt{d}. \tag{4.6}$$

Collecting the above inequalities, we have

$$\begin{aligned}
\|x_i - \eta - \tilde{x}_i\| &= \|x_i - c(u_i) - \eta + c(u_i) - \tilde{x}_i\| \\
&= \|x_i - c^*(u_i) + c(u_i) - \tilde{x}_i\| && \text{Equation (4.4)} \\
&\le \|x_i - c^*(u_i)\| + \|c(u_i) - \tilde{x}_i\| && \text{triangle inequality} \\
&\le 2 \cdot 2^{\ell(u_i)}\sqrt{d} && \text{Equations (4.5) and (4.6)} \\
&\le 2 \cdot 2^{\ell(w)-\Lambda}\sqrt{d} && \text{Equation (4.3).}
\end{aligned}$$

Similarly for j we have $\|x_i - \eta - \tilde{x}_i\| \le 2 \cdot 2^{\ell(w)-\Lambda}\sqrt{d}$. Together, by the triangle inequality,

$$\begin{aligned}
\|\tilde{x}_i - \tilde{x}_j\| &= \|\tilde{x}_i + \eta - x_i + x_i - x_j + x_j - \eta - \tilde{x}_j\| \\
&= \|x_i - x_j\| \pm (\|x_i - \eta - \tilde{x}_i\| + \|x_i - \eta - \tilde{x}_i\|) \\
&= \|x_i - x_j\| \pm 4 \cdot 2^{\ell(w)-\Lambda}\sqrt{d}.
\end{aligned}$$

To complete the proof of Lemma 4.4.4, it now suffices to show $4 \cdot 2^{\ell(w)-\Lambda}\sqrt{d} \le \epsilon \cdot \|x_i - x_j\|$. This follows from Equation (4.2). □

The distance estimation guarantee of Theorem 4.1.1 follows from combining Lemma 4.4.3 and Lemma 4.4.4.

4.4.2 Sketch Size and Construction Time

In this section we prove the sketch size bound and sketching runtime of QuadSketch, which together complete the proof of Theorem 4.1.1.

Lemma 4.4.6. *QuadSketch produces a sketch of size $O(n(d\Lambda + \log n + \log\log\Phi))$ bits.*

Proof. The tree T has n leaves, and we have pruned each non-branching path in it to length Λ. Hence its total size is $O(n\Lambda)$, and its structure can be stored with this many bits using (say) the Eulerian Tour Technique described in Section 4.2. Each short edge label is d bits long, so together they consume $O(nd\Lambda)$ bits. As for the long edges, there can be at most

$O(n)$ of them, since the bottom of each long edge is either a branching node (i.e., a node with more than one child) or a leaf. The long edge labels are lengths of downward paths in the non-pruned tree T^*, whose height bounded by is $L = O(\log \Phi + \Lambda)$. Together the long edge labels consume $O(n \log(\log \Phi + \Lambda))$ bits. Finally, for each point x_i we store the index of its corresponding leaf v_i. Since there are n leaves, this requires $O(n \log n)$ additional bits to store. □

Note that in the statement of Theorem 4.1.1 we set $\Lambda = O(\log(d\Phi/(\epsilon\delta))$, so the $n \log \log \Phi$ term is the above lemma is dominated by $nd\Lambda$.

Lemma 4.4.7. *The QuadSketch construction algorithm runs in time $O(ndL)$.*

Proof. Given a quadtree cell and a point contained in it, in order to bucket the point into a cell in the next level, we need to check for each coordinate whether the point falls in the upper or lower half of the cell. This takes time $O(d)$. Since each point is bucketed once in every level, and we generate T^* for L levels, the quadtree construction time is $O(ndL)$. The pruning step requires just a linear scan of T^*, in time $O(nL)$. □

4.4.3 Metric Compression

In this section we show that QuadSketch can induce a formal solution to the metric sketching problem from Definition 2.1.1. It is looser than our optimal bound from Theorem 2.1.2 by a small factor of $O(\log \log n + \log \log \Phi + \log(1/\epsilon))$. Yet, it is better than the discretized dimension reduction upper bound of $O(\epsilon^{-2} \log(n) \log \Phi)$ bits per point (except in pathological settings where Φ is extremely small). Note that the difference between Theorem 2.1.2 and Theorem 4.1.1 is that here we need the sketch to accurately estimate all distance simultaneously, rather than from a single given points as in Theorem 4.1.1.

Theorem 4.4.8. *Let $\epsilon > 0$. Suppose we are given a point set $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ with Euclidean distances in the range $[1, \Phi]$. Using the QuadSketch algorithm from Section 4.2 as a black-box, we can compute in expected time $\tilde{O}(nd \log(d\Phi/\epsilon))$ a sketch of size*

$$O(\epsilon^{-2} n \log(n) \cdot (\log \log n + \log \log \Phi + \log(1/\epsilon))) \quad \textit{bits},$$

such that with high probability, every distance $\|x_i - x_j\|$ can be recovered from the sketch up to distortion $1 \pm \epsilon$.

Sketching algorithm. As usual, by [JL84] we can assume w.l.o.g. that $O(\epsilon^{-2} \log n)$, by preliminary dimension reduction, which succeeds with high probability.

The sketching algorithm will use multiple invocations of QuadSketch. For all of them, we set $\Lambda = O(\log(d \log \Phi/\epsilon))$ and $L = \log \Phi + \Lambda$. The randomness in each invocation (i.e., the random shift of the quadtree) is independent of all other invocations.

Given a point set X, apply QuadSketch to X and let T_1 be the resulting tree. Let $Q \subset X$ be the padded points in T_1 (meaning those for which the condition of Lemma 4.4.3 is satisfied for T_1). If $|Q| < \frac{1}{2}|X|$, discard the tree and run QuadSketch again, and so on until $|Q| \geq \frac{1}{2}|X|$. Continue by recursion on $X \setminus Q$, until we have constructed a sequence of trees $T_1, \ldots, T_k$ such that,

- In each tree, at least half the points are padded.
- Each tree contains less than half of the points of the previous tree.
- Every point in X is padded in some tree.

The returned sketch contains all trees $T_1, \ldots, T_k$.

For every $i \in [n]$, let $\gamma[i] \in [k]$ denote the index of the first tree in which x_i is padded. Note that we can easily recover $\gamma(i)$ from the sketch, as it is the index of the last tree in which i appears (i.e., the last tree in which x_i was included in the sketched point set and associated with a leaf).

Query algorithm. Given two point indices $i, j \in [n]$, assume w.l.o.g. $\gamma(i) \leq \gamma(j)$. Then, the tree $T_{\gamma(i)}$ has corresponding leaves for both x_i and x_j. We decompress $\tilde{x}_i$ and $\tilde{x}_j$ from $T_{\gamma(i)}$ and return $\|\tilde{x}_i - \tilde{x}_j\|$.

Proof of Theorem 4.4.8. The correctness of the estimate up to distortion $1 \pm \epsilon$ follows from Lemma 4.4.4, since for every pair $i, j \in [n]$, either x_i or x_j is padded in the tree that we use to report the distance between them.

We now bound the sketch size and the running time. Lemma 4.4.3 with $\delta = 0.25$ implies that in each of the trees $T_1, \dots, T_k$, the expected fraction of padded points is 0.75. Hence by Markov's inequality, with probability 0.5 at least half the points are padded. Therefore the number of times we need to build each T_i until a successful attempt (that allows us to proceed to T_{i+1} instead of discarding T_i and building it again) is distributed geometrically with parameter 0.5, and the expected number of attempts is 2. After $k = O(\log n)$ successfully built trees, every point is padded in some tree, Q becomes empty, and the sketching algorithm terminates. Overall we need in expectation $O(\log n)$ invocations of QuadSketch, which by Lemma 4.4.7 takes $\tilde{O}(ndL)$ expected time.

Furthermore, since every tree T_i sketches less than half as many points as the previous one, the total size of the sketches $T_1, \dots, T_k$, by Lemma 4.4.6 is

$$O\left(\sum_{k'=0}^{k-1} \frac{n}{2^{k'}}(d\Lambda + \log \frac{n}{2^{k'}})\right) = O(n(d\Lambda + \log n)).$$

Recalling that $d = O(\epsilon^{-2} \log n)$, $\Lambda = O(\log(d \log \Phi/\epsilon))$, and $L = \log \Phi + \Lambda$, we get the stated bound. □

4.4.4 Nearest Neighbor Search

In this section we prove that QuadSketch is guaranteed to report approximate nearest neighbors even for query points outside the dataset, as formally defined in Definition 3.1.1. The proof is by adapting our techniques developed in Chapter 3. As in the case of metric sketching (Section 4.4.3), the compression bound obtained by QuadSketch is looser than our bound from Theorem 3.1.3 by a small factor, but still asymptotically better than discretized dimension reduction, while being simple and practical.

Theorem 4.4.9. *Let $\epsilon > 0$. Suppose we are given a point set $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ with Euclidean distances in the range $[1, \Phi]$. QuadSketch, with the middle-out compression step described below, computes in time $O(nd \log(d\Phi q/(\epsilon\delta)))$ a sketch of size*

$$O(\epsilon^{-2} n \log(n) \cdot (\log\log n + \log\log \Phi + \log(q/(\epsilon\delta))) + d \log \Phi) \ \textit{ bits,}$$

such that with probability at least $1-\delta$, simultaneously for every q query points in $\mathbb{R}^d$, it is possible to report from the sketch a $(1+\epsilon)$-approximate nearest neighbor in X for each query point.

Technique: Middle-out compression. We now describe how to apply the techniques from Chapter 3 to QuadSketch to support new query points. In that chapter we introduced three techniques: *top-out compression*, *grid quantization* and *root surrogate hashing*. The latter two will turn out to be unnecessary for QuadSketch, since as we will see, they are already "organically" built into the quadtree. As for top-out compression, we combine it into the path compression step of QuadSketch from Section 4.2, which prunes every maximal 1-path except for its top Λ nodes (i.e., "bottom-out" compression). Combining the two yields *middle-out compression*: every maximal 1-path longer than 2Λ is replaced by a long edge, except for its top and its bottom Λ nodes. Note that the QuadSketch algorithm is nearly unchanged, and the sketch size is at most twice as big. In the remainder of this section we show that this modification to QuadSketch allows it to accurately report approximate nearest neighbors for new query points, thereby proving Theorem 4.4.9.

Notation. For every node v in the quadtree, let $C(v)$ denotes the subset of points in X (the dataset) that are contained in the grid cell associated with v. As in Chapters 2 and 3, the quadtree is partitioned into a set $\mathcal{F}(T)$ of *subtrees* by removing the long edges.

For every node v in the quadtree, we let $x_{c(v)}$ denote an arbitrarily chosen fixed point in $C(v)$. We also denote by $s^*(v)$ the "bottom-left" (i.e. minimal in all dimensions) corner of the grid cell associated with v. (We use these notations as they are analogous to *centers* and *surrogates* from Chapters 2 and 3, except here these notions are considerably simplified.)

In the QuadSketch description from Section 4.2, we use the same setting of Λ from Equation (4.1):

$$\Lambda = \log\left(\frac{16 \cdot d^{1.5} \cdot \log\Phi}{\epsilon\delta}\right),$$

and

$$L = \lceil \log\Phi \rceil + 2 + \lceil\Lambda\rceil.$$

Basic properties. We start by stating some basic properties of the sketch.

Claim 4.4.10. *For every point $x \in X$, with probability $1-\delta$, the following holds. If $z \in \mathbb{R}^d$ is any point outside the grid cell that contains x in level ℓ of the quadtree, then $\|x-z\| \geq 8\epsilon^{-1} \cdot 2^{\ell-\Lambda}\sqrt{d}$.*

Proof. This is a restatement of the padding property from Lemma 4.4.3. If x is (ϵ, Λ)-padded in the quadtree, then its grid cell in level ℓ also contains the ball of radius $8\epsilon^{-1} \cdot 2^{\ell-\Lambda}\sqrt{d}$ around x (see Definitions 4.4.1 and 4.4.2). If z is outside the grid cell then it is outside the ball. □

Claim 4.4.11. *Let v be a node in the quadtree, and $x, x' \in \mathbb{R}^d$ points contained in the grid cell associated with v. Then $\|x - x'\| \leq 2^{\ell(v)}\sqrt{d}$.*

Proof. The grid cell associated with v is a hypercube with side $2^{\ell(v)}$ and diameter $2^{\ell(v)}\sqrt{d}$. □

Claim 4.4.12. *For every leaf v of the quadtree, $C(v)$ contains a single point of X, and v is the bottom of a 1-path of length at least Λ.*

Proof. Recall that the top level in the quadtree has side length $4\Delta = 2^{\lceil \log \Phi \rceil + 2}$, and we build it for $L = \lceil \log \Phi \rceil + 2 + \lceil \Lambda \rceil$ levels. After $\lceil \log \Phi \rceil + 2$ levels, the grid has side length 1, and since this is the minimum distance between the data points, every grid cell contains at most a single point in that level. Then we proceed for $\lceil \Lambda \rceil$ additional levels, ensuring that every leaf in the quadtree is the bottom of a 1-path of length at least Λ. □

Claim 4.4.13. *Every subtree leaf in the quadtree is the bottom of a 1-path of length at least Λ.*

Proof. If v is a leaf of the quadtree, this follows from Claim 4.4.12. Otherwise this follows from middle-out compression. □

Query algorithm. Let y be a query point for which we need to report an approximate nearest neighbor from the sketch. The query algorithm is the same as in Section 3.3: starting with the subtree that contains the quadtree root, it recovers the surrogates in the current subtree and chooses the subtree v whose surrogate is the closest to y. If v is a quadtree leaf, its center is returned as the approximate nearest neighbor. Otherwise, the algorithm proceeds by recursion on the subtree under v.

The difference is in the way we recover the surrogates of a given subtree. In Section 3.3 this was done using the root surrogate hashes. Here we can use a simpler, deterministic surrogate recovery subroutine. Let $s^*(H) \in \mathbb{R}^d$ the surrogate of the quadtree root. (We store this point explicitly in the sketch, and it will be convenient to think of it w.l.o.g. as the the origin in $\mathbb{R}^d$.) As observed in Claim 4.4.5, for every tree node v, if we concatenate the bits annotating the edges on the path from the root to v, we get the binary expansion of the point $s^*(H) + s^*(v)$. Therefore, we can recover $s^*(v)$ from the sketch, as long as the path from the root to v does not traverse a long edge.

If the path to v contains long edges (and thus missing bits in the binary expansion of $s^*(v)$), the algorithm completes these bits from the binary expansion of y. Let $r_0, r_1, \ldots$ be the subtree roots traversed by the algorithm, and let $T_0, T_1, \ldots$ be the corresponding subtrees. Let t be the smallest such that the algorithm does not recover the surrogates in T_t correctly (because the bits missing on the long edge connecting T_{t-1} to T_t are not truly equal to those of y). As in Section 3.3, the query algorithm does not know t (it simply always assumes that the bits of y are the correct missing ones), but we will use it for analysis. Note that by definition of t, all surrogates in the subtrees rooted at $r_0, \ldots, r_{t-1}$ are recovered correctly. (The analog of this fact in Chapter 3 was Lemma 3.3.2, whereas here its conclusion holds deterministically.)

Proof of Theorem 4.4.9. Let $x^* \in X$ be a fixed true nearest neighbor of y in X (chosen arbitrarily if there is more than one). We shall assume that the event in Claim 4.4.10 occurs for x^*.

Lemma 4.4.14 (analog of Lemma 3.3.3)**.** *Let $T' \in \mathcal{F}(T)$ be a subtree rooted in r, such that $x^* \in C(r)$. Let v be a leaf of T' that minimizes $\|y - s^*(v)\|$. Then either $x^* \in C(v)$, or every $z \in C(v)$ is a $(1 + O(\epsilon))$-approximate nearest neighbor of y.*

Proof. If $x^* \in C(v)$ then we are done. Assume now that $x^* \in C(u)$ for a leaf $u \neq v$ of T'. Let $\ell = \max\{\ell(v), \ell(u)\}$.

We start by showing that $\|y - x^*\| > \epsilon^{-1}2^\ell\sqrt{d}$. Assume by contradiction this is not the case. Since x^*, $x_{c(u)}$ and $s^*(u)$ are all points in $C(u)$, by Claim 4.4.11 we have,

$$\|x^* - x_{c(u)}\| \leq 2^\ell\sqrt{d} \quad \text{and} \quad \|x_{c(u)} - s^*(u)\| \leq 2^\ell\sqrt{d}.$$

Therefore, by the triangle inequality,

$$\|y - s^*(u)\| \leq \|y - x^*\| + \|x^* - x_{c(u)}\| + \|x_{c(u)} - s^*(u)\| \leq (\epsilon^{-1} + 2)2^\ell\sqrt{d}, \tag{4.7}$$

where we have used the above together with the contradiction hypothesis $\|y-x^*\| \leq \epsilon^{-1}2^\ell\sqrt{d}$.

On the other hand, by Claim 4.4.13, both v and u are the bottom of 1-paths of length at least Λ. This means that x^* and $x_{c(v)}$ are separated already at level $\ell+\Lambda$, and by Claim 4.4.10, this implies

$$\|x^* - x_{c(v)}\| \geq 8\epsilon^{-1} \cdot 2^\ell\sqrt{d}.$$

Moreover, by Claim 4.4.11,

$$\|x_{c(v)} - s^*(v)\| \leq 2^\ell\sqrt{d}.$$

Again using that $\|y - x^*\| \leq \epsilon^{-1}2^\ell\sqrt{d}$ by contradiction, we get by the triangle inequality,

$$\|y - s^*(v)\| \geq \|x^* - x_{c(v)}\| - \|y - x^*\| - \|x_{c(v)} - s^*(v)\| \geq (7\epsilon^{-1} - 1) \cdot 2^\ell\sqrt{d}. \tag{4.8}$$

For $\epsilon < 1/2$, Equations (4.7) and (4.8) together imply $\|y - s^*(v)\| > \|y - s^*(u)\|$, which contradicts the choice of v. Thus we have shown $\|y - x^*\| > \epsilon^{-1}2^\ell\sqrt{d}$.

The lemma now follows because for every $z \in C(v)$,

$$\begin{aligned}
\|y - z\| &\leq \|y - s^*(v)\| + \|s^*(v) - x_{c(v)}\| + \|x_{c(v)} - z\| && (4.9)\\
&\leq \|y - s^*(u)\| + \|s^*(v) - x_{c(v)}\| + \|x_{c(v)} - z\| && (4.10)\\
&\leq \|y - x^*\| + \|x^* - x_{c(u)}\| + \|x_{c(u)} - s^*(u)\| + \|s^*(v) - x_{c(v)}\| + \|x_{c(v)} - z\| && (4.11)\\
&\leq \|y - x^*\| + 4 \cdot 2^\ell\sqrt{d} && (4.12)\\
&\leq (1 + 4\epsilon)\|y - x^*\|, && (4.13)
\end{aligned}$$

where (4.9) and (4.11) are by the triangle inequality, (4.10) is since $\|y-s^*(v)\| \leq \|y-s^*(u)\|$ by choice of v, (4.12) is by applying Lemma 4.4.11 to each of the last four summands, and (4.13) is since we have shown that $\|y - x^*\| > \epsilon^{-1}2^{\ell}\sqrt{d}$. Therefore z is a $(1+4\epsilon)$-approximate nearest neighbor of y. □

Now we prove that the query algorithm returns an approximate nearest neighbor for y. We may assume w.l.o.g. that ϵ is smaller than a sufficiently small constant. Let t be as defined in the surrogate recovery part of the query algorithm above.

We consider two cases. In the first case, $x^* \notin C(r_t)$. Let $i \in \{1,\dots,t\}$ be the smallest such that $x^* \notin C(r_i)$. By applying Lemma 4.4.14 on r_{i-1}, we have that every point in $C(r_i)$ is a $(1+O(\epsilon))$-approximate nearest neighbor of y. After reaching r_i, the algorithm would return the center of some leaf reachable from r_i, and it would be a correct output.

In the second case, $x^* \in C(r_t)$. We will show that every point in $C(r_t)$ is a $(1+O(\epsilon))$-approximate nearest neighbor of y, so once again, once the algorithm arrives at r_t it can return anything.

By definition of t, we know that y does not reside in the grid cell associated with r_t. Therefore, by Claim 4.4.10,

$$\|y - x^*\| \geq 8\epsilon^{-1}2^{\ell(r_t)-\Lambda}\sqrt{d}.$$

On the other hand, by Claim 4.4.13, r_t is the bottom of a 1-path of length at least Λ. Therefore, any two points in $C(r_t)$ are contained in the same grid cell at level $\ell(r_t)-\Lambda$, whose diameter is $2^{\ell(r_t)-\Lambda}\sqrt{d}$. In particular, for every $x \in C(r_t)$ we have,

$$\|x - x^*\| \leq 2^{\ell(r_t)-\Lambda}\sqrt{d}.$$

The above inequalities together imply $\|x-x^*\| \leq \frac{1}{8}\epsilon\|y-x^*\|$, and therefore by the triangle inequality,

$$\|y-x\| \leq \|y-x^*\| + \|x^*-x\| \leq (1+\tfrac{1}{8}\epsilon)\|y-x^*\|.$$

Thus every $x \in C(r_t)$ is a $(1+\epsilon)$-approximate nearest neighbor of y in X.

In the above proof we have assumed that the event in Claim 4.4.10 holds for x^*. This happens with probability $1-\delta$, which proves Theorem 4.4.9 for one query point. To handle q

query points, we can scale δ down to δ/q and take a union bound over the q nearest neighbors of the q query points. Finally, ϵ can be scaled by a constant to yield the theorem. $\square$

Chapter 5

Low-Rank Approximation of Distance Matrices

A natural way to represent a metric space is by a *distance matrix*, in which every entry indicates the distance between a pair of points corresponding to its row and column. This view of the metric space motivates notions of efficient representation that arise from linear algebra, such as approximating the distance matrix by a low-rank matrix. In this chapter, we give a sublinear time algorithm for this problem, which is both simpler and more efficient than previous work. Furthermore, we prove that the algorithm is optimal, up to constants, in terms of the number of entries it reads from the input matrix.

5.1 Introduction

A distance matrix is formally defined as follows:

Definition 5.1.1 (distance matrix)**.** *A matrix $A \in \mathbb{R}^{n\times m}$ is called a* distance matrix *if there is an associated metric space $(\mathcal{Z}, \mathrm{d})$ with $\mathcal{X} = \{x_1, \ldots, x_n\} \subset \mathcal{Z}$ and $\mathcal{Y} = \{y_1, \ldots, y_m\} \subset \mathcal{Z}$, such that $A_{ij} = \mathrm{d}(x_i, y_j)$ for every i, j.*

The intersection between $\mathcal{X}$ and $\mathcal{Y}$ can be arbitrary. In the special case where $n = m$ and $x_i = y_i$ for every $i \in [n]$, we call A a *symmetric* distance matrix. Otherwise we call it *asymmetric*.

Distance matrices are a natural way of representing metric data. Indeed, they arise in various computational contexts, such as learning image manifolds [WS06], image understanding [TDSL00], protein structure analysis [HS93], and more. The recent survey [DPRV15] provides a comprehensive list. Common software packages such as Julia, MATLAB or R include operations specifically design to produce or process such matrices.

Viewing a metric space as a matrix naturally leads to linear algebraic notions of efficient representation, like *low-rank approximation*. Given an integer $k > 0$, the goal is to find a matrix A' of rank k which is as close as possible to A (whose rank can be as large as $\min\{n, m\}$). Specifically, we aim to minimize the squared Frobenius norm[1] error $\|A - A'\|_F^2$. When k is small, using A' as a proxy for A leads to significant saving of computational resources, with only a limited loss in accuracy. For example, storing A requires $\Omega(mn)$ space, while storing A' requires only $O(k(m + n))$ space; likewise, multiplying A by a vector takes $\Omega(mn)$, while multiplying A' by the same vector takes only $O(k(m + n))$ time.[2] Generally, computing low-rank approximations of matrices is a classical computational problem, with a remarkable number of applications in science and engineering.

The best rank-k approximation of A, denote henceforth A_k, can be found in polynomial time $O(\min\{m^2n, mn^2\})$ by computing the singular value decomposition (SVD) of A. However, since the input matrix A is often very large, this running time is rendered infeasible or undesirable. As a result, faster approximate algorithms for computing low-rank approximations have been studied extensively — see the surveys [Mah11, Woo14] and references therein. In particular, this line of work has shown that for any matrix A and an arbitrarily small $\epsilon > 0$, it is possible to compute a matrix A' that attains the optimal Frobenius norm error up to an additive error of $\epsilon\|A\|_F$ (that is, A' satisfies $\|A - A'\|_F^2 \leq \|A - A_k\|_F^2 + \epsilon\|A\|_F^2$), in time nearly linear in the size of A. Furthermore, this additive error guarantee can strengthened to a relative error guarantee, $\|A - A'\|_F^2 \leq (1 + \epsilon)\|A - A_k\|_F^2$. For distance matrices, the running time is $\Omega(mn)$.

Can we get an even faster, *sublinear* running time? Since merely writing down the

[1]The squared Frobenius norm of a matrix M is the sum of squares of its entries, $\|M\|_F^2 = \sum_{i,j} M_{ij}^2$.

[2]In fact, both the storage space and vector multiplication time of A can be made proportional to the number of nonzero entries in A. However, distance matrices are generally dense — their number of zero entries is equal to the intersection size between $\mathcal{X}$ and $\mathcal{Y}$, and thus they have $\Omega(mn)$ nonzero entries.

output A' may take $\Omega(mn)$ time, in this case the algorithm is required to output a low-rank factorization of A', in the form of two matrices $V \in \mathbb{R}^{n\times k}$ and $U \in \mathbb{R}^{k\times m}$ such that $A' = VU$. Nonetheless, for general matrices the answer remains negative, as a single large entry in the input matrix can significantly influence the output, and finding such an entry could take linear time. However, it is known for that for several interesting special classes of matrices, one can find an approximate low-rank solution using only a sublinear amount of time or samples from the input matrix. This includes algorithms for incoherent matrices [CR09], positive semidefinite matrices [MW17], and distance matrices [BW18].

Bakshi and Woodruff [BW18] were the first to consider low-rank approximation of distance matrices. They gave a sublinear time algorithm, that runs in time $O((n+m)^{1+\gamma}) \cdot \mathrm{poly}\,(k, 1/\epsilon)$ where $\gamma > 0$ is an arbitrarily small constant, and finds a rank-k approximation that satisfies the additive error guarantee. Furthermore, they showed that the relative error guarantee cannot be achieved for general distance matrices in sublinear time.

5.1.1 Our Results

In this chapter we present an algorithm for low-rank approximation of distance matrices with an additive error guarantee, that is both simpler and more efficient than prior work. Specifically, we show:

Theorem 5.1.2 (upper bound). *There is a randomized algorithm that given a distance matrix $A \in \mathbb{R}^{n\times m}$, reads $O((n+m)k/\epsilon)$ entries of A, runs in time $\tilde{O}(n+m)\cdot \mathrm{poly}\,(k,1/\epsilon)$, and computes matrices $V \in \mathbb{R}^{n\times k}, U \in \mathbb{R}^{k\times m}$ that with probability 0.99 satisfy*

$$\|A - VU\|_F^2 \le \|A - A_k\|_F^2 + \epsilon\|A\|_F^2. \tag{5.1}$$

We complement the sample complexity of our algorithm with a matching lower bound on the number of entries of the input matrix that must be read by any algorithm.

Theorem 5.1.3 (lower bound). *Let $k \le m \le n$ and $\epsilon > 0$ be such that $k/\epsilon = O(\min(m, n^{1/3}))$. Any randomized and possibly adaptive algorithm that given a distance matrix $A \in \mathbb{R}^{n\times m}$, computes $V \in \mathbb{R}^{n\times k}, U \in \mathbb{R}^{k\times m}$ that satisfy $\|A - VU\|_F^2 \le \|A - A_k\|_F^2 + \epsilon\|A\|_F^2$, must read at* *least $\Omega((n+m)k/\epsilon)$ entries of A in expectation. The lower bound holds even for symmetric distance matrices.*

We include an empirical evaluation of our algorithm on synthetic and real data. The results validate that our approach attains good approximation with faster running time than existing methods.

5.1.2 Technical Overview

Upper bound. Our high level approach is similar to [BW18]: use a classical result of Frieze, Kannan and Vempala [FKV04], which shows how to compute a solution satisfying Equation 5.1 in $\tilde{O}(n+m)\cdot \mathrm{poly}\,(k, 1/\epsilon)$ time, assuming it is possible to sample a row (or a column) of the input matrix with probability proportional to its squared norm. Moreover, the [FKV04] result is robust in the following sense: Suppose we estimate the sampling probabilities instead of computing them exactly. If for some $0 < \delta < 1$ it is guaranteed that the actual sampling probability of each row is no smaller than δ times its correct probability, then the foregoing result works as long as we oversample rows by a factor of $O(1/\delta)$.

Thus, the main challenge is to estimate the row norms in sublinear time. Although this cannot be done for general matrices, distance matrices have additional structure (imposed by the triangle inequality), which makes the problem easier. Specifically, in a distance matrix, the squared norm of a row corresponding to a point $x \in \mathcal{X}$ takes the form

$$r_x = \sum_{y\in\mathcal{Y}} \mathrm{d}(x,y)^2,$$

which is seen to be related to k-means clustering. Indeed, various techniques from the approximate clustering literature can be applied here [Ind99, Che09, CCK15, CCK18]; see a detailed discussion in Section 5.1.3.

Our row norm estimation procedure is considerably simpler than prior methods, and reads only a single row and column from the input matrix, resulting in an optimal sample bound. Let us describe it here in the symmetric case $\mathcal{X} = \mathcal{Y}$. We pick a point x^* uniformly at random from $\mathcal{X}$, and read its corresponding row from the input matrix, thus learning the distance from x^* to any other point in $\mathcal{X}$. For every pair $x, y \in \mathcal{X}$, we estimate the squared distance $\mathrm{d}(x,y)^2$ by the detour through x^*, i.e., by $\mathrm{d}(x,x^*)^2+\mathrm{d}(x^*,y)^2$. Thus, the squared norm r_x of every row x is estimated by

$$\tilde{r}_x=\sum_{y\in\mathcal{X}}(\mathrm{d}(x,x^*)^2+\mathrm{d}(x^*,y)^2)=|\mathcal{X}|\cdot\mathrm{d}(x,x^*)^2+\sum_{y\in\mathcal{X}}\mathrm{d}(x^*,y)^2.$$

As an immediate consequence of the triangle inequality (for squared distances, see Claim 5.2.3), $\tilde{r}_x\geq\frac{1}{2}r_x$. On the other hand, it is easily seen that in expectation over x^* we have $\mathbb{E}_{x^*}[\sum_{x\in\mathcal{X}}\tilde{r}_x]=2\sum_{x\in X}r_x$. Therefore, if we normalize the $\tilde{r}_x$'s to sum to 1, they yield sampling probabilities that satisfy the robust condition stated above (with $\delta=\Omega(1)$). Namely, if we let $p_x=\tilde{r}_x/\sum_{x'}\tilde{r}_{x'}$, then $p_x\geq\Omega(1)\cdot r_x/\sum_{x'}r_{x'}$ while $\sum_x p_x=1$. This is sufficient to support a reduction to [FKV04] as long as we oversample the rows by a constant factor.

After executing the algorithm of [FKV04], we still need one more step to compute the solution (as their method reports U but not V). This amounts to a regression problem that admits efficient approximate solutions by standard techniques. For example, using leverage score sampling (see [Mah11, Woo14]) would result in nearly tight sample complexity, up to a factor of $\log k$. To avoid this factor and obtain the tight sample bound, we use a recent solver of Chen and Price [CP17].

Lower bound. First let us note that an $\Omega(nk)$ lower bound can be easily obtained. It is not hard to see that any $n\times k$ matrix with entries in $\{1,2\}$ is an (asymmetric) distance matrix, since the triangle inequality is satisfied trivially. If we choose a uniformly random matrix from $\{1,2\}^{n\times k}$, then any algorithm that computes a matrix satisfying Equation (5.1) with $\epsilon=\Omega(1)$ must match a $1-\Omega(\epsilon)$ fraction of the entries exactly, yielding the lower bound.

Our $\Omega(nk/\epsilon)$ lower bound is considerably more involved, and uses tools from communication complexity and random matrix theory. For simplicity, let us describe our techniques in the case $k=1$. Consider the problem of reporting the majority bit of a random binary string of length $r=\Theta(1/\epsilon)$. This requires reading $\Omega(r)$ of the input bits. If we stack together n instances into an $n\times r$ random binary matrix A, then reporting the majority bit for a large fraction of rows requires reading $\Omega(nr)$ input bits. This is our target lower bound.

The reduction proceeds by first shifting the values in A from $\{0,1\}$ to $\{1,2\}$, so that it becomes an (asymmetric) distance matrix. A naïve rank-1 approximation would be to replace each entry with 1.5, yielding a total squared Frobenius error of $\frac{1}{4}nr$. However, the optimal rank-1 approximation is (essentially) to replace each row by its true mean value instead of 1.5. By anti-concentration of the binomial distribution, in most rows the majority element appears $\Omega(\sqrt{r})$ times more often than the minority element. A simple calculation shows this leads to a constant additive advantage per row, and $\Omega(n)$ advantage over the whole matrix, of the optimal rank-1 approximation over the naïve one. Since $\epsilon\|A\|_F^2 = O(n)$, any algorithm that satisfies Equation (5.1) must attain a similar advantage.

By spectral properties of random matrices, the optimal rank-1 approximation of A is essentially unique. In particular, the largest singular value of A is much larger than the second-largest one. For technical reasons we need to sharpen this separation even further. We accomplish this by augmenting the matrix with an extra row with very large values, which corresponds to augmenting the metric space with an extra very far point. The upshot is that any algorithm that satisfies Equation (5.1) must approximately recover the mean values for a large fraction of the rows. This allows us to solve the majority problem, by reporting whether each row mean in the rank-1 approximation matrix is smaller or larger than 1.5. This yields the desired lower bound for asymmetric distance matrices. The result for symmetric distance matrices, and for general values of k, builds on similar techniques.

5.1.3 Additional Related Work

As pointed out above, the low rank approximation problem for distance matrices is reduced by [FKV04] to estimating the 1-means clustering cost of $\mathcal{Y}$ with each centroid in $\mathcal{X}$, i.e., estimating $r_x = \sum_{y\in\mathcal{Y}} \mathrm{d}(x,y)^2$ for every $x \in \mathcal{X}$. There are numerous approaches in the literature for this problem. Bakshi and Woodruff [BW18] use a uniform sample of $\mathcal{Y}$ to obtain a coarse estimate, which is then refined recursively. Alternatively, one could obtain a constant approximation of each r_x using the approximate comparator technique of [Ind99] (developed there for k-median clustering, but applicable to k-means as well), or coreset techniques like [Che09]. For our purpose, these approaches exceed the optimal number of samples by at least a few logarithmic factors.

The most closely related approach to ours is the Probability Proportional to Size (PPS) technique of Cohen, Chechik and Kaplan [CCK15, CCK18], although we had not been aware of it at the time of the original publication of our results ([IVWW19]). We now briefly describe it. The technique in [CCK15] is presented in the symmetric case ($\mathcal{X} = \mathcal{Y}$) and for 1-median, i.e., the goal estimate the quantities $m_x = \sum_{x' \in \mathcal{X}} \mathrm{d}(x, x')$ for every x. They let

$$\gamma_x = \max_{s \in S} \frac{\mathrm{d}(x, s)}{\sum_{x' \in \mathcal{X}} \mathrm{d}(x', s)},$$

where S is a uniform sample of $\mathcal{X}$ of size at least 2. They prove that with high probability, on one hand $\sum_x \gamma_x = O(1)$, while on the other hand,

$$\gamma_x \geq \Omega(1) \cdot \max_{s \in \mathcal{X}} \frac{\mathrm{d}(x, s)}{\sum_{x' \in \mathcal{X}} \mathrm{d}(x', s)}.$$

By a known inequality ($\max_i a_i / b_i \geq \sum_i a_i / \sum_i b_i$ for all positive $a_1, \ldots, a_n$, $b_1, \ldots, b_n$), this implies

$$\gamma_x \geq \Omega(1) \cdot \frac{\sum_{s \in \mathcal{X}} \mathrm{d}(x, s)}{\sum_{s \in \mathcal{X}} \sum_{x' \in \mathcal{X}} \mathrm{d}(x', s)} = \frac{m_x}{\sum_{x' \in \mathcal{X}} m_{x'}}.$$

This is the same result as in our Theorem 5.2.4 below, except that it is in the symmetric 1-median case; nonetheless, their techniques extend to asymmetric 1-means as well. We prove the result directly with a simpler argument.

5.2 Upper Bound

5.2.1 Preliminaries

Our algorithm uses two existing sublinear time algorithms as subroutines. They are formalized in the following two theorems. The first theorem, due to Frieze, Kannan and Vempala, reduces low-rank approximation to sampling proportionally to row (or column) norms. We use $A_{i,*}$ to denote the ith row of A.

Theorem 5.2.1 ([FKV04])**.** *Let $A \in \mathbb{R}^{n \times m}$ be any matrix. Let $k > 0$ be an integer, and let $0 < \epsilon < 1$. Let S be a sample of $O(k/\epsilon)$ rows according to a probability distribution $(p_1, \ldots, p_n)$ that satisfies $p_i \geq \Omega(1) \cdot \|A_{i,*}\|_2^2 / \|A\|_F^2$ for every $i = 1, \ldots, n$. Then, in time*

Input: Distance matrix $A \in \mathbb{R}^{n \times m}$. **Output:** Matrices $V \in \mathbb{R}^{n \times k}$ and $U \in \mathbb{R}^{k \times m}$.

```
1: Choose $i^* \in [n]$ and $j^* \in [m]$ uniformly at random.
2: For each $i = 1, \dots, n$: $p_i \leftarrow A_{i,j^*}^2 + A_{i^*,j^*}^2 + \frac{1}{m}\sum_{j=1}^m A_{i^*,j}^2$.
3: Sample $O(k/\epsilon)$ rows of $A$ according to the distribution proportional to $(p_1, \dots, p_n)$.
4: Compute $U$ from the sample, using Theorem 5.2.1.
5: Compute $V$ from $A$ and $U$, using Theorem 5.2.2.
6: Return $V, U$.
```

Algorithm 1: Low-rank approximation for distance matrices

$O(mk/\epsilon + \mathrm{poly}(k, 1/\epsilon))$ *we can compute from* S *a matrix* $U \in \mathbb{R}^{k \times m}$*, that with probability* 0.99 *satisfies*

$$\|A - AU^TU\|_F^2 \le \|A - A_k\|_F^2 + \epsilon\|A\|_F^2. \tag{5.2}$$

The second theorem, due to Chen and Price, approximately solves a regression problem while reading only a small number of columns of the input matrix.

Theorem 5.2.2 ([CP17]). *Let* $k > 0$ *be an integer, and let* $0 < \epsilon < 1$*. There is a randomized algorithm that given matrices* $A \in \mathbb{R}^{n \times m}$ *and* $U \in \mathbb{R}^{k \times m}$*, reads only* $O(k/\epsilon)$ *columns of* A*, runs in time* $\tilde{O}(m+n) \cdot \mathrm{poly}(k, 1/\epsilon)$*, and returns* $V \in \mathbb{R}^{n \times k}$ *that with probability* 0.99 *satisfies*

$$\|A - VU\|_F^2 \le (1+\epsilon) \min_{X \in \mathbb{R}^{n \times k}} \|A - XU\|_F^2. \tag{5.3}$$

Since our sampling procedure evaluates the sum of squared distances (rather than just distances), we need the following approximate version of the triangle inequality.

Claim 5.2.3. *For every* $x, y, z \in \mathcal{Z}$ *in a metric space* $(\mathcal{Z}, \mathrm{d})$, $\mathrm{d}(x,y)^2 \le 2(\mathrm{d}(x,z)^2 + \mathrm{d}(z,y)^2)$.

Proof. By the triangle inequality, $\mathrm{d}(x,y)^2 \le (\mathrm{d}(x,z) + \mathrm{d}(z,y))^2 = \mathrm{d}(x,z)^2 + 2\mathrm{d}(x,z)\mathrm{d}(z,y) + \mathrm{d}(z,y)^2$. By the inequality of means, $\mathrm{d}(x,z)\mathrm{d}(z,y) \le \frac{1}{2}(\mathrm{d}(x,z)^2 + \mathrm{d}(z,y)^2)$. □

5.2.2 Algorithm

In this section we prove Theorem 5.1.2. The algorithm is stated in Algorithm 1. The main step in the analysis is to provide guarantees for the sampling probabilities p_i computed in Steps 1 and 2 of the algorithm. They are specified by the following theorem.

Theorem 5.2.4. *There is a randomized algorithm that given a distance matrix $A \in \mathbb{R}^{n\times m}$, runs in time $O(m+n)$, reads $O(m+n)$ entries of A, and outputs sampling probabilities $(p_1,\dots,p_n)$, that with probability $1-\delta$ satisfy $p_i \ge \Omega(\delta)\cdot\|A_{i,*}\|_2^2/\|A\|_F^2$ for every $i=1,\dots,n$.*

Proof. Let $(\mathcal{Z},\mathrm{d})$ be the metric space associated with A. Let $\mathcal{X}=\{x_1,\dots,x_n\}$ and $\mathcal{Y}=\{y_1,\dots,y_m\}$ be the pointsets associated with its rows and its columns, respectively. Choose a uniformly random $i^*\in[n]$ and a uniformly random $j^*\in[m]$. For every $i\in[n]$, the output sampling probabilities are given by

$$p_i = \mathrm{d}(x_i,y_{j^*})^2 + \mathrm{d}(x_{i^*},y_{j^*})^2 + \frac{1}{m}\sum_{j=1}^m \mathrm{d}(x_{i^*},y_j)^2.$$

All p_i's can be computed in time $O(n+m)$ and by reading $n+m$ entries of A, since they only involve distances between x_{i^*} to $\mathcal{Y}$ and between y_{j^*} to $\mathcal{X}$. For every $i\in[n]$,

$$\begin{aligned}
\|A_{i,*}\|_2^2 &= \sum_{j=1}^m \mathrm{d}(x_i,y_j)^2 \\
&\le 2\sum_{j=1}^m \left(\mathrm{d}(x_i,y_{j^*})^2 + \mathrm{d}(y_{j^*},y_j)^2\right) && \text{by Claim 5.2.3} \\
&\le 2\sum_{j=1}^m \left(\mathrm{d}(x_i,y_{j^*})^2 + 2\mathrm{d}(x_{i^*},y_{j^*})^2 + 2\mathrm{d}(x_{i^*},y_j)^2\right) && \text{by Claim 5.2.3} \\
&= 2m\cdot\mathrm{d}(x_i,y_{j^*})^2 + 4m\cdot\mathrm{d}(x_{i^*},y_{j^*})^2 + 4\sum_{j=1}^m \mathrm{d}(x_{i^*},y_j)^2 \\
&\le 4m\cdot p_i.
\end{aligned}$$

On the other hand:

- For every fixed i, in expectation over j^*, we have $\mathbb{E}\left[\mathrm{d}(x_i,y_{j^*})^2\right] = \frac{1}{m}\sum_{j=1}^m \mathrm{d}(x_i,y_j)^2$.
- For every fixed j, in expectation over i^*, we have $\mathbb{E}\left[\mathrm{d}(x_{i^*},y_j)^2\right] = \frac{1}{n}\sum_{i=1}^n \mathrm{d}(x_i,y_j)^2$.
- In expectation over i^* and j^*, we have $\mathbb{E}\left[\mathrm{d}(x_{i^*},y_{j^*})^2\right] = \frac{1}{nm}\sum_{i=1}^n\sum_{j=1}^m \mathrm{d}(x_i,y_j)^2$.

Thus,

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{n} p_i\right] &= \sum_{i=1}^{n}\left(\mathbb{E}\left[\mathrm{d}(x_i, y_{j^*})^2\right] + \mathbb{E}\left[\mathrm{d}(x_{i^*}, y_{j^*})^2\right] + \mathbb{E}\left[\frac{1}{m}\sum_{j=1}^{m}\mathrm{d}(x_{i^*}, y_j)^2\right]\right)\\ &= 3n \cdot \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\mathrm{d}(x_i, y_j)^2\\ &= \frac{3}{m}\|A\|_F^2.\end{aligned}$$

By Markov's inequality, $\sum_{i=1}^{n} p_i \leq \frac{3}{\delta m}\|A\|_F^2$ with probability $1-\delta$. Normalizing the p_i's by their sum yields the theorem. □

We remark that if A is a symmetric distance matrix, i.e., $\mathcal{X} = \mathcal{Y}$, the sampling probabilities can be simplified to choosing a single $i^* \in [n]$ uniformly at random, and letting $p_i = \mathrm{d}(x_i, x_{i^*})^2 + \frac{1}{n}\sum_{j=1}^{m}\mathrm{d}(x_{i^*}, x_j)^2$. The proof is similar to the above.

We also remark that essentially the same result follows from [CCK15] (see Section 5.1.3 for details).

Proof of Theorem 5.1.2. Consider Algorithm 1. By Theorem 5.2.4, the probabilities computed in Steps 1–2 are suitable for invoking Theorem 5.2.1. This ensures that the matrix U computed in Steps 3–4 satisfies Equation (5.2). Theorem 5.2.2 guarantees that the matrix V computed in Step 5 satisfies Equation (5.3). Putting these together, we have

$$\begin{aligned}\|A - VU\|_F^2 &\leq (1+\epsilon)\min_{X\in\mathbb{R}^{n\times k}}\|A - X^T U\|_F^2 && \text{by Equation (5.3)}\\ &\leq (1+\epsilon)\|A - AU^T U\|_F^2 \\ &\leq (1+\epsilon)\left(\|A - A_k\|_F^2 + \epsilon\|A\|_F^2\right) && \text{by Equation (5.2)}\\ &\leq \|A - A_k\|_F^2 + \epsilon\cdot(2+\epsilon)\cdot\|A\|_F^2 && \text{since } \|A - A_k\|_F^2 \leq \|A\|_F^2,\end{aligned}$$

and we can scale ϵ by a constant. This proves Equation (5.1). For the query complexity bound, note that Steps 1 and 2 in Algorithm 1 read exactly $m+n-1$ entries of A (the i^*-th row and the j^*-th column). Theorem 5.2.1 reads $O(k/\epsilon)$ rows of A, and Theorem 5.2.2 reads $O(k/\epsilon)$ columns of A, yielding a total of $O((n+m)k/\epsilon)$ entries read. Finally, the running time is the sum of running times of Theorems 5.2.1, 5.2.2 and 5.2.4. □

5.3 Lower Bound for Rank-1 Approximation

We now begin proving Theorem 5.1.3. For a clearer presentation, in this section we prove it in the special case $k = 1$, and for asymmetric distance matrices. This case encompasses the main ideas, and will serve as the backbone for the full proof of Theorem 5.1.3. In Section 5.4 we extend the proof to any rank k, and in Section 5.5 we show it applies also to symmetric distance matrices, thus completing the proof of Theorem 5.1.3 in its full generality. For concreteness, let us formally state the special case that will be proven in this section.

Theorem 5.3.1. *Let n, r, ϵ be such that $r \leq n$ and $1 > \epsilon \geq \Omega(n^{-1/3})$. Any randomized algorithm that given a distance matrix $A \in \mathbb{R}^{n\times r}$, computes $V \in \mathbb{R}^{n\times k}, U \in \mathbb{R}^{k\times r}$ that with probability $2/3$ satisfy $\|A - VU\|_F^2 \leq \|A - A_1\|_F^2 + \epsilon\|A\|_F^2$, must read at least $\Omega(n/\epsilon)$ entries of A in expectation.*

5.3.1 Preliminaries: Linear Algebra

Matrix norms. Let $M = (M_{ij}) \in \mathbb{R}^{m\times n}$ be a matrix of rank r. Let $\sigma_1 \geq \ldots \geq \sigma_r$ be its non-zero singular values, sorted from largest to smallest. The *Frobenius norm* of M is defined as

$$\|M\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} M_{ij}^2},$$

and is also known to be equal to $\sqrt{\sum_{i=1}^{r}\sigma_i^2}$. The *spectral norm* of M is defined as

$$\|M\|_2 = \max_{x\in\mathbb{R}^n : x\neq\mathbf{0}} \frac{\|Ax\|_2}{\|x\|_2},$$

and is also known to be equal to σ_1.

Optimal low-rank approximation by the SVD. We recall the singular value decomposition (SVD): M can be written as $M = U\Sigma V^T$, where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ have orthonormal columns, and $\Sigma \in \mathbb{R}^{r\times r}$ is a diagonal matrix whose diagonal entries are $\sigma_1, \ldots, \sigma_r$.

Let $k \in [r]$. Denote by $U_k \in \mathbb{R}^{m\times k}$ and $V_k \in \mathbb{R}^{n\times k}$ the restriction of U and V, respectively, to their first (left) k columns. Denote by $\Sigma_k \in \mathbb{R}^{k\times k}$ the restriction of Σ to its top-left $k \times k$

principal submatrix. Denote:

$$M_k = U_k \Sigma_k V_k^T.$$

We will often use this notation for any matrix M and integer k.

The following theorem is standard and well-known.

Fact 5.3.2 (Eckart-Young-Mirski theorem)**.** *M_k is the optimal rank-k approximation of M, both in the Frobenius and in the spectral norms. More precisely:*

$$\min_{M' \text{ of rank } k} \|M - M'\|_F^2 = \|M - M_k\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2,$$

and

$$\min_{M' \text{ of rank } k} \|M - M'\|_2^2 = \|M - M_k\|_2^2 = \sigma_{k+1}^2.$$

Projections. A square matrix Π is a *projection matrix* if $\Pi^2 = \Pi$. It is furthermore an *orthogonal* projection if $\Pi = \Pi^T$. Let $\mathrm{im}(\Pi)$ be the image subspace of Π, and let $\mathrm{im}(\Pi)^\perp$ denote its orthogonal subspace. The orthogonal projection on $\mathrm{im}(\Pi)^\perp$ equals $I - \Pi$, and we will often denote it by $\Pi^\perp$.

We remark that if U is a matrix with orthonormal columns, then the orthogonal projection matrix on the subspace spanned by its columns is UU^T.

The following is a well-known extension of the Pythagorean theorem.

Fact 5.3.3 (Pythagorean theorem)**.** *Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m' \times n}$ be such that $A^T B = 0$. Then,*

$$\|A + B\|_F^2 = \|A\|_F^2 + \|B\|_F^2.$$

In particular, if $M \in \mathbb{R}^{m \times n}$ and $\Pi \in \mathbb{R}^{n \times n}$ is an orthogonal projection, then

$$\|M\|_F^2 = \|M\Pi\|_F^2 + \|M\Pi^\perp\|_F^2.$$

Singular values under perturbation. The following fact, about singular values under a perturbation of restricted rank, is a consequence of [Tho76, Theorem 1].

Fact 5.3.4. *Let $\beta > \alpha > 0$. Let M be a matrix of rank r, such that each of its nonzero singular values lies in the interval $[\alpha, \beta]$. If L is a matrix with rank at most ℓ, then $M + L$ has at least $r - \ell$ singular values in $[\alpha, \infty)$, and at least $r - \ell$ singular values in $[0, \beta]$.*

5.3.2 Preliminaries: The Majority Problem on Random Instances

In the majority problem, the goal is to compute the majority bit of an input bitstring. We will show the hardness of low-rank approximation via reduction from solving multiple random instances of the majority problem. The sample complexity hardness of this problem is well-known, and is stated in the following lemma. The proof is included for completeness.

Lemma 5.3.5. *Let $r, t > 0$ be integers. Any deterministic algorithm that gets a uniformly random matrix $S \in \{0,1\}^{t \times r}$ as input, and outputs $s^* \in \{0,1\}^t$ such that for every $i \in [t]$, $\Pr[s^*(i) = \text{majority element of } i\text{th row of } S] \geq 2/3$, must read in expectation at least $\Omega(rt)$ entries of S.*

Proof. The reduction is from the distributional Gap-Hamming communication problem, which is defined as follows. Alice has a bit string x in $\{0,1\}^r$ and Bob has a bit string y in $\{0,1\}^r$, where x and y are independent and uniformly distributed. Let $\Delta(x, y)$ denote their Hamming distance. The goal is to decide whether $\Delta(x, y) \geq \frac{1}{2}r + \sqrt{r}$ or $\Delta(x, y) \leq \frac{1}{2}r - \sqrt{r}$. If neither case holds, then any output is considered successful. The information cost under this distribution is $\Omega(r)$ [BGPW16].

Next consider the r-fold version of the same problem, i.e., Alice and Bob are given r instances of distributional Gap-Hamming, and they need to solve a constant fraction of them. By a standard direct sum theorem (see e.g. [BR14]), this requires $\Omega(rt)$ bits of communication.

Finally we reduce this problem to the majority problem in the lemma statement. Let X denote the xor matrix of Alice's and Bob's matrices. The Gap Hamming problem is equivalent to finding the majority bit over rows of X in which the majority bit appears at least $\frac{1}{2}r + \sqrt{r}$ times (call these rows "typical"). By binomial anti-concentration (Lemma 5.3.6 below), this happens in a large constant fraction of the rows. Given a black-box algorithm for the majority problem that queries q entries of the input matrix, Alice and Bob can simulate it on M by communicating to each other only those entries of their matrices, which costs them $\Theta(q)$. The algorithm solves a large fraction of the rows, and thus a large fraction of the typical rows. Hence they have solved the Gap Hamming problem, and $q = \Omega(rt)$. $\square$

We will also need the following standard fact about anti-concentration of the binomial distribution: in a random length-r bitstring, the majority bit is likely appear $\Omega(\sqrt{r})$ times more than the other bit.

Lemma 5.3.6 (anti-concentration)**.** *Let $0 < \delta < 1$. Let $s \in \{1,2\}^r$ be a uniformly random majority instance. Then, for $\gamma = \Omega(\delta)$, the majority element of s appears in it at least $\frac{1}{2}r + \gamma\sqrt{r}$ times with probability at least $1-\delta$.*

Proof. Let $X \sim \text{Binomial}(r, \frac{1}{2})$. The statement we need to show is equivalent to $\Pr[|X - \frac{1}{2}r| < \frac{1}{2}\gamma\sqrt{r}] < \delta$, or equivalently,

$$\sum_{i=\lfloor \frac{1}{2}r - \frac{1}{2}\gamma\sqrt{r} \rfloor}^{\lceil \frac{1}{2}r + \frac{1}{2}\gamma\sqrt{r} \rceil} \Pr[X = i] < \delta.$$

Let $\gamma = \sqrt{\pi/2} \cdot \delta$. Note that $\Pr[X = i] \le \Pr[X = \lfloor r/2 \rfloor]$ for every i, and $\Pr[X = \lfloor r/2 \rfloor] = 2^{-r}\binom{r}{\lfloor r/2 \rfloor} \le 1/\sqrt{2\pi r}$ by a known estimate. Therefore, the above left-hand side sum is upper-bounded by $2\gamma\sqrt{r}/\sqrt{2\pi r} = \delta$ as needed. $\square$

5.3.3 Hard Distribution over Distance Matrices

By Yao's principle, in order to prove Theorem 5.3.1 it suffices to construct a distribution over distance matrices, and prove the sampling lower bound for any deterministic algorithm that operates on inputs from that distribution. We now define a suitable hard distribution and prove some of its properties.

Random majority instances. Given n and $\epsilon > 0$, let $\beta, C > 0$ be constants that will be chosen later. (β will be sufficiently small and C sufficiently large.) Let $r = \beta/\epsilon$, and assume w.l.o.g. this is an integer by letting ϵ be sufficiently smaller. Note that in Lemma 5.3.5, we can symbolically replace the majority alphabet $\{0,1\}$ with any alphabet of size 2, and here we will use $\{1,2\}$. Let $S \in \{1,2\}^{n\times r}$ be a uniformly random matrix. Let $s_1, \ldots, s_n$ be its rows. We call each of its rows an *instance* (of the majority problem). Thus S is an instance of the random multi-instance majority problem from Lemma 5.3.5 (with $t = n$). For every instance s_i, let $\mu_i = \frac{1}{r}\sum_{j=1}^{r} s_{ij}$ denote its mean.

Typical and atypical instances. We call an instance s in S *typical* if its majority element appears in it at least $\frac{1}{2}r + \gamma\sqrt{r}$ times, where γ is the constant from Lemma 5.3.6. Otherwise, we call the instance *atypical*. Let Ψ_{typical} denote the event there are at least $0.9n$ typical instances. By Markov's inequality, $\Pr[\Psi_{\text{typical}}] \geq 1 - 10\delta$.

The hard distribution over distance matrices. Our goal is to solve S via reduction to rank-1 approximation of distance matrices. To this end, we transform S into a random distance matrix. First, we randomly permute the rows of S to obtain a matrix A. The random permutation is denoted by $\pi : [n] \to [n]$.[3] Then, we add an additional $(n+1)$-th row to A, whose entries are all equal $M = \sqrt{Cn}$. The matrix with the added row is denoted by $\bar{A}$, and this is the matrix sampled from our hard distribution. We now show that $\bar{A}$ is indeed an (asymmetric) distance matrix.

Lemma 5.3.7. *Every supported $\bar{A}$ is a distance matrix.*

Proof. Consider a symbolic pointset $X = P \cup Q$ where $P \cap Q = \emptyset$, such that $P = \{p_1, \dots, p_{n+1}\}$ corresponds to the rows of $\bar{A}$, and $Q = \{q_1, \dots, q_r\}$ to the columns of $\bar{A}$. Our goal is to define a metric d on X such that $\mathrm{d}(p_i, q_j) = \bar{A}_{ij}$ for every $i \in [n+1]$ and $j \in [r]$. We need to set the rest of the distances such that d is indeed a metric – that is, such that d satisfies the triangle inequality. For every $i, i' \in [n]$ we set $\mathrm{d}(p_i, p_{i'}) = 1$. For every $j, j' \in [r]$ we set $\mathrm{d}(q_j, q_{j'}) = 1$. Finally we need to set the distances from p_{n+1}. By construction of $\bar{A}$ we already have $\mathrm{d}(p_{n+1}, q_j) = M$ for every $j \in [r]$. We set all the remaining distances, $\mathrm{d}(p_i, p_{n+1})$ for every $i \in [n]$, to also be M.

We need to verify that for all distinct triplets $x, y, z \in X$, $\mathrm{d}(x, y) \leq \mathrm{d}(x, z) + \mathrm{d}(z, y)$. Indeed, all distances are in $\{1, 2, M\}$. If $\mathrm{d}(x, y) \in \{1, 2\}$ then the inequality holds for any

[3]The random permutation is for a technical reason and does not change the distribution. Specifically, our reduction is assumes that we have a black-box algorithm $\mathcal{ALG}$ that computes a rank-1 approximation of an input distance matrix. The random permutation of the rows would ensure that $\mathcal{ALG}$ solves each majority instance s_i with the same probability, rather than, say, focusing on a few fixed instances $\{s_i\}_{i=1}^{n'}$, $n' \ll n$, and never attempting the rest.

setting of $\mathrm{d}(x, z)$ and $\mathrm{d}(z, y)$. Otherwise $\mathrm{d}(x, y) = M$, hence necessarily either $x = p_{n+1}$ or $y = p_{n+1}$, and in both cases $\mathrm{d}(x, z) + \mathrm{d}(z, y) \geq \max\{\mathrm{d}(x, z), \mathrm{d}(z, y)\} \geq \mathrm{d}(p_{n+1}, z) = M = \mathrm{d}(x, y)$ as needed. $\square$

Next, we state certain bounds on the best rank-1 approximation $\bar{A}_1$ error of $\bar{A}$.

Lemma 5.3.8. *$\|\bar{A} - \bar{A}_1\|_F^2 + \epsilon\|\bar{A}\|_F^2 \leq \sum_{i=1}^n \|s_i - \mu_i \mathbf{1}\|_2^2 + (4 + C)\beta n$.*

Proof. Let $\bar{A}^*$ be the matrix in which each row equals $\mathbf{1}$ times the mean of the corresponding row of A. Then $\|\bar{A} - \bar{A}_1\|_F^2 \leq \|\bar{A} - \bar{A}^*\|_F^2 = \sum_{i=1}^n \|s_i - \mu_i \mathbf{1}\|_2^2$, where the first inequality is since $\bar{A}^*$ has rank 1 (each of its rows is a multiple of $\mathbf{1}$). Note that the sum ranges only up to n and not $n+1$, since in the $(n+1)$th row all entries are equal (to M) and thus it contributes 0 to $\|\bar{A} - \bar{A}^*\|_F^2$. This bounds the first summand in the lemma. To bound the second summand, note that each entry in the first n rows of $\bar{A}$ is at most 2, thus contributing in total $4rn$ to $\|\bar{A}\|_F^2$. The final row contributes $rM^2 = Crn$. Recalling that $\epsilon r = \beta$, we have $\epsilon\|A\|_F^2 \leq (4 + C)\beta n$. $\square$

Corollary 5.3.9. *$\|\bar{A} - \bar{A}_1\|_F^2 + \epsilon\|\bar{A}\|_F^2 \leq \frac{1}{4}nr + (4 + C)\beta n$.*

Proof. For every s_i, its mean μ_i minimizes the sum of squared differences from a single value, namely $\|s_i - \mu_i \mathbf{1}\|_2^2 = \min_{\nu \in \mathbb{R}} \|s_i - \nu \mathbf{1}\|_2^2$. In particular, $\|s_i - \mu_i \mathbf{1}\|_2^2 \leq \|s_i - 1.5 \cdot \mathbf{1}\|_2^2$. Furthermore, since $s_i \in \{1, 2\}^r$, we have $\|s_i - 1.5 \cdot \mathbf{1}\|_2^2 = r \cdot (\frac{1}{2})^2 = \frac{1}{4}r$. Hence $\sum_{i=1}^n \|s_i - \mu_i \mathbf{1}\|_2^2 \leq \frac{1}{4}nr$, and the corollary follows from Lemma 5.3.8. $\square$

5.3.4 Spectral Properties

We will require some facts from random matrix theory about the spectrum of A. For the next two lemmas, write $A = 1.5J + B$ where J is the all-1's matrix and B is a matrix with i.i.d. random entries chosen uniformly from $\{-\frac{1}{2}, \frac{1}{2}\}$. Let $P_{\mathbf{1}}$ denote the orthogonal projection on the subspace spanned by $\mathbf{1}$.

Lemma 5.3.10. *Suppose $r^3 = O(n)$. With probability $1 - e^{-\Omega(n/r^{3/2})}$, $\|A - A_1\|_F^2 \geq \frac{1}{4}nr - O(n)$.*

Proof. We use a sharp estimate of [FS10] on the smallest singular value of B (see also eq. (2.5) in [RV10]). It states that with probability $1-\exp(-\Omega(n/r^{3/2}))$, all r singular values of B are at least $\frac{1}{4}n - O(\sqrt{nr})$. Since A is obtained from B by adding a rank-1 matrix (namely $1.5J$), then by Fact 5.3.4, A has at least $r-1$ singular values which are at least $\frac{1}{4}n - O(\sqrt{nr})$. Therefore $\|A - A_1\|_F^2$, which is the sum of all squared singular values of A except the largest, is at least $(r-2)\cdot(\frac{1}{4}n - O(\sqrt{nr})) = \frac{1}{4}nr - O(n) - O(r^{3/2}\sqrt{n})$. □

Lemma 5.3.11. *Let $\zeta > 0$. Let Z be a rank-1 matrix such that $\|Z\|_2 \leq \zeta\sqrt{n}$. Then with probability $1-o(1)$, $\|AP_{\mathbf{1}}^{\perp} - Z\|_F^2 \geq \|AP_{\mathbf{1}}^{\perp}\|_F^2 - \zeta \cdot O(n)$.*

Proof. By the Hoffman-Wielandt inequality [HW03] for singular values (see also Exercise 22(v) in [Tao10]), any $X, Y \in \mathbb{R}^{n\times r}$ satisfy $\|X - Y\|_F^2 \geq \sum_{j=1}^{r}(\sigma_j(X) - \sigma_j(Y))^2$, where $\sigma_1(X) \geq \ldots \geq \sigma_r(X)$ and $\sigma_1(Y) \geq \ldots \geq \sigma_r(Y)$ are their respective sorted singular values. In particular, letting $X = AP_{\mathbf{1}}^{\perp}$ and $Y = Z$, since Z has rank 1, we have

$$\begin{aligned}
\|AP_{\mathbf{1}}^{\perp} - Z\|_F^2 &\geq \sum_{j=2}^{r}(\sigma_j(AP_{\mathbf{1}}^{\perp}))^2 + \Big(\sigma_1(AP_{\mathbf{1}}^{\perp}) - \sigma_1(Z)\Big)^2 \\
&= \|AP_{\mathbf{1}}^{\perp}\|_F^2 - 2\sigma_1(AP_{\mathbf{1}}^{\perp})\sigma_1(Z) + (\sigma_1(Z))^2 \\
&= \|AP_{\mathbf{1}}^{\perp}\|_F^2 - 2\|AP_{\mathbf{1}}^{\perp}\|_2\|Z\|_2 + \|Z\|_2^2.
\end{aligned}$$

Therefore it suffices to show that $\|AP_{\mathbf{1}}^{\perp}\|_2 = O(\sqrt{n})$. Indeed, $\|AP_{\mathbf{1}}^{\perp}\|_2 = \|(1.5J+B)P_{\mathbf{1}}^{\perp}\|_2 = \|BP_{\mathbf{1}}^{\perp}\|_2 \leq \|B\|_2$, and an upper bound $\|B\|_2 = O(\sqrt{n})$ is well known, e.g., see Proposition 2.4 in [RV10]. □

Since $r = \beta/\epsilon$ and in Theorem 5.3.1 we assume $\epsilon^{-3} = O(n)$, Lemma 5.3.10 is satisfied with probability $1 - o(1)$. Therefore,

Corollary 5.3.12. *Denote by $\Psi_{spectral}$ the event that the conclusions of both Lemma 5.3.10 and Lemma 5.3.11 hold. Then $\Pr[\Psi_{spectral}] \geq 1 - o(1)$, and therefore $\Pr[\Psi_{typical} \wedge \Psi_{spectral}] \geq 1 - 10\delta - o(1)$.*

5.3.5 The Reduction

Suppose we have a deterministic algorithm for the rank-1 approximation problem of distance matrices from the distribution from Section 5.3.3. That is, given $\bar{A}$, it returns $\bar{A}' = \bar{a}b^T$, where $\bar{a} \in \mathbb{R}^{n+1}$ and $b \in \mathbb{R}^r$, such that

$$\|\bar{A} - \bar{a}b^T\|_F^2 \leq \|\bar{A} - \bar{A}_1\|_F^2 + \epsilon\|\bar{A}\|_F^2. \tag{5.4}$$

Let $a \in \mathbb{R}^n$ denote the restriction of $\bar{a}$ to the first n entries. By scaling (i.e., multiplying $\bar{a}$ by a constant and b by its reciprocal), we can assume w.l.o.g. that $\bar{a}_{n+1} = M$. Since the $(n+1)$th row of $\bar{A}$ equals $M \cdot \mathbf{1}$, we have

$$\|\bar{A} - \bar{a}b^T\|_F^2 = \|A - ab^T\|_F^2 + \|M \cdot \mathbf{1} - \bar{a}_{n+1}b\|_2^2 = \|A - ab^T\|_F^2 + M^2\|\mathbf{1} - b\|_2^2.$$

If we rearrange this, and use Equation (5.4) and Corollary 5.3.9 as an upper bound on $\|\bar{A} - \bar{a}b^T\|_F^2$ and Lemma 5.3.10 as a lower bound on $\|A - ab^T\|_F^2$, we get $M^2\|\mathbf{1} - b\|_2^2 \leq (4 + C)\beta n + O(n)$. Plugging $M = \sqrt{Cn}$,

$$\|\mathbf{1} - b\|_2^2 \leq \left(\frac{4}{C} + 1\right)\beta + \frac{O(1)}{C} = \frac{O(1)}{C}. \tag{5.5}$$

This inequality yields the following two lemmas.

Lemma 5.3.13. *We have $1 - \eta/\sqrt{r} \leq \|b^T P_{\mathbf{1}}\|/\|\mathbf{1}\| \leq 1 + \eta/\sqrt{r}$, where $\eta > 0$ is a constant that can be made arbitrarily small by choosing $C > 0$ sufficiently large.*

Proof. By the triangle inequality we have

$$\|\mathbf{1}\|_2 - \|\mathbf{1} - b\|_2 \leq \|b\|_2 \leq \|\mathbf{1}\|_2 + \|\mathbf{1} - b\|_2.$$

The upper bound implies

$$\|b^T P_{\mathbf{1}}\| \leq \|b\| \leq \|\mathbf{1}\|_2 + \|\mathbf{1} - b\|_2. \tag{5.6}$$

The lower bound implies

$$\|\mathbf{1}-b\|_2^2 = \|\mathbf{1}\|_2^2 + \|b\|_2^2 - 2b^T\mathbf{1} \geq \|\mathbf{1}\|_2^2 + \|\mathbf{1}\|_2^2 + \|\mathbf{1}-b\|_2^2 - 2\|\mathbf{1}\|_2\|\mathbf{1}-b\|_2 - 2b^T\mathbf{1},$$

which rearranges to $b^T\mathbf{1} \geq \|\mathbf{1}\|_2^2 - \|\mathbf{1}\|_2\|\mathbf{1}-b\|_2$, implying

$$\|b^T P_{\mathbf{1}}\| = b^T(\frac{1}{\|\mathbf{1}\|_2}\mathbf{1}) \geq \|\mathbf{1}\|_2 - \|\mathbf{1}-b\|_2. \tag{5.7}$$

Putting Equations (5.6) and (5.7) together,

$$\|\mathbf{1}\|_2 - \|\mathbf{1}-b\|_2 \leq \|b^T P_{\mathbf{1}}\| \leq \|\mathbf{1}\|_2 + \|\mathbf{1}-b\|_2,$$

and the lemma follows since $\|\mathbf{1}\|_2 = \sqrt{r}$ and since by eq. (5.5), $\|\mathbf{1}-b\|_2$ is a constant that can be made arbitrarily small by choosing $C > 0$ sufficiently large. □

Lemma 5.3.14. *$\|a\| = O(\sqrt{n})$.*

Proof. By the triangle inequality, $\|b\|_2 \geq \|\mathbf{1}\|_2 - \|b-\mathbf{1}\|_2$. Since $\|\mathbf{1}\|_2 = \sqrt{r}$ and $\|b-\mathbf{1}\|_2$ is an arbitrarily small constant by Equation (5.5), $\|b\|_2 \geq \frac{1}{2}\sqrt{r}$. Thus

$$\|ab^T\|_F^2 = \|a\|_2^2\|b\|_2^2 \geq \tfrac{1}{4}r\|a\|_2^2. \tag{5.8}$$

We finish by showing that $\|ab^T\|_F^2 = O(nr)$. Indeed,

$$\begin{aligned}\|A - ab^T\|_F^2 &\leq \|\bar{A} - \bar{a}b^T\|_F^2 \\ &\leq \|\bar{A} - \bar{A}_1\|_F^2 + \epsilon\|\bar{A}\|_F^2 && \text{by Equation (5.4)} \\ &\leq \tfrac{1}{4}nr + (4+C)\beta n. && \text{by Corollary 5.3.9}\end{aligned}$$

Furthermore $\|A\|_F^2 \leq 4nr$ since each entry of A has absolute value at most 2. Finally, by approximate triangle inequality (Claim 5.2.3),

$$\|ab^T\|_F^2 \leq 2\|A\|_F^2 + 2\|A - ab^T\|_F^2 = O(nr).$$

With Equation (5.8) this implies the lemma. □

We now show how to use $\bar{A}' = \bar{a}b^T$ to solve the majority instance S of the problem in Lemma 5.3.5. We condition on the intersection of the events Ψ_{typical} and Ψ_{spectral}. By Corollary 5.3.12 it occurs with probability at least $1 - 10\delta - o(1)$.

Let $s_1, \ldots, s_n \in \{1,2\}^r$ denote the random instances in S. Recall that we assigned them to rows of A by a uniformly random permutation π, that is, the $\pi(i)$-th row of A equals s_i.

We use a to solve the majority problem as follows. For each s_i, if $a_{\pi(i)} \le 1.5$ then we output that the majority is 1, and otherwise we output that the majority is 2. We say that a *solves* the instance s_i if the output is correct. Due to π being random, the probability that a solves any instance s_i is identical. Denote this probability by p. We need to show that $p \ge 2/3$.

Assume by contradiction that $p < 2/3$. By Markov's inequality, with probability at least $4/5$ we have at least $n/6$ unsolved instances. Since by Ψ_{typical} there are only $0.1n$ atypical instances, we have at least $n/15$ unsolved typical instances. Denote by S' the set of unsolved typical instances. Consider such instance $s_i \in S'$. Suppose its majority element is 1. Then, since it is typical,

$$\begin{aligned} \|s_i - \mu_i \mathbf{1}\|_2^2 &\le \|s_i - (1.5 - \gamma/\sqrt{r})\mathbf{1}\|_2^2 \\ &\le \left(\tfrac{1}{2}r + \gamma\sqrt{r}\right)\left(\tfrac{1}{2} - \gamma/\sqrt{r}\right)^2 + \left(\tfrac{1}{2}r - \gamma\sqrt{r}\right)\left(\tfrac{1}{2} + \gamma/\sqrt{r}\right)^2 \\ &= \tfrac{1}{4}r - \gamma^2. \end{aligned} \tag{5.9}$$

On the other hand, since s_i is unsolved, $a_{\pi(i)} \ge 1.5$. Hence by Lemma 5.3.13,

$$\frac{\|b^T P_{\mathbf{1}}\|_2}{\|\mathbf{1}\|_2} \cdot a_{\pi(i)} \ge 1.5 - \eta/\sqrt{r}.$$

Therefore, noting that $b^T P_{\mathbf{1}} = \frac{\|b^T P_{\mathbf{1}}\|_2}{\|\mathbf{1}\|_2} \cdot \mathbf{1}$, we have

$$\begin{aligned}
\|s_i - a_{\pi(i)} b^T P_{\mathbf{1}}\|_2^2 &= \|s_i - \frac{\|b^T P_{\mathbf{1}}\|_2}{\|\mathbf{1}\|_2} \cdot a_{\pi(i)} \mathbf{1}\|_2^2 \\
&\geq \|s_i - (1.5 - \tfrac{\eta}{\sqrt{r}}) \cdot \mathbf{1}\|_2^2 \\
&\geq \left(\tfrac{1}{2}r + \gamma\sqrt{r}\right)\left(\tfrac{1}{2} - \eta/\sqrt{r}\right)^2 + \left(\tfrac{1}{2}r - \gamma\sqrt{r}\right)\left(\tfrac{1}{2} + \eta/\sqrt{r}\right)^2 \\
&= \tfrac{1}{4}r + \eta^2 - 2\gamma\eta. \qquad (5.10)
\end{aligned}$$

Similar calculations yield the same bounds when the majority element is 2. From Equation (5.9), together with Equation (5.4) and Lemma 5.3.8, we get:

$$\|\bar{A} - \bar{a}b^T\|_F^2 \leq \sum_{i=1}^{n} \|s_i - \mu_i \mathbf{1}\|_2^2 \leq \tfrac{1}{15} n (\tfrac{1}{4} r - \gamma^2) + \sum_{s_i \notin S'} \|s_i - \mu_i \mathbf{1}\|_2^2 + (4 + C)\beta n. \qquad (5.11)$$

On the other hand, by Equation (5.10),

$$\|A - ab^T P_{\mathbf{1}}\|_F^2 = \sum_{i=1}^{n} \|s_i - a_{\pi(i)} b^T P_{\mathbf{1}}\|_2^2 \geq \tfrac{1}{15} n (\tfrac{1}{4} r + \eta^2 - 2\gamma\eta) + \sum_{s_i \notin S'} \|s_i - \mu_i \mathbf{1}\|_2^2. \qquad (5.12)$$

It remains to relate Equations (5.11) and (5.12) to derive a contradiction. By the Pythagorean identity, $\|A - ab^T\|_F^2 = \|AP_{\mathbf{1}} - ab^T P_{\mathbf{1}}\|_F^2 + \|AP_{\mathbf{1}}^{\perp} - ab^T P_{\mathbf{1}}^{\perp}\|_F^2$, and

$$\|A - ab^T P_{\mathbf{1}}\|_F^2 = \|AP_{\mathbf{1}} - ab^T P_{\mathbf{1}}^2\|_F^2 + \|AP_{\mathbf{1}}^{\perp} - ab^T P_{\mathbf{1}} P_{\mathbf{1}}^{\perp}\|_F^2 = \|AP_{\mathbf{1}} - ab^T P_{\mathbf{1}}\|_F^2 + \|AP_{\mathbf{1}}^{\perp}\|_F^2.$$

Together,

$$\|A - ab^T P_{\mathbf{1}}\|_F^2 = \|A - ab^T\|_F^2 + \left(\|AP_{\mathbf{1}}^{\perp}\|_F^2 - \|AP_{\mathbf{1}}^{\perp} - ab^T P_{\mathbf{1}}^{\perp}\|_F^2\right). \qquad (5.13)$$

Let us upper-bound both terms in the right-hand side of Equation (5.13). For the first term, we simply use $\|A - ab^T\|_F^2 \leq \|\bar{A} - \bar{a}b^T\|_F^2$. For the second term, note that $\|b^T P_{\mathbf{1}}^{\perp}\|_2 = \|\mathbf{1} P_{\mathbf{1}}^{\perp} - b^T P_{\mathbf{1}}^{\perp}\|_2 \leq \|\mathbf{1} - b\|_2$. Together with Lemma 5.3.14, $\|ab^T P_{\mathbf{1}}^{\perp}\|_2 \leq O(\sqrt{n}) \cdot \|1 - b\|_2$. Thus by Lemma 5.3.11,

$$\left(\|AP_{\mathbf{1}}^{\perp}\|_F^2 - \|AP_{\mathbf{1}}^{\perp} - ab^T P_{\mathbf{1}}^{\perp}\|_F^2\right) \leq O(n) \cdot \|\mathbf{1} - b\|_2.$$

By Equation (5.5), the latter is $O(n)/\sqrt{C}$. Plugging both upper bounds into Equation (5.13),

$$\|A - ab^T P_{\mathbf{1}}\|_F^2 \leq \|\bar{A} - \bar{a}b^T\|_F^2 + O(n) \cdot C^{-1/2}.$$

This relates Equations (5.11) and (5.12), yielding

$$\tfrac{1}{15}(\gamma^2 + \eta^2) \leq \tfrac{2}{15} \cdot \gamma\eta + (4 + C)\beta + O(1) \cdot C^{-1/2}.$$

Since γ is fixed, choosing β, η sufficiently small and C sufficiently large leads to a contradiction.

Thus $p \geq 2/3$, meaning the reduction solves each instance in the majority problem S with probability at least $2/3$. Accounting for the conditioning on Ψ_{typical} and Ψ_{spectral}, the determined low-rank approximation algorithm from Section 5.3.5 solves a random instance of Lemma 5.3.5 with probability at least $2/3 - 10\delta - o(1)$ (the constants can be scaled without changing the lower bound). Hence, it requires reading at least $\Omega(n/\epsilon)$ bits from the matrix, which proves Theorem 5.3.1.

5.4 Lower Bound for General Rank-k Approximation

In this section we prove the full statement of Theorem 5.1.3. The proof largely goes by reduction to the $k = 1$ case, described as follows. We take k copies (referred to as *blocks*) of the hard distribution from the $k = 1$ case, and concatenate them horizontally into an $n \times (kr)$ matrix (where as previously, $r = \Theta(1/\epsilon)$). Then, for each block we pick a Hadamard vector, and add it to all columns in that block. This renders the blocks nearly orthogonal, forcing any low-rank approximation algorithm to compute the majority element of most rows in most blocks, thus solving $\Omega(nk/\epsilon)$ instances of random majority, yielding the desired lower bound. Even though the description is straightforward, the formal proof requires some elaborate technical work, as given in the rest of this section.

Another rather minor difference is that due to having k blocks (which correspond to k clusters of points in the metric space), we cannot add a "heavy row" (which would correspond to a very far point), with all entries set to a large $M > 0$, to each block as we did in the

$k = 1$ case. The reason is that the clusters are close (the distance between every two clusters is at most 2), so any point which is far from one cluster must be far from all of them. Thus it would sharpen the spectrum separation of the entire matrix, but not of each block separately, which is the effect we wish to achieve (namely, it would increase the top singular value, but not all top-k singular values.). This is solved by adding M^2 light rows instead of a single heavy row. In a light row, we can set all distances to a given cluster $i \in [k]$ to 2, and the rest of the distances to 1. This makes the corresponding point slightly further from cluster i than from the rest of the clusters. Over many similar light rows, this yields the desired effect.

5.4.1 Hard Distribution

Given n, k, ϵ, let $\beta, C > 0$ be constants that will be chosen later. (β will be sufficiently small and C sufficiently large.) Let $r = \beta/\epsilon$, and assume w.l.o.g. this is an integer by letting ϵ be sufficiently smaller. Let $N = (1 + C)n$.

Next we use the Walsh construction of Hadamard vectors. Recall these are vectors with entries in $\{\pm 1\}^N$ which are pairwise orthogonal. Let $v_1, \ldots, v_k \in \mathbb{R}^N$ be k Hadamard column vectors, which are different than all-1's. We rescale them to have entries in $\{\pm\frac{1}{2}\}$. Let $V^i \in \mathbb{R}^{N\times r}$ be made of r copies of v^i concatenated horizontally.

For every $i = 1, \ldots, k$ let $S^i \in \{\pm\frac{1}{2}\}^{n\times r}$ be made of n vertically stacked random instances of majority, after a random permutation of the rows. We complete it to a matrix $\bar{S}^i \in \{0, \pm\frac{1}{2}\}^{N\times r}$ by adding all-0's lines at the bottom. We use J to denote the all-1's matrix of dimensions implied by context. We form a matrix $\bar{A}^i \in \mathbb{R}^{N\times r}$ by

$$\bar{A}^i = 2J + V^i + \bar{S}^i.$$

We concatenate the $\bar{A}^i$'s horizontally to obtain a matrix $\bar{A} \in \mathbb{R}^{N\times kr}$. This defines the hard distribution over distance matrices $\bar{A} \in \mathbb{R}^{N\times kr}$.

Claim 5.4.1. *Every supported $\bar{A}$ is an (asymmetric) distance matrix.*

Proof. Observe that all entries of $\bar{A}$ are in $\{1, 1\frac{1}{2}, 2, 2\frac{1}{2}, 3\} \subset [1, 3]$. Let $\mathcal{X}$ and $\mathcal{Y}$ be disjoint symbolic point sets, which correspond to the rows and columns of $\bar{A}$ respectively. For every

$x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the distance $\mathrm{d}(x,y)$ is determined by $\bar{A}$. For every $x, x' \in \mathcal{X}$ we set the distance to $\mathrm{d}(x,x') = 2$, and for every $y, y' \in \mathcal{Y}$ we set the distance to $\mathrm{d}(y,y') = 2$. It can be easily checked that the triangle inequality is satisfied for the metric space $(\mathcal{X} \cup \mathcal{Y}, \mathrm{d})$. $\square$

Let $\bar{B} = \bar{A} - 2J$ and $\bar{B}^i = V^i + \bar{S}^i$ (not the $\bar{B}$ is the horizontal concatenation of the $\bar{B}_i$'s). Moreover, let $B \in \mathbb{R}^{n \times kr}$ denote the restriction of $\bar{B}$ to its top n rows. In most of the proof we will actually work with the matrix $\bar{B}$ instead of $\bar{A}$ (cf. Lemma 5.4.6 later on).

For every $i \in [k]$ and $j \in [n]$ let us denote by s^i_j the majority instance which is at the jth line of S^i. Let μ^i_j denote its mean. As in the $k = 1$ case, let $\mathbf{1}$ denote the all-1's vector in $\mathbb{R}^r$, and let $P_{\mathbf{1}}$ denote the orthogonal projection on the subspace spanned by it.

Lemma 5.4.2. *$\|\bar{B} - \bar{B}_k\|_F^2 + \epsilon \|\bar{B}\|_F^2 \le \sum_{i=1}^k \sum_{j=1}^n \|s^i_j - \mu^i_j \mathbf{1}\|_2^2 + (1+C)\beta nk$.*

Proof. For the first summand, consider the rank-k matrix given by replacing each majority instance s^i_j in B by $\mu^i_j \mathbf{1}$. For the second summand, note that each entry in $\bar{B}$ has magnitude at most 1, thus $\epsilon \|\bar{B}\|_F^2 \le \epsilon \cdot (1+C)n \cdot kr = (1+C)\beta nk$. $\square$

Corollary 5.4.3. *$\|\bar{B} - \bar{B}_k\|_F^2 + \epsilon \|\bar{B}\|_F^2 \le \frac{1}{4} nkr + (1+C)\beta kn$.*

Proof. In the term $\sum_{i=1}^n \|s_i - \mu_i \mathbf{1}\|_2^2$ in the above lemma, if we replace μ_i by an all-0 vector (which does not decrease the term, since the means μ_i are optimal for it), we pay exactly $(\frac{1}{2})^2$ per entry. $\square$

5.4.2 Spectral Properties

Lemma 5.4.4. *Suppose $(kr)^3 = O(n)$. Let $\sigma_j(\bar{B})_{j=1}^{rk}$ denote the sorted singular values of $\bar{B}$ (so that $\sigma_1(\bar{B}) \ge \sigma_2(\bar{B}) \ge \ldots$). With probability at least $1 - e^{-\Omega(n/(kr)^{3/2}}$,*

1. *For every $j \ge k+1$, $\sigma_j(\bar{B})^2 \le (\frac{1}{4} + o(1))n$.*
2. *For every $j \le rk - k$, $\sigma_j(\bar{B})^2 \ge (\frac{1}{4} - o(1))n$.*

Proof. Consider the random portion of $\bar{B}$, which is the horizontal concatenations the blocks S^i. This is a matrix of order $n \times rk$ with entries distributed uniformly i.i.d. in $\{\pm\frac{1}{2}\}$. Thus, similarly to Lemma 5.3.10, all of its squared singular values are $(\frac{1}{4} \pm o(1))n$ with high probability. Since $\bar{B}$ is obtained by adding k rank-1 matrices to that random portion, both parts of the lemma follow from Fact 5.3.4. □

Lemma 5.4.5. *Suppose $(kr)^3 = O(n)$. With probability at least $1 - e^{-\Omega(n/(kr)^{3/2}}$, for every $k' \leq k$, the sum of squares of the top k' singular values of $\bar{B}$ is at most $(1 + o(1)) \cdot \frac{1}{4}((C + 1)nk'r + 2nk)$.*

Proof. Fix a block i. Its corresponding Hadamard vector is v^i. Denote by $\bar{e}_i$ its corresponding block-indicator vector (this is a column vector in $\mathbb{R}^{kr}$ which has 1 in entries corresponding to block i and 0 everywhere else). Note that $\|v^i\| = \sqrt{\frac{1}{4}(C+1)n}$ and $\|\bar{e}_i\| = \sqrt{r}$.

Consider the multiplication of $\bar{B}$ on the left by v^i and on the right by $\bar{e}_i$. This yields a scalar value ϕ_i which we now calculate. It equals r times the inner product of v^i with itself, which is $\frac{1}{4}(C+1)n$, plus r times the inner product of the first n entries of v^i with a column of S^i, which has entries i.i.d. uniformly random in $\{\pm\frac{1}{2}\}$. Each summand in this latter inner product is i.i.d. uniformly random from $\{\pm\frac{1}{4}\}$, and thus with high probability, the inner product is between $-O(\sqrt{n})$ and $O(\sqrt{n})$. Thus, in total, $\phi_i \geq (1 - o(1)) \cdot \frac{1}{4}(C+1)n$. This implies that $\bar{B}$ has a singular value which is at least

$$\frac{(1 - o(1)) \cdot \frac{1}{4}(C+1)nr}{\|v\| \cdot \|\bar{e}_i\|} = \sqrt{(1 - o(1)) \cdot \tfrac{1}{4}(C+1)nr}.$$

Since we have k blocks, and their Hadamard vectors $\{v_i\}$ are orthogonal, and their block indicators $\{\bar{e}_i\}$ are orthogonal, this implies we have at least k singular values whose square is at least $(1-o(1)) \cdot \frac{1}{4}(C+1)nr$. Consequently, if we let $\{\sigma_j(\bar{B})\}_{j=1}^{kr}$ denote the sorted singular values of $\bar{B}$, then

$$\sum_{j=k'+1}^{k} \sigma_j(\bar{B})^2 \geq (k - k') \cdot (1 - o(1)) \cdot \tfrac{1}{4}(C+1)nr.$$

By Lemma 5.4.4, $\bar{B}$ has at least $rk - 2k$ additional singular values whose squared value is at least $(1-o(1))\cdot\frac{1}{4}n$. Together,

$$\begin{aligned}\sum_{j=1}^{k}\sigma_j(\bar{B})^2 &= \sum_{j=1}^{k'}\sigma_j(\bar{B})^2+\sum_{j=k'+1}^{k}\sigma_j(\bar{B})^2+\sum_{j=k+1}^{rk}\sigma_j(\bar{B})^2\\ &\geq \sum_{j=1}^{k'}\sigma_j(\bar{B})^2+(k-k')\cdot(1-o(1))\cdot\tfrac{1}{4}(C+1)nr+(rk-2k)\cdot(1-o(1))\cdot\tfrac{1}{4}n.\end{aligned}$$

On the other hand, consider the squared Frobenius norm of $\bar{B}$. In the top n rows, the absolute value of each entry is i.i.d. uniform in $\{0,1\}$ (since it is the sum of a fixed entry from the Hadamard vector, either $\frac{1}{2}$ or $-\frac{1}{2}$, with a uniformly random value in $\{\pm\frac{1}{2}\}$, so their total contribution is tightly concentrated at $\frac{1}{2}nkr$. In the bottom Cn rows each entry has absolute value of $\frac{1}{2}$ (an entry of a fixed Hadamard vector) so their total contribution is $\frac{1}{4}Cnkr$. Together,

$$\sum_{j=1}^{k}\sigma_j(\bar{B})^2=\|\bar{B}\|_F^2\leq\tfrac{1}{4}Cnkr+(1+o(1))\cdot\tfrac{1}{2}nkr.$$

Rearranging this with the above, $\sum_{j=1}^{k'}\sigma_j(\bar{B})^2\leq(1+o(1))\cdot\frac{1}{4}((C+1)nk'r+2nk)$ as needed. □

5.4.3 The Reduction

Suppose we have a deterministic algorithm that given $\bar{A}$, returns $\bar{A}'$ of rank $k+1$ that with probability at least $(2/3)+\delta$ satisfies

$$\|\bar{A}-\bar{A}'\|_F^2\leq\|\bar{A}-\bar{A}_{k+1}\|_F^2+\epsilon\|\bar{A}\|_F^2. \tag{5.14}$$

This the hypothesis of Theorem 5.1.3, except with $k+1$ instead of k; this will be more convenient to work with, and does not change the theorem statement (one can shift k by 1 everywhere).

We now move from working with $\bar{A}$ to working with $\bar{B}$.

Lemma 5.4.6. *We can obtain an approximation $\bar{B}'$ of rank $k+2$ that satifies*

$$\|\bar{B}-\bar{B}'\|_F^2\leq\|\bar{B}-\bar{B}_k\|_F^2+O(\epsilon)\cdot\|\bar{B}\|_F^2. \tag{5.15}$$

Proof. Recall that $\bar{B} = \bar{A} - 2J$. We take $\bar{B}' = \bar{A}' - 2J$. Then, for a constant $c > 0$,

$$\begin{aligned}
\|\bar{B} - \bar{B}'\|_F^2 &= \|(\bar{A} - 2J) - (\bar{A}' - 2J)\|_F^2 \\
&= \|\bar{A} - \bar{A}'\|_F^2 \\
&\le \|\bar{A} - \bar{A}_{k+1}\|_F^2 + \epsilon\|\bar{A}\|_F^2 && \text{Equation (5.14)} \\
&\le \|\bar{A} - \bar{B}_k - 2J\|_F^2 + \epsilon\|\bar{A}\|_F^2 && (*) \\
&\le \|\bar{B} - \bar{B}_k\|_F^2 + c\epsilon\|\bar{B}\|_F^2, && (**)
\end{aligned}$$

where $(*)$ is since $\bar{B}_k + 2J$ has rank $k+1$ and $\bar{A}_{k+1}$ is the optimal rank-$(k+1)$ approximation of $\bar{A}$, and $(**)$ is since $\|\bar{A}\|_F^2$ and $\|\bar{B}\|_F^2$ are equal up to a constant (both are $\Theta(nkr)$). The lemma follows since we can scale ϵ down by the constant c. □

Combined with Corollary 5.4.3, we get

Corollary 5.4.7. $\|\bar{B} - \bar{B}'\|_F^2 \le \frac{1}{4}nkr + (1 + C)\beta kn$.

Recall that $\bar{B}'_k$ denotes the optimal rank-k approximation of $\bar{B}'$, and thus $\|\bar{B}'_k\|_F^2$ is the sum of squares of the top-k singular values of $\bar{B}'$ (which has a total of $k+2$ singular values).

Lemma 5.4.8. *The singular values of $\bar{B}'$ satisfy*

$$\sum_{i=1}^{k+2} \Big(\sigma_i(\bar{B}) - \sigma_i(\bar{B}')\Big)^2 \le O(C\beta nk).$$

Furthermore, the two bottom squared singular values are each $O(C\beta nk)$.

Proof. Using Lemma 5.4.6 as an upper bound and the Hoffman-Weilandt inequality as a lower bound on $\|\bar{B} - \bar{B}'\|_F^2$,

$$\sum_{i=1}^{k+2} \Big(\sigma_i(\bar{B}) - \sigma_i(\bar{B}')\Big)^2 + \|\bar{B} - \bar{B}_{k+2}\|_F^2 \le \|\bar{B} - \bar{B}'\|_F^2 \le \|\bar{B} - \bar{B}_k\|_F^2 + O(C\beta nk).$$

Observe that $\|\bar{B} - \bar{B}_k\|_F^2 - \|\bar{B} - \bar{B}_{k+2}\|_F^2 = \sigma_{k+1}(\bar{B})^2 + \sigma_{k+2}(\bar{B})^2$, and by Lemma 5.4.4 each of these two summands is $O(n)$, which is less than $O(C\beta nk)$.[4] Plugging this above yields

[4]Note that β and C are constants that will eventually be chosen such that $C\beta$ is smaller than a sufficiently small constant. It holds that $n = O(C\beta nk)$ if k is larger than a sufficiently large constant, which we can assume w.l.o.g. since we have already proven the $k = 1$ case.

the desired inequality, $\sum_{i=1}^{k+2}\left(\sigma_i(\bar{B})-\sigma_i(\bar{B}')\right)^2 \le O(C\beta nk)$.

As for the bottom two singular values of $\bar{B}'$, the inequality just proven yields in particular $\left(\sigma_{k+1}(\bar{B})-\sigma_{k+1}(\bar{B}')\right)^2 \le O(C\beta nk)$, hence $\sigma_{k+1}(\bar{B}') \le \sigma_{k+1}(\bar{B}) + O(\sqrt{C\beta nk})$. As already mentioned above, $\sigma_{k+1}(\bar{B}) = O(\sqrt{n}) = O(\sqrt{C\beta nk})$ by Lemma 5.4.4. Thus $\sigma_{k+1}(\bar{B}')^2 \le O(C\beta nk)$. The same holds for $\sigma_{k+2}(\bar{B}')$. □

5.4.4 Averaging Columns in Blocks

We now carry out the main part of the reduction to the rank-1 case. We do this by showing that $\bar{B}$ can be approximated by the matrix resulting from taking $\bar{B}'$ and replacing each column in each block by the average of columns in that block. Note that in the resulting matrix, each block has rank-1 since its columns are identical. Therefore, by averaging, we could get a rank-1 approximation for a large constant fraction of the blocks. Let us now argue this formally.

Let $\Pi_{\mathbf{1}}$ be the orthogonal projection of $\mathbb{R}^{kr}$ on the subspace spanned by block indicators. Note that for a matrix $Z \in \mathbb{R}^{N\times kr}$, the operation $Z\Pi_{\mathbf{1}}$ averages the columns in each block. The main lemma for this part of this following.

Lemma 5.4.9. $\|\bar{B}-\bar{B}'\Pi_{\mathbf{1}}\|_F^2 \le \|\bar{B}-\bar{B}'\|_F^2 + O(\sqrt{C\beta}\cdot nk)$.

The proof will go by showing that the row space of $\bar{B}'$ has to be close to the span of the block indicators, which is the subspace on which $\Pi_{\mathbf{1}}$ projects. (This would yield $\bar{B}' \approx \bar{B}'\Pi_{\mathbf{1}}$ and hence $\|\bar{B}-\bar{B}'\|_F^2 \approx \|\bar{B}-\bar{B}'\Pi_{\mathbf{1}}\|_F^2$, as the lemma asserts). The way we show this is by transitivity, by showing that both subspaces are close to the top-k row space of $\bar{B}$. We will require the following technical linear algebraic claims, whose proofs are deferred to Section 5.4.6 for better readability.

Lemma 5.4.10. *Let $A, B \in \mathbb{R}^{n\times m}$. Let $B = U\Sigma V^T$ be the SVD of B. Suppose $\|A-B\|_F^2 \le \|A\|_F^2 - \Delta$. Then $\|AV\|_F^2 \ge \Delta$.*

Lemma 5.4.11. *Let $A \in \mathbb{R}^{n\times n}$ and let $A = U\Sigma V^T$ be its SVD. Let V_k be the restriction of V to the top-k right singular vectors of A. Let Π an orthogonal projection on some k-dimensional subspace. If*

$$(1-\epsilon)k \le \|V_k^T\Pi\|_F^2 \le k,$$

then

$$\|(A\Pi^{\perp})_k\|_F^2 \le \|A_{\epsilon k}\|_F^2 + \|(A_{n-k})_k\|_F^2.$$

(Recall that $\|X_k\|_F^2$ is the sum of squared top-k singular values for every matrix X.)

Remark. As a small digression, let us preview that we will use this lemma twice, on the matrices $\bar{B}$ and $\bar{B}'$. In both cases the projection would be $\Pi_{\mathbf{1}}$. In the former case the bound yielded by the lemma would be $O(nk)$, and it the latter case it would be (the better bound) $O(C\beta nk)$. That is, we will get $\|(\bar{B}\Pi^{\perp})_k\|_F^2 = O(nk)$ and $\|(\bar{B}'\Pi^{\perp})_k\|_F^2 = O(C\beta nk)$ (see Section 5.4.6 for an elaboration why). However we still need to establish the condition $(1-\epsilon)k \le \|V_k^T\Pi\|_F^2 \le k$ for both invocations, which we will do shortly.

Lemma 5.4.12. *Let $V, U, W \in \mathbb{R}^{n\times k}$ matrices such that each has orthonormal columns. Suppose $\|V^TU\|_F^2 \ge (1-\epsilon)k$ and $\|U^TW\|_F^2 \ge (1-\epsilon)k$. Then $\|V^TW\|_F^2 \ge (1-O(\epsilon))k$.*

We now prove Lemma 5.4.9. Let $\bar{B} = U\Sigma V^T$ denote the SVD of $\bar{B}$. Write it as $\bar{B} = U\Sigma V^T = U_T\Sigma_T V_T^T + U_B\Sigma_B V_B^T$ where Σ_T are the top k singular values and Σ_B are the remaining (bottom) singular values.

Lemma 5.4.13. *Let $\bar{B}^* \in \mathbb{R}^{N\times kr}$ be a rank-k' matrix such that $\|\bar{B}-\bar{B}^*\|_F^2 \le \frac{1}{4}nkr + O(nk)$. Let $W \in \mathbb{R}^{k'\times kr}$ be an orthonormal basis for the row span of $\bar{B}^*$. Then $\|WV_T\|_F^2 \ge k - O(\epsilon(k+k'))$.*

Proof. On one hand, since $\|\bar{B}\|_F^2 \ge \frac{1}{2}nkr - O(kn)$, the hypothesis $\|\bar{B}-\bar{B}^*\|_F^2 \le \frac{1}{4}nkr + O(kn)$ implies, by Lemma 5.4.10, $\|\bar{B}W^T\|_F^2 \ge \frac{1}{4}nkr - O(kn)$. On the other hand,

$$\begin{aligned}
\tfrac{1}{4}nkr - O(kn) \le \|\bar{B}W^T\|_F^2 &= \|U_T\Sigma_T V_T^T W^T + U_B\Sigma_B V_B^T W^T\|_F^2 \\
&= \|\Sigma_T V_T^T W^T\|_F^2 + \|\Sigma_B V_B^T W^T\|_F^2 \\
&\le \|\Sigma_T\|_2^2 \|V_T^T W^T\|_F^2 + \|\Sigma_B\|_2^2 \|V_B^T W^T\|_F^2 \\
&\le (\tfrac{1}{4}nr + O(n)) \cdot \|V_T^T W^T\|_F^2 + O(n) \cdot \|V_B^T W^T\|_F^2 \\
&\le (\tfrac{1}{4}nr + O(n)) \cdot \|V_T^T W^T\|_F^2 + O(n) \cdot k'.
\end{aligned}$$

The lemma follows by rearranging and recalling that $r = \Theta(1/\epsilon)$. □

We apply the above lemma twice: once with $\bar{B}^*$ being $\bar{B}'$ (whose rank is $k+2$), and once with $\bar{B}^*$ being the matrix obtained from averaging the columns in each block of $\bar{B}$ (note that this matrix has rank is k). Since $\Pi_{\mathbf{1}}$ is the orthogonal projection on the row space of that matrix, then by the latter application of Lemma 5.4.13 we have

$$k \geq \|V_T^T \Pi_{\mathbf{1}}\|_F^2 \geq k - O(\epsilon k), \tag{5.16}$$

which establishes the condition of Lemma 5.4.11, yielding

$$\|(\bar{B}\Pi_{\mathbf{1}}^{\perp})_k\|_F^2 \leq O(nk).$$

For the former application, let $\bar{B}' = U'\Sigma'(V')^T$ denote the SVD of $\bar{B}'$. Corollary 5.4.7 provides the requirement of Lemma 5.4.13, which in turn yields $\|(V')^T V_T\|_F^2 \geq k - O(\epsilon k)$. Together with Equation (5.16), by transitivity (Lemma 5.4.12),

$$k \geq \|(V')^T \Pi_{\mathbf{1}}\|_F^2 \geq k - O(\epsilon k),$$

which establishes the condition of Lemma 5.4.11, yielding

$$\|(\bar{B}'\Pi_{\mathbf{1}}^{\perp})_k\|_F^2 \leq O(C\beta nk).$$

Together with the above,

$$\|(\bar{B}\Pi_{\mathbf{1}}^{\perp})_k\|_F \|(\bar{B}'\Pi_{\mathbf{1}}^{\perp})_k\|_F \leq O(\sqrt{C\beta} \cdot nk). \tag{5.17}$$

We can extend this from rank-k to rank-$(k+2)$ since the additional two square singular value of each of the matrices is $O(C\beta nk)$ (cf. Lemmas 5.4.4 and 5.4.8).[5] Since $\bar{B}'\Pi_{\mathbf{1}}^{\perp}$ has

[5]Recall again that we set $C\beta < 1$.

rank $k+2$, then by Hoffman-Weilandt,

$$\begin{aligned}
\|\bar{B}\Pi_{\mathbf{1}}^{\perp} - \bar{B}'\Pi_{\mathbf{1}}^{\perp}\|_F^2 &\geq \sum_i \left(\sigma_i(\bar{B}\Pi_{\mathbf{1}}^{\perp}) - \sigma_i(\bar{B}'\Pi_{\mathbf{1}}^{\perp})\right)^2 && \text{by Hoffman-Weilandt} \\
&= \|\bar{B}\Pi_{\mathbf{1}}^{\perp}\|_F^2 - \sum_{i=1}^{k+2} \sigma_i(\bar{B}\Pi_{\mathbf{1}}^{\perp})\sigma_i(\bar{B}'\Pi_{\mathbf{1}}^{\perp}) + \|\bar{B}'\Pi_{\mathbf{1}}^{\perp}\|_F^2 && \operatorname{rank}(\bar{B}'\Pi_{\mathbf{1}}^{\perp}) = k+2 \\
&\geq \|\bar{B}\Pi_{\mathbf{1}}^{\perp}\|_F^2 - \sum_{i=1}^{k+2} \sigma_i(\bar{B}\Pi_{\mathbf{1}}^{\perp})\sigma_i(\bar{B}'\Pi_{\mathbf{1}}^{\perp}) \\
&\geq \|\bar{B}\Pi_{\mathbf{1}}^{\perp}\|_F^2 - \|(\bar{B}\Pi_{\mathbf{1}}^{\perp})_{k+2}\|_F \|(\bar{B}'\Pi_{\mathbf{1}}^{\perp})_{k+2}\|_F && \text{by Cauchy-Schwartz} \\
&\geq \|\bar{B}\Pi_{\mathbf{1}}^{\perp}\|_F^2 - O(\sqrt{C\beta} \cdot nk) && \text{by Equation (5.17) .}
\end{aligned}$$

Finally, by Pythagorean identities,

$$\begin{aligned}
\|\bar{B} - \bar{B}'\Pi_{\mathbf{1}}\|_F^2 &= \|\bar{B}\Pi_{\mathbf{1}} - \bar{B}'\Pi_{\mathbf{1}}\|_F^2 + \|\bar{B}\Pi_{\mathbf{1}}^{\perp}\|_F^2 \\
&= \|\bar{B} - \bar{B}'\|_F^2 + \left(\|\bar{B}\Pi_{\mathbf{1}}^{\perp}\|_F^2 - \|\bar{B}\Pi_{\mathbf{1}}^{\perp} - \bar{B}'\Pi_{\mathbf{1}}^{\perp}\|_F^2\right) \\
&\leq \|\bar{B} - \bar{B}'\|_F^2 + O(\sqrt{C\beta} \cdot nk).
\end{aligned}$$

This proves Lemma 5.4.9.

Relevant Blocks

Lemma 5.4.14. *There is a subset $I \subset [k]$ of size at least $|I| \geq 0.99k$ such that for every $i \in I$,*

$$\|\bar{B}^i - (\bar{B}')^i P_{\mathbf{1}}\|_F^2 \leq \|\bar{B}^i = (\bar{B}^i)_1\|_F^2 + O(\sqrt{C\beta} \cdot n), \tag{5.18}$$

where $(\bar{B}^i)_1$ is (as usual) the optimal rank-1 approximation of $\bar{B}^i$. We refer to blocks B_i with $i \in I$ as relevant blocks*.*

Proof. By Lemma 5.4.9, $\|\bar{B} - \bar{B}'\Pi_{\mathbf{1}}\|_F^2 \leq \|\bar{B} - \bar{B}'\|_F^2 + O(\sqrt{C\beta} \cdot nk)$. Note that the left-hand side equals $\sum_{i=1}^k \|\bar{B}^i - (\bar{B}')^i P_{\mathbf{1}}\|_F^2$. As for the right-hand side, by Equation (5.15) we have $\|\bar{B} - \bar{B}'\|_F^2 \leq \|\bar{B} - \bar{B}_k\|_F^2 + O(\epsilon) \cdot \|B\|_F^2$, and we recall that $\|B\|_F^2 = O(Cnkr) = O(C\beta nk/\epsilon)$. Furthermore, $\|\bar{B} - \bar{B}_k\|_F^2 \leq \sum_{i=1}^k \|\bar{B}^i - (\bar{B}^i)_1\|_F^2$. Putting it all together yields

$\sum_{i=1}^{k} \|\bar{B}^i - (\bar{B}')^i P_{\mathbf{1}}\|_F^2 \le \sum_{i=1}^{k} \|\bar{B}^i - (\bar{B}^i)_1\|_F^2 + O(\sqrt{C\beta} \cdot nk)$, or rearranging,

$$\sum_{i=1}^{k} \left(\|\bar{B}^i - (\bar{B}')^i P_{\mathbf{1}}\|_F^2 - \|\bar{B}^i - (\bar{B}^i)_1\|_F^2 \right) \le O(\sqrt{C\beta} \cdot nk).$$

Each term in the sum on the left-hand side is non-negative, by the optimality of $(\bar{B}^i)_1$ for rank-1 approximation of $\bar{B}^i$. Therefore we can use an averaging argument (Markov's inequality) and conclude that at least $0.99k$ of the k summands on the left-hand side are at most $100/k$ times the right-hand side. The lemma follows. □

Fix $i \in I$. Since $(\bar{B}')^i P_{\mathbf{1}}$ is a rank-1 matrix we can write it as $\bar{a}^i (b^i)^T$ where $\bar{a}^i \in \mathbb{R}^N$ and $b^i \in \mathbb{R}^r$. Let $a^i \in \mathbb{R}^n$ denote the restriction of $\bar{a}^i$ to the first n entries. Consider $\|\bar{B}^i - \bar{a}^i (b^i)^T\|_F^2$. Note that the last Cn rows of are either all 1 or all -1, depending on the Hadamard vector v^i. Let $\sigma_j^i \in \{\pm 1\}$ denote the sign of row j. Then the contribution of the last Cn rows is $\sum_{j=n+1}^{Cn} \|\sigma_j^i \mathbf{1} - a_j^i b^i\|_2^2$ which can be rewritten as $\sum_{j=n+1}^{Cn} \|\mathbf{1} - \sigma_j^i a_j^i b^i\|_2^2$. Pick the j that minimizes the term $\|\mathbf{1} - \sigma_j^i a_j^i b^i\|_2^2$ and set all entries $a_{n+1}^i, \ldots, a_{Cn}^i$ to a_j^i with the appropriate sign, to obtain a vector $\hat{a}^i$. By choice of j we have

$$\|\bar{B}^i - \hat{a}^i (b^i)^T\|_F^2 \le \|\bar{B}^i - \bar{a}^i (b^i)^T\|_F^2. \tag{5.19}$$

Furthermore,

$$\|\bar{B}^i - \hat{a}^i (b^i)^T\|_F^2 = \|B^i - a^i (b^i)^T\|_F^2 + Cn\|b^i - \mathbf{1}\|_2^2. \tag{5.20}$$

Combining Equations (5.19) and (5.20),

$$\|B^i - a^i (b^i)^T\|_F^2 + Cn\|b^i - \mathbf{1}\|_2^2 \le \|\bar{B}^i - \bar{a}^i (b^i)^T\|_F^2. \tag{5.21}$$

If we use Lemma 5.4.14 as an upper bound on $\|\bar{B}^i - \bar{a}^i (b^i)^T\|_F^2$ and Lemma 5.3.10 as a lower bound on $\|B^i - a^i (b^i)^T\|_F^2$, we get $Cn\|\mathbf{1} - b^i\|_2^2 \le O(n)$, which rearranges to

$$\|\mathbf{1} - b^i\|_2^2 \le \frac{O(1)}{C}. \tag{5.22}$$

This implies Lemmas 5.3.13 and 5.3.14 for every relevant block, by the same proofs as their original proofs.

5.4.5 Solving Majority

Recall we have a total of nk majority instances (each of length r) embedded in $\bar{B}$. Note by the construction of $\bar{B}$, each of them has alphabet either $\{0,1\}$ or $\{0,-1\}$, depending on the sign of the corresponding entry of the Hadamard vector v^i, where i the block in which the instance is embedded.

By Lemma 5.3.6 and Markov's inequality, at least $0.9nk$ of the instances are typical. For an instance with alphabet $\{0,1\}$, we solve it using $\bar{B}'$ by reporting that the majority element is 1 if the average over the corresponding entries in $\bar{B}'$ is larger than 0.5, and reporting 0 if it is smaller than 0.5. Instances with alphabet $\{0,-1\}$ are solved similarly with threshold -0.5. Note that the solution procedure compares the threshold to the mutual value of the corresponding entries of $\bar{B}'\Pi_{\mathbf{1}}$. If we are correct on an instance, we say it is *solved*, and otherwise *unsolved*. For relevant block $i \in I$, let S'_i denote the subset of majority instances which are both typical and unsolved. Let $S' = \cup_{i\in I} S'_i$ be the subset of all instances which are typical, unsolved, and embedded in a relevant block.

Our goal is to show that we solve each instance with probability at least $2/3$. Since the instances were placed in $\bar{B}$ by random permutation, every instance has the same probability p to be solved, thus we need to show $p \geq 2/3$. Suppose by contradiction that $p < 2/3$. Since at least $0.9nk$ instances are typical, and at least $0.99k$ blocks are relevant, then there is a fixed constant $\zeta > 0$ (this was $\frac{1}{15}$ in the $k = 1$ case) such that $|S'| \geq \zeta nk$.

For every $i \in I$ we have (by definition of relevant blocks),

$$\|\bar{B}^i - (\bar{B}'\Pi_{\mathbf{1}})^i\|_F^2 \leq \|\bar{B}^i - (\bar{B}^i)_1\|_F^2 + O(\sqrt{C\beta}\cdot n) \leq \sum_{j=1}^{n} \|s_j - \mu_j \mathbf{1}\|_2^2 + O(\sqrt{C\beta}\cdot n).$$

By bounding $\sum_{j=1}^{n} \|s_j - \mu_j \mathbf{1}\|_2^2$ in the same way as in the $k = 1$ case,

$$\|\bar{B}^i - (\bar{B}'\Pi_{\mathbf{1}})^i\|_F^2 \leq |S'_i|(\tfrac{1}{4}r - \gamma^2) + \sum_{s_j \notin S'_i} \|s_j - \mu_j \mathbf{1}\|_2^2 + O(\sqrt{C\beta}\cdot n). \tag{5.23}$$

Similarly, if we denote $(\bar{B}')^i P_{\mathbf{1}} = \bar{a}^i (b^i)^T$ (since this is a rank-1 matrix), then as in the $k = 1$

case,

$$\|B^i - (B'\Pi_{\mathbf{1}})^i\|_F^2 = \sum_{i=1}^{n} \|s_i - \bar{a}_j^i (b^i)^T\|_2^2 \geq |S_i'|(\tfrac{1}{4}r + \eta^2 - 2\gamma\eta) + \sum_{s_j \notin S_i'} \|s_j - \mu_j \mathbf{1}\|_2^2. \quad (5.24)$$

(We remark that the latter inequality relies on Lemmas 5.3.13 and 5.3.14, which were proven in the previous section for relevant blocks, based on eq. (5.22); as per Lemma 5.3.13, $\eta = \Theta(1/\sqrt{C})$.)

Together,

$$|S_i'|(\tfrac{1}{4}r + \eta^2 - 2\gamma\eta) \leq |S_i'|(\tfrac{1}{4}r - \gamma^2) + O(\sqrt{C\beta} \cdot n).$$

We sum this over all $i \in I$, and recall that $\sum_{i \in I} |S_i'| = |S'|$. This yields,

$$|S'|(\tfrac{1}{4}r + \eta^2 - 2\gamma\eta) \leq |S'|(\tfrac{1}{4}r - \gamma^2) + O(\sqrt{C\beta} \cdot kn),$$

which rearranges to $|S'|(\gamma - \eta)^2 \leq O(\sqrt{C\beta} \cdot nk)$. Recalling that $|S'| \geq \zeta nk$, we get $\zeta(\gamma-\eta)^2 \leq O(\sqrt{C\beta} \cdot)$. Since ζ and γ are fixed constant, we can take η and β to be sufficiently small, and arrive at the desired contradiction. $\square$

5.4.6 Deferred Proofs from Appendix 5.4.4

Lemma 5.4.15. *Let $A, B \in \mathbb{R}^{n \times m}$. Let $B = U\Sigma V^T$ be the SVD of B. Suppose $\|A - B\|_F^2 \leq \|A\|_F^2 - \Delta$. Then $\|AV\|_F^2 \geq \Delta$.*

Proof.

$$\begin{aligned}
\|AV\|_F^2 &= \|AVV^T\|_F^2 \\
&= \|A\|_F^2 - \|A(I - VV^T)\|_F^2 && \text{Pythagorean theorem} \\
&\geq \Delta + \|A - B\|_F^2 - \|A(I - VV^T)\|_F^2 \\
&= \Delta + \|AVV^T - BVV^T\|_F^2 + \|A(I - VV^T)\|_F^2 - \|A(I - VV^T)\|_F^2 && \text{Pythagorean theorem} \\
&\geq \Delta.
\end{aligned}$$

$\square$

Lemma 5.4.16. *Let $A \in \mathbb{R}^{n\times n}$ and let $A = U\Sigma V^T$ be its SVD. Let V_k be the restriction of V to the top-k right singular vectors of A. Let Π an orthogonal projection on some k-dimensional subspace. If*

$$(1-\epsilon)k \leq \|V_k^T \Pi\|_F^2 \leq k,$$

then

$$\|(A\Pi^\perp)_k\|_F^2 \leq \|A_{\epsilon k}\|_F^2 + \|(A_{n-k})_k\|_F^2.$$

(Recall again that $\|X_k\|_F^2$ denotes the sum of squared top-k singular values for every matrix X.)

Proof. We have $\|(A\Pi^\perp)_k\|_F^2 = \|(A_{n-k}\Pi^\perp + A_k\Pi^\perp)_k\|_F^2$. We can write $A\Pi^\perp$ as $A_{n-k}\Pi^\perp + A_k\Pi^\perp$, and for any vector x, $A_{n-k}\Pi^\perp x$ and $A_k\Pi^\perp x$ are orthogonal, and so $\|A\Pi^\perp x\|_2^2 = \|A_{n-k}\Pi^\perp x\|_2^2 + \|A_k\Pi^\perp x\|_2^2$. It follows that

$$\|(A\Pi^\perp)_k\|_F^2 \leq \|A_k\Pi^\perp\|_F^2 + \|(A_{n-k}\Pi^\perp)_k\|_F^2.$$

Note that $\|(A_{n-k}\Pi^\perp)_k\|_F^2 \leq \|(A_{n-k})_k\|_F^2$. Thus it remains to show $\|A_k\Pi^\perp\|_F^2 \leq \|A_{\epsilon k}\|_F^2$, or equivalently, by the Pythagorean theorem, $\|A_k\Pi\|_F^2 \geq \|A_k\|_F^2 - \|A_{\epsilon k}\|_F^2 = \sum_{i=\epsilon k+1}^k \sigma_i^2(A)$.

We have $\|A_k\Pi\|_F^2 = \|\Sigma_k V_k^T \Pi\|_F^2$ and we know $\|V_k^T\Pi\|_F^2 \geq k(1-\epsilon)$. Let R be an $n \times n$ rotation matrix that takes V_k^T to $[I_k\ 0]$, where here I_k is the identity matrix of order k and 0 is an $k \times (n-k)$ zero matrix. Replace Π with $R^T\Pi$. Then $\|A_k\Pi\|_F^2 = \|\Sigma_k(V_k^T R)(R^T\Pi)\|_F^2$ and $\|(V_k^T R)(R^T\Pi)\|_F^2 \geq k(1-\epsilon)$. Thus, we can assume w.l.o.g. that $V_k^T = [I_k\ 0]$, and so $R^T\Pi$ has the form $[\Phi\ 0]$, where Φ is $k \times k$.

Thus $\|A_k\Pi\|_F^2 = \|\Sigma_k\Phi\|_F^2$ subject to $\|\Phi\|_F^2 \geq k(1-\epsilon)$. Also each row of Φ has squared norm at most 1 since it is a submatrix of a rotation matrix. Consequently, since Σ_k is a diagonal matrix, $\|\Sigma_k\Phi\|_F^2$ is minimized when placing all mass of Φ on the bottom $k(1-\epsilon)$ rows, and in this case it is exactly $\sum_{i=\epsilon k+1}^k \sigma_i^2(A)$. □

We have applied this lemma in Appendix 5.4.4 to both $\bar{B}_k$ and $\bar{B}'$. Let us show the resulting upper bound $\|A_{\epsilon k}\|_F^2 + \|(A_{n-k})_k\|_F^2$ in each case.

For $\bar{B}_k$, by Lemma 5.4.5, the top $k' = \lceil \epsilon k \rceil$ squared singular values of $\bar{B}$ sum to $O(nk'r + 2nk)$. Since $r = O(1/\epsilon)$, this bound is $O(nk)$. By Lemma 5.4.4, the rest of the squared singular values are $\Theta(n)$, thus $\|(\bar{B}_{n-k})_k\|_F^2 = O(kn)$, and the total bound is $O(nk)$.

For $\bar{B}'$,

$$\begin{aligned}
\|\bar{B}'_{\epsilon k}\|_F^2 &= \sum_{i=1}^{\epsilon k} \sigma_i(\bar{B}')^2 \\
&= \sum_{i=1}^{\epsilon k} \left(\sigma_i(\bar{B}') - \sigma_i(\bar{B}) + \sigma_i(\bar{B})\right)^2 \\
&\le \sum_{i=1}^{\epsilon k} 2\left(\sigma_i(\bar{B})^2 + (\sigma_i(\bar{B}) - \sigma_i(\bar{B}'))^2\right) && \text{Similarly to Claim 5.2.3} \\
&= 2\|\bar{B}_{\epsilon k}\|_F^2 + 2\sum_{i=1}^{\epsilon k}(\sigma_i(\bar{B}) - \sigma_i(\bar{B}'))^2.
\end{aligned}$$

The first term was already upper bounded by $O(C\beta nk)$ above, and the second sum is upper bounded by $O(C\beta nk)$ by Lemma 5.4.8. The term $\|(\bar{B}'_{n-k})_k\|_F^2$ is $O(C\beta nk)$ since there are two remaining eigenvalues and each is $O(C\beta nk)$ by Lemma 5.4.8. The total bound is $O(C\beta nk)$.

Lemma 5.4.17. *Let $V, U, W \in \mathbb{R}^{n\times k}$ matrices such that each has orthonormal columns. Suppose $\|V^TU\|_F^2 \ge (1-\epsilon)k$ and $\|U^TW\|_F^2 \ge (1-\epsilon)k$. Then $\|V^TW\|_F^2 \ge (1-O(\epsilon))k$.*

Proof. Note we can replace V^T with V^TR and U with R^TU and W with R^TW, where R is an $n \times n$ rotation which takes U to the top k standard unit vectors. All norms in the premise and goal of the claim are preserved. So we can assume that $\|V_{top}\|_F^2 \ge k(1-\epsilon)$, $\|W_{top}\|_F^2 \ge k(1-\epsilon)$, and need to show $\|V^TW\|_F^2 \ge k - O(k\epsilon)$, where "top" means the top $k\times k$ submatrix with remaining rows replaced with 0s. Let $W = W_{top} + W_{rest}$ and $V = V_{top} + V_{rest}$.

Then,

$$\begin{aligned}
\|V^TW\|_F^2 &= \|V_{top}^T W_{top} + V_{rest}^T W_{rest}\|_F^2 \\
&\geq \|V_{top}^T W_{top}\|_F^2 - 2Tr(W_{rest}^T V_{rest} V_{top}^T W_{top}) && \text{(i)} \\
&= \|V_{top}^T W_{top}\|_F^2 - 2Tr(W_{top} W_{rest}^T V_{rest} V_{top}^T) && \text{(ii)} \\
&\geq \|V_{top}^T W_{top}\|_F^2 - 2\|W_{top} W_{rest}\|_F \|V_{rest} V_{top}^T\|_F && \text{(iii)} \\
&\geq \|V_{top}^T W_{top}\|_F^2 - 2\|W_{top}\|_2 \|W_{rest}\|_F \|V_{top}\|_2 \|V_{rest}\|_F && \text{(iv)} \\
&\geq \|V_{top}^T W_{top}\|_F^2 - 2\|W_{rest}\|_F \|V_{rest}\|_F && \text{(v)} \\
&= \|V_{top}^T W_{top}\|_F^2 - 2(1 - \|W_{top}\|_F^2)^{1/2}(1 - \|V_{top}\|_F^2)^{1/2} \\
&= \|V_{top}^T W_{top}\|_F^2 - 2(\epsilon k)^{1/2}(\epsilon k)^{1/2} \\
&= \|V_{top}^T W_{top}\|_F^2 - 2\epsilon k, && (*)
\end{aligned}$$

where,

- (i) is by expanding the square and dropping a non-negative term;
- (ii) is by cyclicity of trace;
- (iii) is since $Tr(AB) <= \|A\|_F \|B\|_F$;
- (iv) is by submultiplicativity of operator and Frobenius norm;
- (v) is since W and V are orthonormal so their operator norm is 1, and operator norm does not decrease by taking submatrices.

So we just need to lower bound $\|V_{top}^T W_{top}\|_F^2$, and we can drop the last $n-k$ rows of V_{top} and W_{top} since they are zeros. Next, we write V_{top} in its SVD, $V_{top} = A\Sigma B^T$. Then since $\|V_{top}\|_F^2 \geq k(1-\epsilon)$ and $\|V_{top}\|_2^2 \leq 1$ (since it is a submatrix of V), necessarily, there are at least $k(1-2\epsilon)$ singular values of squared value at least $1-2\epsilon$. Indeed, otherwise $\|V_{top}\|_F^2 \leq k(1-2\epsilon)(1-2\epsilon) + 2\epsilon k \cdot 1 < k(1-\epsilon)$ for ϵ less than a small enough constant. Let Σ_h be these singular values and Σ_l be the remaining ones. Then

$$\|V_{top}^T W_{top}\|_F^2 = \|\Sigma B^T W_{top}\|_F^2 \geq (1-2\epsilon)\|B_h^T W_{top}\|_F^2.$$

Now $\|W_{top}\|_F^2 \geq k(1-\epsilon)$, and B_h is a $k(1-2\epsilon)$-dimensional subspace of $\text{span}(e_1, ..., e_k)$ (the standard unit vectors), and so we can extend it with an orthonormal basis B' so that $\text{span}(B_h, B') = \text{span}(e_1, ..., e_k)$. Then by the Pythagorean theorem

$$k(1-\epsilon) \leq \|W_{top}\|_F^2 = \|B_h^T W_{top}\|_F^2 + \|B' W_{top}\|_F^2,$$

and since $\|W_{top}\|_2^2 \leq 1$, we have $\|B' W_{top}\|_F^2 \leq \|B'\|_F^2 \leq 2\epsilon k$. Consequently, $\|B_h^T W_{top}\|_F^2 \geq k(1-\epsilon) - 2\epsilon k = k - 3\epsilon k$. Hence, $\|V_{top}^T W_{top}\|_F^2 \geq (1-2\epsilon)k(1-3\epsilon) \geq k(1-O(\epsilon))$. Plugging into (*) gives us our desired $k(1-O(\epsilon))$ lower bound on $\|V^T W\|_F^2$. $\square$

5.5 Lower Bound for Symmetric Distance Matrices

The lower bounds in Sections 5.3 and 5.4 are proven for asymmetric distance matrices. In this section we show that they hold also for symmetric distance matrices, completing the proof of Theorem 5.1.3. The proof is by a reduction of the asymmetric case to the symmetric case.

5.5.1 General Rank k

We start by reducing rank-k approximation of asymmetric distance matrices to rank-$(2k+2)$ approximation of symmetric distance matrices. The lower bound in Section 5.4 defined a hard distribution over asymmetric distance matrices of order $N \times kr$, where $N = O(n)$ and $r = \Omega(1/\epsilon)$. For clarity let us use n for N (as they are only a constant apart, this does not change the result asymptotically). Furthermore, we suppose w.l.o.g. that kr is an integer divisor of n. This does lose generality since we can increase n by up to $O(n)$ without changing the result asymptotically, and we need to add at most kr in order to arrive at an integer multiple of kr. By hypothesis of Theorem 5.1.3, $kr \leq k/\epsilon \ll n$.

Let $B \in \mathbb{R}^{n \times kr}$ be an asymmetric distance matrix drawn from the hard distribution defined in Section 5.4. Recall that all of its entries are in $\{1, 2, 3\}$. We can scale them by half so they are all in the interval $[1, 2]$.

We construct a symmetric distance matrix $A \in \mathbb{R}^{2n \times 2n}$. It is partitioned into $n \times n$ blocks,

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}.$$

We set its entries as follows. Its main diagonal is all-zeros. A_{11} and A_{22} both have all off-diagonal entries set to 1. A_{21} consists of $n/(kr)$ copies of B, concatenated horizontally. A_{12} is determined symmetrically. Since all entries of A are in the interval $[1, 2]$, the triangle inequality is satisfied trivially and thus A is a distance matrix.

We will show here that any rank-$(2k + 2)$ approximation algorithm for A must read at least $\Omega(nk/\epsilon)$ of its entries. Let $\mathbf{0}$ and $\mathbf{1}$ denote the all-0's and all-1's vectors in $\mathbb{R}^n$, respectively. For any $x, y \in \mathbb{R}^n$ let $[x\ y]$ denote their concatenation into a vector in $\mathbb{R}^{2n}$. Let B_k be the optimal rank-k approximation of B. Write B_k as $B_k = UV^T$ where $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{kr \times k}$. Let $u_1, \ldots, u_k$ be the columns of U, and let $v_1, \ldots, v_k$ be the columns of V^T. For every $i \in [k]$, let $\bar{v}_i$ be the vector given by concatenating $n/(kr)$ copies of v_i. Consider the rank-$(2k+2)$ approximation of A given by the column vectors $\{[\mathbf{0}\ u_i] : i \in [k]\} \cup \{[\bar{v}_i\ \mathbf{0}] : i \in [k]\} \cup \{[\mathbf{1}\ \mathbf{0}], [\mathbf{0}\ \mathbf{1}]\}$. This means we aim to bound the error $\|A - A''\|_F^2$, where $A'' \in \mathbb{R}^{2n \times 2n}$ is a matrix of our choice whose columns are spanned by these $2k + 2$ column vectors. We partition A'' into $n \times n$ blocks:

$$A'' = \begin{bmatrix} A''_{11} & A''_{12} \\ A''_{21} & A''_{22} \end{bmatrix},$$

and define them as follows: in A''_{11} and A''_{22} all entries are set to 1; A''_{21} consists of $n/(kr)$ copies of B_k, concatenated horizontally; A''_{12} is the transpose of A''_{21}. It is not hard to see that the columns of A'' are indeed spanned by the $2k + 2$ column vectors specified above.

Let us bound the error $\|A - A''\|_F^2$. On A_{12} and A_{21}, which contain concatenated copies of B, the error on each copy of B is exactly $\|B - B_k\|_F^2$. On A_{11} and A_{11}, which contain 0's on the diagonal and 1's off the diagonal, the error is zero on the off-diagonal entries, and $2n$ in total over the diagonal entries. Consequently,

$$\|A - A_{2k+2}\|_F^2 \leq \|A - A''\|_F^2 \leq t \cdot \|B - B_k\|_F^2 + 2n,$$

where $t = 2n/(kr)$ is the number of copies of B embedded in A.

Suppose we have an algorithm $\mathcal{ALG}$ that given A, returns a rank-$(2k+2)$ matrix A' that satisfies Equation (5.1). Let $A'_1, \ldots, A'_t$ denote the restriction of A' to the blocks matching the copies of B embedded in A. Thus, $\|A - A'\|_F^2 \geq \sum_{i=1}^t \|B - A'_i\|_F^2$. Furthermore, since $\|A\|_F^2 = \Theta(n^2) = \Theta(tnkr)$ and $\|B\|_F^2 = \Theta(nkr)$, we have $\|A\|_F^2 = O(1) \cdot t \cdot \|B\|_F^2$. Putting everything into Equation (5.1),

$$\sum_{i=1}^{t} \|B - A'_i\|_F^2 \leq t \cdot \left(\|B - B_k\|_F^2 + O(n) + O(\epsilon) \cdot \|B\|_F^2 \right).$$

By averaging, for at least one $i \in [t]$ we have $\|B - A'_i\|_F^2 \leq \|B - B_k\|_F^2 + O(\epsilon) \cdot \|B\|_F^2$. By scaling ϵ by a constant, A'_i satisfies Equation (5.1) as a rank-k

solves the rank-k approximation problem for B. By the proof for the asymmetric case, this requires reading $\Omega(nk/\epsilon)$ entries of B. $\mathcal{ALG}$

5.5.2 Rank 1

The previous section proves hardness for rank-k approximation of symmetric distance matrices, for $k \geq 4$. For completeness let us also show hardness for the $k = 1$ case, by a somewhat more refined analysis of the reduction in that case.

To this end we slightly modify the construction of A from the previous section. We draw $B \in \mathbb{R}^{n \times r}$ from the hard distribution for asymmetric distance matrices in the $k = 1$ case (Section 5.3.3). $A \in \mathbb{R}^{2n \times 2n}$ is constructed as above, except that in A_{22}, we change the off-diagonal entries from 1 to $1.5^2 = 2.25$. To make sure A is a distance matrix, we can scale it by $8/9$; it is not hard to verify that this renders A a distance matrix (the constant $8/9$ is chosen to "squeeze" the new 2.25-valued entries into the interval $[1, 2]$, which ensures the triangle inequality is satisfied), and for the sake of clarity we ignore this scaling below.

Let $B_1 = uv^T$ be the best rank-1 approximation of B, where $u \in \mathbb{R}^n$ and $v \in \mathbb{R}^r$. As above we let $\bar{v}$ denote n/r copies of v concatenated to form a vector in $\mathbb{R}^n$. Let $A' \in \mathbb{R}^{2n \times 2n}$

be the outer product of $[\bar{v}\ u]$ with itself, meaning,

$$A' = \begin{bmatrix} \bar{v}\bar{v}^T & \bar{v}u^T \\ u\bar{v}^T & uu^T \end{bmatrix},$$

and consider it as a rank-1 approximation of A. Let us bound the error $\|A - A'\|_F^2$. The error on A_{12} and A_{21} is optimal by construction. We need to show that the error on A_{11} and A_{22} is at most $\epsilon \cdot O(n^2)$. To this end we use the fact that in our hard distribution that generated B in Section 5.3, for all supported B, the top left and right singular vectors are nearly the same. Namely, the top-right one is close to $\mathbf{1}$, and the top-left one is close to $1.5 \cdot \mathbf{1}$, for any matrix B in the support.

Concretely, consider an entry in A_{22} whose value is 2.25. The corresponding entry in A' is $v_i v_j$ for $i \neq j$. Each v_i is the mean of a uniformly random vector in $\{1,2\}^r$. Thus it is a scaled binomial random variable with mean 1.5 and variance $1/(4r) = \frac{1}{4}\beta\epsilon$. (Recall that $r = \beta/\epsilon$ where $\beta > 0$ is a small constant.) Furthermore, v_i and v_j are independent. Therefore, the expected squared Frobenius norm error on that entry is

$$\mathbb{E}[(2.25 - v_i v_j)^2] = \mathbf{Var}[v_i v_j] = \mathbf{Var}[v_i] \cdot \mathbf{Var}[v_j] = (\tfrac{1}{4}\beta\epsilon)^2.$$

Thus the expected total error over all of the 2.25 entries (of which there are $O(n^2)$) is $\epsilon \cdot O(n^2)$. The concentration for the 1-entries in A_{11} is even stronger. This completes the proof of the symmetric case for $k = 1$.

5.6 Experiments

In this section, we evaluate the empirical performance of Algorithm 1 compared to the existing methods in the literature:

- Conventional SVD, from NumPy's linear algebra package.[6]

[6]https://docs.scipy.org/doc/numpy-1.15.1/reference/routines.linalg.html. This performs full SVD. The iterative SVD algorithms built into MATLAB and Python yielded errors larger by a few orders of magnitude than the reported methods, so they are not included.

- the input-sparsity time algorithm of Clarkson and Woodruff [CW17], referred to below as IS, which computes a low-rank approximation for arbitrary matrices in near-linear time (proportional to the number of nonzero entries in the input matrix).

- The algorithm of [BW18], referred to below as BW, which is the previous best algorithm for distance matrices.

The experimental setup is analogous to that in [BW18]. Specifically, we consider two datasets:

- Synthetic clustering dataset: This data set is generated using the `scikit-learn` package. We generate $10,000$ points with 200 features and partition the points into 20 clusters. As observed in our experiments, the dataset is expected to have a good rank-20 approximation.

- MNIST dataset: The dataset contains $70,000$ handwritten characters, and each is considered a point. We subsample $10,000$ points.

For each dataset we construct a symmetric distance matrix $A_{i,j} = \mathrm{d}(p_i, p_j)$. We use four distances d: Manhattan (ℓ_1), Euclidean (ℓ_2), Chebyshev (ℓ_∞) and Canberra[7] (ℓ_c). Figures 5-1 and 5-2 show the approximation error for each distance on each dataset, for varying values of the rank k. Note that the rank-k approximation error attained by SVD is the best possible. Table 5.1 lists the running times for $k = 40$. Figure 5-3 shows the running time of our algorithm on MNIST subsampled to varying sizes, for target rank $k = 40$.

Overall, the results show that our algorithm achieves similar accuracy to previous low-rank approximation algorithms, and close to the optimal SVD error, while being faster.

[7]The Canberra distance d_c between vectors $p, q \in \mathbb{R}^n$ is defined as $\mathrm{d}_c(p,q) = \sum_{i=1}^n \frac{|p_i - q_i|}{|p_i| + |q_i|}$.

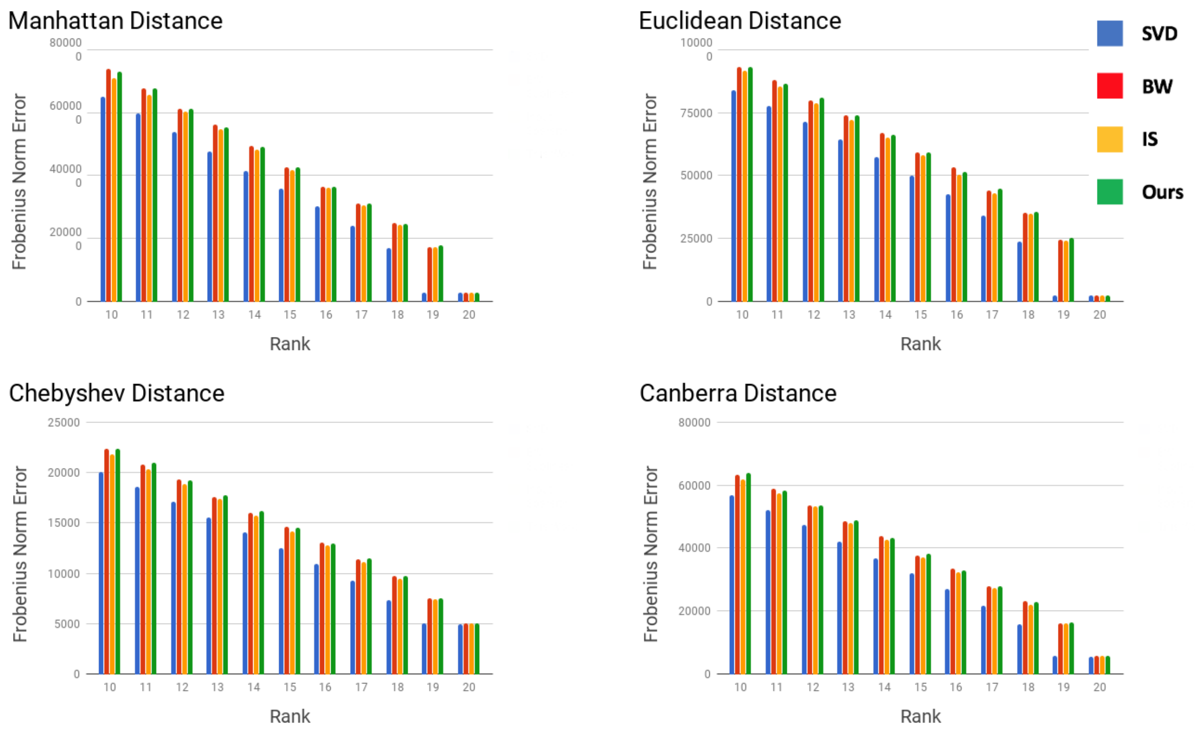


Figure 5-1: Approximation error of low-rank approximation algorithms on the synthetic clustering dataset

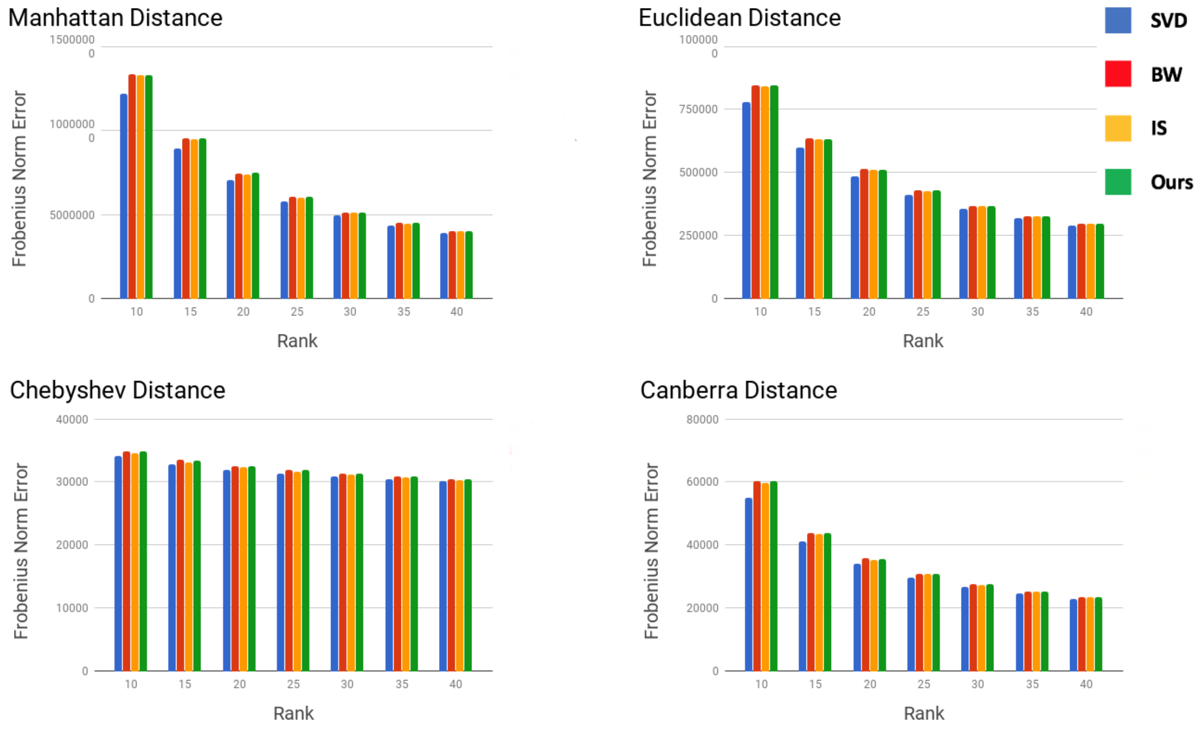


Figure 5-2: Approximation error of low-rank approximation algorithms on MNIST

	Synthetic				MNIST			
Metric	**SVD**	**IS**	**BW**	**Ours**	**SVD**	**IS**	**BW**	**Ours**
ℓ_2	398.77	8.95	1.70	1.17	398.50	34.32	4.17	1.23
ℓ_1	410.60	8.16	1.82	1.197	560.91	39.50	3.71	1.23
ℓ_∞	427.90	9.18	1.63	1.16	418.01	39.33	4.00	1.14
ℓ_c	452.17	8.49	1.76	1.15	390.07	38.34	3.91	1.24

Table 5.1: Running times (in seconds) of the compared methods for rank $k = 40$ approximation

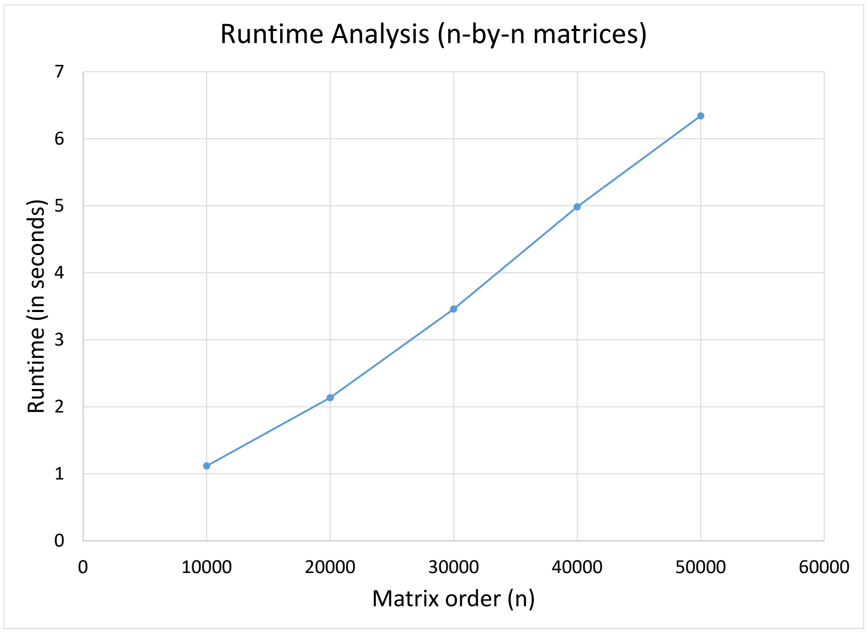


Figure 5-3: Running time of our algorithm on subsets of MNIST, with $k = 40$.

Chapter 6

Scalable Nearest Neighbor Search for Optimal Transport

The Optimal Transport distance (a.k.a. Wasserstein or Earth Mover distance) is a notion of similarity between *subsets* or *distributions* of points in a ground metric space. It is increasingly popular in applications involving rich data domains, such as images or text documents. However, it is costly to compute, which hinders its applicability to large datasets.

In this chapter we introduce *Flowtree*, an approximation algorithm for the Optimal Transport distance, based on efficient tree representations of the ground metric space. It is provably accurate for approximate nearest neighbor search, while also being fast and accurate in practice. Our extensive experimental evaluation on real data shows that compared to previous state of the art, Flowtree achieves up to 7.4 times faster running time.

6.1 Introduction

Given a finite metric space $\mathcal{M} = (X, \mathrm{d})$, and two distributions μ and ν supported on X, the Wasserstein-1 distance (a.k.a. Earth Mover's Distance or Optimal Transport) between μ and ν is defined as

$$W_1(\mu, \nu) = \min_{\tau} \sum_{x_1, x_2 \in X} \tau(x_1, x_2) \cdot \mathrm{d}(x_1, x_2), \tag{6.1}$$

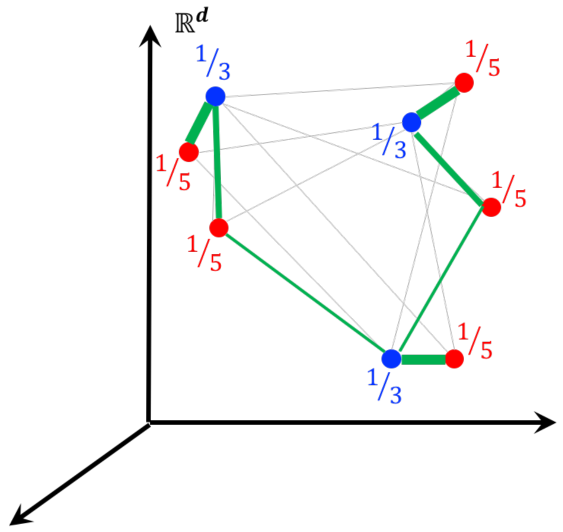


Figure 6-1: Illustration of the Optimal Transport distance, computed in $\mathbb{R}^d$ between a uniform distribution over a red set of 5 points, and a uniform distribution over a blue set of 3 points. The Optimal Transport distance is the cost of the minimum cost flow on the bipartite graph between the blue and red points (in gray), where the cost of each edge is the Euclidean distance between its endpoints. The flow is marked in green.

where the minimum is taken over all distributions τ on $X \times X$ whose marginals are equal to μ and ν.[1] See Figure 6-1 for illustration. The Wasserstein-1 distance and its variants are heavily used in applications to measure similarity in structured data domains, such as images [RTG00] and natural language text [KSKW15]. In particular, Kusner et al. [KSKW15] proposed the *Word Mover Distance* (WMD) for text documents. Each document is seen as a uniform distribution over the words it contains, and the underlying metric between words is induced by high-dimensional word embeddings such as word2vec [MSC+13] or GloVe [PSM14]. It is shown in [KSKW15] (see also [LYFC19, YCC+19, WYX+18]) that the Wasserstein-1 distance between the two distributions is a high-quality measure of semantic similarity between the associated documents.

To leverage the Wasserstein-1 distance for classification tasks, the above line of work uses the k-nearest neighbor classifier. For example, given a query document, [KSKW15] perform nearest neighbor search with respect to WMD on a dataset of documents labeled by topics, and the query document is given the same topic label as its nearest neighbor. This poses a

[1]For mathematical foundations of Wasserstein distances, see [Vil03].

notorious bottleneck for large datasets, necessitating the use of fast approximate similarity search algorithms. While such algorithms are widely studied for ℓ_p distances (chiefly ℓ_2; see [AIR18] for a survey), much less is known for Wasserstein distances, and a comprehensive study appears to be lacking. In particular, two properties of the W_1 distance make the nearest neighbor search problem very challenging. First, the W_1 distance is fairly difficult to compute (the most common approaches are combinatorial flow algorithms [Kuh55] or approximate iterative methods [Cut13]). Second, the W_1 distance is strongly incompatible with Euclidean (and more generally, with ℓ_p) geometries [Bou86, KN06, NS07, AIK08, ANN15, AKR18], which renders many of the existing techniques for nearest neighbor search inadequate (e.g., Johnson-Lindenstrauss-type random projections).

In this chapter, we systematically study the k-nearest neighbor search (k-NNS) problem with respect to the W_1 distance. In accordance with the above applications, we focus on the case where the ground set X is a finite subset of $\mathbb{R}^d$, endowed with the Euclidean distance, where d can be a high dimension, and each distribution over X has finite support of size at most s.[2] Given a dataset of n distributions $\mu_1, \mu_2, \ldots, \mu_n$, the goal is to preprocess it, such that given a query distribution ν (also supported on X), we can quickly find the k distributions μ_i closest to ν in the W_1 distance. To speed up search, the algorithms we consider rely on efficient estimates of the distances $W_1(\mu_i, \nu)$. This may lead to retrieving approximate nearest neighbors rather than the exact ones, which is often sufficient for practical applications.

6.1.1 Prior Work

Kusner et al. [KSKW15] sped up k-NNS for WMD by designing two approximations of W_1. The first algorithm estimates $W_1(\mu, \nu)$ as the Euclidean distance between their respective means. The second algorithm, called "Relaxed WMD" (abbrev. R-WMD), assigns every point in the support of μ to its closest point in the support of ν, and vice versa, and returns the maximum of the two assignments. Both of these methods produce an estimate no larger than the true distance $W_1(\mu, \nu)$. The former is much faster to compute, while the

[2] In the application to [KSKW15], X is the set word embeddings of (say) all terms in the English language, and s is the maximum number of terms per text document.

latter has a much better empirical quality of approximation. The overall k-NNS pipeline in [KSKW15] consists of the combination of both algorithms, together with exact W_1 distance computation. Recently, [AM19] proposed modifications to R-WMD by instating additional capacity constraints, resulting in more accurate estimates that can be computed almost as efficiently as R-WMD.

Indyk and Thaper [IT03] studied the approximate NNS problem for the W_1 distance in the context of image retrieval. Their approach capitalizes on a long line of work of *tree-based* methods, in which the given metric space is embedded at random into a tree metric. This is a famously fruitful approach for many algorithmic and structural statements [Bar96, Bar98, CCG+98, Ind01, GKL03, FRT04, CKR05, MN06]. It is useful in particular for Wasserstein distances, since the optimal flow (τ in (6.1)) on a tree can be computed in linear time, and since a tree embedding of the underlying metric yields an ℓ_1-embedding of the Wasserstein distance, as shown by [KT02, Cha02]. This allowed [IT03] to design an efficient NNS algorithm for W_1 based on classical locality-sensitive hashing (LSH). Recently, [LYFC19] introduced a kernel similarity measure based on the same approach, and showed promising empirical results for additional application domains.

6.1.2 Our Results

Flowtree. The tree-based method used in [IT03, LYFC19] is the classical Quadtree algorithm, which we have already used in Chapter 4. Here its application is even more straightforward, as we describe in detail Section 6.2. In this method, the ground metric X is embedded into a randomly shifted quadtree, and the cost of the optimal flow is computed with respect to the tree metric. We suggest a modification to this algorithm, which we call *Flowtree*: It computes the optimal *flow* on the same random tree, but evaluates the *cost* of that flow in the original ground metric.

While this may initially seem like a small modification, it in fact leads to an algorithm with vastly different properties. On one hand, while both algorithms run asymptotically in time $O(s)$ (where we recall that s is an upper bound on the support sizes of the input distributions), Quadtree is much faster in practice. The reason is that the *cost* of the optimal flow on the tree can be computed very efficiently, without actually computing the flow itself.

On the other hand, Flowtree is *dramatically more accurate.* Formally, we prove it has an asymptotically better approximation factor than Quadtree. Empirically, our experiments show that Flowtree is as accurate as state-of-the-art $O(s^2)$ time methods, while being much faster.

Theoretical results. A key difference between Flowtree and Quadtree is that the approximation quality of Flowtree is *independent of the dataset size*, i.e., of the number n of distributions $\mu_1, \ldots, \mu_n$ that need to be searched. Quadtree, on the other hand, degrades in quality as n grows. We expose this phenomenon in two senses:

- *Worst-case analysis:* We prove that Flowtree reports an $O(\log^2 s)$-approximate nearest neighbor w.h.p if the input distributions are uniform, and an $O(\log(d\Phi) \cdot \log s)$-approximate nearest neighbor (where d is the dimension and Φ is the coordinate range of X) even if they are non-uniform. Quadtree, on the other hand, reports an $O(\log(d\Phi) \cdot \log(sn))$-approximate nearest neighbor, and we show the dependence on n is necessary.

- *Random model:* We analyze a popular random data model, in which both Flowtree and Quadtree recover the exact nearest neighbor with high probability. Nonetheless, here too, we show that Flowtree's success probability is independent of n, while Quadtree's degrades as n grows.

Empirical results. We evaluate Flowtree, as well as several baselines and state-of-the-art methods, for nearest neighbor search in the W_1 distance on real-world datasets.

Our first set of experiments evaluates each algorithm individually. Our results yield a sharp divide among existing algorithms: The linear time ones are very fast in practice but only moderately accurate, while the quadratic time ones are much slower but far more accurate. Flowtree forms an intermediate category: it is slower and more accurate than the other linear time algorithms, and is at least 5.5 (and up to 30) times faster than the quadratic time algorithms, while attaining similar or better accuracy.

The above results motivate a sequential combination of algorithms, that starts with a fast and coarse algorithm to focus on the most promising candidates nearest neighbors, and gradually refines the candidate list by slower and more accurate algorithms. Such pipelines are commonly used in practice, and in particular were used in [KSKW15] (termed “prefetch and prune”). Our second set of experiments evaluates pipelines of various algorithms. We show that incorporating Flowtree into pipelines substantially improves the overall running times, by a factor of up to 7.4.

6.2 Preliminaries: Quadtree

In this section we explain the classical Quadtree algorithm for Optimal Transport. It has been described as part of our QuadSketch algorithm in Chapter 4 with various modifications and annotations. As it forms the basis for Flowtree, we now describe it in detail in its most generic form, and then its application to Optimal Transport.

Generic Quadtree. Let $X \subset \mathbb{R}^d$ be a finite set of points. Our goal is to embed X into a random tree metric, so as to approximately preserve each pairwise distance in X. To simplify the description, suppose that the minimum pairwise distance in X is exactly 1, and that all points in X have coordinates in $[0, \Phi]$. This is without loss of generality, as we can set the minimum distance to 1 by scaling, and we can shift all the points to have non-negative coordinates without changing internal distances.

The first step is to obtain a randomly shifted hypercube that encloses all points in X. To this end, let $H_0 = [-\Phi, \Phi]^d$ be the hypercube with side length 2Φ centered at the origin. Let $\sigma \in \mathbb{R}^d$ be a random vector with i.i.d. coordinates uniformly distributed in $[0, \Phi]$. We shift H_0 by σ, obtaining the hypercube $H = [-\Phi, \Phi]^d + \sigma$. Observe that H has side length 2Φ and encloses X. The random shift is needed in order to obtain formal guarantees for arbitrary X.

Now, we construct a tree of hypercubes by letting H be the root, halving H along each dimension, and recursing on the resulting sub-hypercubes. We add to the tree only those hypercubes that are non-empty (i.e., contain at least one point from X). Furthermore, we do not partition hypercubes that contain exactly one point from X; they become leaves.

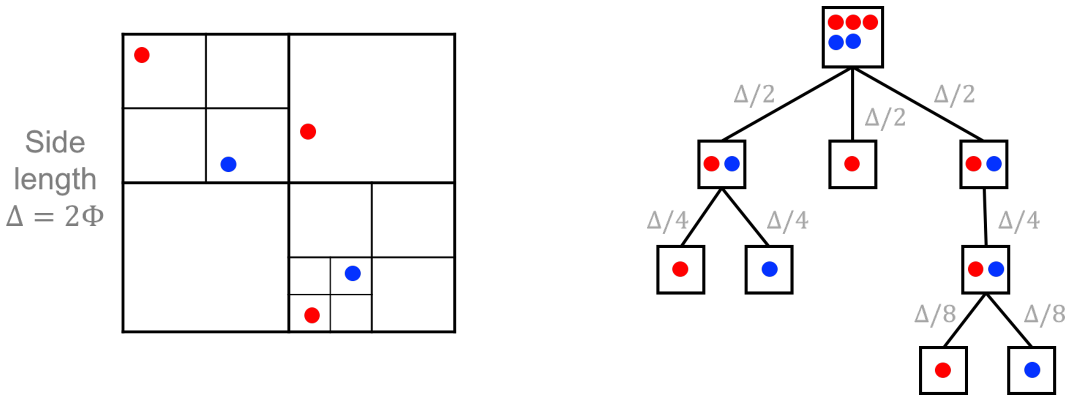


Figure 6-2: Quadtree illustration. Left: Input metric space. Right: Corresponding quadtree.

The resulting tree has at most $O(\log(d\Phi))$ levels and exactly $|X|$ leaves, one per point in X.[3] We number the root level as $\log \Phi + 1$, and the rest of the levels are numbered downward accordingly $(\log \Phi, \log \Phi - 1, \ldots)$. We set the weight of each tree edge between level $\ell + 1$ and level ℓ to be 2^{ℓ}. See Figure 6-2 for illustration.

The resulting quadtree has $O(|X|d \cdot \log(d\Phi))$ nodes, and it is straightforward to build it in time $\widetilde{O}(|X|d \cdot \log(d\Phi))$. Note that although the construction partitions each hypercube into 2^d sub-hypercubes, eliminating empty hypercubes ensures that the tree size does not depend exponentially on d.

Wasserstein-1 on Quadtree. The tree distance between each pair $x, x' \in X$ is defined as the total edge weight on the unique path between their corresponding leaves in the quadtree. Given two distributions μ, ν on X, the Wasserstein-1 distance with this underlying metric (as a proxy for the Euclidean metric on X) admits the closed-form

$$\sum_v 2^{\ell(v)} |\mu(v) - \nu(v)|,$$

where v ranges over all nodes in the tree, $\ell(v)$ is the level of v, $\mu(v)$ is the total μ-mass of points enclosed in the hypercube associated with v, and $\nu(v)$ is defined similarly for the

[3]This is since the diameter of the root hypercube H is $\sqrt{d}\Phi$, and the diameter of a leaf is no less than $1/2$, since by scaling the minimal distance in X to 1 we have assured that a hypercube of diameter $1/2$ contains a single point and thus becomes a leaf. Since the diameter is halved in each level, there are at most $O(\log(d\Phi))$ levels.

ν-mass. If μ, ν have supports of size at most s, then this quantity can be computed in time $O(s \cdot \log(d\Phi))$.

The above closed-form implies, in particular, that W_1 on the quadtree metric embeds isometrically into ℓ_1, as originally observed by [Cha02] following [KT02]. Namely, the ℓ_1 space has a coordinate associated with each tree node v, and a distribution μ is embedded in that space by setting the value of each coordinate v to $2^{\ell(v)}\mu(v)$, where $\mu(v)$ is defined as above. Furthermore, observe that if μ has support size at most s, then its corresponding ℓ_1 embedding w.r.t the tree metric has at most sh non-zero entries, where h is the height of the tree. Thus, computing W_1 on the tree metric amounts to computing the ℓ_1 distance between sparse vectors, which further facilitates fast implementation in practice.

6.3 Flowtree

The Flowtree algorithm for k-NNS w.r.t. the W_1 distance is as follows. In the preprocessing stage, we build a quadtree T on the ground set X, as described in Section 6.2. Let $t(x, x')$ denote the quadtree distance between every pair $x, x' \in X$. In the query stage, in order to estimate $W_1(\mu, \nu)$ between two distributions μ, ν, we compute the optimal flow f w.r.t. the tree metric, that is,

$$f = \operatorname{argmin}_{\tilde{f}} \sum_{x, x' \in X} \tilde{f}(x, x') \cdot t(x, x'),$$

where the argmin is taken over all distributions on $X \times X$ with marginals μ, ν. Then, the estimate of the distance between μ and ν is given by

$$\widetilde{W}_1(\mu, \nu) = \sum_{x, x' \in X} f(x, x') \cdot \|x - x'\|.$$

See Figure 6-3 for illustration.

Note that if the support sizes of μ and ν are upper-bounded by s, then the Flowtree estimate of their distance can be computed in time linear in s. The precise algorithm, and the proof of the next lemma, are given in Section 6.5.1.

Lemma 6.3.1. *$\widetilde{W}_1(\mu, \nu)$ can be computed in time $O(s(d + \log(d\Phi)))$.*

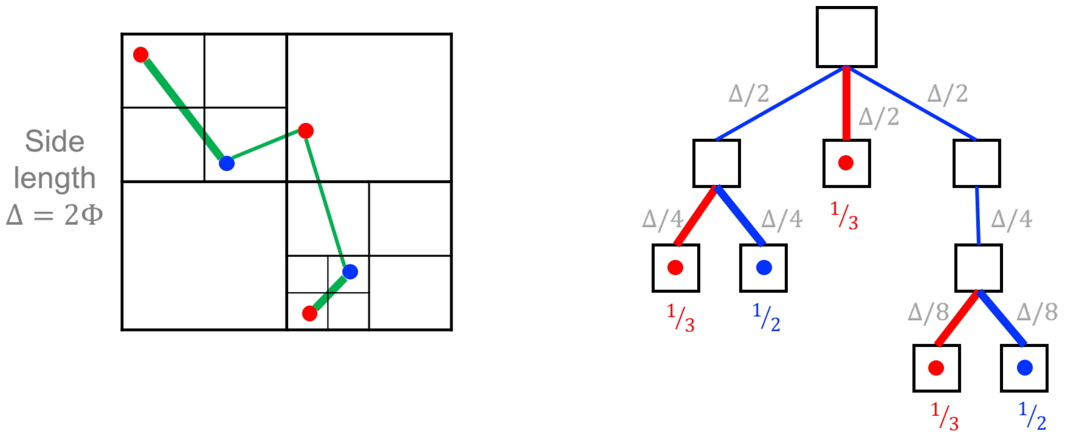


Figure 6-3: Flowtree illustration. Right: The optimal flow on the quadtree is computed by a greedy bottom-up process. Left: The flow cost is measured in the original metric space (the flow is marked in green).

Unlike Quadtree, Flowtree does not reduce to sparse ℓ_1 distance computation. Instead, one needs to compute the optimal flow tree f explicitly by bottom-up greedy algorithm, and then use it to compute $\widetilde{W}_1(\mu,\nu)$. (See Section 6.5.1.) On the other hand, Flowtree has the notable property mentioned earlier: its NNS approximation factor is *independent* of the dataset size n. In comparison, the classical Quadtree does not possess this property, and its accuracy deteriorates as the dataset becomes larger. We formally establish this distinction in two senses: first by analyzing worst-case bounds, and then by analyzing a popular random data model.

6.3.1 Worst-Case Bounds

We start with an analytic worst-case bound on the performance of quadtree. Let us recall notation: X is a finite subset of $\mathbb{R}^d$, and $\Phi > 0$ is the side length of a hypercube enclosing X. We are given a dataset of n distributions $\mu_1, \ldots, \mu_n$, and a query distribution ν, where each of these distributions is supported on a subset of X of size at most s. Our goal is to find a near neighbor of ν among $\mu_1, \ldots \mu_n$. A distribution μ_i is called a *c-approximate nearest neighbor* of ν if $W_1(\mu_i,\nu) \le c \cdot \min_{i^*} W_1(\mu_{i^*},\nu)$.

The following theorem adapts a result by Andoni, Indyk and Krauthgamer [AIK08] (where it is proven for a somewhat different algorithm, with similar analysis). All proofs appear in Section 6.5.

Theorem 6.3.2 (Quadtree upper bound)**.** *With probability* ≥ 0.99*, the nearest neighbor of* ν *among* $\mu_1, \ldots \mu_n$ *in the Quadtree distance is an* $O(\log(\min\{sn, |X|\}) \log(d\Phi))$*-approximate nearest neighbor in the* W_1 *distance.*

Next, we show that the $\log n$ factor in the above upper bound is *necessary* for Quadtree.

Theorem 6.3.3 (Quadtree lower bound)**.** *Suppose* c *is such that Quadtree is guaranteed to return a* c*-approximate nearest neighbor, for any dataset, with probability more than (say)* $1/2$*. Then* $c = \Omega(\log n)$*.*

In contrast, Flowtree attains an approximation factor that does not depend on n.

Theorem 6.3.4 (Flowtree upper bound)**.** *With probability* ≥ 0.99*, the nearest neighbor of* ν *among* $\mu_1, \ldots \mu_n$ *in the Flowtree distance is an* $O(\log(s) \log(d\Phi))$*-approximate nearest neighbor for the* W_1 *distance.*

Finally, we combine ideas from [AIK08] and Backurs and Indyk [BI14] to prove another upper bound for Flowtree, which is also independent of the dimension d and the numerical range Φ. No such result is known for Quadtree (nor does it follow from our techniques).

Theorem 6.3.5 (Flowtree upper bound for uniform distributions[4])**.** *For an integer* s*, assume that for every distribution there exists an integer* $s' \leq s$ *such that the weights of all elements in the support are integer multiples of* $1/s'$*. With probability* ≥ 0.99*, the nearest neighbor of* ν *among* $\mu_1, \ldots \mu_n$ *in the Flowtree distance is an* $O(\log^2 s)$*-approximate nearest neighbor for the* W_1 *distance.*

6.3.2 Random Model

The above worst-case results appear to be overly pessimistic for real data. Indeed, in practice we observe that Quadtree and especially Flowtree often recover the exact nearest neighbor.

[4]For simplicity, Theorem 6.3.5 is stated for uniform distribution (or close to uniform), such as documents in [KSKW15]. A similar result holds for any distribution, with additional dependence on the numerical range of mass values.

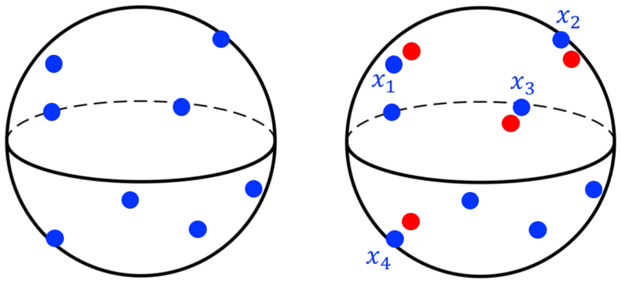


Figure 6-4: Random model illustration with $s = 4$. Left: The blue points are the N random data points. The data distributions are all subsets of 4 points. Right: The red points form a query distribution whose planted nearest neighbor is the distribution supported on $\{x_1, x_2, x_3, x_4\}$.

This motivates us to study their performance on a simple model of random data, which is standard in the study of nearest neighbor search.

The data is generated as follows. We choose a ground set X of N points i.i.d. uniformly at random on the d-dimensional unit sphere $\mathcal{S}^{d-1}$. For each subset of N of size s, we form a uniform distribution supported on that subset. These distributions make up the dataset $\mu_1, \ldots, \mu_n$ (so $n = \binom{N}{s}$).

To generate a query, pick any μ_i as the "planted" nearest neighbor, and let $x_1, \ldots, x_s$ denote its support. For $k = 1, \ldots, s$, choose a uniformly random point y_k among the points on $\mathcal{S}^{d-1}$ at distance at most ϵ from x_k, where ϵ is a model parameter. The query distribution ν is defined as the uniform distribution over $y_1, \ldots, y_s$. By known concentration of measure results, the distance from y_k to every point in X except x_k is $\sqrt{2} - o(1)$ with high probability. Thus, the optimal flow from ν to μ_i is the perfect matching $\{(x_k, y_k)\}_{k=1}^{s}$, and μ_i is the nearest neighbor of ν. The model is illustrated in Figure 6-4.

Theorem 6.3.6. *In the above model, the success probability of Quadtree in recovering the planted nearest neighbor decays exponentially with N, while the success probability of Flowtree is independent of N.*

6.4 Experiments

In this section we empirically evaluate Flowtree and compare it to various existing methods.

6.4.1 Synthetic Data

We implement the random model from Section 6.3.2. The results are in Figure 6-5. The x-axis is N (the number of points in the ground metric), and the y-axis is the fraction of successes over 100 independent repetitions of planting a query and recovering its nearest neighbor. As predicted by Theorem 6.3.6, Quadtree's success rate degrades as N increases (and we recall that $n = \binom{N}{s}$), while Flowtree's does not.

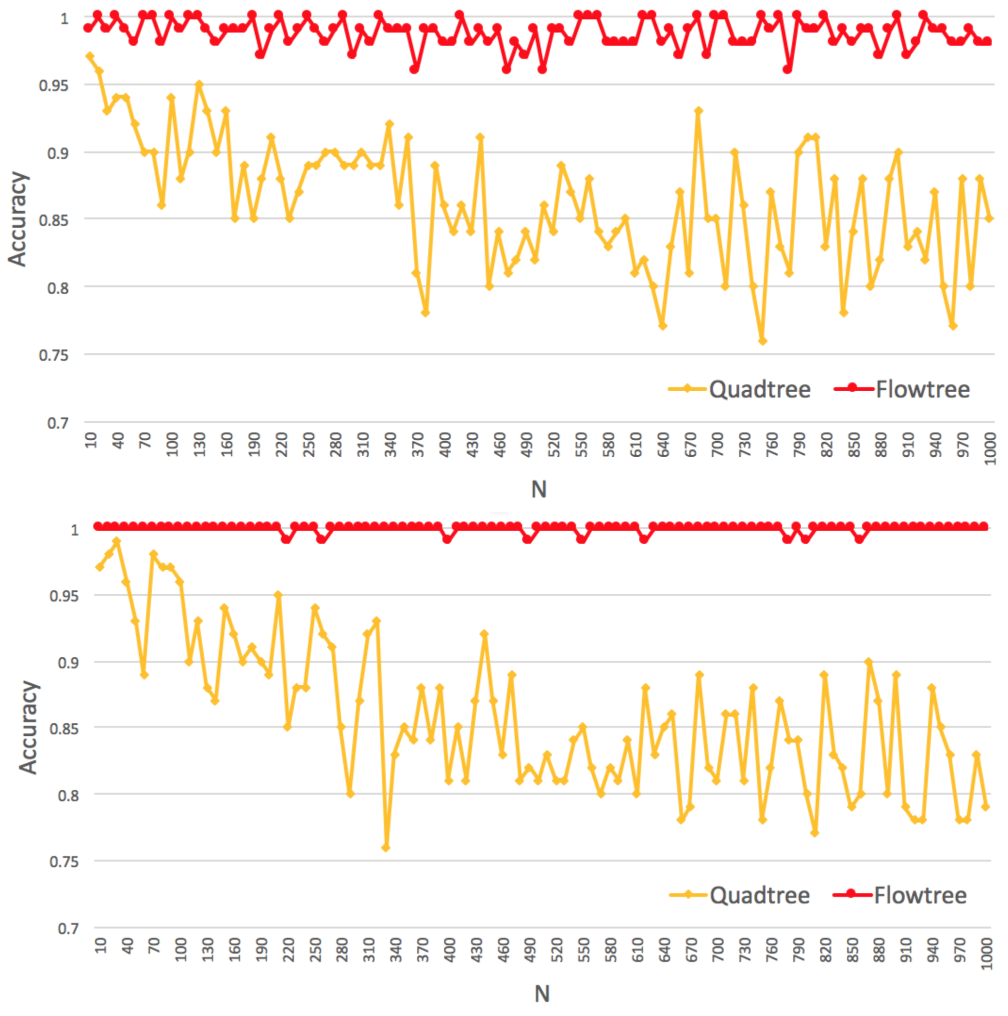


Figure 6-5: Results on random data, with two settings of parameters. Top: $d = s = 10$, $\epsilon = 0.25$. Bottom: $d = 10$, $s = 100$, $\epsilon = 0.4$.

Name	**Size**	**Queries**	**Underlying metric**	**Avg. support size** (s)
20news	$11,314$	$1,000$	Word embedding	115.9
Amazon	$10,000$	$1,000$	Word embedding	57.44
MNIST	$60,000$	$10,000$	2D Euclidean	150.07

Table 6.1: Dataset properties

6.4.2 Real Data

Datasets. We use three datasets from two application domains. Their properties are summarized in Table 6.1.

- *Text documents:* We use the standard benchmark 20news dataset of news-related online discussion groups, and a dataset of Amazon reviews split evenly over 4 product categories. Both have been used in [KSKW15] to evaluate the Word-Move Distance. Each document is interpreted as a uniform distribution supported on the terms it contains (after stopword removal). For the underlying metric, we use GloVe word embeddings [PSM14] with $400,000$ terms and 50 dimensions.

- *Image recognition:* We use the MNIST dataset of handwritten digits. As in [Cut13], each image is interpreted as a distribution over 28×28 pixels, with mass proportional to the greyscale intensity of the pixel (normalized so that the mass sums to 1). Note that the distribution is supported on only the non-white pixels in the image. The underlying metric is the 2-dimensional Euclidean distance between the 28×28 pixels, where they are identified with the points $\{(i,j)\}_{i,j=1}^{28}$ on the plane.

Algorithms. We evaluate the following algorithms:

- *Mean:* $W_1(\mu, \nu)$ is estimated as the Euclidean distance between the means of μ and ν. This method has been suggested and used in [KSKW15].[5]

- *Overlap:* A simple baseline that estimates $W_1(\mu, \nu)$ by the size of the intersection of their supports.

[5]There it is called Word Centroid Distance (WCD).

- *TF-IDF:* A well-known similarity measure for text documents. It is closely related to Overlap.[6] For MNIST we omit this baseline since it is not a text dataset.

- *Quadtree:* See Section 6.2.

- *Flowtree:* See Section 6.3.

- *R-WMD:* The Relaxed WMD method of [KSKW15], described in Section 6.1.1. We remark that this method does not produce an admissible flow (i.e., it does not adhere to the capacity and demand constraints of W_1).

- *ACT-1*: The Approximate Constrained Transfers method of [AM19] gradually adds constraints to R-WMD over i iterations, for a parameter i. The $i = 0$ case is identical to R-WMD, and increasing i leads to increasing both the accuracy and the running time. Like R-WMD, this method does not produce an admissible flow. In our experiments, the optimal setting for this method is $i = 1$,[7] which we denote by ACT-1. The appendix (Section 6.6) contains additional results for larger i.

- *Sinkhorn with few iterations:* The iterative Sinkhorn method of [Cut13] is designed to converge to a near-perfect approximation of W_1. Nonetheless, it can be adapted into a fast approximation algorithm by invoking it with a fixed small number of iterations. We use 1 and 3 iterations, referred to as Sinkhorn-1 and Sinkhorn-3 respectively. Since the Sinkhorn method requires tuning certain parameters (the number of iterations as well as the regularization parameter), the experiments in this section evaluate the method at its optimal setting, and the appendix (Section 6.6) includes experiments with other parameter settings.

As mentioned in Section 6.1.2, these methods can be grouped by their running time dependence on s:

- *"Fast" linear-time:* Mean, Overlap, TF-IDF, Quadtree

- *"Slow" linear-time:* Flowtree

[6] Namely, it is a weighted variant of Overlap, where terms are weighted according to their frequency in the dataset.

[7] This coincides with the results reported in [AM19].

- *Quadratic time:* R-WMD, ACT-1, Sinkhorn

The difference between "fast" and "slow" linear time is that the former algorithms reduce to certain simple cache-efficient operations, and furthermore, Mean greatly benefits from SIMD vectorization. In particular, Overlap, TF-IDF and Quadtree require computing a single ℓ_1 distance between sparse vectors, while Mean requires computing a single Euclidean distance in the ground metric. This renders them an order of magnitude faster than the other methods, as our empirical results will show.

Runtime measurement. All running times are measured on a "Standard F72s_v2" Microsoft Azure instance equipped with Intel Xeon Platinum 8168 CPU. In our implementations, we use NumPy linked with OpenBLAS, which is used in a single-threaded mode.

Implementation. We implement R-WMD, ACT and Sinkhorn in Python with NumPy, as they amount to standard matrix operations which are handled efficiently by the underlying BLAS implementation. We implement Mean, Overlap, TF-IDF, Quadtree and Flowtree in C++ (wrapped in Python for evaluation). For Mean we use the Eigen library to compute dense ℓ_2 distances efficiently. For Exact W_1 we use the POT library in Python, which in turn calls the Lemon graph library written in C++. The accuracy and pipeline evaluation code is in Python.

6.4.3 Individual Accuracy Experiments

Our first set of experiments evaluates the runtime and accuracy of each algorithm individually. The results are depicted in Figures 6-6 to 6-8. The plots report the recall@m accuracy as m grows. The recall@m accuracy is defined as the fraction of queries for which the true nearest neighbors is included in the top-m ranked neighbors (called *candidates*) by the evaluated method.

For each dataset, the left plot reports the accuracy of all of the methods for large values of m. The right plot reports the accuracy of the high-accuracy methods for smaller values of m (since they cannot be discerned in the left plots). The high-accuracy methods are Flowtree, R-WMD, ACT, Sinkhorn, and on MNIST also Quadtree. For Quadtree and Flowtree, which

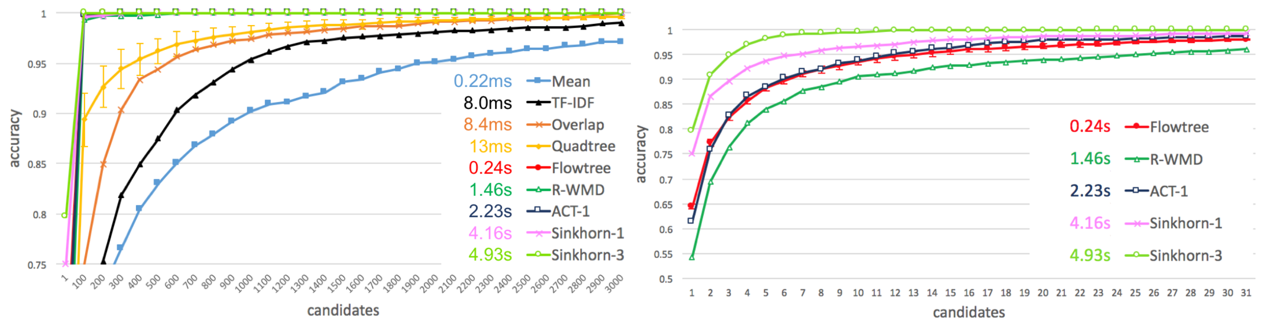


Figure 6-6: Individual accuracy and runtime results on 20news

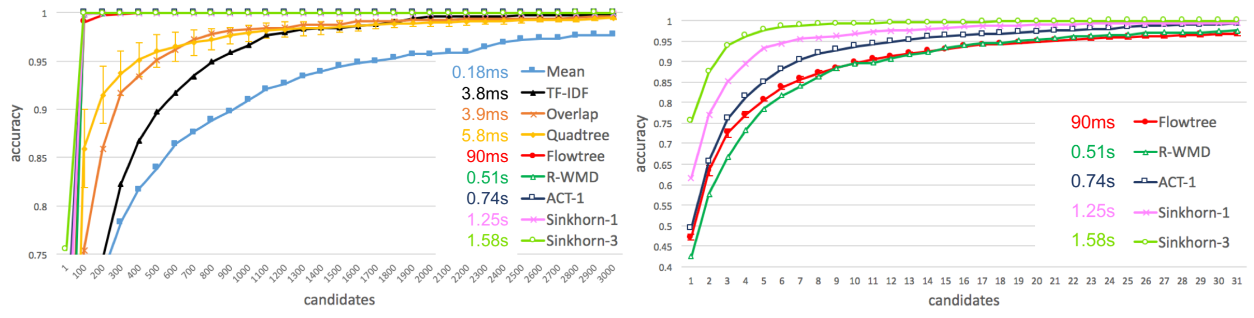


Figure 6-7: Individual accuracy and runtime results on Amazon

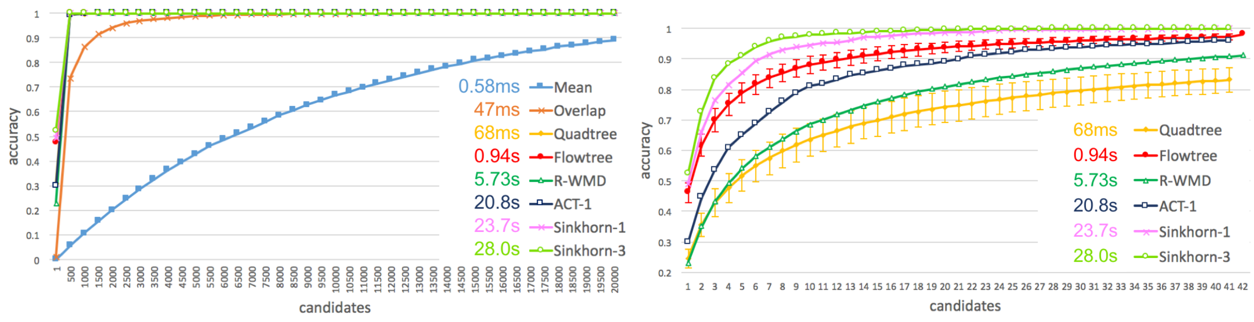


Figure 6-8: Individual accuracy and runtime results on MNIST (see footnote (*) in Table 6.2 regarding runtimes)

Dataset	**Mean**	**TF-IDF**	**Overlap**	**Quadtree**	**Flowtree**
20news	0.22ms	8.0ms	8.4ms	13ms	0.24s
Amazon	0.18ms	3.8ms	3.9ms	5.8ms	90ms
MNIST*	0.58ms	—	47ms	68ms	0.94s

Dataset	**R-WMD**	**ACT-1****	**Sinkhorn-1**	**Sinkhorn-3**	**Exact** W_1
20news	1.46s	2.23s	4.16s	4.93s	41.5s
Amazon	0.51s	0.74s	1.25s	1.58s	4.23s
MNIST*	5.73s	20.8s	23.7s	28.0s	154.0s

* On MNIST, the accuracy of R-WMD, ACT and Sinkhorn is evaluated on $1,000$ random queries. The running time of R-WMD, ACT, Sinkhorn and Exact W_1 is measured on 100 random queries. The running time of Flowtree is measured $1,000$ random queries.

** ACT takes a faster form when applied to uniform distributions. We use a separate implementation for this case. This accounts for the large difference in its performance on 20news and Amazon (where distributions are uniform) compared to MNIST (where they are not).

Table 6.2: Running times

are randomized methods, we report the mean and standard deviation (shown as error bars) of 5 executions. The other methods are deterministic. The legend of each plot is annotated with the running time of each method, also summarized in Table 6.2.

Results. The tested algorithms yield a wide spectrum of different time-accuracy tradeoffs. The "fast" linear time methods (Mean, TF-IDF, Overlap and Quadtree) run in order of milliseconds, but are less accurate than the rest. The quadratic time methods (R-WMD, ACT-1 and Sinkhorn) are much slower, running in order of seconds, but are dramatically more accurate.

Flowtree achieves comparable accuracy to the quadratic time baselines, while being faster by a margin. In particular, its accuracy is either similar to or better than R-WMD, while being 5.5 to 6 times faster. Compared to ACT-1, Flowtree is either somewhat less or more accurate (depending on the dataset), while being at least 8 times faster. Compared to Sinkhorn, Flowtree achieves somewhat lower accuracy, but is at least 13.8 and up to 30 times faster.

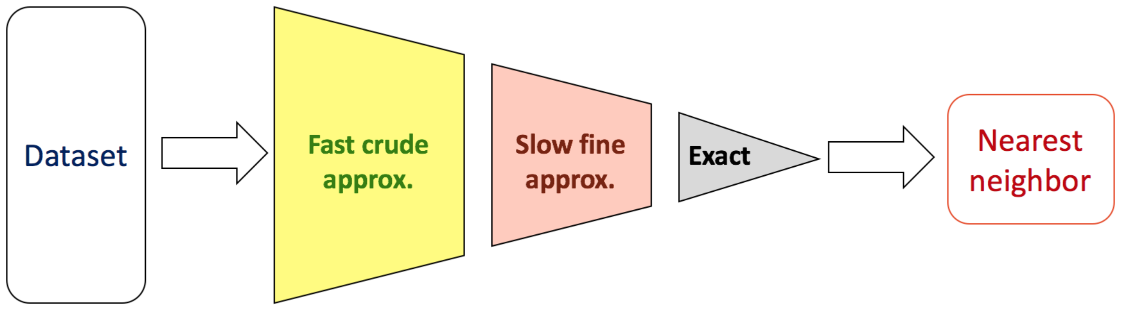


Figure 6-9: Nearest neighbor search pipeline illustration

6.4.4 Pipeline Experiments

The above results exhibit a sharp divide between fast and coarse algorithms to slow and accurate ones. In practical nearest neighbor search system, both types of algorithms are often combined sequentially as a *pipeline* (e.g., [SZ03, JDS08, JDS11]). First, a fast and coarse method is applied to all points, pruning most of them; then a slower and more accurate method is applied to the surviving points, pruning them further; and so on, until finally exact computation is performed on a small number of surviving points. See Figure 6-9 for illustration. In particular, [KSKW15] employ such a pipeline for the Word Mover Distance, which combines Mean, R-WMD, and exact W_1 computation.

In this section, we systematically evaluate pipelines built of the algorithms tested above, on the 20news dataset.

Experimental setup. We perform two sets of experiments: In one, the pipeline reports one candidate, and its goal is to output the true nearest neighbor (i.e., recall@1). In the other, the pipeline reports 5 candidates, and its goal is to include the true nearest neighbor among them (i.e., recall@5). We fix the target accuracy to 0.9 (i.e., the pipeline must achieve the recall goal on 90% of the queries), and report its median running time over 3 identical runs.

Evaluated pipelines. The baseline pipelines we consider contain up to three methods:

- First: Mean, Overlap or Quadtree.

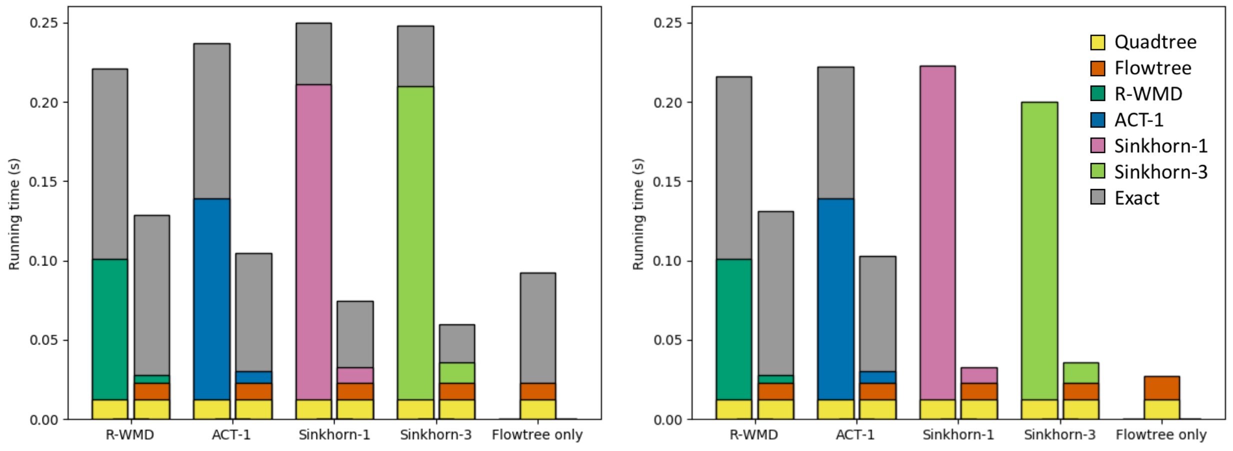


Figure 6-10: Best performing pipelines for recall@1 ≥ 0.9 (on the left) and recall@5 ≥ 0.9 (on the right). Each vertical bar denotes a pipeline, built of the methods indicated by the color encoding, bottom-up. The y-axis measures the running time up to each step of the pipeline. The plot depicts 4 baseline pipelines, consisting of Quadtree, then the method X indicated on the x-axis (R-WMD, ACT-1, Sinkhorn-1 or Sinkhorn-3), and then (optionally) Exact W_1. Next to each baseline bar we show the bar obtained by adding Flowtree to the pipeline as an intermediate algorithm between Quadtree and X. The rightmost bar in each plot shows the pipeline obtained by using Flowtree instead of X.

- Second: R-WMD, ACT-1, Sinkhorn-1, or Sinkhorn-3.
- Third: Exact W_1 computation. For recall@5 pipelines whose second method is Sinkhorn, this third step is omitted, since they already attain the accuracy goal without it.

To introduce Flowtree into the pipelines, we evaluate it both as an intermediate stage between the first and second methods, and as a replacement for the second method.

Pipeline parameters. A pipeline with ℓ algorithms has parameters $c_1, \ldots, c_{\ell-1}$, where c_i is the number of output candidates (non-pruned points) of the i^{th} algorithm in the pipeline.[8] We tune the parameters of each pipeline optimally on a random subset of 300 queries (fixed for all pipelines). The optimal parameters are listed in the appendix.

Results. We found that in the first step of the pipeline, Quadtree is significantly preferable to Mean and Overlap, and the results reported in this section are restricted to it. More results

[8]The final algorithm always outputs either 1 or 5 points, according to the recall goal.

	Recall@1 ≥ 0.9	Recall@5 ≥ 0.9
Without Flowtree	0.221s	0.200s
With Flowtree	0.059s	0.027s

Table 6.3: Best pipeline runtime results

are included in the appendix.

Figure 6-10 shows the runtimes of pipelines that start with Quadtree. Note that each pipeline is optimized by different parameters, not depicted in the figure. For example, Sinkhorn-3 is faster than Sinkhorn-1 on the right plot, even though it is generally a slower algorithm. However, it is also more accurate, which allows the preceding Quadtree step to be less accurate and report fewer candidates, while still attaining overall accuracy of 0.9. Specifically, in the optimal recall@5 setting, Sinkhorn-3 runs on 227 candidates reported by Quadtree, while Sinkhorn-1 runs on 295.

The best runtimes are summarized in Table 6.3. The results show that introducing Flowtree improves the best runtimes by a factor of 3.7 for recall@1 pipelines, and by a factor of 7.4 for recall@5 pipelines.

In the recall@1 experiments, the optimally tuned baseline pipelines attain runtimes between 0.22 to 0.25 seconds. Introducing Flowtree before the second method in each pipeline improves its running time by a factor of 1.7 to 4.15. Introducing Flowtree instead of the second method improves the runtime by a factor of 2.4 to 2.7.

Once Flowtree is introduced into the recall@1 pipelines, the primary bottleneck becomes the final stage of exact W_1 computations. In the recall@5 experiments, this step is not always required, which enables larger gains for Flowtree. In these experiments, the optimally tuned baseline pipelines attain runtimes between 0.2 to 0.22 seconds. Introducing Flowtree before the second method in each pipeline improves its running time by a factor of 1.64 to 6.75. Introducing Flowtree instead of the second method improves the runtime by a factor of 7.4 to 8.

Overall, Flowtree significantly improves the running time of every pipeline, both as an addition and as a replacement.

6.5 Proofs

6.5.1 Flowtree Computation

In this section we prove Lemma 6.3.1. We begin by specifying the greedy flow computation algorithm on the tree. Let h denote the height of the tree (for the quadtree the height is $h = O(\log(d\Phi))$. Suppose we are given a pair of distributions μ, ν, each supported on at most s leaves of the tree. For every node v in the tree, let $C_\mu(v)$ denote the set of points in $x \in X$ such that $\mu(x) > 0$ and the tree leaf that contains x is a descendant of v. Similarly define $C_\nu(v)$. Note that we only need to consider nodes for which either $C_\mu(v)$ or $C_\nu(v)$ is non-empty, and there are at most $2sh$ such nodes.

The algorithm starts with a zero flow f, and processes the nodes in a bottom-up order starting at the leaf. In each node, the unmatched demands collected from its children are matched arbitrarily, and the demands that cannot be matched are passed on to the parent. In more detail, a node is processed as follows:

1. Collect from the children the list of unmatched μ-demands for the nodes in $C_\mu(v)$ and the list of unmatched ν-demands for the nodes in $C_\nu(v)$. Let $\{\mu_v(x) : x \in C_\mu(v)\}$ denote the unmatched ν-demands and let $\{\nu_v(x) : x \in C_\nu(v)\}$ denote the unmatched ν-demands.

2. While there is a pair $x \in C_\mu(v)$ and $x' \in C_\mu(v)$ with $\mu_v(x) > 0$ and $\nu_v(x') > 0$, let $\eta = \min\{\mu_v(x), \nu_v(x')\}$, and update:

   (a) $f(x, x') \mathrel{+}= \eta$,

   (b) $\mu_v(x) \mathrel{-}= \eta$,

   (c) $\nu_v(x') \mathrel{-}= \eta$.

3. Now either μ_v or ν_v is all-zeros. If the other one is not all-zeros (i.e., there is either remaining unmatched μ-demand or remaining unmatched ν-demand), pass it on to the parent.

A leaf v contains a single point $x \in X$ with either $\mu(x) > 0$ or $\nu(x) > 0$; it simply passes it on to its parent without processing.

It is well known that the above algorithm computes an optimal flow on the tree (with respect to tree distance costs), see, e.g., [KK95]. Let us now bound its running time. The processing time per node v in the above algorithm is $O(|C_\mu(v)|+|C_\nu(v)|)$. In every given level in the tree, if $v_1,\dots,v_k$ are the nodes in that level, then $\{C_\mu(v_1),\dots,C_\mu(v_k)\}$ is a partition of the support of μ, and $\{C_\nu(v_1),\dots,C_\nu(v_k)\}$ is a partition of the support of ν. Therefore the total processing time per level is $O(s)$, and since there are h levels, the flow computation time is $O(sh)$. Then we need to compute the Flowtree output $\widetilde{W}_1(\mu,\nu)$. Observe that in the above algorithm, whenever we match demands between a pair x, x', we fully satisfy the unmatched demand of one of them. Therefore the output flow f puts non-zero flow between at most $2s$ pairs. For each such pair we need to compute the Euclidean distance in time $O(d)$, and the overall running time is $O(s(d+h))$.

6.5.2 Quadtree Upper Bound

Proof of Theorem 6.3.2. Let $x, y \in X$. Let $p_\ell(x,y)$ be the probability that x, y fall into the same cell (hypercube) in level ℓ of the quadtree. It satisfies,

$$1-\frac{\|x-y\|_1}{2^\ell} \le p_\ell(x,y) \le \exp\left(-\frac{\|x-y\|_1}{2^\ell}\right). \tag{6.2}$$

To see this, recall that in level ℓ we impose a grid with side length 2^ℓ, shifted at random by an i.i.d. uniform shift in $[0,2^\ell]$ in each coordinate. The probability that x, y are separated in coordinate i is $2^{-\ell}|x_i-y_i|$, and thus $p_\ell(x,y)=\prod_{i=1}^d(1-2^{-\ell}|x_i-y_i|)$. The lower bound in Equation (6.2) follows by a union bound, and the upper bound follows by applying the general estimate $1-z\le\exp(-z)$ to each term in the product.

Let t be the tree metric induced on X by the quadtree. Note that for $t(x,y)$ to be at most $O(2^\ell)$, x, y must fall into the same hypercube in level ℓ. For any $\delta>0$, we can round $\|x-y\|_1/\log(1/\delta)$ to its nearest power of 2 and obtain ℓ such that

$$2^\ell=\Theta\left(\frac{\|x-y\|_1}{\log(1/\delta)}\right).$$

It satisfies,

$$\Pr\left[t(x,y) < \frac{O(1)}{\log(1/\delta)}\|x-y\|_1\right] \le \delta.$$

By letting $\delta = \Omega(\min\{1/|X|, 1/(s^2n)\})$, we can take union bound either over all pairwise distances in X (of which there are $\binom{|X|}{2}$), or over all distances between the support of the query ν and the union of supports of the dataset $\mu_1, \ldots, \mu_n$ (of which there are at most s^2n, if every support has size at most s). Then, with probability say 0.995, all those distances are contracted by at most $O(\log(\min\{sn, |X|\}))$, i.e.,

$$t(x,y) \ge \frac{1}{O(\log(1/\delta))}\|x-y\|_1. \tag{6.3}$$

On the other hand,

$$\begin{aligned}\mathbb{E}[t(x,y)] &= \sum_\ell 2^\ell \cdot (1 - p_\ell(x,y)) \\ &\le \sum_\ell 2^\ell \cdot \frac{\|x-y\|_1}{2^\ell} \\ &\le O(\log(d\Phi)) \cdot \|x-y\|_1.\end{aligned}$$

Let μ^* be the true nearest neighbor of ν in $\mu_1, \ldots, \mu_n$. Let $f^*_{\mu^*,\nu}$ be the optimal flow between them. Then by the above,

$$\mathbb{E}\left[\sum_{(x,y)\in X\times X} f^*_{\mu^*,\nu}(x,y)t(x,y)\right] \le O(\log(d\Phi)) \sum_{(x,y)\in X\times X} f^*_{\mu^*,\nu}(x,y)\|x-y\|_1.$$

By Markov, with probability say 0.995,

$$\sum_{(x,y)\in X\times X} f^*_{\mu^*,\nu}(x,y)\cdot t(x,y) \le O(\log(d\Phi)) \sum_{(x,y)\in X\times X} f^*_{\mu^*,\nu}(x,y)\cdot\|x-y\|_1. \tag{6.4}$$

Let μ' be the nearest neighbor of ν in the dataset according to the quadtree distance. Let $f^*_{\mu',\nu}$ be the optimal flow between them in the true underlying metric (ℓ_1 on X), and let $f_{\mu,\nu}$ be the optimal flow in the quadtree. Finally let W_t denote the Wasserstein-1 distance on the quadtree. Then,

$$\begin{aligned}
& W_1(\mu', \nu) \\
& = \sum_{(x,y)\in X\times X} f^*_{\mu',\nu}(x,y)\cdot \|x-y\|_1 \\
& \leq \sum_{(x,y)\in X\times X} f_{\mu',\nu}\cdot \|x-y\|_1 && f^*_{\mu^*,\nu} \text{ is optimal for } \|\cdot\|_1 \\
& \leq O(\log(\min\{sn,|X|\})) \sum_{(x,x')\in X\times X} f_{\mu',\nu}\cdot t(x,y) && \text{eq. (6.3)} \\
& = O(\log(\min\{sn,|X|\}))\cdot W_t(\mu',\nu) && \text{definition of } W_t \\
& \leq O(\log(\min\{sn,|X|\}))\cdot W_t(\mu^*,\nu) && \mu' \text{ is the nearest neighbor in } W_t \\
& = O(\log(\min\{sn,|X|\})) \sum_{(x,y)\in X\times X} f_{\mu^*,\nu}\cdot t(x,y) && \text{definition of } W_t \\
& \leq O(\log(\min\{sn,|X|\})) \sum_{(x,y)\in X\times X} f^*_{\mu^*,\nu}\cdot t(x,y) && f_{\mu^*,\nu} \text{ is optimal for } t(\cdot,\cdot) \\
& \leq O(\log(\min\{sn,|X|\})\log(d\Phi)) \sum_{(x,y)\in X\times X} f^*_{\mu^*,\nu}\cdot \|x-y\|_1 && \text{eq. (6.4)} \\
& = O(\log(\min\{sn,|X|\})\log(d\Phi))\cdot W_1(\mu^*,\nu),
\end{aligned}$$

so μ' is a $O(\log(\min\{sn,|X|\})\log(d\Phi))$-approximate nearest neighbor. □

6.5.3 Quadtree Lower Bound

Proof of Theorem 6.3.3. It suffices to prove the claim for $s = 1$ (i.e., the standard ℓ_1-distance). Let $d > 0$ be an even integer. Consider the d-dimensional hypercube. Our query point is the origin. The true nearest neighbor is e_1 (standard basis vector). The other data points are the hypercube nodes whose hamming weight is exactly $d/2$. The number of such points is $\Theta(2^d/\sqrt{d})$, and this is our n.

Consider imposing the grid with cell side 2 on the hypercube. The probability that 0 and 1 are uncut in a given axis is exactly 1/2, and since the shifts in different axes are independent, the number of uncut axes is distributed as $Bin(d, 1/2)$. Thus with probability 1/2 there are at least $d/2$ uncut dimensions. If this happens, we have a data point hashed into the same grid cell as the origin (to get such data point, put 1 in any $d/2$ uncut dimensions and 0 in the rest), so its quadtree distance from the origin is 1. On the other hand, the distance of the origin to its true nearest neighbor e_1 is at least 1, since they will necessarily be separated in the next level (when the grid cells have side 1). Thus the quadtree cannot tell between the true nearest neighbor and the one at distance $d/2$, and we get the lower bound $c \geq d/2$. Since $n = \Theta(2^d/\sqrt{d})$, we have $d/2 = \Omega(\log n)$ as desired. $\square$

6.5.4 Flowtree Upper Bound

Proof of Theorem 6.3.4. The proof is the same as for Theorem 6.3.2, except that in eq. (6.3), we take a union bound only over the s^2 distances between the supports of ν and μ^* (the query and its true nearest neighbor). Thus each distance between μ^* and ν is contracted by at most $O(\log s)$.

Let W_F denote the Flowtree distance estimate of W_1. Let μ' be the nearest neighbor of ν in the Flowtree distance. With the same notation in the proof of Theorem 6.3.2,

$$\begin{aligned}
W_1(\mu', \nu) &= \sum_{(x,y)\in X\times X} f^*_{\mu',\nu}(x,y)\cdot \|x-y\|_1 \\
&\leq \sum_{(x,y)\times X\times X} f_{\mu',\nu}(x,y)\cdot \|x-y\|_1 && f^*_{\mu',\nu} \text{ is optimal for } \|\cdot\|_1 \\
&= W_F(\mu', \nu) && \text{Flowtree definition} \\
&\leq W_F(\mu^*, \nu) && \mu' \text{ is nearest in Flowtree distance} \\
&= \sum_{(x,y)\in X\times X} f_{\mu^*,\nu}(x,y)\cdot \|x-y\|_1 && \text{Flowtree definition} \\
&\leq O(\log s) \sum_{(x,y)\in X\times X} f_{\mu^*,\nu}(x,y)\cdot t(x,y) && \text{eq. (6.3)} \\
&\leq O(\log s) \sum_{(x,y)\in X\times X} f^*_{\mu^*,\nu}(x,y)\cdot t(x,y) && f_{\mu^*,\nu} \text{ is optimal for } t(\cdot,\cdot) \\
&\leq O(\log(d\Phi)\log s) \sum_{(x,y)\in X\times X} f^*_{\mu^*,\nu}(x,y)\cdot \|x-y\|_1 && \text{eq. (6.4)} \\
&= O(\log(d\Phi)\log s)\cdot W_1(\mu^*, \nu),
\end{aligned}$$

as needed. Note that the difference from the proof of Theorem 6.3.2 is that we only needed the contraction bound (eq. (6.3)) for distances between μ^* and ν. $\square$

6.5.5 Flowtree Upper Bound for Uniform Distributions

Proof of Theorem 6.3.5. We set $\varepsilon = 1/\log s$. Let $t'(x,y)$ denote the quadtree distance where the weight corresponding to a cell v in level $\ell(v)$ is $2^{\ell(v)(1-\varepsilon)}$ instead of $2^{\ell(v)}$. Let $f_{\mu,\nu}$ be the optimal flow in the quadtree defined by weights t'.

Let $\delta = c/s^2$ where $c > 0$ is a sufficiently small constant. For a every x, y, let ℓ_{xy} be the largest integer such that

$$2^{\ell_{xy}} \leq \frac{\|x-y\|_1}{(\log(1/\delta))^{1/(1-\epsilon)}}.$$

The probability that x, y are separated (i.e., they are in different quadtree cells) in level ℓ_{xy} is

$$1 - p_{\ell_{xy}}(x,y) \geq 1 - \exp\left(-\frac{\|x-y\|_1}{2^{\ell_{xy}}}\right) \geq 1 - \frac{\delta}{1-\epsilon}.$$

By the setting of δ, we can take a union bound over all $x \in \text{support}(\mu^*)$ and $y \in \text{support}(\nu)$ and obtain that with say 0.99 probability, simultaneously, every pair x, y is separated at level ℓ_{xy}. We denote this event by $\mathcal{E}_{lower}$ and suppose it occurs. Then for every x, y we have

$$t'(x,y) \geq 2 \cdot 2^{\ell_{xy}(1-\epsilon)} \geq 2 \cdot \left(\frac{1}{2} \cdot \frac{\|x-y\|_1}{(\log(1/\delta))^{1/(1-\epsilon)}}\right)^{1-\epsilon} \geq \frac{\|x-y\|_1^{1-\epsilon}}{\log(1/\delta)} = \frac{\|x-y\|_1^{1-\epsilon}}{\Theta(\log s)}.$$

Next we upper-bound the expected tree distance $t'(x,y)$. (Note that we are not conditioning on $\mathcal{E}_{lower}$.) Observe that

$$t'(x,y) = 2 \sum_{\ell=-\infty}^{\infty} 2^{\ell(1-\epsilon)} \cdot \mathbf{1}\{x, y \text{ are separated at level } \ell\}.$$

Let $L_{x,y}$ be the largest integer such that $2^{L_{xy}} \leq \|x-y\|_1$. We break up $t'(x,y)$ into two terms,

$$t'_{lower}(x,y) = 2 \sum_{\ell=-\infty}^{L_{xy}} 2^{\ell(1-\epsilon)} \cdot \mathbf{1}\{x, y \text{ are separated at level } \ell\},$$

and

$$t'_{upper}(x,y) = 2 \sum_{\ell=L_{xy}+1}^{\infty} 2^{\ell(1-\epsilon)} \cdot \mathbf{1}\{x, y \text{ are separated at level } \ell\},$$

thus $t'(x,y) = t'_{lower}(x,y) + t'_{upper}(x,y)$. For $t'_{lower}(x,y)$ it is clear that deterministically,

$$t'_{lower}(x,y) \le 2 \sum_{\ell=-\infty}^{L_{xy}} 2^{\ell(1-\epsilon)} = O\left(2^{L_{xy}(1-\epsilon)}\right) = O\left(\|x-y\|_1^{1-\epsilon}\right).$$

For $t'_{upper}(x,y)$, we have

$$\begin{aligned}
\mathbb{E}[t'_{upper}(x,y)] &= 2 \sum_{\ell=L_{xy}+1}^{\infty} 2^{\ell(1-\epsilon)} p_\ell(x,y) \\
&\le 2 \sum_{\ell=L_{xy}}^{\infty} 2^{\ell(1-\epsilon)} \cdot \frac{\|x-y\|_1}{2^\ell} \\
&= 2\|x-y\|_1 \sum_{\ell=L_{xy}}^{\infty} 2^{-\epsilon\ell} \\
&= 2\|x-y\|_1 \cdot \frac{2^{-L_{xy}\cdot\epsilon}}{1-2^{-\epsilon}} \\
&\le O(\log s) \cdot \|x-y\|_1^{1-\epsilon},
\end{aligned}$$

where in the final bound we have used that $2^{L_{xy}} = \Theta(\|x-y\|_1)$ and $1-2^{-\epsilon} = \Theta(\epsilon) = \Theta(\log s)$. Together,

$$\mathbb{E}[t'(x,y)] = \mathbb{E}[t'_{lower}(x,y) + t'_{upper}(x,y)] \le \Theta(\log s) \cdot \|x-y\|_1^{1-\epsilon}. \qquad (6.5)$$

Now we are ready to show the $O(\log^2 s)$ upper bound on the approximation factor. We start with,

$$\begin{aligned}
W_1(\mu',\nu) &= \sum_{(x,y)\in X\times X} f^*_{\mu',\nu}(x,y) \cdot \|x-y\|_1 && f^*_{\mu',\nu} \text{ is optimal for } \|\cdot\|_1 \\
&\le \sum_{(x,y)\times X\times X} f_{\mu',\nu}(x,y) \cdot \|x-y\|_1 \\
&= W_F(\mu',\nu) && \text{Flowtree definition} \\
&\le W_F(\mu^*,\nu) && \mu' \text{ is nearest to } \nu \text{ in Flowtree distance} \\
&= \sum_{(x,y)\in X\times X} f_{\mu^*,\nu}(x,y) \cdot \|x-y\|_1 && \text{Flowtree definition.}
\end{aligned}$$

By subadditivity of $(\cdot)^{1-\varepsilon}$,

$$\sum_{(x,y)\in X\times X} f_{\mu^*,\nu}(x,y)\cdot\|x-y\|_1 \le \left(\sum_{(x,y)\in X\times X} f^{1-\varepsilon}_{\mu^*,\nu}(x,y)\cdot\|x-y\|_1^{1-\varepsilon}\right)^{1/(1-\varepsilon)}.$$

Next we use the assumption of the lemma, that there are integers $1\le s', s''\le s$ such each nonzero probability mass in μ^* is an integer multiple if $1/s'$, and each nonzero probability mass in ν is an integer multiple of $1/s''$. This implies that for every x, y, the flow value $f_{\mu^*,\nu}(x,y)$ is of the form $i/(s's'')$ for some integer $0\le i\le s's''$. Consequently, for every x, y we have two cases:

- If $f_{\mu^*,\nu}(x,y) > 0$, then necessarily $f_{\mu^*,\nu}(x,y)\ge 1/s^2$, and hence, recalling that $\epsilon = 1/\log s$, we have $(f_{\mu^*,\nu}(x,y))^{\epsilon}\ge s^{2/\log s} = 4$. Hence, $(f_{\mu^*,\nu}(x,y))^{1-\epsilon}\le 4f_{\mu^*,\nu}(x,y)$.

- If $f_{\mu^*,\nu}(x,y) = 0$, then trivially $(f_{\mu^*,\nu}(x,y))^{1-\epsilon}\le 4f_{\mu^*,\nu}(x,y)$.

Therefore,

$$\left(\sum_{(x,y)\in X\times X} f^{1-\varepsilon}_{\mu^*,\nu}(x,y)\cdot\|x-y\|_1^{1-\varepsilon}\right)^{1/(1-\varepsilon)} \le O(1)\left(\sum_{(x,y)\in X\times X} f_{\mu^*,\nu}(x,y)\cdot\|x-y\|_1^{1-\varepsilon}\right)^{1/(1-\varepsilon)}.$$

Continuing the chain of upper bounds, by Equation (6.5),

$$\left(\sum_{(x,y)\in X\times X} f_{\mu^*,\nu}(x,y)\cdot\|x-y\|_1^{1-\varepsilon}\right)^{1/(1-\varepsilon)} \le O(\log s)\left(\sum_{(x,y)\in X\times X} f_{\mu^*,\nu}(x,y)\cdot t'(x,y)\right)^{1/(1-\varepsilon)}.$$

Since the flow $f_{\mu^*,\nu}$ is optimal with respect to the distance $t'(\cdot,\cdot)$,

$$\left(\sum_{(x,y)\in X\times X} f_{\mu^*,\nu}(x,y)\cdot t'(x,y)\right)^{1/(1-\varepsilon)} \le \left(\sum_{(x,y)\in X\times X} f^*_{\mu^*,\nu}(x,y)\cdot t'(x,y)\right)^{1/(1-\varepsilon)}.$$

By Equation (6.5),

$$\left(\sum_{(x,y)\in X\times X} f^*_{\mu^*,\nu}(x,y)\cdot t'(x,y)\right)^{1/(1-\varepsilon)} \le O(\log s)\left(\sum_{(x,y)\in X\times X} f^*_{\mu^*,\nu}(x,y)\cdot\|x-y\|_1^{1-\varepsilon}\right)^{1/(1-\varepsilon)}.$$

Finally, by the concavity of $(\cdot)^{1-\varepsilon}$, and since $\sum f^*_{\mu^*,\nu}(x,y) = 1$, we have

$$\left(\sum_{(x,y)\in X\times X} f^*_{\mu^*,\nu}(x,y) \cdot \|x-y\|_1^{1-\varepsilon} \right)^{1/(1-\varepsilon)} \leq \cdot W_1(\mu^*,\nu).$$

Following the chain of upper bounds, we see that

$$W_1(\mu',\nu) \leq O(\log^2 s) \cdot W_1(\mu^*,\nu),$$

as was to be shown. □

6.5.6 Random Model

Proof of Theorem 6.3.6. **Quadtree.** For every $k = 1,\ldots,s$, let H_k be the smallest hypercube in the quadtree that contains both x_k and y_k. (Note that H_k is a random variable, determined by the initial random shift in the Quadtree construction.) In order for Quadtree to correctly identify μ_i as the nearest neighbor of ν, every H_k must not contain any additional points from X. Otherwise, if say H_1 contains a point $x' \neq x_1$, the W_1 distance on the quadtree from ν to μ_i is equal to its distance to the uniform distribution over $\{x', x_2, \ldots, x_s\}$. Since the points in X are chosen uniformly i.i.d. over $\mathcal{S}^{d-1}$, the probability of the above event, and thus the success probability of Quadtree, is upper bounded by $\mathbb{E}[(1-V)^{N-s}]$, where $V = \text{volume}(\cup_{k=1}^{s} H_k \cap \mathcal{S}^{d-1})$. This V is a random variable whose distribution depends only on d, s, ϵ, and is independent of N. Thus the success probability decays exponentially with N.

Flowtree. On the other hand, suppose that each H_k contains no other points from $\{x_1,\ldots,x_s\}$ other than x_k (but is allowed to contain any other points from X). This event guarantees that the optimal flow on the tree between μ_i and ν is the planted perfect matching, i.e., the true optimal flow, and thus the estimated Flowtree distance between them *equals* $W_1(\mu_i,\nu)$. This guarantees that Flowtree recovers the planted nearest neighbor, and this event depends only on d, s, ϵ, and is independent of N. □

6.6 Appendix: Additional Experiments

6.6.1 Additional Sinkhorn and ACT Experiments

Number of iterations. Both ACT and Sinkhorn are iterative algorithms, and the number of iterations is a parameter to set. Our main experiments use ACT with 1 iteration and Sinkhorn with 1 or 3 iterations. The next experiments motivate these choices. Figures 6-11(a)–(c) depict the accuracy and running time of ACT-1, ACT-7, Sinkhorn-1, Sinkhorn-3 and Sinkhorn-5 on each of our datasets.[9] It can be seen that for both algorithms, increasing the number of iterations beyond the settings used in Section 6.4 yields comparable accuracy with a slower running time. Therefore in Section 6.4 we restrict our evaluation to ACT-1, Sinkhorn-1 and Sinkhorn-3. We also remark that in the pipeline experiments, we have evaluated Sinkhorn with up to 9 iterations. In those experiments too, the best results are achieved with either 1 or 3 iterations

Sinkhorn regularization parameter. Sinkhorn has a regularization parameter λ that needs to be tuned per dataset. We set $\lambda = \eta \cdot M$, where M is the maximum value in the cost matrix (of the currently evaluated pair of distributions), and tune η. In all of our three datasets the optimal setting is $\eta = 30$, which is the setting we use in Section 6.4. As an example, Figure 6-11(d) depicts the 1-NN accuracy (y-axis) of Sinkhorn-1 per η (x-axis).

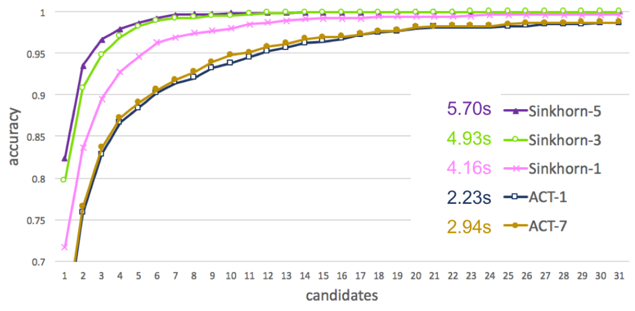
6.6.2 Additional Pipeline Results

The next tables summarize the running times and parameters settings of all pipelines considered in our experiments (whereas the main text focuses on pipelines that start with Quadtree, since it is superior as a first step to Mean and Overlap). The listed parameters are the number of output candidates of each step in the pipeline.

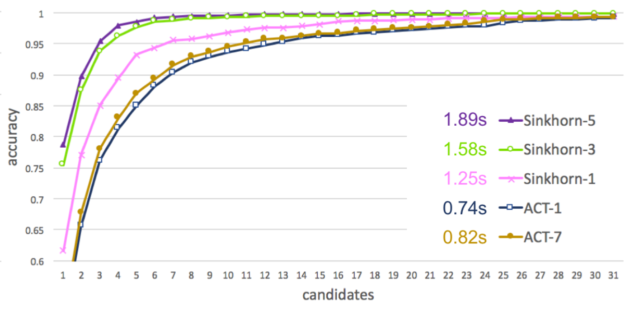
In the baseline pipelines, parameters are tuned to achieve optimal performance (i.e., minimize the running time while attaining the recall goal on at least 90% of the queries). The details of the tuning procedure are as follows. For all pipelines we use the same random subset of 300 queries for tuning. Suppose the pipeline has ℓ algorithms. For $i = 1, \dots, \ell$,

[9]ACT-1 and ACT-7 are the settings reported in [AM19].

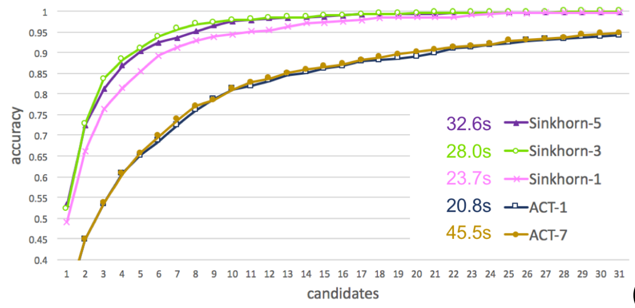
Figure 6-11: Additional Sinkhorn and ACT experiments



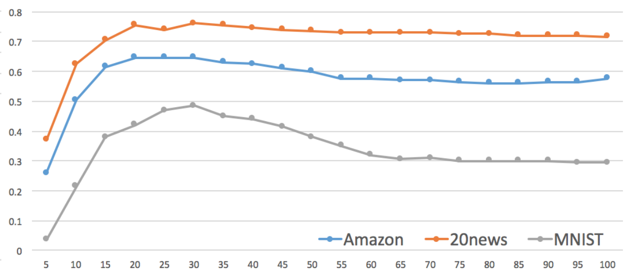
(a) 20news dataset



(b) Amazon dataset



(c) MNIST dataset



(d) 1-NN accuracy of Sinkhorn-1 with varying regularization

let c_i the output number of candidates of the ith algorithm in the pipeline. Note that c_ℓ always equals either 1 or 5, according to the recall goal of the pipeline, so we need to set $c_1, \ldots, c_{\ell-1}$. Let p_1 be the recall@1 accuracy of the first algorithm in the pipeline. Namely, p_1 is the fraction of queries such that the top-ranked c_1 candidates by the first algorithm contain the true nearest neighbor. We calculate 10 possible values of c_1, corresponding to $p_1 \in \{0.9, 0.91, \ldots, 0.99\}$. We optimize the pipeline by a full grid search over those values of c_1 and all possible values of $c_2, \ldots, c_{\ell-1}$.

When introducing Flowtree into a pipeline as an intermediate method, we do not re-optimize the parameters, but rather set its output number of candidates to the maximum between 10 and twice the output number of candidates of the subsequent algorithm in the pipeline. Re-optimizing the parameters could possibly improve results.

Pipeline methods	**Candidates**	**Time**
Mean, Sinkhorn-1, Exact	1476, 11, 1	0.543
Mean, Sinkhorn-3, Exact	1476, 5, 1	0.598
Mean, R-WMD, Exact	1850, 28, 1	0.428
Mean, ACT-1, Exact	1677, 14, 1	0.420
Overlap, Sinkhorn-1, Exact	391, 6, 1	0.610
Overlap, Sinkhorn-3, Exact	391, 5, 1	0.691
Overlap, R-WMD, Exact	576, 14, 1	0.367
Overlap, ACT-1, Exact	434, 10, 1	0.429
Quadtree, Sinkhorn-1, Exact	295, 5, 1	0.250
Quadtree, Sinkhorn-3, Exact	227, 3, 1	0.248
Quadtree, R-WMD, Exact	424, 12, 1	0.221
Quadtree, ACT-1, Exact	424, 8, 1	0.236

Table 6.4: Recall@1, no Flowtree.

Pipeline methods	**Candidates**	**Time**
Mean, **Flowtree**, Sinkhorn-1, Exact	1850, 10, 5, 1	0.089
Mean, **Flowtree**, Sinkhorn-3, Exact	1677, 10, 4, 1	0.077
Mean, **Flowtree**, R-WMD, Exact	2128, 48, 24, 1	0.242
Mean, **Flowtree**, ACT-1, Exact	2128, 20, 10, 1	0.138
Overlap, **Flowtree**, Sinkhorn-1, Exact	489, 10, 5, 1	0.087
Overlap, **Flowtree**, Sinkhorn-3, Exact	576, 10, 3, 1	0.076
Overlap, **Flowtree**, R-WMD, Exact	576, 28, 14, 1	0.173
Overlap, **Flowtree**, ACT-1, Exact	576, 16, 8, 1	0.119
Quadtree, **Flowtree**, Sinkhorn-1, Exact	424, 10, 5, 1	0.074
Quadtree, **Flowtree**, Sinkhorn-3, Exact	424, 10, 3, 1	0.059
Quadtree, **Flowtree**, R-WMD, Exact	424, 22, 11, 1	0.129
Quadtree, **Flowtree**, ACT-1, Exact	424, 16, 8, 1	0.104
Mean, **Flowtree**, Exact	1850, 9, 1	0.105
Overlap, **Flowtree**, Exact	489, 9, 1	0.100
Quadtree, **Flowtree**, Exact	424, 9, 1	0.092

Table 6.5: Recall@1, with Flowtree.

Pipeline methods	**Candidates**	**Time**
Mean, Sinkhorn-1	1476, 5	0.464
Mean, Sinkhorn-3	1476, 5	0.549
Mean, R-WMD, Exact	1850, 28, 5	0.426
Mean, ACT-1, Exact	1677, 14, 5	0.423
Overlap, Sinkhorn-1	391, 5	0.560
Overlap, Sinkhorn-3	391, 5	0.650
Overlap, R-WMD, Exact	576, 14, 5	0.368
Overlap, ACT-1, Exact	434, 10, 5	0.428
Quadtree, Sinkhorn-1	295, 5	0.222
Quadtree, Sinkhorn-3	227, 5	0.200
Quadtree, R-WMD, Exact	424, 11, 5	0.216
Quadtree, ACT-1, Exact	424, 7, 5	0.222

Table 6.6: Recall@5, no Flowtree.

Pipeline methods	**Candidates**	**Time**
Mean, **Flowtree**, Sinkhorn-1	1850, 10, 5	0.046
Mean, **Flowtree**, Sinkhorn-3	1476, 10, 5	0.043
Mean, **Flowtree**, R-WMD, Exact	2128, 48, 24, 5	0.237
Mean, **Flowtree**, ACT-1	2128, 10, 5	0.048
Overlap, **Flowtree**, Sinkhorn-1	391, 10, 5	0.042
Overlap, **Flowtree**, Sinkhorn-3	391, 10, 5	0.044
Overlap, **Flowtree**, R-WMD, Exact	576, 28, 14, 5	0.173
Overlap, **Flowtree**, ACT-1	576, 10, 5	0.046
Quadtree, **Flowtree**, Sinkhorn-1	424, 10, 5	0.033
Quadtree, **Flowtree**, Sinkhorn-3	424, 10, 5	0.034
Quadtree, **Flowtree**, ACT-1	424, 10, 5	0.029
Mean, **Flowtree**	2128, 5	0.043
Overlap, **Flowtree**	576, 5	0.039
Quadtree, **Flowtree**	645, 5	0.027
Quadtree, **Flowtree**, R-WMD, Exact	424, 22, 11, 5	0.131
Quadtree, **Flowtree**, ACT-1, Exact	424, 16, 8, 5	0.103

Table 6.7: Recall@5, with Flowtree.

Bibliography

[AC09] Nir Ailon and Bernard Chazelle, *The fast johnson–lindenstrauss transform and approximate nearest neighbors*, SIAM J. Comput. **39** (2009), no. 1, 302–322.

[Ach03] Dimitris Achlioptas, *Database-friendly random projections: Johnson-lindenstrauss with binary coins*, J. Comput. Syst. Sci. **66** (2003), no. 4, 671–687.

[ADD$^+$93] Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares, *On sparse spanners of weighted graphs*, Discrete & Computational Geometry **9** (1993), no. 1, 81–100.

[AI06] Alexandr Andoni and Piotr Indyk, *Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions*, 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings, 2006, pp. 459–468.

[AI17] ______, *Nearest neighbors in high-dimensional spaces*, CRC Handbook of Discrete and Computational Geometry (2017).

[AIK08] Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer, *Earth mover distance over high-dimensional spaces*, Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2008, pp. 343–352.

[AIR18] Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn, *Approximate nearest neighbor search in high dimensions*, arXiv preprint arXiv:1806.09823 (2018).

[AK16] Noga Alon and Bo'az Klartag, *Optimal compression of approximate euclidean distances*, arXiv preprint arXiv:1610.00239. In IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS 2017) (2016).

[AKR18] Alexandr Andoni, Robert Krauthgamer, and Ilya Razenshteyn, *Sketching and embedding are equivalent for norms*, SIAM Journal on Computing **47** (2018), no. 3, 890–916.

[Alo03] Noga Alon, *Problems and results in extremal combinatoricsâĂŤi*, Discrete Mathematics **273** (2003), no. 1âĂŞ3, 31 – 53, EuroComb'01.

[AM19] Kubilay Atasu and Thomas Mittelholzer, *Linear-complexity data-parallel earth moverâĂŹs distance approximations*, International Conference on Machine Learning, 2019, pp. 364–373.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy, *The space complexity of approximating the frequency moments*, Journal of Computer and system sciences **58** (1999), no. 1, 137–147.

[ANN15] Alexandr Andoni, Assaf Naor, and Ofer Neiman, *Snowflake universality of wasserstein spaces*, arXiv preprint arXiv:1509.08677 (2015).

[Bar96] Yair Bartal, *Probabilistic approximation of metric spaces and its algorithmic applications*, Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on, IEEE, 1996, pp. 184–193.

[Bar98] ———, *On approximating arbitrary metrices by tree metrics.*, STOC, vol. 98, 1998, pp. 161–168.

[BDI$^+$20] Arturs Backurs, Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner, *Scalable nearest neighbor search for optimal transport*, International Conference on Machine Learning, 2020.

[BGPW16] Mark Braverman, Ankit Garg, Denis Pankratov, and Omri Weinstein, *Information lower bounds via self-reducibility*, Theory of Computing Systems **59** (2016), no. 2, 377–396.

[BI14] Artūrs Bačkurs and Piotr Indyk, *Better embeddings for planar earth-mover distance over sparse sets*, Proceedings of the thirtieth annual symposium on Computational geometry, 2014, pp. 280–289.

[BIO$^+$19] Arturs Backurs, Piotr Indyk, Krzysztof Onak, Baruch Schieber, Ali Vakilian, and Tal Wagner, *Scalable fair clustering*, International Conference on Machine Learning, 2019, pp. 405–413.

[BIW19] Arturs Backurs, Piotr Indyk, and Tal Wagner, *Space and time efficient kernel density estimation in high dimensions*, Advances in Neural Information Processing Systems, 2019, pp. 15799–15808.

[Bou86] Jean Bourgain, *The metrical interpretation of superreflexivity in banach spaces*, Israel Journal of Mathematics **56** (1986), no. 2, 222–230.

[BR14] Mark Braverman and Anup Rao, *Information equals amortized communication*, IEEE Transactions on Information Theory **60** (2014), no. 10, 6058–6069.

[Bro97] Andrei Z Broder, *On the resemblance and containment of documents*, Compression and Complexity of Sequences 1997. Proceedings, IEEE, 1997, pp. 21–29.

[BW18] Ainesh Bakshi and David Woodruff, *Sublinear time low-rank approximation of distance matrices*, Advances in Neural Information Processing Systems, 2018, pp. 3786–3796.

[CCFC02] Moses Charikar, Kevin Chen, and Martin Farach-Colton, *Finding frequent items in data streams*, International Colloquium on Automata, Languages, and Programming, Springer, 2002, pp. 693–703.

[CCG$^+$98] Moses Charikar, Chandra Chekuri, Ashish Goel, Sudipto Guha, and Serge Plotkin, *Approximating a finite metric by a small number of tree metrics*, Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280), IEEE, 1998, pp. 379–388.

[CCK15] Shiri Chechik, Edith Cohen, and Haim Kaplan, *Average distance queries through weighted samples in graphs and metric spaces: High scalability with tight statistical guarantees*, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[CCK18] Edith Cohen, Shiri Chechik, and Haim Kaplan, *Clustering small samples with quality guarantees: Adaptivity with one2all pps*, Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[CG15] John P Cunningham and Zoubin Ghahramani, *Linear dimensionality reduction: Survey, insights, and generalizations*, The Journal of Machine Learning Research **16** (2015), no. 1, 2859–2900.

[Cha02] Moses S Charikar, *Similarity estimation techniques from rounding algorithms*, Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, ACM, 2002, pp. 380–388.

[Che09] Ke Chen, *On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications*, SIAM Journal on Computing **39** (2009), no. 3, 923–947.

[Che15] Shiri Chechik, *Approximate distance oracles with improved bounds*, Proceedings of the forty-seventh annual ACM symposium on Theory of Computing, 2015, pp. 1–10.

[CKR05] Gruia Calinescu, Howard Karloff, and Yuval Rabani, *Approximation algorithms for the 0-extension problem*, SIAM Journal on Computing **34** (2005), no. 2, 358–372.

[CP17] Xue Chen and Eric Price, *Active regression via linear-sample sparsification*, arXiv preprint arXiv:1711.10051 (2017).

[CR09] Emmanuel J Candès and Benjamin Recht, *Exact matrix completion via convex optimization*, Foundations of Computational mathematics **9** (2009), no. 6, 717.

[Cut13] Marco Cuturi, *Sinkhorn distances: Lightspeed computation of optimal transport*, Advances in neural information processing systems, 2013, pp. 2292–2300.

[CW79] J Lawrence Carter and Mark N Wegman, *Universal classes of hash functions*, Journal of computer and system sciences **18** (1979), no. 2, 143–154.

[CW17] Kenneth L Clarkson and David P Woodruff, *Low-rank approximation and regression in input sparsity time*, Journal of the ACM (JACM) **63** (2017), no. 6, 54, (first appeared in STOC'13).

[DG03] Sanjoy Dasgupta and Anupam Gupta, *An elementary proof of a theorem of johnson and lindenstrauss*, Random Structures & Algorithms **22** (2003), no. 1, 60–65.

[DIRW20] Yihe Dong, Piotr Indyk, Ilya P. Razenshteyn, and Tal Wagner, *Learning space partitions for nearest neighbor search*, 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, OpenReview.net, 2020.

[DKW15] Michael Dinitz, Robert Krauthgamer, and Tal Wagner, *Towards resistance sparsifiers*, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA (Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim, eds.), LIPIcs, vol. 40, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015, pp. 738–755.

[DPRV15] Ivan Dokmanic, Reza Parhizkar, Juri Ranieri, and Martin Vetterli, *Euclidean distance matrices: essential theory, algorithms, and applications*, IEEE Signal Processing Magazine **32** (2015), no. 6, 12–30.

[Efr17] A. Efros, *How to stop worrying and learn to love nearest neighbors*, https://nn2017.mit.edu/wp-content/uploads/sites/5/2017/12/Efros-NIPS-NN-17.pdf (2017).

[FKV04] Alan Frieze, Ravi Kannan, and Santosh Vempala, *Fast monte-carlo algorithms for finding low-rank approximations*, Journal of the ACM (JACM) **51** (2004), no. 6, 1025–1041.

[FM88] Peter Frankl and Hiroshi Maehara, *The johnson-lindenstrauss lemma and the sphericity of some graphs*, Journal of Combinatorial Theory, Series B **44** (1988), no. 3, 355–362.

[Fod02] Imola K Fodor, *A survey of dimension reduction techniques*, Tech. report, Lawrence Livermore National Lab., CA (US), 2002.

[FRT04] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar, *A tight bound on approximating arbitrary metrics by tree metrics*, Journal of Computer and System Sciences **69** (2004), no. 3, 485–497.

[FS10] Ohad N Feldheim and Sasha Sodin, *A universality result for the smallest eigenvalues of certain sample covariance matrices*, Geometric And Functional Analysis **20** (2010), no. 1, 88–123.

[GKL03] Anupam Gupta, Robert Krauthgamer, and James R Lee, *Bounded geometries, fractals, and low-distortion embeddings*, 44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings., IEEE, 2003, pp. 534–543.

[GMRS16] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers, *Robust random cut forest based anomaly detection on streams*, International Conference on Machine Learning, 2016, pp. 2712–2721.

[HP11] Sariel Har-Peled, *Geometric approximation algorithms*, no. 173, American Mathematical Soc., 2011.

[HPIM12] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani, *Approximate nearest neighbor: Towards removing the curse of dimensionality*, Theory of Computing **8** (2012), no. 14, 321–350.

[HS93] Liisa Holm and Chris Sander, *Protein structure comparison by alignment of distance matrices*, Journal of molecular biology **233** (1993), no. 1, 123–138.

[HW03] Alan J Hoffman and Helmut W Wielandt, *The variation of the spectrum of a normal matrix*, Selected Papers Of Alan J Hoffman: With Commentary, World Scientific, 2003, pp. 118–120.

[IM98] Piotr Indyk and Rajeev Motwani, *Approximate nearest neighbors: towards removing the curse of dimensionality*, Proceedings of the thirtieth annual ACM symposium on Theory of computing, 1998, pp. 604–613.

[Ind99] Piotr Indyk, *A sublinear time approximation scheme for clustering in metric spaces*, Foundations of Computer Science, 1999. 40th Annual Symposium on, IEEE, 1999, pp. 154–159.

[Ind01] ______, *Algorithmic applications of low-distortion geometric embeddings*, Proceedings 42nd IEEE Symposium on Foundations of Computer Science, IEEE, 2001, pp. 10–33.

[Ind06] ______, *Stable distributions, pseudorandom generators, embeddings, and data stream computation*, Journal of the ACM (JACM) **53** (2006), no. 3, 307–323.

[IRW17] Piotr Indyk, Ilya Razenshteyn, and Tal Wagner, *Practical data-dependent metric compression with provable guarantees*, Advances in Neural Information Processing Systems, 2017, pp. 2614–2623.

[IT03] Piotr Indyk and Nitin Thaper, *Fast image retrieval via embeddings*, 3rd international workshop on statistical and computational theories of vision, vol. 2, 2003, p. 5.

[IVWW19] Piotr Indyk, Ali Vakilian, Tal Wagner, and David P Woodruff, *Sample-optimal low-rank approximation of distance matrices*, COLT, 2019.

[IW17] Piotr Indyk and Tal Wagner, *Near-optimal (euclidean) metric compression*, Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2017, pp. 710–723.

[IW18] ______, *Approximate nearest neighbors in limited space*, Conference On Learning Theory, 2018, pp. 2012–2036.

[IW20] ______, *Optimal (euclidean) metric compression*, in preparation (2020).

[JDJ17a] Jeff Johnson, Matthijs Douze, and Hervé Jégou, *Billion-scale similarity search with gpus*, CoRR **abs/1702.08734** (2017).

[JDJ17b] Jeff Johnson, Matthijs Douze, and Hervé Jégou, *Faiss: A library for efficient similarity search*, https://code.facebook.com/posts/1373769912645926/faiss-a-library-for-efficient-similarity-search/ (2017).

[JDS08] Herve Jegou, Matthijs Douze, and Cordelia Schmid, *Hamming embedding and weak geometric consistency for large scale image search*, European conference on computer vision, Springer, 2008, pp. 304–317.

[JDS11] ______, *Product quantization for nearest neighbor search*, IEEE transactions on pattern analysis and machine intelligence **33** (2011), no. 1, 117–128.

[JL84] William B Johnson and Joram Lindenstrauss, *Extensions of lipschitz mappings into a hilbert space*, Contemporary mathematics **26** (1984), no. 189-206, 1–1.

[JW13] Thathachar S Jayram and David P Woodruff, *Optimal bounds for johnson-lindenstrauss transforms and streaming problems with subconstant error*, ACM Transactions on Algorithms (TALG) **9** (2013), no. 3, 26.

[KK95] Bahman Kalantari and Iraj Kalantari, *A linear-time algorithm for minimum cost flow on undirected one-trees*, Combinatorics Advances, Springer, 1995, pp. 217–223.

[KMN11] Daniel Kane, Raghu Meka, and Jelani Nelson, *Almost optimal explicit johnson-lindenstrauss families*, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Springer, 2011, pp. 628–639.

[KN06] Subhash Khot and Assaf Naor, *Nonembeddability theorems via fourier analysis*, Mathematische Annalen **334** (2006), no. 4, 821–852.

[KOR00] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani, *Efficient search for approximate nearest neighbor in high dimensional spaces*, SIAM Journal on Computing **30** (2000), no. 2, 457–474.

[KSKW15] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger, *From word embeddings to document distances*, International conference on machine learning, 2015, pp. 957–966.

[KT02] Jon Kleinberg and Eva Tardos, *Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields*, Journal of the ACM (JACM) **49** (2002), no. 5, 616–639.

[Kuh55] Harold W Kuhn, *The hungarian method for the assignment problem*, Naval research logistics quarterly **2** (1955), no. 1-2, 83–97.

[LC98] Yann LeCun and Corinna Cortes, *The mnist database of handwritten digits*, 1998.

[LN14] Huy Le Nguyen, *Algorithms for high dimensional data*, Ph.D. thesis, Princeton University, 2014.

[LN17] Kasper Green Larsen and Jelani Nelson, *Optimality of the johnson-lindenstrauss lemma*, 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2017, pp. 633–638.

[LYFC19] Tam Le, Makoto Yamada, Kenji Fukumizu, and Marco Cuturi, *Tree-sliced approximation of wasserstein distances*, arXiv preprint arXiv:1902.00342 (2019).

[Mah11] Michael W Mahoney, *Randomized algorithms for matrices and data*, Foundations and Trends® in Machine Learning **3** (2011), no. 2, 123–224.

[Mat96] Jiří Matoušek, *On the distortion required for embedding finite metric spaces into normed spaces*, Israel Journal of Mathematics **93** (1996), no. 1, 333–344.

[Mat08] ———, *On variants of the johnson–lindenstrauss lemma*, Random Structures & Algorithms **33** (2008), no. 2, 142–156.

[Mat13] Jirı Matoušek, *Lecture notes on metric embeddings*, Tech. report, 2013.

[MMMR18] Sepideh Mahabadi, Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn, *Nonlinear dimension reduction via outer bi-lipschitz extensions*, Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, 2018, pp. 1088–1101.

[MN04] Manor Mendel and Assaf Naor, *Euclidean quotients of finite metric spaces*, Advances in Mathematics **189** (2004), no. 2, 451–494.

[MN06] ______, *Ramsey partitions and proximity data structures*, 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), IEEE, 2006, pp. 109–118.

[MSC$^+$13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, *Distributed representations of words and phrases and their compositionality*, Advances in neural information processing systems, 2013, pp. 3111–3119.

[MW17] Cameron Musco and David P Woodruff, *Sublinear time low-rank approximation of positive semidefinite matrices*, Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on, IEEE, 2017, pp. 672–683.

[MWY13] Marco Molinaro, David P Woodruff, and Grigory Yaroslavtsev, *Beating the direct sum theorem in communication complexity with implications for sketching*, Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2013, pp. 1738–1756.

[Nao17] Assaf Naor, *Probabilistic clustering of high dimensional norms*, Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2017, pp. 690–709.

[New91] Ilan Newman, *Private vs. common random bits in communication complexity*, Information processing letters **39** (1991), no. 2, 67–71.

[NN19] Shyam Narayanan and Jelani Nelson, *Optimal terminal dimensionality reduction in euclidean space*, Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, 2019, pp. 1064–1069.

[NS07] Assaf Naor and Gideon Schechtman, *Planar earthmover is not in l_1*, SIAM Journal on Computing **37** (2007), no. 3, 804–826.

[PS89] David Peleg and Alejandro A Schäffer, *Graph spanners*, Journal of graph theory **13** (1989), no. 1, 99–116.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning, *Glove: Global vectors for word representation*, Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[RTG00] 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** (2000), no. 2, 99–121.

[RV10] Mark Rudelson and Roman Vershynin, *Non-asymptotic theory of random matrices: extreme singular values*, Proceedings of the International Congress of Mathematicians 2010 (ICM 2010) (In 4 Volumes) Vol. I: Plenary Lectures and Ceremonies Vols. II–IV: Invited Lectures, World Scientific, 2010, pp. 1576–1602.

[Sam84] Hanan Samet, *The quadtree and related hierarchical data structures*, ACM Computing Surveys (CSUR) **16** (1984), no. 2, 187–260.

[SDI06] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk, *Nearest-neighbor methods in learning and vision: theory and practice (neural information processing)*, The MIT press, 2006.

[SH09] Ruslan Salakhutdinov and Geoffrey Hinton, *Semantic hashing*, International Journal of Approximate Reasoning **50** (2009), no. 7, 969–978.

[SVM14] Carlos Oscar Sánchez Sorzano, Javier Vargas, and A Pascual Montano, *A survey of dimensionality reduction techniques*, arXiv preprint arXiv:1403.2877 (2014).

[SZ03] Josef Sivic and Andrew Zisserman, *Video google: A text retrieval approach to object matching in videos*, null, IEEE, 2003, p. 1470.

[Tao10] Terence Tao, *254a, notes 3a: Eigenvalues and sums of hermitian matrices*, https://terrytao.wordpress.com/2010/01/12/254a-notes-3a-eigenvalues-and-sums-of-hermitian-matrices, 2010.

[TDSL00] Joshua B Tenenbaum, Vin De Silva, and John C Langford, *A global geometric framework for nonlinear dimensionality reduction*, science **290** (2000), no. 5500, 2319–2323.

[Tho76] RC Thompson, *The behavior of eigenvalues and singular values under perturbations of restricted rank*, Linear Algebra and its Applications **13** (1976), no. 1-2, 69–78.

[TZ05] Mikkel Thorup and Uri Zwick, *Approximate distance oracles*, Journal of the ACM (JACM) **52** (2005), no. 1, 1–24.

[TZ12] Mikkel Thorup and Yin Zhang, *Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation*, SIAM Journal on Computing **41** (2012), no. 2, 293–331.

[Vil03] Cédric Villani, *Topics in optimal transportation*, no. 58, American Mathematical Soc., 2003.

[Wag20] Tal Wagner, *Eccentricity heuristics through sublinear analysis lenses*, 1st Symposium on Algorithmic Principles of Computer Systems, APOCS@SODA 2020, Salt Lake City, UT, USA, January 8, 2020, SIAM, 2020, pp. 75–89.

[WGKM18] Tal Wagner, Sudipto Guha, Shiva Kasiviswanathan, and Nina Mishra, *Semi-supervised learning on data streams via temporal label propagation*, International Conference on Machine Learning, 2018, pp. 5095–5104.

[WLKC16] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang, *Learning to hash for indexing big data: a survey*, Proceedings of the IEEE **104** (2016), no. 1, 34–57.

[WN12] Christian Wulff-Nilsen, *Approximate distance oracles with improved preprocessing time*, Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms, SIAM, 2012, pp. 202–208.

[Woo14] David P Woodruff, *Sketching as a tool for numerical linear algebra*, Foundations and Trends® in Theoretical Computer Science **10** (2014), no. 1–2, 1–157.

[WS06] Kilian Q Weinberger and Lawrence K Saul, *Unsupervised learning of image manifolds by semidefinite programming*, International journal of computer vision **70** (2006), no. 1, 77–90.

[WTF09] Yair Weiss, Antonio Torralba, and Rob Fergus, *Spectral hashing*, Advances in neural information processing systems, 2009, pp. 1753–1760.

[WYX$^+$18] Lingfei Wu, Ian EH Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar, and Michael J Witbrock, *Word mover's embedding: From word2vec to document embedding*, arXiv preprint arXiv:1811.01713 (2018).

[WZS$^+$18] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al., *A survey on learning to hash*, IEEE Transactions on Pattern Analysis and Machine Intelligence **40** (2018), no. 4, 769–790.

[YCC$^+$19] Mikhail Yurochkin, Sebastian Claici, Edward Chien, Farzaneh Mirzazadeh, and Justin Solomon, *Hierarchical optimal transport for document representation*, arXiv preprint arXiv:1906.10827 (2019).