Machine Learning: Metrics and Embeddings

Timothy Chu

August 19

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Gary Miller, Chair
Anupam Gupta
Daniel Sleator
Satish Rao, UC Berkeley

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

Keywords: Machine learning, Computational Geometry, Clustering, Kernels, Group Theory, Theoretical Foundations

To my mother, who supported me since the day I was born.

Abstract

In this thesis, we analyze new theories of clustering, one of the most fundamental tasks in machine learning. We use methods drawing from multiple disciplines, including metric embeddings, spectral algorithms, and group representation theory.

- We propose a metric that adapts to the shape of data, and show how to quickly compute it. These metrics may be useful for improving k-means clustering methods.
- We build a spectral partition method with provable theoretical guarantees. This may lead to more theoretically principled spectral clustering methods, as existing methods do not have any such guarantees. Spectral clustering is one of the most popular methods of clustering.
- We classify all Manhattan distance kernels. Kernel methods are one of the oldest and most established methods of clustering data. This result is a Manhattan distance analog of one of the fundamental results on machine learning kernels.

Each of these contributions answers natural questions in machine learning theory. We develop multidisciplinary tools from disciplines ranging from linear algebra to group theory, and combine these with key ideas from metric embeddings and computational geometry.

Acknowledgments

First and foremost, I'd like to thank my advisor Dr. Miller, who among many things encouraged me to think about unorthodox problems.

I would like to thank my thesis committee: Professor Gary Miller (my aforementioned advisor), Professor Anupam Gupta, Professor Satish Rao, and Professor Danny Sleator. I would also like to thank Deborah Cavlovich, for her tireless hard work in helping PhD students at Carnegie Mellon.

Research is always better with a great group of collaborators. I've had a great time working with Professor Donald Sheehy, Professor Noel Walkington, Professor Sushant Sachdeva, Professor Josh Alman, Aaron Schild, Mark Sellke, Shyam Narayanan, Alex Wang, Yu Gao, Saurabh Sawlani, Junxing Wang, Jakub Pachocki, Michael Cohen (now passed), Zhao Song, Nathan Pinsker, and Alex X. Chen. I've also gotten considerable math help from Amol Aggarwal and David Yang.

I would like to thank Professor Mikkel Thorup and the graduate students at BARC for a wonderful research experience in Copenhagen, Professor Richard Peng for early mentorship and the opportunity to be included on some cool research projects early in my PhD career, and Professor Lorenzo Orecchia for a research opportunity when I was an undergrad at MIT.

Graduate school wouldn't be the same without a supportive group of friends and loved ones. I've had many great memories with my friends at Carnegie Mellon, including Goran Zuzic, Pedro Paredes, Ainesh Bakshi, Travis Hance, Rajesh Jayaram, Roie Levin, Jonathan Laurent, and Jason Li. Finally, I would like to thank my parents Iris Li and Weishang Chu, my brother Joseph Chu, and Liting Chen for encouraging me to finish my PhD when times were hard.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern machine learning has seen unprecedented practical success in fields ranging from language processing, image recognition, data clustering, and more. In this thesis, we aim to understand the principles behind existing machine learning methods, particularly for the task of fundamental ML task of clustering. We leverage classical ideas from computational geometry and metric embeddings to gain deeper insight into machine learning methods like clustering, kernel methods, natural language processing (transformers), and data mining on graphs.

1.0 Quick Preliminaries: Clustering Definitions and Geometry in ML

Clustering consists of taking unlabeled data and categorizing it into different clusters. As an example, imagine that one measures the size and weight of k different types of bacteria in a cell culture, and wants to determine what type of bacteria each individual measurement comes from. This task is known as k-way clustering.

One classical insight in machine learning is that data measurements can be represented as points in geometric space, and that information about the data set can be inferred from the geometric configuration of the resulting data points. For example, if one measures the size and weight of a single bacterium in a cell culture, one can plot this as a two dimensional point with x coordinate equal to the size, and y coordinate equal to the weight. The representation of data as points in geometric space is a classical one in machine learning, and is widely used for most clustering methods. This geometric representation of data is one reason computational geometry can be expected to lead to a greater understanding of machine learning methods.

1.1 Thesis Overview

In this thesis, we examine metric-based clustering, spectral clustering on large data sets, and kernel methods: each of these are widely used in practice. We prove theorems about the principles underlying each method. Once we have a greater understanding of these principles, we then present new machine learning methods based on these principles, where these new methods

have theoretical guarantees backing them. The tools in each of our explorations rely heavily on metrics and embeddings. For some applications, we supplement this with insights from spectral graph algorithms and group theory.

Chapter 1.2 of this thesis is about group theory in kernels and natural language processing. Chapter 1.3 is about clustering with data-sensitive distances. Chapter 1.4 is about a variant of classical spectral clustering backed by additional mathematical principles and provable guarantees on cluster quality. Chapter 1.5 is about spectral graph theory can create fast algorithms for classical clustering methods.

1.2 Metric Embeddings and Group Theory in Kernels and Natural Language Processing

Smola developed kernel methods in the 1990s to improve machine learning performance [261]. The key idea is that given a set of data in d dimensional space, one can find a function F to map this data into an infinite dimensional space in which natural clusters can be more easily discovered. Then, one can perform clustering methods on this new space. The key insight is that clustering in the new space can be done quickly and efficiently given a fast computational oracle for $\langle F(x), F(y)\rangle$ for any x, y, even when F itself is not known or not computable. In some settings, F is designed so that an easily computable function $f : \mathbb{R}^d \to \mathbb{R}$ satisfies $f(\|x - y\|_p) = \langle F(x), F(y)\rangle$. The set of functions f for which such an F exists, is known as the set of **positive definite ℓ_p kernel functions.** When $p = 2$, we refer to such kernels as **positive definite Euclidean kernels**, and when $p = 1$, we refer to such functions as **positive definite Manhattan kernels**. The function e^{-x^2} is an example of a positive definite Euclidean kernel, and the function $e^{-|x|}$ is a positive definite Manhattan kernel [35, 251].

Kernel methods have been widely used ever since their introduction by Smola, and have seen practical applications in almost all areas of machine learning. They have also been shown to have deep connections to neural networks [108].

Separately, the field of natural language processing has undergone a revolution with the rise of deep neural networks. One of the most powerful neural network architectures for natural language processing is the *transformer* [185]. Despite impressive results in translation, audio transcription, and speech synthesis, the theory behind transformers is not well understood.

In this dissertation, we develop novel bridges between the mathematical discipline of group representation theory and metric embeddings, and apply it to kernel methods and natural language processing. Group representation theory has previously been used throughout complexity theory, algorithm design, combinatorics, algebraic geometry, chemistry, quantum physics, and more. This theory is arguably the foundation of all of modern algebra [118, 129].

In our work, we will show the following results:

1. A function is a positive definite Manhattan kernel if and only if it is a completely monotone function. Positive definite Manhattan kernels are widely used across machine learning; one example is the Laplace kernel which is widely used in machine learning for chemistry. This completes the theory of Manhattan metric kernels initiated by Schoenberg in 1942, and serves as a Manhattan metric analog of the fundamental theory of positive definite

Euclidean kernels as pioneered by Smola [254, 261, 262].

2. A function transforms Manhattan distances to Manhattan distances if and only if it is a Bernstein function. This work completes the theory of Manhattan to Manhattan metric transforms initiated by Assouad in 1980 [35].
3. A function applied entry-wise to any square matrix of rank r always results in a matrix of rank $< 2^{r-1}$ if and only if it is a polynomial of sufficiently low degree. This gives a converse to a the polynomial method in algorithm design [1]. Moreover, this result will also illuminate the theory behind key steps of transformers in natural language processing.

Our work combines finite group representation theory, the study of group symmetries (which is widely used across physics and chemistry), with metric embedding results and linear algebra. To my knowledge, this is one of the first results that uses finite group representation theory with metric embeddings to understand machine learning. We elaborate on this in Chapter 2.

1.3 Data-Sensitive Distances in Clustering

In this chapter, we consider a metric-based approach for clustering. We consider a suite of distance measures, or metrics, between data points embedded in geometric space. Ideally, these metrics should consider two points in the same cluster to be close, even if their Euclidean distance is long. Likewise, these metrics should consider two points in different clusters to be far apart, even if their Euclidean distance is short. We call metrics with this property **data-sensitive**.

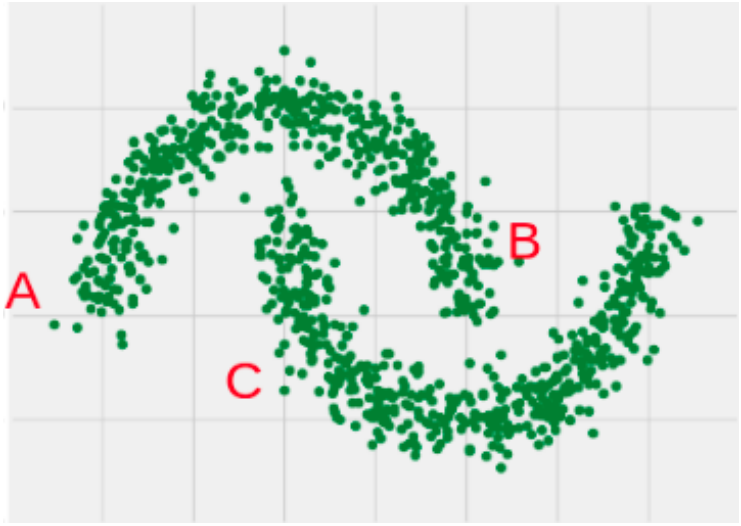

Figure 1.1: In this picture, $A, B,$ and C are data points in a two dimensional data set. Data-sensitive metrics have the property that the distance between A and B is short, while the distance between A and C is long.

In this thesis, we explore how to efficiently compute two previously considered data-sensitive

[1]The polynomial method in algorithm design states that applying a low degree polynomial to the entries of a low rank matrix results in another low rank matrix

metrics: the edge-squared metric [289], and the nearest-neighbor metric [94]. Computing these metrics efficiently allows clustering to be performed on them in practice.

The results we show on clustering are:

1. A $(1+\epsilon)$-spanner of the nearest neighbor metric, a notable example of a data sensitive distance, can be computed in nearly linear time in constant dimension. This will allow us to perform clustering using these metrics in a more efficient fashion than was previously known.
2. The nearest neighbor metric and edge-squared metric are exactly identical in all cases.

Prior to these results, it was not known how to compute either metric efficiently, and it was not even suspected that these two metrics were secretly identical. For this work, we make heavy use of the Kirszbraun theorem from metric embedding and computational geometry, also known as the Lipschitz extension theorem. We go into detail about these results in Chapter 3.

1.4 Spectral Clustering in Large Datasets

One of the most widely used clustering algorithms is known as spectral clustering [230]. We explore how it clusters data as the number of samples grows large. In this perspective, we make the assumption that the points are drawn from a probability density function. We show that spectral clustering has a key deficiency, and then devise a spectral clustering variant that addresses this deficiency.

1. Given a large set of data drawn from a well-chosen and simple Lipschitz probability density, the most popular version of spectral clustering will converge to a bad cut of the data set, with unboundedly bad sparsity guarantees.
2. Given a large data set drawn from any Lipschitz probability density, we present a new variant of spectral clustering that will converge to a cut of the data set with provable guarantees on the cut sparsity.

Traditional spectral clustering is one of the most robust and widely used methods of clustering data. However, it has a major theoretical issue. When a large number of points are drawn from a probability density, then spectral clustering converges to something called a **spectral sweep cut** of the underlying density. However, this spectral sweep cut can partition the probability density poorly, and thus spectral clustering can converge to a poor partition of the data.

To remedy this, we build a new variant of the classical spectral clustering method (splitting data into two clusters), and show that it will, in the large data regime, generate a cut of the underlying probability density with provable guarantees on the cut sparsity. At a high level, this means that the generated cut cuts through low surface area while splitting the density into two pieces of relatively high volume.

Spectral clustering itself is based on embedding data into new geometric spaces, and thus the use of geometric embeddings features prominently in our work. We will prove our variant of spectral clustering has good sparsity guarantees through new extensions of the Cheeger and Buser inequalities, applied to probability densities. Cheeger and Buser inequalities are famous inequalities traditionally used in the graph and manifold setting, with the goal of relating fundamental eigenvalues with sparsest cut in each of these settings. We will go into detail about

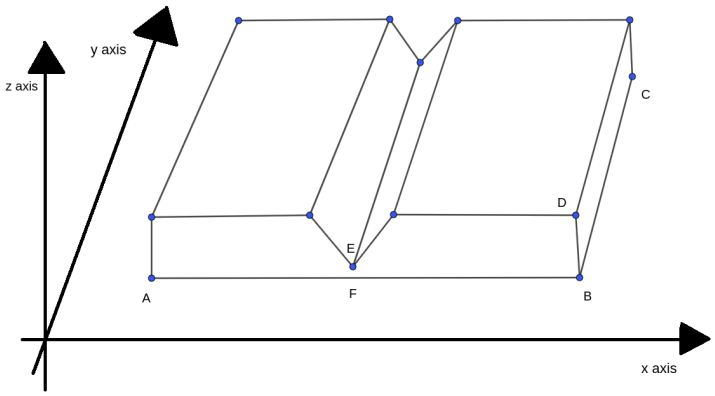

Figure 1.2: We show a probability density function with support on the xy plane, and value on the z coordinate. Traditional spectral clustering converges to a cut parallel to the x axis, instead of the cut along the valley. In this figure, BC is a constant factor longer than AB, and the height of the lowest point in the valley is $1/(BC)^{5/2}$.

spectral clustering in Chapter 4

1.5 Spectral Graph Theory in Machine Learning

Graphs are often used in machine learning to encode the relationship between data points. There is a long history of applying graph theory algorithms on such graphs to perform machine learning primitives such as clustering, semi-supervised learning, and more.

In Chapter 5, we develop spectral graph algorithms and complexity results on graphs arising from points in geometric space. Since data is often represented as points in geometric space, these results can be naturally applied to the above-mentioned machine learning questions. Our results can be used to speed up certain forms of spectral clustering, semi-supervised learning, and more.

Results we will show on geometric spectral graph theory include:

1. For certain complete geometric graphs arising from n points in d dimensions, we can compute a spectral sparsifier of the graph in $O(nd)$ time, where n is the number of points and d is the dimension.
2. For a large class of complete geometric graphs, it is impossible to solve the Laplacian to high accuracy in sub-quadratic time (assuming the Strong Exponential Time Hypothesis, or SETH).

These results have applications to machine learning and physical simulations. In particular, the first algorithmic result will enable us to perform spectral clustering on these geometric graphs more efficiently, while the second result shows that the widely used fast-multipole method for computing forces between electrostatic charges in physics, has a run-time with unavoidable exponential dependence on dimension. We elaborate on these results in Chapter 5.

Chapter 2

Metric Embeddings and Group Theory in Kernels and Natural Language Processing

In this chapter, we develop a new technique which we call representation theory of the real hyperrectangle, which describes how to compute the eigenvectors and eigenvalues of certain matrices arising from hyperrectangles. We show that these matrices arise naturally when analyzing a number of different algorithmic tasks such as kernel methods, neural network training, natural language processing, and the design of algorithms using the polynomial method. We then use our new technique along with these connections to prove several new structural results in these areas, including:

1. A function is a positive definite Manhattan kernel if and only if it is a completely monotone function. These kernels are widely used across machine learning; one example is the Laplace kernel which is widely used in machine learning for chemistry.
2. A function transforms Manhattan distances to Manhattan distances if and only if it is a Bernstein function. This completes the theory of Manhattan to Manhattan metric transforms initiated by Assouad in 1980.
3. A function applied entry-wise to any square matrix of rank r always results in a matrix of rank $< 2^{r-1}$ if and only if it is a polynomial of sufficiently low degree. This gives a converse to a key lemma used by the polynomial method in algorithm design.

Our work includes a sophisticated combination of techniques from different fields, including metric embeddings, the polynomial method, and group representation theory.

2.1 Introduction

In this chapter, we introduce a new analytic technique we call 'representation theory of the real hyperrectangle'. At a high level, this technique gives simple expressions for computing the eigenvectors and eigenvalues of a large class of matrices which are defined in terms of hyperrectangles (high-dimensional analogues of rectangles). We will see that this class of matrices arises frequently in the study of linear algebraic tools for modern machine learning and algorithm design. As a result, we use our new technique to prove a number of new structural results in these

areas.

Before getting into the representation theory of the real hyperrectangle in more detail, we first describe our three main applications. First, in Section 2.1.1, we give a classification of positive definite kernels with Manhattan distance input. Second, in Section 2.1.2, we categorize all functions which transform Manhattan distances to Manhattan distances or squared Euclidean distances. Third, in Section 2.1.3, we prove that the only functions which always yield a low-rank matrix when applied entry-wise to a low-rank matrix are low-degree polynomials; this is a converse of a key idea behind the polynomial method in algorithm design and in the training of transformers in natural language processing. Afterwards, in Section 2.2, we describe our new tool, representation theory of the real hyperrectangle, and how we use to to yield these applications.

2.1.1 Kernel Methods

Our first application is to the study of kernel methods in machine learning. Much of the prior work on kernels methods focuses in the Euclidean distance setting. Our new application shows how to classify kernels in the Manhattan distance setting.

We start with defining positive definite kernel under Euclidean space.

Definition 2.1.1 (Positive definite Euclidean kernel)**.** *A function f is a positive definite Euclidean kernel if, for any $x_1, \ldots x_n \in \mathbb{R}^d$ for any n and d, the matrix $M \in \mathbb{R}^{n \times n}$ with*

$$M_{i,j} = f(\|x_i - x_j\|_2)$$

is positive semi-definite. Equivalently, f is a positive definite Euclidean kernel if and only if there exists a function $F : \mathbb{R}^d \to \mathcal{H}$[1] such that:

$$\langle F(x), F(y) \rangle = f(\|x - y\|_2)$$

for all $x, y \in \mathbb{R}^d$ for all d.

The proof of the equivalence can be found in [251]. Positive definite kernels are used in machine learning to separate data embedded in $\mathbb{R}^d$ using linear separator techniques, when the initial data is not linearly separable [254, 261, 262]. In other words, a positive definite kernel can map points in $\mathbb{R}^d$ which are not linearly separable, to points in potentially higher dimensions which are linearly separable. Finding such an embedding is not an easy task in general, but kernel methods solve this problem [254, 261, 262]. The key idea is to pick a function f based on the application so that a function F like the one in Definition 2.1.1 can be found which maps the data points to vectors of possible higher dimensions, after which linear separation can be performed efficiently on these higher dimensional points.

Interestingly, linear separator algorithms such as the widely used *Support Vector Machines* (SVMs) [97] can separate the data efficiently as long as $\langle F(x), F(y) \rangle$ is easily computed for any $x, y \in \mathbb{R}^d$, even if F itself cannot be easily computed. By definition of the positive-definite kernel f, we know that $\langle F(x), F(y) \rangle = f(\|x - y\|_2)$, which allows us to compute $\langle F(x), F(y) \rangle$ quickly by instead computing $f(\|x - y\|_2)$. In other words, in order to apply linear separator

[1] $\mathcal{H}$ represents Hilbert space.

algorithms, it suffices to know that a F *exists*, and not necessarily know what it is or how to compute it.

The core result behind kernel methods is a full classification of all positive-definite Euclidean kernels, showing that a function $f : \mathbb{R} \to \mathbb{R}$ is a positive-definite Euclidean kernel if and only if $f(\sqrt{x})$ is a completely monotone function [251, 262]:

Definition 2.1.2 (Completely monotone functions [47])**.** *A function $f : \mathbb{R}^+ \to \mathbb{R}_{\geq 0}$ is completely monotone if*

$$(-1)^k f^{(k)}(x) \geq 0$$

for all $k \geq 0, x > 0$. A function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is completely monotone if $f(0) \geq \lim_{x \to 0^+} f(x)$ and $f|_{\mathbb{R}^+}$ is completely monotone.

An example of a completely monotone function is $f(x) = e^{-x}$.

Theorem 2.1.3 (Classification of all positive definite Euclidean kernels [251, 262])**.** *Function $f : \mathbb{R} \to \mathbb{R}$ is a positive-definite Euclidean kernel (Definition 2.1.1) if and only if $f(\sqrt{x})$ is a completely monotone function.*

This theorem gives a simple criterion to test whether *any* given function is a positive-definite Euclidean kernel. One famous example of these kernels include The Gaussian kernel: $f_\sigma(x) = e^{-\sigma x^2}$ for $\sigma > 0$. Another famous example is called neural tangent kernel: $f(x) = (\frac{1}{2} - \frac{1}{2\pi} \arccos(\frac{1}{2} - \frac{1}{2}x^2)) \cdot (\frac{1}{2} - \frac{1}{2}x^2)^2$, which recently proposed by machine learning community [164] and it plays a crucial role in showing the convergence of deep neural networks with non-linear activation functions [14, 15, 59, 111, 195, 203, 265]. This theory allows practitioners to describe all positive definite Euclidean kernels.

Main Result

In this chapter, we classify all positive-definite Manhattan kernels. These kernels are widely used in machine learning for physical and chemical applications [120, 204, 205]. A notable example of such a kernel is the Laplace kernel $f_\sigma(x) = e^{-\sigma x}$ which is commonly used in classification tasks [43]. However, a full description of all positive-definite Manhattan kernels was not known before our work.

Definition 2.1.4 (Positive definite Manhattan kernel)**.** *A function f is a* positive definite Manhattan kernel *if, for any $x_1, \ldots x_n \in \mathbb{R}^d$ for any n and d, the $n \times n$ matrix M with*

$$M_{i,j} = f(\|x_i - x_j\|_1)$$

is positive semi-definite.

Our main result is as follows:

Theorem 2.1.5 (Main result, informal statement of Theorem E.2)**.** *f is a positive definite Manhattan kernel (Definition 2.1.4) if only if f is completely monotone (Definition 2.1.2).*

Theorem 2.1.5 classifies all positive-definite kernels when the input distance is Manhattan. It was previously known that completely monotone functions are positive definite Manhattan kernels [35, 263], but it was not known these were the only such functions. Interestingly, our

[2]This equation is corresponding to the ReLU activation function. For other activation functions, the equation will be different.

new classification is similar to Theorem 2.1.3, but without a square root applied to the input. Prior to our result, one could have imagined that there are other positive definite Manhattan kernels to use in SVMs than were previously known. However, our result shows that there are no other such kernels.

2.1.2 Metric Transforms

Our second application is to *metric transforms*, a mathematical notion introduced by Schoenberg and Von Neumann [229].

Definition 2.1.6 (Metric transform)**.** *Suppose $\mathcal{X}$ and $\mathcal{Y}$ are semi-metric spaces*[3]*. Function f* ***transforms*** *$\mathcal{X}$ to $\mathcal{Y}$ if, for any finite set $S \subseteq \mathcal{X}$, there is a function $F : \mathcal{X} \to \mathcal{Y}$ such that*

$$f(d_{\mathcal{X}}(x_1, x_2)) = d_{\mathcal{Y}}(F(x_1), F(x_2)),$$

for all $x_1, x_2 \in S$.

Metric transforms arise naturally in many settings where one wants to transform a set of points from a metric space while maintaining some of the metric structure between them. They have proven useful in many areas including sketching and embedding norms [28], algorithms to compute a manifold geodesic [90], machine learning [237, 255], harmonic analysis [31, 177, 215], complex analysis [31], and PDE theory [79, 127]. Typically we have particular metric spaces $\mathcal{X}$ and $\mathcal{Y}$ of interest, as well as certain constraints on the function f, and would like to determine whether any function which satisfies those constraints and maps $\mathcal{X}$ to $\mathcal{Y}$. This leads to the key question in metric transforms:

Question 2.1.7. *For a given semi-metric space $\mathcal{X}$ and a given semi-metric space $\mathcal{Y}$, what is the full classification of functions f that transform $\mathcal{X}$ to $\mathcal{Y}$?*

Much work has been done on metric transforms in the special case where $\mathcal{X}$ and $\mathcal{Y}$ are both Euclidean distances[4] or close variants. Building on Schoenberg and Von Neumann's work [229], Schoenberg [263] classified all functions that transform Euclidean distances to Euclidean distances. Interestingly, it is known that there is a close connection between these metric transforms and positive definite Euclidean kernels [251, 254]

One natural question arises: what is the theory of metric transforms for non-Euclidean metrics? Surprisingly little attention has been paid to this question.

In the case when $\mathcal{X}$ is Manhattan (or ℓ_1) distance, and $\mathcal{Y}$ is Euclidean distance, Schoenberg [263] provided a partial categorization of functions that transform Manhattan distance to Euclidean distance. This was followed by Assouad's work in 1980, which provided a partial categorization of functions that transform Manhattan distances to Manhattan distances [35]. Our work on metric transforms completes the partial categorizations of Schoenberg and Assouad, and proves their partial categorization is a full categorization.

[3] A semi-metric satisfies all the axioms for a metric except possibly the triangle inequality; the square of the Euclidean distance gives rise to a semi-metric.

[4] When we refer to Euclidean or Manhattan distance in the remainder of this section, we always refer to distances in infinite dimensional Euclidean metric space and infinite dimensional Manhattan metric spaces, respectively.

Main Result

Our main result about metric transforms is a complete classification of functions that transform Manhattan distances to Manhattan distances. First, we need to define Bernstein functions:

Definition 2.1.8 (Bernstein functions [47])**.** *A function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is Bernstein if $f(0) = 0$ and its derivative f' is completely monotone (see Definition 2.1.2) when restricted to $\mathbb{R}^+$. Equivalently, a function f is Bernstein if:*

1. $(-1)^k \frac{\mathrm{d}^k f(x)}{\mathrm{d}x^k} \leq 0$ *for all* $k \geq 1, x \geq 0$,
2. $f(x) \geq 0$ *for all* $x \geq 0$, *and*
3. $f(0) = 0$.[5]

Now we are ready to state our main result:

Theorem 2.1.9 (Main result, classifying all Manhattan metric transforms, informal version and combination of Theorem C.2 and D.4)**.** *For a function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, the following are equivalent:*

1. *f is Bernstein.*
2. *f transforms Manhattan distances to Manhattan distances.*
3. *f transforms Manhattan distances to squared Euclidean distances.*

It was previously known that Bernstein functions transform Manhattan distances to Manhattan distances [35], and that they transform Manhattan distances to squared Euclidean distances [263], but in both cases, it was not previously known that these were the only such functions. It was previously conceivable that, in situations where one needs a metric transform involving Manhattan spaces, but Bernstein functions do not suffice, one could find other suitable metric transforms; our Theorem 2.1.9 rules out such a possibility. This also has a number of simple consequences, for instance: given any n points $x_1, \ldots x_n$ in the metric space $(\mathbb{R}^d, \ell_1)$ for any d, one can use our construction in Theorem 2.1.9 to explicitly calculate $F : \mathbb{R}^d \to \ell_1$ such that $\|F(x_i) - F(x_j)\|_1 = f(\|x_i - x_j\|_1)$.

2.1.3 Only Polynomials Preserve Low Rank Matrices

The *polynomial method* is a powerful technique for designing algorithms and constructing combinatorial objects. A key insight behind many of these results is the following fact, that applying a low-degree polynomial entry-wise to a low-rank matrix yields another low-rank matrix:

Fact 2.1.10 (The polynomial method, folklore; see e.g. [100])**.** *Suppose $f : \mathbb{R} \to \mathbb{R}$ is a polynomial of degree d. Then, for any matrix $M \in \mathbb{R}^{n \times n}$ of rank r, the matrix $M^f \in \mathbb{R}^{n \times n}$ given by $M^f_{i,j} := f(M_{i,j})$ has $\mathrm{rank}(M^f) \leq 2\binom{r + \lfloor d/2 \rfloor - 1}{\lfloor d/2 \rfloor}$. For instance, if $r = \log_2 n$ and $d < o(\log_2 n)$, then $\mathrm{rank}(M^f) < n$.*

For one example, consider the fastest known algorithm for batch Hamming Nearest Neighbor Search due to Alman, Chan, and Williams [20]. In this problem, one is given as input $2n$ vectors $x_1, \ldots, x_n, y_1, \ldots, y_n \in \{0,1\}^d$ for $d = \Theta(\log n)$, and a threshold value $t \in \{0, 1, \ldots, d\}$, and one wants to find a pair $(i, j) \in [n] \times [n]$ such that the Hamming distance between x_i

[5] We remark that the special attention on $f(0)$ in the definitions above is a bit non-standard but are convenient for our purposes.

and y_j is at most t. [20] takes an algebraic approach to this problem, by first considering the matrix $M \in \mathbb{R}^{n \times n}$ where $M_{i,j}$ is the Hamming distance between x_i and y_j. One can see that $\mathrm{rank}(M) \leq 2d$, and one could use fast matrix multiplication to quickly compute all the entries of M[6]. However, since M itself has n^2 entries, this could not improve much on the straightforward $O(n^2 \log n)$ time algorithm. They instead take the following approach:

1. Pick a parameter $g = n^\delta$ for a constant $\delta > 0$, and a function $f : \mathbb{R} \to \mathbb{R}$ such that $f(x) > g^2$ for all $x \in \{0, 1, \ldots, t\}$, and $f(x) \in [0, 1]$ for all $x \in \{t+1, t+2, \ldots, d\}$. [20] use Chebyshev polynomials to construct such an f which is a low-degree polynomial, so that the matrix M^f has low rank by Fact 2.1.10.
2. Let $S_1, \ldots, S_{n/g}$ be a partition of $[n]$ into n/g groups of size g, and consider the matrix $F \in \mathbb{R}^{\frac{n}{g} \times \frac{n}{g}}$ given by $F_{a,b} = \sum_{i \in S_a} \sum_{j \in S_b} M^f_{i,j}$. It is not hard to verify that $\mathrm{rank}(F) \leq \mathrm{rank}(M^f)$. Moreover, by the way f was defined, an entry $F_{a,b}$ is larger than g^2 if and only if there is an $(i, j) \in S_a \times S_b$ such that the Hamming distance between x_i and y_j is at most t.

There is a trade-off between the parameter δ and the degree of f, and hence the rank of F. [20] balance this trade-off to yield a matrix F of low rank[7] and dimensions $n^{1-\delta} \times n^{1-\delta}$ for some $\delta > 0$. Since F now has a subquadratic total number of entries, fast matrix multiplication can be used to compute all its entries and solve the problem, in roughly $O(n^{2-2\delta})$ time.

The polynomial method in algorithm design is used like this to design the fastest known algorithms for a variety of different, important problems, including: the Orthogonal Vectors problem from fine-grained complexity [2, 67], All-Pairs Shortest Paths [67, 295], the lightbulb problem in which one wants to find a planted pair of correlated vectors among a collection of random vectors [16, 174, 286], computational problems related to kernel methods in spectral clustering and semi-supervised learning [21], and some stable matching problems [224]. In all these works, one starts with a matrix M describing the input data which has low rank, and one transforms it into a matrix like M^f which 'amplifies' the key properties of the data while still having low rank. A similar approach has also been used to bound the ranks of matrices which arise in other settings, such as in the recent resolution of the Cap Set Conjecture from extremal combinatorics [100, 115], and in recent proofs that Hadamard and Fourier transforms have low Matrix Rigidity [18, 112, 113].

This motivates the question:

Question 2.1.11. *Is it possible to generalize the polynomial method (Fact 2.1.10) to functions f other than polynomials?*

In other words, are there functions f which are not polynomials, but such that if one starts with any low-rank matrix M, and applies it entry-wise yielding the matrix M^f, then M^f also has low rank? This would allow algorithm designers to expand the efficacy and reach of the polynomial method, both by expanding the set of constraints on the function f (such as those in step 1 of the algorithm above) that one could use in the recipe above, and by potentially allowing us to find new functions which satisfy those constraints but lead to lower rank bounds, and hence

[6] We first construct the matrices $X \in \mathbb{R}^{n \times 2d}$ and $Y \in \mathbb{R}^{2d \times n}$ such that $M = X \times Y$. We can then compute the product $X \times Y$ in $\widetilde{O}(n^2)$ time using fast rectangular matrix multiplication [96, 295] as long as $d < n^{0.1}$.

[7] They pick rank $\approx n^{0.1}$ in order to apply fast rectangular matrix multiplication as in footnote 6, although different applications of the polynomial method have aimed for different target ranks.

faster algorithms.

Application to Transformers in NLP Question 2.1.11 is also important in the study of *transformers* in machine learning. Transformers are a type of neural network structure that has been widely applied to many natural language processing (NLP) tasks [287]. A common computational task which arises when training transformers is to calculate the 'self attention' [185]; formally, in this task, we are given three matrices $A, B, C \in \mathbb{R}^{n \times d}$ where $n \gg d$,[8] and we would like to compute

$$(AB^\top)^f \cdot C \tag{2.1}$$

where $f : \mathbb{R} \to \mathbb{R}$ is a non-linear function that we apply entry-wise to the matrix $AB^\top \in \mathbb{R}^{n \times n}$, then we multiply the result on the right by C. In many applications, f is the soft-max function.

Naively evaluating Eq. (2.1) takes time $O(n^2 d)$ (without using fast matrix multiplication). However, if we can quickly find matrices $\widetilde{A}, \widetilde{B} \in \mathbb{R}^{n \times \widetilde{d}}$ for some $\widetilde{d} < n$ such that

$$(AB^\top)^f = \widetilde{A} \times \widetilde{B}^\top,$$

then we can evaluate Eq. (2.1) more quickly by first computing $\widetilde{B}^\top \times C$ and then computing $\widetilde{A} \times (\widetilde{B}^\top \times C)$, for a total running time of just $O(nd\widetilde{d})$.

Since $AB^\top$ can be any rank d matrix, and $\widetilde{A} \times \widetilde{B}^\top$ has rank at most $\widetilde{d}$, it follows that an upper bound on the best $\widetilde{d}$ we can achieve is the maximum, over all matrices M of rank d, of $\mathrm{rank}(M^f)$. Question 2.1.11 asks whether it is possible to achieve $d' < n$ for functions f like the soft-max function which are not a polynomial. If not, then we can only hope to carry out this plan of attack if we can find a low-degree polynomial approximation to our function f.

Main Result

More formally, the functions f we are interested in are those which preserve low-rank matrices. We first define applying a function to a matrix entry-wise, then the matrices of interest.

Definition 2.1.12 (Entry-wise application)**.** *For a function $f : \mathbb{R} \to \mathbb{R}$ and matrix $M \in \mathbb{R}^{a \times b}$, the* entry-wise application of f to M *is the matrix $M^f \in \mathbb{R}^{a \times b}$ where $M^f_{i,j} := f(M_{i,j})$, for $(i, j) \in [a] \times [b]$.*

Definition 2.1.13 (Preserve low-rank matrices)**.** *For a function $f : \mathbb{R} \to \mathbb{R}$ and positive integer n, we say f* preserves low-rank $n \times n$ matrices *if, for every matrix $M \in \mathbb{R}^{n \times n}$ with $\mathrm{rank}(M) \leq \lceil \log_2(n) \rceil + 1$, we have $\mathrm{rank}(M^f) < n$.*

For a function f to be effective in the polynomial method as described above, it is necessary (but usually not sufficient) that f preserves low-rank $n \times n$ matrices in the sense of Definition 2.1.13. Indeed, in all the aforementioned applications of the polynomial method, such as the algorithm of [20] and the application to transformers that we described above, the original matrix M describing the data can have rank greater than $\log_2 n$. The details of how low the rank

[8] The matrices A, B and C correspond to the query, key, and value matrices, respectively, when training transformers in NLP applications. For more background, we refer the reader to the post by Kulshrestha [185] and more followups [88, 121, 179, 293].

of M^f must be can vary in the different applications, but it is always necessary that M^f has less than *full* rank (i.e., $\mathrm{rank}(M^f) < n$).

Our main result answers Question 2.1.11 in the negative, showing that Fact 2.1.10 cannot be generalized.

Theorem 2.1.14 (Main result, informal statement of Theorem B.6)**.** *For any positive integer $n \geq 2$, if the real analytic function $f : \mathbb{R} \to \mathbb{R}$ preserves low-rank $n \times n$ matrices, then f is a polynomial of degree at most $\lceil \log_2(n) \rceil$.*

This shows that real analytic functions f which are not polynomials do not preserve low-rank $n \times n$ matrices, and only polynomials of degree less than $\lceil \log_2(n) \rceil$ can preserve low-rank $n \times n$ matrices. Hence, one cannot hope to improve on the polynomial method by extending it to any classes of real analytic functions other than low-degree polynomials.

We note that there is a small constant-factor gap between the degree which Fact 2.1.10 tells us is sufficient for a polynomial to preserve low-rank $n \times n$ matrices, and the degree that Theorem 2.1.14 says is necessary: for instance, Fact 2.1.10 says that polynomials of degree at most $\frac{1}{2}\log_2(n)$ suffice, since $\binom{\frac{5}{4}\log_2(n)}{\frac{1}{4}\log_2(n)} \ll n$, whereas Theorem 2.1.14 says that degree less than $\log_2(n)$ is necessary. We leave open the question of closing this gap, although we note that the constant factor in front of the polynomial degree does not play a major role in most of the aforementioned applications of Fact 2.1.10.[9]

2.2 Technique Overview

2.2.1 Key Lemma

In this section, we introduce our key new technical idea, representation theory of the real hyperrectangle. This technique computes eigenvectors and eigenvalues of a large class of matrices which are defined in terms of hyperrectangles. We first describe and provide intuition about this technique, then we will explain how it leads to our applications by demonstrating why these matrices and their eigenvalues are relevant to kernel methods, metric transforms, and the converse of the polynomial method.

Our new technique concerns matrices defined in terms of a real hyperrectangle.

Definition 2.2.1 (Real hyperrectangle)**.** *The d-dimensional real hyperrectangle parameterized by d variables $a_1, \ldots a_d > 0$ is the convex hull of the 2^d points $\{\pm a_1/2, \ldots \pm a_d/2\}$.*

The eigenvectors of the family of matrices we define shortly will come from columns of Walsh-Hadamard matrices.

Definition 2.2.2 (Walsh-Hadamard matrices)**.** *For a positive integer d, let $v_1, \ldots v_{2^d} \in \{0,1\}^d$ be the enumeration of all n-bit vectors in lexicographical order. The* Walsh-Hadamard *matrix H_d is the $2^d \times 2^d$ matrix defined by $H_d(v_i, v_j) := (-1)^{\langle v_i, v_j \rangle}$, where $\langle v_i, v_j \rangle$ is the inner product between v_i and v_j.*

We now introduce our key new technical lemma:

[9] For instance, our running example algorithm of [20] only uses an asymptotic bound on how the degree grows with the dimension of the input points, and the constant factor in front of the polynomial degree is ultimately subsumed by a 'O' in the running time.

Lemma 2.2.3 (Representation Theory of the Real Hyperrectangle, informal version of Lemma F.2)**.** *Consider a d-dimensional hyperrectangle (Definition 2.2.1) parameterized by $a_1, \dots a_d > 0$. Enumerate the vertices in lexicographical ordering as $p_1, \dots p_{2^d}$.*

For any $f : \mathbb{R} \to \mathbb{R}$, let D be the 2^d by 2^d matrix given by $D_{i,j} = f(\|p_i - p_j\|_1)$. Then:

1. *$\Sigma := H_d D H_d$ is a diagonal matrix whose entries are the eigenvalues of D multiplied by 2^d, and $D = 4^{-d} \cdot H_d \Sigma H_d$.*
2. *Let $v_1, \dots v_{2^d}$ be the columns of the Hadamard matrix H_d. Then $v_1, \dots v_{2^d}$ are the eigenvectors of D. For $i \in [2^d]$, let $B(i) \in \{0,1\}^d$ be the binary representation of i. Then, the eigenvalue corresponding to v_i is:*

$$\lambda_i = \sum_{b \in \{0,1\}^d} (-1)^{\langle B(i), b \rangle} \cdot f(\langle b, a \rangle). \tag{2.2}$$

We will see shortly that this expression for the eigenvalue λ_i can also be rewritten in terms of integrals and derivatives of the function f, allowing us to use analytic techniques when computing or applying these eigenvalues.

Warm-up: d-dimension To illustrate Lemma 2.2.3, consider the case when the dimension of the hyperrectangle is $d = 2$, and the hyperrectangle is parameterized by $a, b > 0$.

Let

$$p_1 = (+a/2, +b/2),\ p_2 = (-a/2, +b/2),\ p_3 = (+a/2, -b/2),\ p_4 = (-a/2, -b/2)$$

be the vertices of the hyperrectangle.

The matrix $D \in \mathbb{R}^{4 \times 4}$ we consider is defined by $D_{i,j} = f(\|p_i - p_j\|_1)$, and is thus given by:

$$D = \begin{bmatrix} f(0) & f(a) & f(b) & f(a+b) \\ f(a) & f(0) & f(a+b) & f(b) \\ f(b) & f(a+b) & f(0) & f(a) \\ f(a+b) & f(b) & f(a) & f(0) \end{bmatrix}. \tag{2.3}$$

Lemma 2.2.3 says that D's eigenvectors are the columns of the 4 by 4 Hadamard matrix H_2:

$$v_1 = \begin{bmatrix} +1 \\ +1 \\ +1 \\ +1 \end{bmatrix}, v_2 = \begin{bmatrix} +1 \\ -1 \\ +1 \\ -1 \end{bmatrix}, v_3 = \begin{bmatrix} +1 \\ +1 \\ -1 \\ -1 \end{bmatrix}, v_4 = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix}.$$

We can verify that these are the eigenvectors, and compute the corresponding eigenvalues $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, by multiplying the first row of D by v_1, v_2, v_3, v_4:

$$\begin{aligned} \lambda_1 &= f(0) + f(a) + f(b) + f(a+b), \\ \lambda_2 &= f(0) - f(a) + f(b) - f(a+b), \\ \lambda_3 &= f(0) + f(a) - f(b) - f(a+b), \\ \lambda_4 &= f(0) - f(a) - f(b) + f(a+b). \end{aligned} \tag{2.4}$$

This is the $d = 2$ version of Eq. (2.2).

A key remark in some of our proofs is that, if f is smooth, then $\lambda_2, \lambda_3, \lambda_4$ can also be written in terms of integrals using the fundamental theorem of calculus:

$$\begin{aligned}
\lambda_2 &= -\int_0^a f'(x)dx - \int_b^{a+b} f'(x)\mathrm{d}x, \\
\lambda_3 &= -\int_0^b f'(x)\mathrm{d}x - \int_a^{a+b} f'(x)\mathrm{d}x, \\
\lambda_4 &= \int_0^a \int_0^b f''(x+y)\mathrm{d}x\mathrm{d}y.
\end{aligned} \tag{2.5}$$

Expressions similar to Eq. (2.5) hold for the general, d-dimensional setting as well.

Proof idea: Representation Theory of the Real Hyperrectangle We call our technique 'representation theory of the real hyperrectangle' since it is proved by using representation theory to calculate the eigenvalues of a large class of matrices. Representation theory in general is used to calculate eigenvalues of matrices associated with objects that have group symmetry [118**?**]. The d-dimensional real hyperrectangle has reflection symmetry about each of its d axes, and Lemma 2.2.3 follows from analyzing this symmetry using Schur's Lemma from representation theory. In other words, Lemma 2.2.3 can be seen as a use of representation theory of the symmetry group of the real hyperrectangle. For more details on representation theory, see Lemma A.10 in Appendix A.6. For a proof of Lemma 2.2.3, see Appendix F.

Related Work Representation Theory is a mathematical field dating back a hundred years, with many applications in physics and computer science. Representation theory is used in physics to calculate the spectra of Hamiltonians and compute molecular and atomic orbitals [123].

In computer science, representation theory is used to compute the vibrational spectra of graph Laplacians where the underlying graph has vertex-transitive group symmetry, a case which covers the boolean cube, cycle, buckyball, and other molecular structures [136, 232, 268]. Guattery and Miller implicitly used representation theory to give structure to the spectra of graphs where there exists a vertex automorphism of order two [146]. Representation theory is also central to the study of quantum tomography [233], Boolean function analysis [232], low-sparsity expander construction [213, 217], random walk theory [106, 125, 249], and more.

Representation Theory is closely related to Fourier transforms [279], which have been extensively studied in theoretical computer science [82, 84, 151, 152, 161, 163, 168, 171, 172, 225, 227, 240].

Use in Applications We next give an overview of how we use Lemma 2.2.3 to derive our three applications. We focus on explaining how the matrices described by Lemma 2.2.3 and their eigenvalues arise in each setting. At a high level, the class of matrices described by Lemma 2.2.3 is sufficiently general that we are able to show it is 'complete' for the matrices or distance functions arising in our applications. At the same time, Lemma 2.2.3 allows us to easily compute the

eigenvalues of these matrices. To our knowledge, a general enough class of matrices which capture our applications but whose eigenvalues are understood was previously not known, and this is what allows us to prove that previous partial categorizations (of positive definite kernels (Definition 2.1.1, 2.1.4), metric transforms (Definition 2.1.6), and functions which preserve low-rank (Definition 2.1.13)) are in fact complete classifications.

2.2.2 Kernel Methods

We begin with an overview of our proof of Theorem 2.1.5. Given any n points in d-dimensional Manhattan space, it is known they can be isometrically embedded into ℓ_1 restricted to the corners of some (possibly high dimensional) hyperrectangle [105]. Therefore, to prove Theorem 2.1.5, it suffices to find all functions f such that the matrix $M \in \mathbb{R}^{2^d} \times \mathbb{R}^{2^d}$ defined as:

$$M_{i,j} = f(\|p_i - p_j\|_1)$$

is positive semi-definite whenever $p_1, \ldots p_{2^d}$ are the vertices of some hyperrectangle.

Fortunately, Lemma 2.2.3 gives us a closed form expression for the eigenvalues of M. For M to be positive semi-definite, the eigenvalues of M must all be nonnegative. We exploit a connection between eigenvalues of M (which are computed by Eq. (2.2)) and discrete derivatives of f to prove that f must be completely monotone (Definition 2.1.2). The details are quite technical; for more details, see Appendix E.

2.2.3 Metric Transforms

We next sketch the proof of Theorem 2.1.9. Schoenberg [263] previously showed that Bernstein functions transform Manhattan distances to squared Euclidean distances, and Assouad [35] previously showed that Bernstein functions transform Manhattan distances to Manhattan distances. It thus suffices to prove that any function that transforms Manhattan to squared Euclidean must be Bernstein, and similarly for any function that transforms Manhattan to Manhattan.

Bernstein functions transform Manhattan to squared Euclidean Our starting point is a classical criterion for determining whether a set of distances is a squared Euclidean distance due to Schoenberg [252]:

Lemma 2.2.4 (Squared Euclidean distance criterion [252])**.** *Given a set of distances $d_{i,j}$ for all $(i,j) \in [n] \times [n]$ satisfying $d_{i,j} = d_{j,i}$ and $d_{i,i} = 0$, then $d_{i,j}$ can be embedded into squared Euclidean distance if and only if matrix D with $D_{i,j} = d_{i,j}$ satisfies $x^\top Dx \leq 0$ for all $x \perp 1$.*

This criterion D must satisfy is known as the *negative type condition* [105]. As in Section 2.2.2 above, we also know that any Manhattan distance can be isometrically embedded into Manhattan distances between a subset of the corners of a (possibly high) dimensional real hyperrectangle. Therefore, by carefully considering the definition of Bernstein functions, one can show: to prove that only Bernstein functions transform Manhattan to squared Euclidean, it suffices to show that the matrix D where

$$D_{i,j} = f(\|p_i - p_j\|_1)$$

satisfies $x^\top Dx \leq 0$ for all $x \perp 1$, whenever $p_1, \ldots p_{2^d}$ are vertices of some hyperrectangle.

Lemma 2.2.3 tells us that that the all ones vector v_1 is an eigenvector of D, since v_1 is the first column of the Hadamard matrix. Therefore, it suffices to show that the eigenvalues of D except for λ_1 are negative. We once again exploit a connection between eigenvalues of D and discrete derivatives of f to prove that f must be Bernstein (Definition 2.1.8). For more details, see Theorem C.2 in Appendix C.

Manhattan to Squared Euclidean $\Leftrightarrow$ Manhattan to Manhattan We next show that functions that transform Manhattan to squared Euclidean must transform Manhattan to Manhattan, and vice versa. It is known that Manhattan distances isometrically embed into squared Euclidean distances [105, 263], which implies that functions that transform Manhattan to Manhattan must transform Manhattan to squared Euclidean.

To prove the other direction, suppose function f transforms Manhattan to squared Euclidean. Consider as before the $2^d \times 2^d$ matrix D where $D_{i,j} = f(\|p_i - p_j\|_1)$, where $p_1, \ldots p_{2^d}$ are vertices in lexicographical order of some real hyperrectangle (Definition 2.2.1).

Using the fact that D contains squared Euclidean distances, we can explicitly find 2^d points whose pairwise squared Euclidean distances are the entries in D (by combining Lemma 2.2.3 and methods of Schoenberg [252]). We show that these 2^d points, themselves, lie on another 2^d-dimensional real hyperrectangle! One can see using the Pythagorean theorem that squared Euclidean distances on the real hyperrectangle can be realized as Manhattan distances, so this shows that f transforms Manhattan to Manhattan as well. For more details, see Appendix D.

2.2.4 Polynomial Method Converse

In this section, we sketch our techniques for Theorem 2.1.14. We also explain how the matrices from hyperrectangles in Lemma 2.2.3 arise in our methods.

Let $d = \lceil \log_2 n \rceil$. Suppose $M : \mathbb{R}^d \to \mathbb{R}^{2^d \times 2^d}$ is a family of matrices defined by $M(a) = \|p_i - p_j\|_1$ where $p_1, \ldots p_{2^d}$ are vertices in lexicographical order of the hyperrectangle paramaterized by $a_1, \ldots a_d$. We show that these matrices have rank at most $d+1$. Therefore if f preserves low rank $n \times n$ matrices, then $M(a)^f$ must have rank $< n$ for all $a \in \mathbb{R}^d$.

Recall that representation theory of the real hyperrectangle (Lemma 2.2.3) gives an algebraic formula for the eigenvalues $\lambda_1^f(a), \ldots \lambda_{2^d}^f(a)$ of $M(a)^f$ in terms of a and f. The fact that $M(a)^f$ does not have full rank for any a means that, for every a, there is an i such that $\lambda_i^f(a) = 0$. However, using Lemma 2.2.3, we prove the stronger statement that there exists an i such that for all a, we have $\lambda_i^f(a) = 0$.

Next, we show that if $\lambda_i^f(a) = 0$ for all a, then $f^{(d)}(x) = 0$ for all $x \in \mathbb{R}$, where $f^{(d)}$ represents the d^{th} derivative of f. We do this by writing $f^{(d)}$ as a linear combination of $\lambda_i^f(a)$ for various settings of a, making use of our integral expressions similar to Eq. (2.5) above. This implies that f is a degree $d = \lceil \log_2 n \rceil$ polynomial if it preserves low rank.

Appendix

Roadmap. In Section 3, we define notations, provide several basic definitions and fundamental tools. In Section B, we prove that non-polynomial function blows up the matrix rank. It proves Theorem 2.1.5. In Section C, we prove condition 1 and condition 2 in Theorem 2.1.9 are equivalent. In Section D, we prove condition 2 and condition 3 in Theorem 2.1.9 are equivalent. Overall, Section C and Section D together prove Theorem 2.1.9. In Section E, we have a proof of Theorem 2.1.5. In Section F, we prove our result about representation theory of real hyperrectangles.

A Preliminaries

This section is organized as follows:

- In Section A.1, we define several basic notations.
- In Section A.2, we provide some definitions about Hadamard matrix, high-dimensional hyperrectangle.
- In Section A.3, we provide some previous work about the classifications of completely monotone and Bernstein function.
- In Section A.4, we state well-known results about metric hierarchies.
- In Section A.5, we define negative metrics and euclidean embeddability.
- In Section A.6, we present previous work about representation theory tools.

A.1 Notations

For a vector x, we use $\|x\|_1$ to denote the entry-wise ℓ_1 norm of x. We use $\|x\|_2$ to denote the entry-wise ℓ_2 norm of x. For two vectors a, b, we use $\langle a, b\rangle$ to denote the inner product between a and b. For a vector x, we use $x^\top$ to denote the transpose of x.

A.2 Definitions

We provide an alternate but equivalent definition of H_d as the square Hadamard matrix with 2^d rows. These matrices consist of ± 1-valued entries and are defined recursively via:

$$H_0 = [1]$$
$$H_{k+1} = \begin{bmatrix} H_k & H_k \\ H_k & -H_k \end{bmatrix}, \quad k \geq 0.$$

For a review of Hadamard matrices, see [153].

Hyperrectangles Often times in our proof, we may say things like "let $x_1, \ldots x_{2^d}$ be the corners of a d dimensional hyperrectangle". For these statements to make sense, we must specify which corner x_i refers to. Scale the d dimensional hyperrectangle to be an axis-aligned hypercube, and place one of the hypercube corners at the origin. Each corner then has a binary number b as its coordinate bit string. We let x_{b+1} refer to the original hyperrectangle corner corresponding to b.

A.3 Alternate Classifications of Completely Monotone and Bernstein Functions

Here we recall the classical Bernstein Theorem from analysis constructively classifying completely monotone (Definition 2.1.2) and Bernstein functions (Definition 2.1.8).

Proposition A.1 (Chapter 14, Theorems 3 and 6 in [190]). *For a function $f : \mathbb{R}_{>0} \to \mathbb{R}_{\geq 0}$, the following are equivalent:*

1. *f is completely monotone.*
2. *Letting $(D_a f)(x) = f(x+a) - f(x)$, for any $(a_1, \ldots, a_n)$ non-negative we have*

$$(-1)^n \left(\prod_{i=1}^{n} D_{a_i} \right) f(x) \geq 0$$

   for all $x > 0$.
3. *There exists a positive finite measure μ on $\mathbb{R}_{\geq 0}$ such that*

$$f(x) = \int_0^\infty e^{-tx} \mathrm{d}\mu(t), \quad x > 0.$$

The part 2 of Proposition A.1 is essentially the definition we gave for completely monotone, except that it does not assume any smoothness or even continuity a priori. The third shows that all completely monotone functions are in fact mixtures of decaying exponentials. From the above one easily derives a corresponding classification of Bernstein functions. If f also has 0 in its domain, then the above result applies the same way, however (with the same measure μ as in part 3 of Proposition A.1) we have

$$f(0) \geq \mu(\mathbb{R}_{\geq 0})$$

since we did not require any continuity at 0.

Proposition A.2 (Theorem 6.7 in [45]). *For a function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with $f(0) = 0$, the following are equivalent:*

1. *f is Bernstein.*
2. *Letting $(D_a f)(x) = f(x+a) - f(x)$, for any $(a_1, \ldots, a_n)$ non-negative we have*

$$(-1)^n \left(\prod_{i=1}^{n} D_{a_i} \right) f(x) \leq 0, \quad x > 0.$$

3. There exists a positive measure μ on $\mathbb{R}^+$ and $a, b \geq 0$ such that

$$f(x) = a + bx + \int_{\mathbb{R}^+} (1 - e^{-tx}) \mathrm{d}\mu(t), \quad x > 0.$$

Here μ must satisfy $\int_{\mathbb{R}_+} \min\{1, t\} \mathrm{d}\mu(t) < \infty$.

Due to the second criterion just above, Bernstein functions are also sometimes called *completely alternating*. We remark that these results apply more generally in the setting of abelian semigroups, where the integral is taken over a measure on the space of positive characters. This general point of view is explained in [45, Chapter 6], and applies, for instance, to the semigroup of compact subsets of $\mathbb{R}$ under union.

A.4 Metric Hierarchies

Here are well-known facts we will use throughout our proof:

Lemma A.3. *For any n points $x_1, \ldots x_n$ in ℓ_1, there exist n points $y_1, \ldots y_n$ such that $\|x_i - x_j\|_1 = \|y_i - y_j\|_1$, and $y_1, \ldots y_n$ are a subset of corners of a d dimensional hyperrectangle for some d.*

Proof. This follows from the equivalence of the cut cone and ℓ_1 distance (Theorem 4.2.2 in [105]). □

Lemma A.4. *The squared Euclidean distance between points in the corners of a hyperrectangle isometrically embeds into Manhattan distance.*

Proof. This follows from the Pythagorean theorem. □

Lemma A.5. *Manhattan distances embed isometrically into squared Euclidean distances.*

Proof. This follows from Corollary 6.1.4 and Lemma 6.1.7 in [105]. □

A.5 Negative Type Metrics and Euclidean Embeddability

We now present a criterion by Schoenberg [252] on when a metric is isometrically embeddable into squared Euclidean distances[10].

Definition A.6 (negative type)**.** *A matrix D is iff $x^\top D x \leq 0$ for all $x \perp 1$.*

Lemma A.7 (Schoenberg [252])**.** *Consider $x_1, \ldots, x_n$ where $d_{i,j}$ is the distance between x_i and x_j. Let D be an n by n matrix where $D_{i,j} = d_{i,j}^2$. The distances $d_{i,j}$ are isometrically embeddable into Euclidean space iff the matrix D is negative type.*

We note that if D happens to have the all ones vector $\mathbf{1}$ as an eigenvector, we have a simpler criterion for testing if D is negative type:

[10] We note that Schoenberg's criteria has a beautiful proof, which one can find one direction of in [239].

Lemma A.8 (Schoenberg Variant)**.** *Consider $x_1, \ldots, x_n$ where $d_{i,j}$ is the distance between x_i and x_j. Let D be an n by n matrix where $D_{i,j} = d_{i,j}^2$.*

If the all ones vector is an eigenvector of D, then the $d_{i,j}$ are isometrically embeddable into Euclidean space iff every eigenvalue of D, excluding the eigenvalue corresponding to the all ones vector, is non-positive.

Proof. Lemma A.8 follows from Lemma A.7 and the fact that every symmetric matrix has an orthonormal set of eigenvectors. □

If d_{ij} is isometrically embeddable into Euclidean space, we can find an explicit embedding:

Lemma A.9. *Consider $x_1, \ldots x_n$ where $d_{i,j}$ is the distance between x_i and x_j. Let D be the matrix where $D_{i,j} = d_{i,j}^2$. Let Π be the projection matrix off the all ones vector, i.e., Π can be expressed explicitly as $I - J/n$, where J is the $n \times n$ all-ones matrix, and I is identity matrix.*

Let $M := -\frac{1}{2}\Pi D \Pi$.

If $y_1, \ldots y_n$ are such that $\|y_i - y_j\|_2 = d_{i,j}$ and $\sum_{i=1}^n y_i = 0$, then $M_{i,j} = \langle y_i, y_j \rangle$. Moreover, if $M = U^\top U$ for some U, then the columns of U are an embedding of $x_1, \ldots x_n$ into Euclidean space.

This follows from Eq. 2 in [99]. A longer exposition of the link between distance matrices and inner product matrices can be found in [99].

A.6 Useful Tools

We present Schur's lemma for Abelian groups G. Schur's lemma is one of the cornerstones of representation theory [118].

Lemma A.10 (Schur's lemma for Abelian groups)**.** *If G is a finite Abelian group of $n \times n$ matrices under multiplication, and M is an $n \times n$ diagonalizable matrix satisfying $Mg = gM$, for all $g \in G$, then there exists a set of linearly independent vectors $v_1, \ldots v_n$ that are eigenvectors of M and all $g \in G$. In other words, M and G are simultaneously diagonalizable.*

Schur's Lemma will be useful in proving our key result about representation theory of the real hyperrectangle, or Lemma 2.2.3.

B Non-Polynomial Functions Blow Up Matrix Rank

The major goal of this section is to prove Theorem B.6. This section is organized as follows

- In Section B.1, we show some basic facts.
- In Section B.2, we show that one eigenvalue is identically zero.
- In Section B.3, we prove that only polynomials have a zero eigenvalue.
- In Section B.4, we rewrite the sum of eigenvalues.
- In Section B.5, we show the convergence via calculating the limit.
- In Section B.6, we state and prove our main result.

B.1 Preliminaries

We start with defining a useful tool.

Lemma B.1. *If $g_1, \dots g_n : \mathbb{R}^d \to \mathbb{R}$ are all Taylor expandable, and the union of the zero-sets of g_i is all of $\mathbb{R}^d$, then one of $g_1, \dots g_n$ is identically zero.*

Proof. Firstly, if $g : \mathbb{R}^d \to \mathbb{R}$ is Taylor expandable, then the zero-set of g has a well-defined measure.

Secondly, if the measure of the zero-set is non-zero, there must exist an open ball in which g is 0. If this is the case, every higher order derivative at the center of the ball must be 0, meaning the Taylor series for that function is identically zero.

Finally, since the union of the zero-sets of g_i is the entire plane, one of their zero-sets has non-zero measure. Thus, it must be identically zero. □

B.2 One Eigenvalue is Identically Zero

The goal of this section is prove Lemma B.2.

Lemma B.2 (One eigenvalue is identically zero)**.** *For any Taylor expandable function f, any n, and $d := \log n + 1$: we can find $M : \mathbb{R}^d \to \mathbb{R}^{n \times n}$ and Taylor expandable $\lambda_i^f : \mathbb{R}^d \to \mathbb{R}$ satisfying:*

1. *$M(a)$ has rank $\leq d$ for all $a \in \mathbb{R}^d$*
2. *$\lambda_1^f(a) \dots \lambda_n^f(a)$ is the full set of eigenvalues of $f(M(a))$, for all $a \in \mathbb{R}^d$.*
3. *If there exists $i \in [n]$ such that $\lambda_i^f(a) = 0$ for all $a \in \mathbb{R}^d$, then f is a degree $d \leq \log n + 1$ polynomial.*

Proof. **Constructing M.** Consider a mapping $B : \{0, 1, \dots 2^d - 1\} \to \{0, 1\}^d$ corresponding to the conversion of integers into d-digit binary strings, which we interpret as d dimensional $0 - 1$ vectors. We set

$$M(a)_{i,j} = \langle a, B(|i - j|) \rangle$$

where $a \in \mathbb{R}^d$.

Constructing λ_i^f. For each matrix $M(a) \in \mathbb{R}^{n \times n}$, we established previously that $f(M(a)) \in \mathbb{R}^{n \times n}$ has eigenvectors equal to the Hadamard matrix columns, and the corresponding eigenvalues are:

$$\lambda_i^f(a) = \sum_{b \in \{0,1\}^d} (-1)^{\langle B(i), b \rangle} \cdot f(\langle b, a \rangle)$$

We note that if f is Taylor expandable, then so is λ_i^f for all i and f.

As noted before, λ_i^f is Taylor expandable if f is Taylor expandable. Also, $\lambda_i^f(a)$ forms the full set of eigenvalues for $M(a)$. Therefore, all that's left to prove is that if any λ_i^f is 0, then so is $f^{(d)}$.

If λ_i^f is 0, then

$$(-1)^{\langle B(i), 1 \rangle} \sum_{b \in \{0,1\}^d} (-1)^{\|b\|_1} \cdot \lambda_i^f(a + \epsilon b) = 0$$

since it is the sum and difference of λ_i^f evaluated at various points. Now:

$$\begin{aligned}
&(-1)^{\langle B(i),1\rangle} \sum_{b\in\{0,1\}^d} (-1)^{\|b\|_1} \cdot \lambda_i^f(a+\epsilon b) \\
&= (-1)^{\langle B(i),1\rangle} \sum_{b_1\in\{0,1\}^d} (-1)^{\|b_1\|_1} \cdot \left(\sum_{b_2\in\{0,1\}^d} (-1)^{\langle B(i),b_2\rangle} f(\langle b_2, a+\epsilon b_1\rangle) \right) \\
&= (-1)^{2\langle B(i),1\rangle} \sum_{b\in\{0,1\}^d} (-1)^{\|b\|_1} \cdot f(\langle a+\epsilon b, \mathbf{1}\rangle) \\
&= \sum_{b\in\{0,1\}^d} (-1)^{\|b\|_1} \cdot f(\langle a+\epsilon b, \mathbf{1}\rangle
\end{aligned}$$

where the first equality follows from the definition of λ_i^f and the second equality follows from Lemma B.4. It follows that if $\lambda_i^f = 0$, then

$$\sum_{b\in\{0,1\}^d} (-1)^{\|b\|_1} \cdot f(\langle a+\epsilon b, \mathbf{1}\rangle = 0$$

for all ϵ and a. By taking the limit as $\epsilon \to 0$ and dividing by ϵ^d, we have for all a:

$$\lim_{\epsilon\to 0} \frac{1}{\epsilon^d} \sum_{b\in\{0,1\}^d} (-1)^{\|b\|_1} \cdot f(\langle a+\epsilon b, \mathbf{1}\rangle) = 0 \tag{2.6}$$

By Lemma B.5, the LHS of Eq. (2.6) is $f^{(d)}(a)$, so $f^{(d)}(a) = 0$ for all a. Therefore, f is at most a degree d polynomial as desired. Thus, we complete the proof. □

B.3 Only Polynomials Have a Zero Eigenvalue

The goal of this section is to prove Lemma B.3.

Lemma B.3 (Only polynomials have a zero eigenvalue)**.** *Given:*

1. *A function $f : \mathbb{R} \to \mathbb{R}$*
2. *A function $M : \mathbb{R}^d \to \mathbb{R}^{n\times n}$, mapping d dimensional vectors to n dimensional matrices.*
3. *A set of n functions $\lambda_1^f, \lambda_2^f, \cdots, \lambda_n^f$ such that each $\lambda_i^f : \mathbb{R}^d \to \mathbb{R}$ is Taylor expandable, and $\lambda_1^f(a) \ldots \lambda_n^f(a)$ is the full set of eigenvalues of f applied entry-wise to $M(a)$ for all $a \in \mathbb{R}^d$,*

Then if f transforms matrices $M(a)$ to rank $< n$ for all $a \in \mathbb{R}^d$, then there exists $i \in [n]$ where function $\lambda_i^f = 0$.

Proof. If f transforms matrix $M(a)$ to rank $< n$, then for any a, there exists an i where $\lambda_i^f(a) = 0$. Thus the union (over i) of the zero sets of λ_i^f is $\mathbb{R}^d$. We can then apply Lemma B.1 to show that one of λ_i^f is identically 0 as desired. □

B.4 Rewriting the Sum

The goal of this section is to prove B.4.

Lemma B.4 (Rewriting the sum)**.**

$$\sum_{b_1 \in \{0,1\}^d} (-1)^{\|b_1\|_1} \left(\sum_{b_2 \in \{0,1\}^d} (-1)^{\langle B(i), b_2 \rangle} f(\langle b_2, a + \epsilon b_1 \rangle) \right)$$
$$= (-1)^{\langle B(i), \mathbf{1} \rangle} \sum_{b \in \{0,1\}^d} (-1)^{\|b\|_1} \cdot f(\langle a + \epsilon b, \mathbf{1} \rangle)$$

where a and b are d-dimensional vectors.

Proof. First, we can show: If b_2 is a d dimensional vector with any 0s in its vector notation, we know

$$\sum_{b_1 \in \{0,1\}^d} (-1)^{\|b_1\|_1} f(\langle b_2, a + \epsilon b_1 \rangle) = 0 \tag{2.7}$$

for any ϵ, and any constant d dimensional vector a. The reason is if b_2 has any 0's in its vector notation, then flipping the corresponding bit in b_1 causes $(-1)^{\|b_1\|_1}$ to change sign, while leaving $\langle b_2, a + \epsilon b_1 \rangle$ unchanged.

Now, we know that:

$$\sum_{b_1 \in \{0,1\}^d} (-1)^{\|b_1\|_1} \left(\sum_{b_2 \in \{0,1\}^d} (-1)^{\langle B(i), b_2 \rangle} f(\langle b_2, a + \epsilon b_1 \rangle) \right)$$
$$= \sum_{b_2 \in \{0,1\}^d} (-1)^{\langle B(i), b_2 \rangle} \left(\sum_{b_1 \in \{0,1\}^d} (-1)^{\|b_1\|_1} f(\langle b_2, a + \epsilon b_1 \rangle) \right)$$
$$= (-1)^{\langle B(i), \mathbf{1} \rangle} \left(\sum_{b_1 \in \{0,1\}^d} (-1)^{\|b_1\|_1} f(\langle \mathbf{1}, a + \epsilon b_1 \rangle) \right).$$

where the first equality follows by rearranging sums, and the second equality follows from Eq. (2.7). This completes the proof.

□

B.5 Calculating the Limit

The goal of this section is to prove Lemma B.5.

Lemma B.5 (Calculating the limit)**.** *Suppose the d^{th} derivative of f, denoted as $f^{(d)}$, is continuous. Then:*

$$\lim_{\epsilon \to 0} \epsilon^{-d} \sum_{b \in \{0,1\}^d} (-1)^{\|b\|_1} \cdot f(\langle a + \epsilon b, \mathbf{1} \rangle) = f^{(d)}(\langle a, \mathbf{1} \rangle).$$

Proof. We have:

$$\sum_{b\in\{0,1\}^d}(-1)^{\|b\|_1}\cdot f(\langle a+\epsilon b,\mathbf{1}\rangle) = \sum_{s=0}^{d}(-1)^s\binom{d}{s}\cdot f(\langle a+\epsilon b,\mathbf{1}\rangle)$$
$$= \sum_{s=0}^{d}(-1)^s\binom{d}{s}\cdot f(\langle a,\mathbf{1}\rangle+s\epsilon)$$
$$= \int_0^\epsilon\int_0^\epsilon\dots\int_0^\epsilon f^{(d)}(\langle a+x,\mathbf{1}\rangle)\mathrm{d}x_1\dots\mathrm{d}x_d \tag{2.8}$$

which we note, is independent of i. The first and second equality follow from grouping b by the number of ones it has, which we denote as s. The last equality follows from the fundamental theorem of calculus.

Thus:

$$\lim_{\epsilon\to0}\epsilon^{-d}\sum_{b\in\{0,1\}^d}(-1)^{\|b\|_1}\cdot f(\langle a+\epsilon b,\mathbf{1}\rangle)$$
$$= \lim_{\epsilon\to0}\epsilon^{-d}\int_0^\epsilon\int_0^\epsilon\dots\int_0^\epsilon f^{(d)}(\langle a+x,\mathbf{1}\rangle)\mathrm{d}x_1\dots\mathrm{d}x_d$$
$$= f^{(d)}(\langle a,\mathbf{1}\rangle)$$

where the first equality follows from Eq. (2.8) and the last equality follows from the continuity of $f^{(d)}$ This completes the proof of Lemma B.5. □

B.6 Main Result

In this section, we prove main result Theorem B.6 using Lemma B.2 and Lemma B.3.

Theorem B.6 (Formal statement of Theorem 2.1.14)**.** *For any positive integer $n\geq 2$, the function $f:\mathbb{R}\to\mathbb{R}$ preserves low rank matrices if and only if f is a polynomial of degree less than $\lceil\log_2(n)\rceil$.*

Proof. Suppose f is a Taylor expandable function. By Lemma B.2, we can find $M:\mathbb{R}^d\to\mathbb{R}^{n\times n}$ and a Taylor expandable $\lambda_i^f:\mathbb{R}^{\log n+1}\to\mathbb{R}$ such that the image of M has rank $\leq\log n+1$, and $\{\lambda_i^f(a)\}_{i\in[n]}$ is the full set of eigenvalues of $M(a)$. Further, if there exists $i\in[n]$ with function $\lambda_i^f=0$, then f is a degree $d\leq\log n+1$ polynomial.

Now, suppose that f is a function that transforms all rank $\log n+1$ matrices to rank $<n$ matrices. Then it must transform all matrices $M(a)$ to rank $<n$ matrices. By Lemma B.3, it must follow that $\lambda_i^f=0$ for some i. However, we just established via Lemma B.2 that if $\lambda_i^f=0$, then f is a degree $d\leq\log n+1$ polynomial. This completes the proof of Theorem B.6. □

C Transforming Manhattan to Euclidean

In this section, we prove Theorem C.2, which states that functions f that transform Manhattan distances to squared Euclidean distances are Bernstein. This section is organized as follows

- In Section C.1, we show that any function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ transforming Manhattan to squared Euclidean is increasing. This serves as a warm-up for the general result which involves higher difference operations.
- In Section C.2, we prove the main result of this section, Theorem C.2.
- In Section C.3, Lemma C.3 shows f must be bounded and continuous. This lemma is used in the proof of our main result.

C.1 Useful Computations

Lemma C.1. *If f transforms Manhattan to squared Euclidean, then f is increasing on $\mathbb{R}_+$.*

Proof. We fix $c > 0$ and show $f'(c) \geq 0$. Consider $\chi : [d] \to \{0, 1\}$ which transforms 1 to 1 and everything else to 0. Let $a_1 = \epsilon$ and $a_2, \ldots a_d = \frac{2c}{d}$. Here, ϵ is a constant which we will adjust later.

The eigenvalue corresponding to χ (by Lemma F.2) is, by straightforward calculation:

$$\sum_{s=0}^{d-1} \binom{d-1}{s} \left(f\left(\frac{2cs}{d}\right) - f\left(\frac{2cs}{d} + \epsilon\right)\right) \tag{2.9}$$

If we divide by 2^{d-1} and take d to to infinity, the quantity in Eq. (2.9) becomes

$$f(c) - f(c + \epsilon)$$

for continuous functions f. Indeed, nearly all of the probability mass in the binomial coefficients concentrates around $s = d/2$ by the law of large numbers and the limit follows from continuity of f and the boundedness of f on bounded sets established below in Lemma C.3.

Applying Lemma A.8, we see that if f transforms Manhattan to squared Euclidean distances, then $f(c) - f(c + \epsilon) \leq 0$ for any $\varepsilon > 0$. This implies the desired result. □

C.2 Main Results

The goal of this section is to prove Theorem C.2.

Theorem C.2 (Manhattan to squared Euclidean, formal version of part (1) $\Leftrightarrow$ part (3) of Theorem 2.1.9)**.** *If f transforms Manhattan distances to squared Euclidean distances, it must be Bernstein.*

Proof. Fix a k-tuple $\epsilon = (\epsilon_1, \ldots, \epsilon_k)$ of positive real numbers and define

$$\Delta_\epsilon^k(f, t) := f(t) - \sum_{i_1 \in [k]} f(t + \epsilon_{i_1}) + \sum_{i_1 < i_2 \in [k]} f(t + \epsilon_{i_1} + \epsilon_{i+2}) + \ldots + (-1)^k f\left(t + \sum_{i=1}^{k} \epsilon_i\right).$$

Consider χ that transforms $1, 2, \ldots k$ to 1 and everything else to 0. Let $a_i = \epsilon_i$ for $i \in [k]$ and $a_{k+1} \ldots a_d = \frac{2c}{d}$ where c, k and ϵ are fixed.

The eigenvalue corresponding to χ is, by direct calculation using Lemma F.2:

$$\lambda_\chi = \sum_{s=0}^{d-k} \binom{d-k}{s} \Delta_\epsilon^k(f, 2sc/d). \tag{2.10}$$

Eq. (2.10) is the d-dimensional analog of Eq. (2.9), and this eigenvalue must satisfy $\lambda_\chi \leq 0$ by Lemma A.8. Dividing by 2^{d-k} and taking d to infinity, we obtain:

$$\Delta_\epsilon^k(f, c) \leq 0.$$

This is because again the probability mass in the binomial coefficients in Eq. (2.10) concentrates around the $s = d/2$ coefficient, where we use continuity and boundedness of f for any compact set (guaranteed by Lemma C.3). By Proposition A.2 this implies f is Bernstein (Definition 2.1.8) since k, c were arbitrary. This completes the proof. □

C.3 Function Should be Bounded

The goal of this section is to prove Lemma C.3.

Lemma C.3. *Any function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that transforms Manhattan to squared Euclidean is bounded on bounded sets and continuous on $(0, \infty)$.*

Proof. By the triangle inequality, $f(x) \leq f(1/2) + f(1/2)$ for all $0 \leq x \leq 1$, so f is bounded on $[0, 1]$. By scaling, from now on we assume f is bounded by 1 on $[0, 1]$.

Now, we show f is continuous on $(0, 1)$. Suppose there is a discontinuity at some point $0 < p < 1$. This means that there exists some ε such that for all $\delta > 0$, there are $a, b \in [p - \delta, p + \delta]$ such that $f(a) - f(b) \geq \varepsilon$. Since $f(x) \leq 1$ for all $x \in [0, 1]$, this means that for all $\delta < \min\{p, 1 - p\}$, we have that $\frac{f(a)}{f(b)} > 1 + \varepsilon$.

Now, fix some ε satisfying the above, and some $n = 2k$. Consider points $x_1, \ldots, x_n$ partitioned into sets $A = x_1, \ldots, x_k$ and $B = x_{k+1}, \ldots, x_n$. For some small δ that we will choose later, pick $a, b \in [p - \delta, p + \delta]$ such that $\frac{f(a)}{f(b)} > 1 + \varepsilon$, and define the metric

$$d(x_i, x_j) := \begin{cases} 0 & i = j \\ a & i, j \in A, i \neq j \\ b & i \text{ or } j \text{ is in } B, i \neq j \end{cases}$$

Now apply f: this gives us some metric $d'(x_i, x_j)$ such that

$$d'(x_i, x_j) := \begin{cases} 0 & i = j \\ f(a) & i, j \in A, i \neq j \\ f(b) & i \text{ or } j \text{ is in } B, i \neq j \end{cases}$$

We show that matrix $D'_{i,j} := d'(x_i, x_j)$ is not negative type if n is sufficiently large (as a function of ε). Consider the vector

$$v = (1, 1, \ldots 1, -1, -1, \ldots -1)$$

with the first k coordinates are ones and the last k coordinates are negative ones. This is orthogonal to the all ones vector, but

$$\begin{aligned} v^\top D' v &= k(k-1)f(a) - 2k^2 f(b) + k(k-1)f(b) \\ &= k(k-1)f(a) - k(k+1)f(b). \end{aligned}$$

Since $\frac{f(a)}{f(b)} > 1+\varepsilon$, if we choose $n > 100/\varepsilon$, we will have that

$$k(k-1)\cdot f(a) - k(k+1)\cdot f(b) > 0.$$

Therefore, by Lemma A.7, d' does not embed into ℓ_2^2, Squared Euclidean space.

However, we show that if δ is sufficiently small (in terms of n, p), then $d(x_i, x_j)$ is embeddable into ℓ_1. First note that the metric $d_1(i,j)$ which equals 0 if $i=j$ and c for some constant $c>0$ is embeddable into ℓ_1, by transforming i to $x_i = \frac{c}{2}\cdot e_i$ for all i, where e_i is the ith unit vector. Likewise, the metric $d_{k,\ell}(i,j)$ which equals 0 if $i=j$ or if $i=k, j=\ell$ or $i=\ell, j=k$ and c otherwise is also embeddable into ℓ_1, by transforming i to $x_i = \frac{c}{2}\cdot e_i$, except ℓ which is sent to $x_\ell = x_k = \frac{c}{2}\cdot e_k$. Now, it is trivial to see that by adding a finite number of these metrics, we still get a metric that is embeddable into ℓ_1.

But, if $\frac{a}{b} \in \left[1-\frac{1}{10n^2}, 1+\frac{1}{10n^2}\right]$, then any metric such that $d(i,j) \in \{a,b\}$ for all a, b can be written as some positive finite combination of d_1 and $d_{k,\ell}$ over all $1 \le k < \ell \le n$.

Therefore, if f is discontinuous at p, we can set $n = \frac{100}{\varepsilon}$, $\delta = \frac{\min(p,1-p)}{100n^2}$, and the metric on $x_1, \ldots, x_n$ as defined previously. We will have that

$$\frac{a}{b} \in \left[1-\frac{1}{10n^2}, 1+\frac{1}{10n^2}\right]$$

whereas $\frac{f(a)}{f(b)} > 1+\varepsilon$, which means that while d is embeddable into ℓ_1, $d' = f(d)$ is not embeddable into ℓ_2^2. Thus, if f is discontinuous at p, we have that f cannot transform Manhattan Distances to Squared Euclidean distances.

By scaling the x-axis, we have that f is bounded on any interval $[0, a]$ and that f is continuous at all $x > 0$. □

D Transforming Manhattan to Manhattan

This section is organized as follows:

- Section D.1 provides some useful tools that are related to ℓ_1 distance, ℓ_2 distance and Hadamard transform.
- In Section D.2, we prove Theorem D.4 which is the main result.
- Section D.3 provide some discussions.

D.1 Useful Tools

Suppose f transforms Manhattan distance to squared Euclidean distance. By definition, f satisfies the following: for any n and any $x_1, \ldots x_n \in (\mathbb{R}^{\mathbb{N}}, \ell_1)$, there exist $p_1, \ldots p_n \in (\mathbb{R}^{\mathbb{N}}, \ell_2)$

such that $f(\|x_i - x_j\|_1) = \|p_i - p_j\|_2^2$. We can assume without loss of generality that points $x_1, x_2, \ldots x_n$ are distinct corners of a d dimensional hyperrectangle (Definition 2.2.1), and $n = 2^d$. This is because any point set in ℓ_1 can be embedded isometrically into ℓ_1 on corners of a hyperrectangle (Lemma A.3).

Lemma D.1. *Let $f : \mathbb{R} \to \mathbb{R}$. If $x_1, \ldots x_{2^d}$ are corners of a hyperrectangle (Definition 2.2.1), the matrix D where $D_{i,j} = f(\|x_i - x_j\|_1)$ must have eigenvectors which are the columns of H_d, where H_d is the Hadamard matrix of size 2^d by 2^d.*

Proof. This follows from Lemma F.1. We note that this lemma does not rely on any assumptions on f. □

Lemma D.2. *Let D be the matrix where $D_{i,j} = f(\|x_i - x_j\|_1)$, and let $M := -\frac{1}{2}\Pi D \Pi$. Then M has eigenvectors H_d.*

Proof. This follows from Lemma D.1 and the definition of M. It is critically important that the columns of H_d are orthogonal to the all ones vector (with the exception of the all ones column in H_d). □

Lemma D.3. *Let $M = H_d \Sigma H_d$ be an eigendecomposition of M, where M is defined as in Lemma D.2. If f transforms ℓ_1 to ℓ_2^2, then Σ has entirely non-negative entries.*

For each i, we use p_i to denote the i-th column of $P = \sqrt{\Sigma} H_d$, we have $\langle p_i, p_j \rangle = M_{i,j}$ and $f(\|x_i - x_j\|_1) = \|p_i - p_j\|_2^2$.

Proof. This follows from Lemma A.9 and Lemma D.2. □

D.2 Main Result

The goal of this section is to prove Theorem D.4.

Theorem D.4 (Manhattan to squared Euclidean, formal version of part (2) $\Leftrightarrow$ part (3) of Theorem 2.1.9)**.** *Any function that transforms Manhattan distances to squared Euclidean distances must transform Manhattan distances to Manhattan distances, and vice versa.*

Proof. Let p_i be defined as in Lemma D.3. By construction, the vectors p_i are a subset of the corners of a 2^d-dimensional hyperrectangle, with side lengths $\sqrt{\Sigma_{i,i}}$. Thus, the pairwise squared Euclidean distances between p_i are isometrically embeddable into ℓ_1 by Lemma A.4. In other words, $f(\|x_i - x_j\|_1) = \|p_i - p_j\|_2^2 = \|q_i - q_j\|_1$ for some $q_i \in \ell_1$ for all i, j. This shows that any f that transforms ℓ_1 to ℓ_2^2 transforms ℓ_1 to ℓ_1 as desired. □

Note that for any x_i, the vectors q_i are finite dimensional and can be explicitly written down in closed form.

D.3 Discussion and Extensions

In our proof of Theorem D.4, we exploited that our points $x_1, \dots x_n$ are points in a hyperrectangle, which has a vertex transitive group symmetry. Similar theories can be generated when the point set lives on any object with a vertex-transitive group symmetry, and the distance measure between points is some function of the Euclidean distance. Such objects include higher dimensional platonic solids, spheres, equilateral triangular prisms, and more.

We remark that the group symmetry must be vertex-transitive to ensure the matrix D in Lemma D.1 has an eigenvector equal to the all ones vector. If this were not the case, Lemma D.2 would no longer hold.

E Positive Definite Manhattan Kernels

The section is organized as follows:

- In Section E.1, we state a useful tool.
- In Section E.2, we present our main result. This result classifies all positive definite Manhattan kernels (Definition 2.1.4), and is a formal restatement of Theorem 2.1.5.

E.1 A Useful Tool

First, we prove the following lemma.

Lemma E.1. *If f is a positive definite Manhattan kernel (Definition 2.1.4), then $f(t) \geq 0$ for all $t \geq 0$.*

Proof. Let $\mathcal{X}$ denote metric space $(\mathbb{R}^N, \ell_1)$. For any $N \geq 0$ we consider the points $x_i = \frac{t}{2} e_i \in \mathcal{X}$ for $i \in [N]$ where $e_i = (0, \dots, 0, 1, 0, \dots, 0)$ is a standard basis vector, so that $\|x_i - x_j\|_1 = t$ for any $i \neq j$. Since the matrix of values $(f(\|x_i - x_j\|_1)_{i,j \in [N]}$ must be positive semidefinite, the sum of all its entries must be positive, hence:

$$N(f(0) + (N-1)f(t)) \geq 0.$$

The above equation implies the following:

$$\frac{f(0)}{N-1} + f(t) \geq 0$$

for all integer $N \geq 0$ and real $t \geq 0$.

Since N can be arbitrarily large, therefore we conclude $f(t) \geq 0$ as claimed.

□

E.2 Main Result

The goal of this section is to prove Theorem E.2.

Theorem E.2 (Formal statement of Theorem 2.1.5)**.** *$f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ is a positive definite Manhattan kernel (Definition 2.1.4) if and only if $f(x)$ is completely monotone (Definition 2.1.2).*

Proof. First, we prove that if f is a positive definite Manhattan kernel, then f must be completely monotone. The converse direction is previously known, and is a consequence of Lemma A.5 and Theorem 3 of [262][11].

Suppose that f is a positive definite Manhattan kernel (Definition 2.1.4). Cauchy-Schwarz easily implies that $f(t) \leq f(0)$ for all t, so f is bounded.. Now, if $x_1, \ldots, x_n$ correspond to $y_1, \ldots, y_n$ then

$$\begin{aligned} f(\|x_i - x_j\|_1) &= \langle y_i, y_j \rangle \\ &= f(0) - \frac{1}{2}\|y_i - y_j\|_2^2. \end{aligned}$$

Therefore $2(f(0) - f(t))$ (equivalently, $f(0) - f(t)$) sends Manhattan distances to squared Euclidean distances. Therefore $f(0) - f(t)$ is Bernstein (Definition 2.1.8), by Theorem 2.1.9. Combining with Lemma E.1 we conclude that f must be completely monotone (Definition 2.1.2). □

F Representation Theory of the Real Hyperrectangles

In this section, we prove Lemma F.2, the formal version of Lemma 2.2.3. This lemma uses representation theoretic ideas to compute the eigenvalues of matrices arising from the real hyperrectangle. We introduce Lemma F.3, which expresses these same eigenvalues in terms of integrals. This integral formulation is useful for proving Theorem B.6.

F.1 Useful Tools

Lemma F.1. *Let $g : (\mathbb{R}^d \times \mathbb{R}^d) \to \mathbb{R}$ such that $g(x, y)$ is invariant under axis reflection. Consider a d-dimensional hyperrectangle with corners $x_1, \ldots x_{2^d}$. Let D be a 2^d by 2^d matrix such that $D_{ij} = g(x_i, x_j)$. Then there is an eigendecomposition of D into $H_d \Sigma H_d$ where Σ is a diagonal matrix.*

Proof. This lemma can be proven directly via computation. However, it is more instructive to view this through the representation theoretic lens. We note that D has the property that for any permutation matrix σ corresponding to a reflection about one of the hyperrectangle's axes, we have $\sigma D = D\sigma$. Schur's lemma from representation theory (see Lemma A.10) states that D and all σ in the reflectional symmetry group of the hyperrectangle have a common set of eigenvectors. It is straightforward to verify that the only common set of eigenvectors for all σ is the columns of the Hadamard matrix, and thus D must have the columns of H_d as its eigenvectors. □

We note that variants of this lemma are used to prove Delsarte's linear programming bound in error correcting codes [104, 232].

[11] Theorem 3 of [262] is a modern restatement of Schoenberg's work in [251]

F.2 Main Result

Lemma F.2 (Representation theory of the real hyperrectangle, formal version of Lemma 2.2.3)**.** *Consider a d-dimensional hyperrectangle (Definition 2.2.1) parameterized by $a_1, \dots a_d > 0$. Enumerate the vertices in lexicographical ordering as $p_1, \dots p_{2^d}$.*

For any $f : \mathbb{R} \to \mathbb{R}$, let D be the 2^d by 2^d matrix given by $D_{i,j} = f(\|p_i - p_j\|_1)$. Then:

1. *$\Sigma := H_d D H_d$ is a diagonal matrix whose entries are the eigenvalues of D multiplied by 2^d, and $D = 4^{-d} \cdot H_d \Sigma H_d$.*
2. *Let $\chi : [d] \to \{0, 1\}$. Let k equal the integer corresponding to transforming χ (written as a d dimensional binary vector) into an integer via binary conversion. For each χ, there is an eigenvector of D equal to the k-th column of Hadamard matrix H_d, and its associated eigenvalue is:*

$$\sum_{T \subseteq [d]} (-1)^{\sum_{t \in T} \chi(t)} f\left(\sum_{t \in T} a_t\right). \tag{2.11}$$

The second part of this theorem on its surface differs from that in Lemma 2.2.3, but the statements are in fact identical via straightforward computation.

Proof. By Lemma F.1, we know that the Hadamard matrix columns are eigenvectors of the matrix D. The result follows by direct computation, noting that the formula in Eq. (2.11) is the d dimensional analog of Eq. (2.4), and can be derived in the same way. □

We now give an alternate formulation of the eigenvalues in Lemma F.2. This lemma is of independent interest.

Lemma F.3. *Given a box with side lengths $a_1, \dots a_d$, each eigenvalue analogous to those in Eq.* (2.4) *corresponds to a function $\chi : [d] \to \{0, 1\}$. Let $Q = \{q_1, \dots q_k\}$ be the full set of values on which χ is* 1*. Then the Eigenvalues in Eq.* (2.11) *equal:*

$$\sum_{T \subseteq [d] \setminus Q} \int_{\sum_{t \in T} a_t}^{a_{q_1} + \sum_{t \in T} a_t} \cdots \int_{\sum_{t \in T} a_t}^{a_{q_k} + \sum_{t \in T} a_k} (-1)^k \frac{\mathrm{d}^k f}{\mathrm{d}x^k}\Big(\sum_{q \in Q} s_q\Big) \mathrm{d}s_1 \dots \mathrm{d}s_k.$$

Proof. The proof is identical to that of Eq. (2.5), but for d dimensions. It follows directly from Lemma F.2 combined with the fundamental theorem of calculus. □

Chapter 3

Data-Sensitive Distances

Data-sensitive metrics adapt distances locally based the density of data points with the goal of aligning distances and some notion of similarity. In this chapter, we give the first exact algorithm for computing a data-sensitive metric called the nearest neighbor metric. In fact, we prove the surprising result that a previously published 3-approximation is an exact algorithm.

The nearest neighbor metric can be viewed as a special case of a density-based distance used in machine learning, or it can be seen as an example of a manifold metric. Previous computational research on such metrics despaired of computing exact distances on account of the apparent difficulty of minimizing over all continuous paths between a pair of points.

We leverage the exact computation of the nearest neighbor metric to compute sparse spanners and persistent homology. We also explore the behavior of the metric built from point sets drawn from an underlying distribution and consider the more general case of inputs that are finite collections of path-connected compact sets.

The main results connect several classical theories such as the conformal change of Riemannian metrics, the theory of positive definite functions of Schoenberg, and screw function theory of Schoenberg and Von Neumann. We also develop some novel proof techniques based on the combination of screw functions and Lipschitz extensions that may be of independent interest.

1 Introduction

The profound success of nonlinear methods in machine learning such as kernels methods, density-based distances, and neural nets reveals that although data are often represented as points in $\mathbb{R}^n$, the shortest path between two points is *not* a straight line. It is widely believed that a more useful metric on the data points would have the property that two points in a dense cluster will be close in some underlying metric, even if the Euclidean distance is far [10, 49, 94, 289]. That is, distances are scaled inversely according to the density of the data along a path between points. We call such a metric **data-sensitive**.

Data-sensitive metrics arise naturally in machine learning, and are implicitly central in celebrated methods such as k-NN graph methods, manifold learning, level-set methods, single-linkage clustering, and Euclidean MST-based clustering (see Section 5 and Appendix G for details). The construction of appropriate data-sensitive metrics is an active area of research. We

consider a simple data-sensitive metric with an underlying manifold structure called the **nearest neighbor metric**. This metric was first introduced in [94]. It and its close variants have been studied in the past by multiple researchers [49, 94, 158, 247, 289]. In this chapter, we show how to compute the nearest neighbor metric exactly for any dimension, which solves one of the most important and challenging problems for any manifold-based metric.

The starting point will be the nearest neighbor function $\mathbf{r}_P$ for the data set P:

$$\mathbf{r}_P(z) = 4 \min_{x \in p} \|x - z\|,$$

where the factor of 4 normalizes and simplifies expressions later. This function is also known as the distance function to the set P and is the basic object of study in the critical point theory of distance functions, a generalization of Morse Theory [145]. This theory has found many recent uses in computational geometry [72, 74] as it is a natural way to infer underlying structure from a sample of points. We have a similar goal of inferring underlying structure when we use $\mathbf{r}_P$ as a cost function for a density-based distance defined as follows (see also Section 4 for explicit inference results).

Definition 1.1. *Given a continuous cost function $c : \mathbb{R}^k \to \mathbb{R}$, we define the density-based cost of a path γ relative to c as:*

$$\ell_c(\gamma) = \int_0^1 c(\gamma(t))\|\gamma'(t)\|dt.$$

*Here, the path γ is defined as a continuous map $\gamma : [0, 1] \to \mathbb{R}^k$. Let $\mathrm{path}(a, b)$ denote the set of piecewise-C_1 paths from a to b. We then define the **density-based distance** between two points $a, b \in \mathbb{R}^k$ as*

$$d_c(a, b) \inf_{\gamma \in \mathrm{path}(a,b)} \ell_c(\gamma)$$

This is a slight simplification of the density-based distances from [247] which included other requirements to facilitate approximation. Conceptually, the density-based cost of a path is the weighted path length, where each infinitesimal path piece is weighted according to c. The cost c is usually some function of an underlying density f (the natural choice would be $c(x) = f(x)^{-\frac{1}{k}}$). Density-based distances have been notable in the machine learning setting for over a decade [49, 247]. To build a data-sensitive metric from density-based distances, we would like a cost function c that is small when close to the data set, and large when far away. The nearest neighbor function $\mathbf{r}_P$ is the most natural candidate, and has been traditionally used as a proximity measure between points and a data set in both the geometry and machine learning settings [49]. It has been used as such in nearest neighbor (and k-NN) classification, k-means/medians/center clustering, finite element methods, and any of the numerous methods that use Voronoi diagrams or Delaunay triangulation as intermediate data structures.

Definition 1.2. *Given any finite set $P \subset \mathbb{R}^k$, the **nearest neighbor cost** function is $\ell_N := \ell_{\mathbf{r}_P}$ and the **nearest neighbor metric** is $\mathbf{d}_N := \mathbf{d}_{\mathbf{r}_P}$. That is, it's the density-based distance with cost function $\mathbf{r}_P$.*

The nearest neighbor metric, and density-based distances in general, are examples of manifold geodesics [247, 278]. Manifold geodesics of data sets are defined by embedding points into a manifold and computing the infimum length path in the manifold. Within computer science,

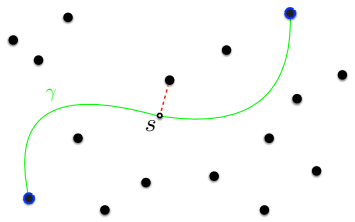

Figure 3.1: In this figure we have a collection of points. The length or cost of the green curve between the two blue points is the integral along the curve scaled by the distance to the nearest point.

dozens of foundational papers in machine learning and surface reconstruction rely on manifold-based metrics to perform clustering, classification, regression, surface reconstruction, persistent homology, and more [10, 49, 94, 114, 214, 247, 278, 289]. Manifold geodesics predate computer science, and are the cornerstone of many fields of physics and mathematics. Exactly computing geodesics is fundamental to countless areas of physics including: the brachistochrone and minimal-drag-bullet problem of Bernoulli and Newton [46], exactly determining a particle's trajectory in classical physics (Hamilton's Principle of Least Action) [98], computing the path of light through a non-homogeneous medium (Snell's law), finding the evolution of wave functions in quantum mechanics over time (Feynman path integrals [122]), and determining the path of light in the presence of gravitational fields (General Relativity, Schwarzschild metric) [257, 277]. In mathematics, manifold geodesics appear in many branches of higher mathematics including differential equations, differential geometry, Lie theory, calculus of variations, algebraic geometry, and topology.

One of the most significant problems on any manifold geodesic is how to compute its length. Exact computation of manifold metrics is considered a fundamental problem in mathematics and physics, dating back for four centuries: entire fields of mathematics, including the celebrated calculus of variations, have arisen to tackle this [98]. Historically, mathematicians placed strong emphasis on exact computation as opposed to constant factor approximations [98]. An algorithmic problem on manifold geodesics, with modern origins, is to $(1+\varepsilon)$ approximate these metrics efficiently on a computer. The core difficulty in the first problem is that geodesics are the minimum cost path out of an uncountable number of paths that can travel 'anywhere' on the manifold structure. This makes exactly computing these metrics challenging, even in the case of the nearest neighbor metric for just four fixed points in two dimensions (the authors are unaware of any easy method for this simplified task). Calculus of variations can show that the optimal nearest neighbor path is piecewise hyperbolic, but this is generally insufficient to exactly compute the nearest neighbor metric—there are point sets where there are many differentiable, piecewise hyperbolic paths between two data points with different costs.

In this chapter, we solve both problems: we exactly compute the Nearest Neighbor metric in all cases, and we $(1+\varepsilon)$ approximate it quickly. Our approach is based on a novel embedding of the data into high dimensions where the geodesics are straight lines. Then we use a Lipschitz extension theorem to relate the lengths of the shortest paths in the original space and the embedding. We combine these tools to prove that the nearest neighbor metric is exactly equal to a

shortest path distance on a geometric graph, the so-called edge-squared metric, in all cases. This allows us to compute the nearest-neighbor metric exactly for any given point set in polynomial time, and it is the only known (non-trivial) density-based distance that can be computed by a discrete algorithm.

Definition 1.3. *For $x \in \mathbb{R}^d$, let $\|x\|$ denote the Euclidean norm. For a set of points $P \subset \mathbb{R}^d$: the **edge-squared metric** for $a, b \in P$ is*

$$\mathbf{d}_2(a,b) = \inf_{(p_0,\ldots,p_k)} \sum_{i=1}^{k} \|p_i - p_{i-1}\|^2,$$

where the infimum is over sequences of points $p_0, \ldots, p_k \in P$ with $p_0 = a$ and $p_k = b$.

Theorem 1.4. *The nearest neighbor metric and edge squared metric are equivalent for any set P in arbitrary dimension that is the finite collection of compact path-connected sets.*

This in particular covers the case of n points in $n-1$ dimension. The exact equality is realized when the nearest neighbor path is piecewise linear, traveling straight from data point to data point. The edge squared metric has been previously studied by multiple researchers in machine learning and power-efficient wireless networks, but previously has only been linked to the nearest neighbor metric by a fairly weak 3-approximation [94]. There are several reasons why it is surprising that these metrics are equal:

1. The optimal nearest neighbor path for two points not in the dataset is generally composed of hyperbolic arcs. This holds true even when the dataset is a single point, and was established by [94] using tools in Riemannian surfaces and the complex plane. Meanwhile, our Theorem implies an optimal nearest neighbor path for two data points (in a dataset of any size) is piecewise linear!
2. There are simple and natural variants of the nearest neighbor metric, for which no analog of Theorem 1.4 is known nor suspected. For example, if one considers powers (other than one) of the distance function as a cost, a corresponding graph-based metric is known to exist only for sets of size at most two.
3. For just three points in a right triangle configuration, there exist an uncountable suite of optimal-cost paths between the two endpoints of the hypotenuse. Each path in this uncountable suite is piecewise hyperbolic, but, surprisingly, they all have the exact same cost as the edge-squared distance. Thus, there shortest paths may not even be unique.
4. The finite union of compact path-connected geometric bodies in arbitrary dimension can have extremely complicated geometry, and the Voronoi diagram on which the nearest neighbor metric depends is poorly understood for even three of these bodies in two dimension. There is no other restriction on the compact geometric objects, and they need not be convex or even simply connected, see figure 3.2.

We can now tackle a second problem of interest for manifold geodesics, which is efficiently $(1+\varepsilon)$ approximating them. In this chapter, we show that the nearest neighbor metric admits $(1+\epsilon)$ spanners computable in nearly-linear time, with linear size, for any point set in constant dimension. Remarkably, these spanners are significantly sparser and faster to compute than the theoretically optimal Euclidean spanners with the same approximation constant, and nearly

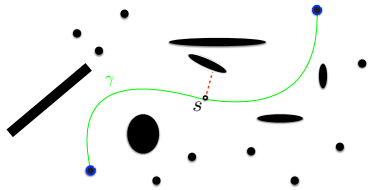

Figure 3.2: In this figure we have a collection of compact bodies in black. The length or cost of the green curve between the two blue points is the integral along the curve scaled by the distance to the nearest body. A curve may traverse a body at no cost. Theorem 1.4 establishes that the shortest path curve between two points goes straight from compact body to compact body.

match the sparsity of the best known Euclidean Steiner spanners. Moreover, if the point set comes from a well-behaved probability distribution in constant dimension (a foundational assumption in machine learning [158]), we show that the nearest neighbor metric has perfect 1-spanners of nearly linear size. The latter result is impossible for many non-density sensitive metrics, such as the Euclidean metric. Both results rely on Theorem 1.4, and significantly improve the nearest neighbor spanners of Cohen et al in [94].

Theorem 1.4 and our spanner theorems solve two core problems of interest for the nearest neighbor metric: exactly computing it for any dimension, and approximating it quickly for both general point sets and point sets arising from a well-behaved probability distribution in constant dimension. This is the first work we know of that computes a manifold metric exactly without calculus of variations, and we hope that our tools can be useful for other metric computations and approximations.

1.1 Contributions and Past Work

Our primary contribution is Theorem 1.4, which lets us exactly compute the nearest neighbor metric. This significantly strengthens a core result of Cohen et al [94]. This theorem should be considered quite surprising: it equates the nearest neighbor metric with the edge-squared metric, even when the point set is a collections of compact, path-connected objects in arbitrarily large dimension. There are no restrictions on the convexity or simple-connectedness of such objects, so in general the Voronoi diagram of these objects (on which the nearest neighbor metric critically depends) can be extremely complicated.

Besides for exactly computing the nearest neighbor metric, we present the following theorems on approximate computation:

Theorem 1.5. *For any set of points in $\mathbb{R}^d$ for constant d, there exists a $(1+\varepsilon)$ spanner of the nearest neighbor metric with size $O\left(n\varepsilon^{-d/2}\right)$ computable in time $O\left(n\log n + n\varepsilon^{-d/2}\log\frac{1}{\varepsilon}\right)$. The $\log\frac{1}{\varepsilon}$ term goes away given access to an algorithm computing floor function in $O(1)$ time.*

Theorem 1.6. *Suppose points P in Euclidean space are drawn i.i.d from a Lipschitz probability density bounded above and below by a constant, with support on a smooth, connected, compact manifold with intrinsic dimension d with boundary of bounded curvature. Then w.h.p. the k-NN graph of P for $k = O(2^d \ln n)$ and edges weighted with Euclidean distance squared, is a*

1-*spanner of the nearest neighbor metric on* P.

These theorems rely on Theorem 1.4 and considerably strengthen the spanner results on the nearest neighbor metric from [94]. They critically rely on Theorem 1.4, which show it suffices to compute spanners of the edge-squared metric. Previously, sparse spanners of the edge-squared metric were shown to exist in two dimensions via Yao graphs and Gabriel graphs [202], but these did not generalize well to constant dimension: Yao graphs are not very efficient to compute, and Gabriel graphs can have quadratically many edges even in 3 dimensions [76]. The spanners we produce are sparser than the theoretical optimal for Euclidean spanners [191].

Theorem 1.6 proves that a 1-spanner of the nearest neighbor metric can be found assuming points are samples from a probability density, by using a k-NN graph for appropriate k. Our result is tight when d is constant. This is not possible for Euclidean distance, as a 1-spanner is almost surely the complete graph. Although the restrictions on the probability density may seem limiting, they are in fact quite flexible and standard in machine learning theory and practice [10, 158]. For example, although they do not cover the case of a Gaussian (unbounded support), they do cover the case of a Gaussian where the very thin tail is cut off, and this recovers most of the relevant data in a Gaussian distribution. Past work on similar results include [38, 133].

Theorem 1.4 will additionally allow us to compute the persistent homology of $\mathbf{d}_N$, a task useful for topological data analysis [114]. We also show how the nearest neighbor metric generalizes Euclidean distance and maximum-edge Euclidean MST distance [202]

The core mathematical contribution of our work is the statement and proof of Theorem 1.4. The techniques to prove our other results are simpler and mostly leverage Theorem 1.4 and past work. We have included them nonetheless to provide a more complete picture of the nearest neighbor metric, and to provide possible directions for future work.

1.2 Definitions and Preliminaries

In this section, we establish additional definitions for this chapter. These are mostly of interest for our spanner and persistent homology results, and are not strictly necessary for Theorem 1.4.

Spanners: For real value $t \geq 1$, a t-spanner of a weighted graph G is a subgraph S such that $d_G(x, y) \leq d_S(x, y) \leq t \cdot d_G(x, y)$ where d_G and d_S represent the shortest path distance functions between vertex pairs in G and S. Spanners of Euclidean distances, and general graph distances, have been studied extensively, and their importance as a data structure is well established. [62, 86, 150, 284].

k-nearest neighbor graphs: The k-nearest neighbor graph (k-NN graph) for a set of objects V is a graph with vertex set V and an edge from $v \in V$ to its k most similar objects in V, under a given distance measure. In this chapter, the underlying distance measure is Euclidean, and the edge weights are Euclidean distance squared. k-NN graph constructions are a key data structure in machine learning [80, 109], clustering [214], and manifold learning [278].

Gabriel Graphs: The Gabriel graph is a graph where two vertices p and q are joined by an edge if and only if the disk with diameter pq has no other points of S in the interior. The Gabriel graph is a subgraph of the Delaunay triangulation [273], and a 1-spanner of the edge-squared metric [273]. Gabriel graphs will be used in the proof of Theorem 1.6.

Persistent Homology: Persistent homology is a popular tool in computational geometry and topology to ascribe quantitative topological invariants to spaces that are stable with respect to perturbation of the input. In particular, it's possible to compare the so-called persistence diagram of a function defined on a sample to that of the complete space [73]. These two aspects of persistence theory—the intrinsic nature of topological invariants and the ability to rigorously compare the discrete and the continuous—are both also present in our theory of nearest neighbor metrics. Indeed, our primary motivation for studying these metrics was to use them as inputs to persistence computations for problems such as persistence-based clustering [75] or metric graph reconstruction [1].

2 Outline

Section 3 contains the proof of Theorem 1.4, equating the edge-squared metric and nearest neighbor metric in all cases. It should be noted that our proof is robust enough to handle not just finite point sets, but also countably infinite collections of disjoint path-connected, compact sets. Remarkably, there is no restriction on the convexity or simply-connectedness of these sets.

As an example of using the nearest neighbor metric to compute intrinsic structure, Section 4 shows how Theorem 1.4 allows us to compute the persistent homology of the nearest neighbor metric.

Section 5 introduces the p-power metrics. We show that Euclidean spanners and Euclidean MSTs are special cases of p-power spanners. We show how clustering algorithms including k-means, level-set methods, and single linkage clustering, are special cases of clustering with p-power metrics. p-power metrics are identical to the Neighbor metric when $p = 2$. This is further detailed in Appendix G.

Section 6.2 outlines a proof of Theorem 1.5, and compares our spanner to new lower bounds on the sparsity of $(1+\varepsilon)$-spanners of the Euclidean metric. We outline a proof of Theorem 1.6 in Section 6.1 and discuss its implications.

Conclusions and open questions are in Section 7. Full proofs for Theorems 1.6, 1.5 are contained in the Appendix.

3 Exactly Computing the nearest neighbor metric

In this section, we prove Theorem 1.4 on finite point sets, and explain in Section 3.1 that our proof strategy applies to finite collections of path-connected compact bodies.

First, lets observe what happens when P has only two points a and b, $\mathbf{d}_2(a,b) = \mathbf{d}_N(a,b)$. This reduces to a high school calculus exercise as the minimum path γ will be a straight line between the points and the nearest neighbor metric is

$$\begin{aligned}\mathbf{d}_N(a,b) &= 4\int_0^1 \mathbf{r}_P(\gamma(t))\|\gamma'(t)\|dt \\ &= 8\int_0^{\frac{1}{2}} t\|a-b\|^2 dt = \|a-b\|^2 = \mathbf{d}_2(a,b).\end{aligned}$$

Now it is easy to observe that the nearest neighbor metric is never greater than the edge-squared distance, as proven in the following lemma.

Lemma 3.1. *For all $s, p \in P$, we have $\mathbf{d}_N(s,p) \leq \mathbf{d}_2(s,p)$.*

Proof. Fix any points $s, p \in P$. Let $q_0, \ldots, q_k \in P$ be such that $q_0 = s$, $q_k = p$ and

$$\mathbf{d}_2(s,p) = \sum_{i=1}^{k} \|q_i - q_{i-1}\|^2.$$

Let $\psi_i(t) = tq_i + (1-t)q_{i-1}$ be the straight line segment from q_{i-1} to q_i. Observe that $\ell(\psi_i) = \|q_i - q_{i-1}\|^2/4$, by the same argument as in the two point case. Then, let ψ be the concatenation of the ψ_i and it follows that

$$\mathbf{d}_2(s,p) = 4\ell(\psi) \geq 4 \inf_{\gamma \in \mathrm{path}(s,p)} \ell(\gamma) = \mathbf{d}_N(s,p).$$

□

By Lemma 3.1, it suffices to show that $\mathbf{d}_N(a,b) \geq \mathbf{d}_2(a,b)$ for all $a, b \in P$.

Let $P \subset \mathbb{R}^d$ be a set of n points. Pick any *source* point $s \in P$. Order the points of P as $p_1, \ldots, p_n$ so that

$$\mathbf{d}_2(s,p_1) \leq \cdots \leq \mathbf{d}_2(s,p_n).$$

This will imply that $p_1 = s$. It will suffice to show that for all $p_i \in P$, we have $\mathbf{d}_2(s,p_i) = \mathbf{d}_N(s,p_i)$. There are three main steps:

1. We first show that when P is a subset of the vertices of an axis-aligned box, $\mathbf{d} = \mathbf{d}_N$. In this case, shortest paths for $\mathbf{d}$ are single edges and shortest paths for $\mathbf{d}_N$ are straight lines.
2. We then show how to lift the points from $\mathbb{R}^d$ to $\mathbb{R}^n$ by a Lipschitz map m that places all the points on the vertices of a box and preserves $\mathbf{d}_2(s,p)$ for all $p \in P$.
3. Finally, we show how the Lipschitz extension of m is also Lipschitz as a function between nearest neighbor metrics. We combine these pieces to show that $\mathbf{d} \leq \mathbf{d}_N$. As $\mathbf{d} \geq \mathbf{d}_N$ (Lemma 3.1), this will conclude the proof that $\mathbf{d} = \mathbf{d}_N$.

The key to the second step, to be elaborated in Section 3, is that if you take points on a line and raise the pairwise distances to the $1/2$ power, you get points on a box. This is a special case of the general theory on screw functions developed by Von Neumann and Schoenberg, which asserts a far more general criterion on when functions applied to pairwise distances between points on a line can be embedded into Euclidean space [228].

Boxes

Let Q be the vertices of a box in $\mathbb{R}^n$. That is, there exist some positive real numbers $\alpha_1, \ldots, \alpha_n$ such that each $q \in Q$ can be written as $q = \sum_{i \in I} \alpha_i e_i$, for some $I \subseteq [n]$.

Let the source s be the origin. Let $\mathbf{r}_Q : \mathbb{R}^n \to \mathbb{R}$ be the distance function to the set Q. Setting $r_i(x) := \min\{x_i, \alpha_i - x_i\}$ (a lower bound on the difference in the ith coordinate to a vertex of

the box), it follows that

$$\mathbf{r}_Q(x) \geq \sqrt{\sum_{i=1}^{n} r_i(x)^2}. \tag{3.1}$$

Let $\gamma : [0,1] \to \mathbb{R}^n$ be a curve in $\mathbb{R}^n$. Define $\gamma_i(t)$ to be the projection of γ onto its ith coordinate. Thus,

$$r_i(\gamma(t)) = \min\{\gamma_i(t), \alpha_i - \gamma_i(t)\} \tag{3.2}$$

and

$$\|\gamma'(t)\| = \sqrt{\sum_{i=1}^{n} \gamma_i'(t)^2}. \tag{3.3}$$

We can bound the length of γ as follows. For simplicity of exposition we only present the case of a path from the origin to the far corner, $p = \sum_{i=1}^{n} \alpha_i e_i$.

$$\begin{aligned}
\ell(\gamma) &= \int_0^1 \mathbf{r}_Q(\gamma(t))\|\gamma'(t)\|dt \\
&\quad \text{[by definition]} \\
&\geq \int_0^1 \left(\sqrt{\sum_{i=1}^{n} r_i(\gamma(t))^2} \sqrt{\sum_{i=1}^{n} \gamma_i'(t)^2} \right) dt \\
&\quad \text{[by (3.1) and (3.3)]} \\
&\geq \sum_{i=1}^{n} \int_0^1 r_i(\gamma(t))\gamma_i'(t)dt \\
&\quad \text{[by Cauchy-Schwarz]} \\
&\geq \sum_{i=1}^{n} \left(\int_0^{\ell_i} \gamma_i(t)\gamma_i'(t)dt + \int_{\ell_i'}^1 (\alpha_i - \gamma_i(t))\gamma_i'(t)dt \right) \\
&\quad \text{[by (3.2) where } \gamma_i(\ell_i) = \alpha_i/2 \text{ for the first time} \\
&\quad \text{and } \gamma_i(\ell_i') = \alpha_i/2 \text{ for the last time.]} \\
&= \sum_{i=1}^{n} 2 \int_0^{\ell_i} \gamma_i(t)\gamma_i'(t)dt \\
&\quad \text{[by symmetry]} \\
&\geq \sum_{i=1}^{n} \frac{\alpha_i^2}{4} \\
&\quad \text{[by basic calculus]}
\end{aligned}$$

It follows that if γ is any curve that starts at s and ends at $p = \sum_{i=1}^{n} \alpha_i e_i$, then $\mathbf{d}_N(s,p) = \mathbf{d}_2(s,p)$.

Lifting the points to $\mathbb{R}^n$

Define a mapping $m : P \to \mathbb{R}^n$. We do this by adding the points $p_1, \ldots, p_n$, as defined above, one point at a time. For each new point we will introduce a new dimension. We start by setting $m(p_1) = 0$ and by induction:

$$m(p_i) = m(p_{i-1}) + \sqrt{\mathbf{d}_2(s, p_i) - \mathbf{d}_2(s, p_{i-1})}e_i, \tag{3.4}$$

where the vectors e_i are the standard basis vectors in $\mathbb{R}^n$. A similar embedding works for some other functions and was extensively studied by Schoenberg and Von Neumann in the theory of screw functions.

Lemma 3.2. *For all $p_i, p_j \in P$, we have*

(i) $\|m(p_j) - m(p_i)\| = \sqrt{|\mathbf{d}_2(s, p_j) - \mathbf{d}_2(s, p_i)|}$, *and*

(ii) $\|m(s) - m(p_j)\|^2 \leq \|m(p_i)\|^2 + \|m(p_i) - m(p_j)\|^2$.

Proof. Proof of (i). Without loss of generality, let $i \leq j$. Then, by the definition of m, expanding the norm, and telescoping the sum, we get the following.

$$\begin{aligned}
&\|m(p_j) - m(p_i)\| \\
&= \left\| \sum_{k=i+1}^{j} \sqrt{\mathbf{d}_2(s, p_k) - \mathbf{d}_2(s, p_{k-1})}e_k \right\| \\
&= \sqrt{\sum_{k=i+1}^{j} (\mathbf{d}_2(s, p_k) - \mathbf{d}_2(s, p_{k-1}))} \\
&= \sqrt{\mathbf{d}_2(s, p_j) - \mathbf{d}_2(s, p_i)}.
\end{aligned}$$

Proof of (ii). As $m(s) = 0$, it suffice to observe that

$$\begin{aligned}
\|m(p_j)\|^2 &= \mathbf{d}_2(s, p_j) && [\text{by } \textit{(i)}] \\
&\leq \mathbf{d}_2(s, p_i) + |\mathbf{d}_2(s, p_j) - \mathbf{d}_2(s, p_i)| \\
&= \|m_(p_i)\|^2 + \|m(p_i) - m(p_j)\|^2 && [\text{by } \textit{(i)}]
\end{aligned}$$

□

We can now show that m has all of the desired properties.

Proposition 3.3. *Let $P \subset \mathbb{R}^d$ be a set of n points, let $s \in P$ be a designated source point, and let $m : P \to \mathbb{R}^n$ be the map defined as in* (3.4)*. Let $\mathbf{d}'$ denote the edge squared metric for the point set $m(P)$ in $\mathbb{R}^n$. Then,*

(i) m is 1-Lipschitz as a map between Euclidean metrics,

(ii) m maps the points of P to the vertices of a box, and

(iii) m preserves the edge squared distance to s, i.e. $\mathbf{d}'(m(s), m(p)) = \mathbf{d}_2(s, p)$ for all $p \in P$.

Proof. Proof of (i). To prove the Lipschitz condition, fix any $a, b \in P$ and bound the distance as follows.

$$\begin{aligned}
\|m(a) - m(b)\| &= \sqrt{|\mathbf{d}_2(s,a) - \mathbf{d}_2(s,b)|} && \text{[Lem. 3.2(i)]} \\
&\leq \sqrt{\mathbf{d}_2(a,b)} && \text{[triangle ineq.]} \\
&\leq \|a - b\|. && \text{[by def. of } \mathbf{d}_2\text{]}
\end{aligned}$$

Proof of (ii). That m maps P to the vertices of a box is immediate from the definition. The box has side lengths $\|m_i - m_{i-1}\|$ for all $i > 1$ and $p_i = \sum_{k=1}^{i} \|m_k - m_{k-1}\| e_k$.
Proof of (iii). We can now show that the edge squared distance to s is preserved. Let $q_0, \ldots, q_k$ be the shortest sequence of points of $m(P)$ that realizes the edge-squared distance from $m(s)$ to $m(p)$, i.e., $q_0 = m(s)$, $q_k = m(p)$, and

$$\mathbf{d}'(m(s), m(p)) = \sum_{i=1}^{k} \|m(q_i) - m(q_{i-1})\|^2.$$

If $k > 1$, then Lemma 3.2(ii) implies that removing q_1 gives a shorter sequence. Thus, we may assume $k = 1$ and therefore, by Lemma 3.2(i),

$$\mathbf{d}'(m(s), m(p)) = \|m(s) - m(p)\|^2 = \mathbf{d}_2(s, p).$$

□

The Lipschitz Extension

Proposition 3.3 and the Kirszbraun theorem on Lipschitz extensions imply that we can extend m to a 1-Lipschitz function $f : \mathbb{R}^d \to \mathbb{R}^n$ such that $f(p) = m(p)$ for all $p \in P$ [60, 178, 285].
Lemma 3.4. *The function f is also 1-Lipschitz as mapping from $\mathbb{R}^d \to \mathbb{R}^n$ with both spaces endowed with the nearest neighbor metric.*

Proof. We are interested in two distance functions $\mathbf{r}_P : \mathbb{R}^d \to \mathbb{R}$ and $\mathbf{r}_{f(P)} : \mathbb{R}^n \to \mathbb{R}$. Recall that each is the distance to the nearest point in P or $f(P)$ respectively.

$$\begin{aligned}
\mathbf{r}_{f(P)}(f(x)) &= \min_{q \in f(P)} \|q - f(x)\| && \text{[by definition]} \\
&= \min_{p \in P} \|f(p) - f(x)\| && [q \in f(P)] \\
&\leq \min_{p \in P} \|p - x\| && [f \text{ is 1-Lipschitz}] \\
&= \mathbf{r}_P(x). && \text{[by definition]}
\end{aligned}$$

For any curve $\gamma : [0,1] \to \mathbb{R}^d$ and for all $t \in [0,1]$, we have $\|(f \circ \gamma)'(t)\| \leq \|\gamma'(t)\|$. It then follows that

$$\ell'(f \circ \gamma) = \int_0^1 \mathbf{r}_{f(P)}(f(\gamma(t)))\|(f \circ \gamma)'(t)\|dt$$
$$\leq \int_0^1 \mathbf{r}_P(\gamma(t))\|\gamma'(t)\|dt = \ell(\gamma), \tag{3.5}$$

where ℓ' denotes the length with respect to $\mathbf{r}_{f(P)}$. Thus, for all $a, b \in P$,

$$\begin{aligned}
\mathbf{d}_N(a,b) &= 4 \inf_{\gamma \in \text{path}(a,b)} \ell(\gamma) && \text{[by definition]} \\
&\geq 4 \inf_{\gamma \in \text{path}(a,b)} \ell'(f \circ \gamma) && \text{[by (3.5)]} \\
&\geq 4 \inf_{\gamma' \in \text{path}(f(a),f(b))} \ell'(\gamma') && [f \circ \gamma \text{ is a path}] \\
&= \mathbf{d}_N(f(a), f(b)). && \text{[by definition]}
\end{aligned}$$

□

We now restate Theorem 1.4 for convenience, and prove it.

Theorem 3.5. *For any point set $P \subset \mathbb{R}^d$, the edge squared metric $\mathbf{d}$ and the nearest neighbor metric $\mathbf{d}_N$ are identical.*

Proof. Fix any pair of points s and p in P. Define the Lipschitz mapping m and its extension f as in (3.4). Let $\mathbf{d}'$ and $\mathbf{d}'_N$ denote the edge-squared and nearest neighbor metrics on $f(P)$ in $\mathbb{R}^n$.

$$\begin{aligned}
\mathbf{d}_2(s,p) &= \mathbf{d}'(m(s), m(p)) && \text{[Proposition 3.3(iii)]} \\
&= \mathbf{d}'_N(m(s), m(p)) && [f(P) \text{ are vertices of a box}] \\
&\leq \mathbf{d}_N(s,p) && \text{[Lemma 3.4]}
\end{aligned}$$

We have just shown that $\mathbf{d} \leq \mathbf{d}_N$ and Lemma 3.1 states that $\mathbf{d} \geq \mathbf{d}_N$, so we conclude that $\mathbf{d} = \mathbf{d}_N$ as desired. □

3.1 From Finite Sets to Finite Collections of Compact Path-Connected Bodies

All of our proof steps hold for finite collections of compact, path-connected bodies in arbitrarily large dimension. Our Lipschitz map m can still be extended to a Lipschitz map f in this setting, largely due to the generality of the Kirszbraun theorem. In this case, the pre-image of the contractive map is the set of all points belonging to some body. Meanwhile, the image is a finite set of points, the corners of a multi-dimensional box. Thus our construction of m contracts each convex body into a single point, and the image of our compact bodies under f is still a finite point set on the corners of a box. Therefore, the remainder of our theorem proof goes through unchanged.

This result is rather remarkable: path-connected compact sets in high dimensional space can have extremely convoluted geometry, and the Voronoi diagrams on these collections (on which the nearest neighbor metric depends) can be massively complex. The key is that our Lipschitz map is robust enough to handle objects of considerable geometric complexity.

4 Persistent Homology of the Nearest-neighbor Geodesic Distance

In this section, we show how to compute the so-called persistent homology [114] of the nearest neighbor metric in two different ways, one ambient and the other intrinsic. The latter relies on Theorem 1.4 and would be quite surprising without it.

The input for persistence computation is a *filtration*—a nested sequence of spaces, usually parameterized by a real number $\alpha \geq 0$. The output is a set of points in the plane called a *persistence diagram* that encodes the birth and death of topological features like connected components, holes, and voids.

The Ambient Persistent Homology Perhaps the most popular filtration to consider on a Euclidean space is the sublevel set filtration of the distance to a sample P. This filtration is $(F_\alpha)_{\alpha \geq 0}$, where

$$F_\alpha := \{x \in \mathbb{R}^d \mid \mathbf{r}_P(x) \leq \alpha\},$$

for all $\alpha \geq 0$. If one wanted to consider instead the nearest neighbor metric $\mathbf{d}_N$, one gets instead a filtration $(G_\alpha)_{\alpha \geq 0}$, where

$$G_\alpha := \{x \in \mathbb{R}^d \mid \min_{p \in P} \mathbf{d}_N(x, p) \leq \alpha\},$$

for all $\alpha \geq 0$.

Both the filtrations (F_α) and (G_α) are unions of metric balls. In the former, they are Euclidean. In the latter, they are the metric balls of $\mathbf{d}_N$. These balls can look very different, for example, for $\mathbf{d}_N$, the metric balls are likely not even convex. However, these filtrations are very closely related.

Lemma 4.1. *For all $\alpha \geq 0$, $F_\alpha = G_{2\alpha^2}$.*

Proof. The key to this exercise is to observe that the nearest point $p \in P$ to a point x is also the point that minimizes $\mathbf{d}_N(x, p)$. To prove this, we will show that for any $p \in P$ and any path $\gamma \in \text{path}(x, p)$, we have $\ell(\gamma) \geq \frac{1}{2}\mathbf{r}_P(x)^2$. Consider any such x, p, and γ. The euclidean length of γ must be at least $\mathbf{r}_P(x)$, so we will assume that $\|\gamma'\| = \mathbf{r}_P(x)$ and will prove the lower bound on the subpath starting at x of length exactly $\mathbf{r}_P(x)$. This will imply a lower bound on the whole path. Because $\mathbf{r}_P$ is 1-Lipschitz, we have $\mathbf{r}_P(\gamma(t)) \geq (1-t)\mathbf{r}_P(x)$ for all $t \in [0, 1]$. It follows that

$$\begin{aligned}\ell(\gamma) &= \int_0^1 \mathbf{r}_P(\gamma(t))\|\gamma'(t)\|dt \\ &\geq \mathbf{r}_P(x)^2 \int_0^1 (1-t)dt = \frac{1}{2}\mathbf{r}_P(x)^2\end{aligned}$$

The bound above applies to any path from x to a point $p \in P$, and so,

$$\mathbf{d}_N(x,p) = 4 \inf_{\gamma \in \text{path}(x,p)} \ell(\gamma) \geq 2\mathbf{r}_P(x).$$

If p is the nearest neighbor of x in P, then $\mathbf{d}_N(x,p) = 2\mathbf{r}_P(x)$, by taking the path to be a straight line. It follows that $\min_{p \in P} \mathbf{d}_N(x,p) = 2\mathbf{r}_P(x)$. □

The preceding lemma shows that the two filtrations are equal up to a monotone change in parameters. By standard results in persistent homology, this means that their persistence diagrams are also equal up to the same change in parameters. This means that one could use standard techniques such as α-complexes [114] to compute the persistence diagram of the Euclidean distance and convert it to the nearest neighbor metric afterwards. Moreover, one observes that the same equivalence will hold for variants of the nearest neighbor metric that take other powers of the distance.

Intrinsic Persistent Homology Recently, several researchers have considered intrinsic nerve complexes on metric data, especially data coming from metric graphs [6, 131]. These complexes are defined in terms of the intersections of metric balls in the input. The vertex set is the input point set. The edges at scale α are pairs of points whose α-radius balls intersect. In the intrinsic Čech complex, triangles are defined for three way intersections, tetrahedra for four-way intersections, etc.

In Euclidean settings, little attention was given to the difference between the intrinsic and the ambient persistence, because a classic result, the Nerve Theorem [51], and its persistent version [73] guaranteed there is no difference. The Nerve theorem, however, requires the common intersections to be contractible, a property easily satisfied by convex sets such as Euclidean balls. However, in many other topological metric spaces, the metric balls might not be so well-behaved. In particular, the nearest neighbor metric has metric balls which may take on very strange shapes, depending on the density of the sample. This is similarly true for graph metrics. So, in these cases, there is a difference between the information in the ambient and the intrinsic persistent homology.

Theorem 4.2. *Let $P \subset \mathbb{R}^d$ be finite and let $\mathbf{d}_N$ be the nearest neighbor metric with respect to P. The edges of the intrinsic Čech filtration with respect to $\mathbf{d}_N$ can be computed exactly in polynomial time.*

Proof. The statement is equivalent to the claim that $\mathbf{d}_N$ can be computed exactly between pairs of points of P, a corollary of Theorem 3.5. Two radius α balls will intersect if and only of the distance between their centers is at most 2α. The bound on the distance necessarily implies a path and the common intersection will be the midpoint of the path. □

5 Relating the nearest neighbor metric to Euclidean MSTs, Euclidean Spanners, and More

The nearest neighbor metric, as seen in Theorem 1.4, is equal to the edge-squared metric. This allows us to connect this manifold distance to a graph distance, which we will in turn show

is a generalization of maximum-edge distance on minimum spanning trees. The results in this section are quite simple to prove, but we nonetheless believe they are important properties of the Nearest Neighbor metric and its variants.

The edge-squared metric on a Euclidean point set, as we recall, is defined by taking the Euclidean distances squared and finding the shortest paths. We could have taken any such power p of the Euclidean distances. We will soon see that taking $p = 1$ gives us the Euclidean distance, and finding spanners of the graph as $\lim p \to \infty$ is the Euclidean MST problem. Let the p-power metric be defined on a Euclidean point set by taking Euclidean distances to the power of p, and performing all-pairs shortest path on the resulting distance graph.

Theorem 5.1. *For all $q > p$, any 1-spanner of the p-power metric is a 1-spanner of the q-power metric on the same point set*

Proof. A 1-spanner of the q-power metric can be made by taking edges uv where

$$\min_{p_0=u,\ldots p_k=v,k\neq 1} \sum_k ||p_i - p_{i-1}||^q > ||u - v||^q. \tag{3.6}$$

If $\sum_{i=1}^{k} ||p_i - p_{i-1}||^q > ||u - v||^q$ for any points $p_1, \ldots p_k$, then $\sum_{i=1}^{k} ||p_i - p_{i-1}||^p > ||u - v||^p$ for any $q > p$. Thus, for all such edges uv satisfying Equation 3.6:

$$\min_{p_0=u,\ldots p_k=v,k\neq 1} \sum_k ||p_i - p_{i-1}||^p > ||u - v||^p.$$

Such edges uv must be included in any 1-spanner of the p-power metric. □

Corollary 5.1.1. *Let P be a set of points in Euclidean space drawn i.i.d. from a Lipschitz probability density bounded above and below, with support on a smooth, compact manifold with intrinsic dimension d, bounded curvature, and smooth boundary of bounded curvature. Then the k-NN graph on P when $k = O(2^d \log n)$ is a 1-spanner of the p-power metric for every $p \geq 2$, w.h.p.*

This follows from combining Theorem 1.6 and Theorem 5.1.

5.1 Relation to the Euclidean MST problem

Definition 5.2. *Let the **normalized p-power metric** between two points in $\mathbb{R}^d$ be the p-power metric between the two points, raised to the $\frac{1}{p}$ power. Define the normalized ∞-power metric as the limit of the normalized p-power metric as $p \to \infty$.*

Lemma 5.3. *The Euclidean MST is a 1-spanner for the normalized ∞-power metric.*

This lemma follows from basic properties of the MST. The normalized p-power metrics give us a suite of metrics such that $p = 1$ is the Euclidean distance and $p = \infty$ gives us the distance of the longest edge on the unique MST-path. Setting $p = 2$ gives the edge-squared metric, which sits between the Euclidean and max-edge-on-MST-path distance. Theorem 5.1 establishes that minimal 1-spanners of the (normalized) p-power metric are contained in each other, as p varies from 1 to ∞. The minimal spanner for a general point set when $p = 1$ is the complete graph, and the Euclidean MST is the minimal spanner for $p = \infty$. Thus:

Theorem 5.4. *For points in $\mathbb{R}^d$, every 1-spanner of the p-power metric on that set of points contains every Euclidean MST.*

Corollary 5.4.1. *Every 1-spanner for the Nearest Neighbor metric contains every Euclidean MST.*

5.2 Generalizing Single Linkage Clustering, Level Sets, and k-Centers clustering

If our point set is drawn from a well-behaved probability density, then the normalized edge-power metrics converge to a nice geodesic distance detailed in [158]. When $p = 1$, clustering with this metric is the same as Euclidean metric clustering (k-means, k-medians, k-centers), and when $p = \infty$, clustering with this metric is the same as the single-linkage clustering and the widely used level-set method [30, 117, 134, 298]. Thus, clustering with normalized edge-power metrics generalizes these two very popular methods, and interpolates between their advantages. Definitions of the level-set method and a full discussion are contained in Appendix G

6 Spanners for the nearest neighbor metric

In this section, we prove our theorems on spanners of the nearest neighbor metric. The proofs of these theorems mostly leverage Theorem 1.4 and past work on geometric spanners. We have nonetheless included them for completeness, and to illustrate that spanners of manifold distances like the nearest neighbor metric can have interesting properties not found in Euclidean spanners (assuming no Steiner points).

6.1 Exact-spanners of nearest neighbor metric in the Probability Density Setting

Theorem 1.6 states that for $k = O(2^d \log n)$, the k-NN graph of n points drawn i.i.d from a nicely behaved probability distribution is a 1-spanner of the nearest neighbor metric. This section is dedicated to outlining a proof of this Theorem, the full result which will be in Appendix I. This result is clearly impossible for Euclidean distances, whose 1-spanner is the complete graph almost surely. Our theorem implies any off-the-shelf k-nearest neighbor graph generator can compute edge-squared metric. We strongly rely on Theorem 1.4 for this result, and the fact that Gabriel graphs are 1-spanners of the edge-squared metric.

First, let us assume that the support of our probability density D has the same dimension as our ambient space. This simplifies our calculations without changing the problem much. Then, we note that as our number of sample points get large, the density inside a k-NN ball around any point x (the ball with radius k^{th}-NN distance, center at x) looks like the uniform distribution on that ball, possibly intersected with a halfspace. The bounding plane of our halfspace represents the boundary of our density D.

For simplicity in the outline, let's suppose that D is convex. If we condition on the radius of the k-NN ball, then the $k - 1^{st}$ nearest neighbors of x are distributed roughly according to the

above distribution, described by the ball intersected with a halfspace. For any other point p in D, we project p onto the k-NN ball to point p', and show that the ball $p'x$ contains a k^{th} nearest neighbor w.h.p, when $k = O(2^d \log n)$. This implies ball with diameter px contains a k^{th} nearest neighbor of x, and thus px is not necessary in any 1-spanner of the edge-squared metric. Then we take union bound over all x. A rigorous proof of Theorem 1.6 requires careful analysis, and is contained in Section I. Our proof can be tweaked to show:

Theorem 6.1. *Given a Lipschitz distribution bounded above and below with support on convex set $C \subset \mathbb{R}^d$, the k-NN graph is Gabriel w.h.p. for $k = O(2^d \log n)$.*

6.2 Fast, Sparse Spanner for the Edge-Squared Metric

Now we outline a proof for Theorem 1.5, which shows that one can construct a $(1+\varepsilon)$ nearest neighbor metric spanner of size $O(n\varepsilon^{-d/2})$ in time $O\left(n \log n + n\varepsilon^{-d/2} \log\left(\frac{1}{\varepsilon}\right)\right)$, for points in constant dimensional space. The full proof is in Appendix H. We critically rely on Theorem 1.4 for this work, which shows a spanner for the edge-squared metric is equivalent to a spanner for the nearest neighbor metric.

Note that this spanner is sparser and faster in terms of epsilon dependency than the theoretical optimal spanner for Euclidean distances [191]. We rely extensively on well-separated pair decompositions (WSPDs), and this outline assumes familiarity with that notation. For a comprehensive set of definitions and notations on well separated pairs, refer to any of [33, 34, 62, 64]. Our proof consists of three parts.

1. Showing that connecting a $(1 + O(\delta^2))$-approximate shortest edge in a $1/\delta$ well separated pair for all the pairs in the decomposition gives a $1 + O(\delta^2)$ edge-squared spanner. The processing for this step takes $O(n \log n + \delta^{-d} n)$ time.
2. Previous work contains an algorithm computing $1 + O(\delta^2)$-approximate shortest edge in a $1/\delta$ well separated pair for all the pairs in a WSPD, and takes $O(1)$ time per pair. The pre-processing for this step will be bounded by $O(\delta^{-d} n \log\left(\frac{1}{\delta}\right))$ time. The $\log\left(\frac{1}{\delta}\right)$ factor goes away given a fast floor function. This procedure was first introduced in [64].
3. Putting these two together, and setting $\epsilon = \delta^2$ gives us a $1 + \epsilon$ spanner with $O(\epsilon^{-d/2} n)$ edges in $O(n \log n + \epsilon^{-d/2} n)$ time.

Full details of this proof are contained in Appendix H

7 Conclusions and Open Questions

We examined the nearest neighbor metric and showed how to compute it exactly, as well as find sparse data structures efficiently for approximate computation in practice. Many problems remain open.

First: are there generalizations of these metrics, for which our proof techniques will still hold? The nearest neighbor metric has many natural generalizations, including the k^{th} nearest neighbor or powers of the nearest neighbor function.

Can we efficiently compute $o(\log n)$-spanners of the nearest neighbor metric in high dimension, such the the spanners have a nearly linear number of edges? The existence of such spanners

has been studied for Euclidean metrics in [150], where the stretch obtained is $\sqrt{\log n}$.

Does computing k-NN graphs with approximate nearest neighbor methods give 1-spanners of the edge-squared metric with high probability? Approximate nearest neighbors have been studied extensively [80, 109, 208], including locality-sensitive hashing for high dimensional point sets [27] and more [189]. Recent work by Andoni et al. [29] showed how to compute approximate nearest neighbors for any non-Euclidean norm. Perhaps there is a rigorous theory about data-sensitive metrics generated from any such norm? Similar to how the edge-squared metric is generated from the Euclidean distance.

It remains an open question how well clustering or classification with nearest neighbor metrics performs on real-world data. Experiments have been done by Bijral, Ratliff, and Srebro in [49]. Theorem 1.6 implies that future experiments can be done using any k-nearest-neighbor graph. We believe that the interest in alternative metrics on Euclidean data will continue to be a rich source of interesting problems.

G Nearest Neighbor Metric and Edge-Power Metrics relate to Single Linkage Clustering, Level Sets, and k-Centers clustering

Many popular clustering algorithms, including k-centers, k-means, and k-medians clustering, use Euclidean distance as a measure of distance between points in $\mathbb{R}^d$. These methods are useful when clusters are spherical and well-separated. However, it is believed by practitioners that data-sensitive distances more accurately capture intrinsic distances between data [10].

The celebrated single-linkage clustering algorithm [134, 305], which is clustering based on an MST, is a widely used tool in machine learning, and gets around many of the problems of the Euclidean distance clustering. In single-linkage clustering, two points are considered similar if the maximum length edge on the path between them in the MST is small. This turns out to be equivalent to computing the normalized ∞-power metric between the two points. Therefore, single linkage clustering can be seen as clustering using the normalized ∞-power metric. Generally, normalized p-power metrics can be seen as an intermediary between Euclidean distances (1-power metrics) and Euclidean MST-based clustering.

Clustering with p-power metric relates to another popular clustering method in machine learning, known as level-set clustering. Loosely speaking, level set clustering involves finding an estimate for the probability density that points are drawn from, finding a cut threshold t, and then taking as clusters all regions with probability density $> t$. Level set clustering has appeared in many incarnations [275, 276, 298], including the celebrated and widely used DBScan method [117] and its considerable number of variations [30]. It is known that level-set clustering is related to single-linkage clustering, as the latter is an approximation of the former [276, 298]. Level-set methods have the advantage that they can find arbitrarily shaped clusters [117], but can cause two points that are very close in Euclidean distance to be considered far apart.

Clustering with the p-power metric incorporates the advantages of both Euclidean distance clustering and level set clustering, as it is both data-sensitive and takes into account overall Euclidean distance between two points. Here, p can be toggled to change the sensitivity of the

metric to the underlying density. As the number of samples drawn from our probability density grows large, it has been proven that the behavior of normalized p-power metrics converges to a natural geodesic distance on the underlying probability density [158]. Clustering with this geodesic distance for $p = 1$ is exactly Euclidean clustering, and for $p = \infty$ is exactly the level set method. Thus, clustering with p-power metric converges to a clustering method that smoothly interpolates between Euclidean-distance clustering and level set clustering.

H Proving Faster and Sparser-than-Euclidean Approximate Spanners

In this appendix, we finish the proof of Theorem 1.5 based on the outline given in Section 6.2.

H.1 $1 + O(\delta^2)$ spanners can be generated from a $1/\delta$ WSPD

Definition H.1. *Let e be a **critical** edge in a shortest path metric on any graph if the (possibly-not-unique) shortest path between the endpoints of e is the edge e.*

Lemma H.2. *The set of critical edges on any graph forms a 1-spanner of the shortest path metric.*

The above lemma is known in the literature.

To check that any graph H is a $(1 + O(\delta^2)$ spanner of any graph G, it suffices to prove that all critical edges in the edge-squared metric have a stretch no larger than $1 + O(\delta^2)$. Let G be the edge-squared graph arising from points $P \subset \mathbb{R}^d$. Build a well-separated pair decomposition on P, with pairs given as $\{A_1, B_1\}, \{A_2, B_2\}, \ldots \{A_m, B_m\}$. Create a spanner H as follows: for each pair $\{A_i, B_i\}$, connect an edge $\{a, b\}, a \in A_i, b \in B_i$ such that the Euclidean distance between a and b is a $(1+c\delta^2)$ approximation of the shortest distance between point sets A_i and B_i, for some constant c independent of i. This can be accomplished in $O(1)$ time assuming a preprocessing step of $O(\delta^{-d} \log\left(\frac{1}{\delta}\right)$ time, as noted in Callahan's paper on constructing a Euclidean MST [64]. Do this for all $1 \leq i \leq m$.

For each critical edge (s, t), consider the well-separated pair $\{A, B\}$ that (s, t) is part of. Let $s \in A$ and $t \in B$. Let (a, b) be a $(1 + c\delta^2)$-approximate shortest edge between A and B ($a \in A, b \in B$). Scale $||a - b||_2$ to be 1. A and B have Euclidean radius at most δ, by the definition of a well separated pair. By induction on Euclidean distance, H is an edge-squared 2-spanner of the edge-squared metric for all points in A and B and all points in B (assuming sufficiently small δ).

Lemma H.3.

$$\begin{aligned} dist_H(s, t) &\leq dist_H(s, a) + dist_H(a, b) + dist_H(b, t) \\ &\leq 1 + O(\delta^2) \end{aligned}$$

Proof. We know $dist_H(a, b)$ = 1 by our scaling, and

$$dist_H(s, a) \leq 2 \cdot (dist_G(s, a)) \leq 2 \cdot ||s - a||^2 \leq 8\delta^2$$

The first inequality follows by the inductive hypothesis that H is a 2-spanner of G in A. The third inequality follows since both s and a are contained in a ball of radius δ.

The same bound applies for $dist_H(b,t)$. □

Lemma H.4.

$$(1+c\delta^2)(dist_G(s,t)) \geq dist_G(a,b) = 1$$

$$\Rightarrow dist_G(s,t) \geq \frac{1}{1+c\delta^2}$$

Lemma H.4 follows from the fact that (a,b) is a $(1+c\delta^2)$ approximate shortest distance between A and B.

Therefore

$$\begin{aligned} stretch_H(s,t) &\leq \frac{dist_H(s,t)}{dist_G(s,t)} \\ &\leq (1+16\delta^2)(1+c\delta^2) \\ &= 1+O(\delta^2) \end{aligned}$$

Thus we have proven that H is a $1+16\delta^2$ spanner. Now set $\epsilon = \delta^2$, which completes proof of Theorem 1.5.

I Spanners in the Probability Density Setting: Full Proof

We prove Theorem 1.6 in full. Through this section, we assume that D is a probability density function with support on smooth connected compact manifold with intrinsic dimension d embedded in ambient space $\mathbb{R}^s$, with smooth boundary of bounded curvature. This probability density function is further assumed to be bounded above and below, and to be Lipschitz. For simplicity, we assume that $s = d$, and we can prove all our results when $s > d$ by taking coordinate charts from the manifold into Euclidean space. We will show at the end of the section that if the distribution is supported on a convex set of full dimension in the ambient space, then the k-NN graph is Gabriel for the same k. It is not difficult to see that Gabriel graphs are 1-spanners of the edge-squared metric [273].

Lemma I.1. *Let M be a compact object in $\mathbb{R}^d$, whose boundary is a smooth manifold of dimension $d-1$ with bounded curvature. Let $\mathbb{B}$ be any ball with sufficiently small radius r_B with center in M, that intersects the boundary of D at some point x. Let H be the halfspace tangent to M at x containing the center of the ball.*

For any point $Q \in M$, let Q' be the point in B closest to Q. If $d(Q', H)/r_B > c$ for arbitrary constant c, then $d(Q,H) \geq c'$ for some constant c'.

This is a basic fact about the smoothness and bounded curvature of the boundary.

Lemma I.2. *Pick n points from D. W.h.p, any two points in $Support(D)$ with Euclidean distance $\geq \Omega(1)$ have nearest neighbor metric of $o(1)$.*

This is implicit in [158].

Lemma I.3. *For any ball $\mathbb{B}$ with center O and any point Q' on the boundary of B, let $B_{Q'O}$ be the ball with diameter $Q'O$. Let H be any halfspace containing O. If $d(Q', H)/r_B \leq c$ for some constant c possibly depending on the dimension d, then $\text{Vol}(\mathbb{B}_{Q'O} \cap H) \geq \frac{1-c'}{2^d} \text{Vol}(\mathbb{B} \cap H)$ for some constant c', where c' goes to 0 as c goes to 0.*

Proof. First, let us consider the case where $d(Q'H) = 0$, that is, Q' is contained in halfspace H'. In this case, dilating $B_{Q'O} \cap H$ by a factor of 2 about point Q' gives a superset of $B \cap H$, as $B_{Q'O}$ maps to B and H maps to a halfspace strictly containing H. In this case, $\text{Vol}(\mathbb{B}_{Q'O} \cap H) \geq \frac{1}{2^d} \text{Vol}(\mathbb{B} \cap H)$ as desired. The case when $d(Q', H)/r_B$ is bounded follows in a straightforward manner. □

This leads us to our following theorem:

Theorem I.4. *For any n point set P picked i.i.d from D, consider any point O. Let $\mathbb{B}$ be the k-NN ball of O. Let $Q \in Support(D)$ be any point outside $\mathbb{B}$, and let the closest point to Q in $\mathbb{B}$ be Q'. For a point x inside B on the boundary of D (assuming such a point exists), let H be the tangent halfplane containing the center of $\mathbb{B}$.*

Then: either $d(Q', H)/r_B \leq c'$ for some constant c' or there exists a constant c where $|QO| > c$. Here, c and c' are independent of the number of points chosen, and c' can be set arbitrarily small.

In the latter case, w.h.p. QO is not in the edge-squared 1-spanner. In the former case, setting c' to be a very small constant ϵ lets us say:

$$\text{Vol}(\mathbb{B}_{Q'O} \cap H) \geq \frac{1-\epsilon}{2^d} \text{Vol}(\mathbb{B} \cap H), \tag{3.7}$$

or equivalently:

$$\mathop{\mathbb{P}}_{x \sim D}[x \in \mathbb{B}_{QO} | x \in \mathbb{B}] \tag{3.8}$$

$$\geq \mathop{\mathbb{P}}_{x \sim D}[x \in \mathbb{B}_{Q'O} | x \in \mathbb{B}] \tag{3.9}$$

$$\geq \frac{1 - \varepsilon - o(1)}{2^d} \tag{3.10}$$

Expression 3.9 $>$ Expression 3.10 follows from Equation 3.7, and the fact that the radius of the k-NN ball goes to 0 as n gets large, and thus the probability density of sampling x from D conditioned on x being in $\mathbb{B}$ approaches the uniform density in $\mathbb{B} \cap Support(D)$. Also, $B \cap H$ approaches $B \cap Support(D)$ as the radius of B goes to 0.

Expression 3.8 $>$ Expression 3.9 since $\mathbb{B}_{QO} \supset B_{Q'O}$. (Here, the k-NN ball B w.r.t. point O is defined as the ball centered at O with radius equal to the distance of the k^{th} nearest neighbor to O).

Note that the $k-1$ nearest neighbors of O, conditioned only on the radius of B, are distributed equivalently to $k-1$ i.i.d samples of D conditioned on containment in $\mathbb{B}$. It follows that for any point Q outside B and in the support of D, where $|QO| < c$:

$$\mathop{\mathbb{P}}_{P\sim D^k}[QO \text{ is not Gabriel w.r.t. } P | Q \notin B]$$
$$\geq 1 - \left(1 - \frac{1-\varepsilon-o(1)}{2^d}\right)^k$$

Thus, setting $\epsilon = 0.1$ and $k > O(\log n/2^d)$, and factoring in the case where $|QO| > c$, then w.h.p.:

$$\mathop{\mathbb{P}}_{P\sim D^k}[QO \text{ is not critical w.r.t. } P | Q \notin B]$$

Here, we recall that an edge AB is Gabriel with respect to a point set P if and only if $\mathbb{B}_{AB}$ does not contain any points in P. Note that every non-Gabriel edge is non-critical, where a critical edge is an edge that must be in the 1-spanner (as in Definition H.1). Thus taking the union bound over $Q, O \in P$ gives us that no edge outside the k-NN graph is critical w.h.p, and thus the k-NN graph contains all critical edges and is a 1-spanner w.h.p.

This proves Theorem 1.6 when the support of D has the same intrinsic dimension as the ambient space. If the support of D has dimension $d < d'$ (where d' is the ambient dimension of the space), simply take coordinate charts from D onto $\mathbb{R}^d$ and the previous arguments will still carry through . We should note that if no point x inside B on the boundary of D exists, then we can ignore H and all the steps of the proof still follow.

Chapter 4

Spectral Clustering in the Limit

Cheeger and Buser inequalities relate fundamental eigenvalues with isoperimetric constants. These inequalities for graphs are the foundation of spectral graph theory.

In this chapter, we introduce Cheeger and Buser inequalities for Lipschitz probability density functions. To do this, we create new definitions of isoperimetry and eigenvalues in this setting. For past definitions, one of the inequalities must fail. We apply our work to give a new spectral algorithm for partitioning probability densities, which is a variant of classical spectral clustering. Classical spectral clustering can fail when data comes from a nicely behaved Lipschitz density function, and our new variant will overcome traditional problems in spectral clustering when points are drawn from general Lipschitz probability densities.

A Introduction

The Cheeger and Buser inequalities relate isoperimetric cuts with eigenvalues. Up to a constant factor, the Cheeger inequality lower bounds the fundamental eigenvalue of a Laplacian with the square of the isoperimetric constant [22, 77], and the Buser inequality upper bounds the eigenvalue with the isoperimetric constant [22, 61].

These inequalities appear in two settings: the graph setting [22] and the manifold setting [61, 77]. In graphs, these inequalities are the foundation of spectral graph theory [22, 92]. Spectral graph theory has provided surprising insights on problems including maximum flow [89], fast Laplacian solving [180], expander decompositions [250, 301], sparse cuts [22, 32, 71], and long-standing mathematical conjectures like the Kadison Singer conjecture [216]. In this setting, the Buser inequality is trivial, and the Cheeger inequality is mathematically substantial [22, 92]. In manifold theory, the Cheeger and Buser inequalities have proven useful for probability theory on manifolds [192], machine learning [40], and more [41, 192]. Unlike in the graph setting, the Buser inequality on manifolds is highly nontrivial, and historically came as a surprise to manifold theorists [61, 192].

This chapter introduces Cheeger and Buser inequalities in a new setting: Lipschitz probability density functions. Here, a probability density function refers to a function $\rho : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ where $\int_{\mathbb{R}^d} \rho = 1$. A Lipschitz probability density refers to a probability density ρ where $|\rho(x) - \rho(y)| < L\|x - y\|_2$ for some constant L. Cheeger inequalities have been used in the probability density setting in the past [200], but primarily in the setting where there are strong parametric assumptions on the density (such as when the density is Gaussian or log-concave) [200]. We show that for general Lipschitz probability densities, either the Cheeger or the Buser inequality must fail using past definitions of eigenvalue and isoperimetric constant.

Since past definitions of eigenvalue and isoperimetric constant are inadequate in the probability density setting, we present new definitions of these quantities for which a Cheeger and Buser inequality will hold. Using our new definitions, the Cheeger inequality can be proven using standard techniques. Akin to the manifold setting, the Buser inequality here is more difficult to prove. New mathematical ideas are required to prove the Buser inequality. We prove both the Cheeger and Buser inequalities on Lipschitz probability densities, using our new definitions.

Applications

We will use our inequalities to show that a spectral sweep cut of a Lipschitz probability density functions will partition it into two pieces with good sparsity guarantees. We then discuss potential applications to machine learning and spectral clustering.

Spectral clustering [230, 259] is one of the most widely used techniques in machine learning [291]. It is known to have a close connection to past definitions for eigenvalues of a probability density function [280]. However, spectral clustering is not known to have any good theoretical sparsity guarantees on partition quality as the number of samples grows large, in part due to the lack of a Cheeger and Buser inequality for past definitions of eigenvalues. Our new Cheeger and Buser inequalities may motivate theoretically principled spectral clustering methods.

A.1 Definitions

To establish our Cheeger and Buser inequalities, we define new notions of isoperimetric constant (equivalently, sparsity), Rayleigh quotients, eigenvalues, eigenvectors, and sweep cuts. Appropriate definitions will let us establish basic spectral theory in the Lipschitz probability density setting.

Definition A.1. *Let ρ be a probability density function with domain $\mathbb{R}^d$, and let A be a subset of $\mathbb{R}^d$.*

*The (α, β)-**sparsity** of the cut induced by A is denoted by $\Phi(A)$. It is defined as $(d-1)$ dimensional integral of ρ^β on the cut, divided by the d dimensional integral of ρ^α on the side of the cut where this integral is smaller.*

For nice smooth cuts this intuitive definition is fine but a more general and precise definition using total variation is given in definition D.2

Definition A.2. *The (α, β)-**isoperimetric constant** of ρ is defined as the infimum of $\Phi(A)$ over all A.*

Definition A.3. *The (α, γ)-**Rayleigh quotient** of u with respect to $\rho : \Re^d \to \Re_{\geq 0}$ is:*

$$R_{\alpha,\gamma}(u) := \frac{\int_{\Re^d} \rho^\gamma |\nabla u|^2}{\int_{\Re^d} \rho^\alpha |u|^2}$$

*A (α, γ)-**principal eigenvalue** of ρ is λ_2, where:*

$$\lambda_2 := \inf_{\int \rho^\alpha u = 0} R_{\alpha,\gamma}(u).$$

*Define a (α, γ)-**principal eigenfunction** of ρ to be a function u such that $R_{\alpha,\gamma}(u) = \lambda_2$, if such u exists.*

Now we define a sweep cut for a given function with respect to a a positive valued function supported on $\mathbb{R}^d$:

Definition A.4. *Let α, β be two real numbers, and ρ be any function from $\Re^d$ to $\Re_{\geq 0}$. Let u be any function from $\Re^d \to \Re$, and let $C_{t,u}$ be the cut defined by the set $\{s \in \Re^d \mid u(s) > t\}$.*

*The **sweep-cut** algorithm for u with respect to ρ returns the cut $C_{t,u}$ of minimum (α, β) sparsity, where this sparsity is measured with respect to ρ.*

*When u is a (α, γ)-principal eigenfunction, the sweep cut is called a (α, γ)-**spectral sweep cut** of ρ.*

Additional Definitions:

A function $\rho : \Re^d \to \Re_{\geq 0}$ is L-**Lipschitz** if $|\rho(x) - \rho(y)|_2 \leq L|x - y|_2$ for all $x, y \in \Re^d$.

A function is $\rho : \Re^d \to \Re_{\geq 0}$ is α-**integrable** if $\int_{\Re^d} \rho^\alpha$ is well defined and finite. Throughout this chapter, we assume ρ is always α-integrable.

Our definitions depend on three constants: α, β, and γ. Informally, these constants can be thought of as the mass constant, the cut constant, and the spring constant respectively. In the graph and manifold setting, the spring and cut constant are the same. One key contribution of

our definitions is the decoupling of the cut and spring constants which will allow us to get tight Cheeger and Buser results.

A.2 Theorems

Theorem A.5. ***Probability Density Cheeger and Buser:***

Let $\rho : \mathbb{R}^d \to \mathbb{R} \geq \nvDash$ be an L-Lipschitz density function. Let $\alpha = \beta - 1 = \gamma - 2$.

Let Φ be the infimum (α, β)-sparsity of a cut through ρ, and let λ_2 be the (α, γ)-principal eigenvalue of ρ. Then:

$$\Phi^2/4 \leq \lambda_2$$

and

$$\lambda_2 \leq O_{\alpha,\beta}(d \max(L\Phi, \Phi^2)).$$

*The first inequality is **Probability Density Cheeger**, and the second inequality is **Probability Density Buser**.*

In particular, a Cheeger and Buser inequality exist when $\alpha = 1, \beta = 2, \gamma = 3$. Note that we don't need ρ to have a total mass of 1 for any of our proofs. The overall probability mass of ρ can be arbitrary.

Theorem A.5 has partial converses:

Lemma A.6. *If $\alpha + \gamma > 2\beta$, the Cheeger inequality in Theorem A.5 does not hold.*

Lemma A.7. *If $\gamma \geq 1$ and $\gamma - 1 < \beta$, then the Buser inequality in Theorem A.5 does not hold.*

In particular, if $\alpha = \beta = \gamma = 1$, the Buser inequality fails. If $\alpha = 1, \gamma = 2$, no Cheeger-Buser inequality can hold for any β. These settings encompass most past work on sparse cuts and eigenvectors in probability densities, as mentioned in Section A.3.

We apply these inequalities to show that spectral sweep cuts of Lipschitz probability density functions give sparse cuts of the density. This contrasts with past work on sweep cuts in probability densities, as mentioned in Section A.3.

Theorem A.8. ***Spectral Sweep Cuts give Sparse Cuts:***

Let $\rho : \Re^d \to \Re_{\geq 0}$ be an L-Lipschitz probability density function, and let $\alpha = \beta - 1 = \gamma - 2$.

The (α, γ)-spectral sweep cut of ρ has (α, β) sparsity Φ satisfying:

$$\Phi_{OPT} \leq \Phi \leq O(\sqrt{dL\Phi_{OPT}}).$$

Here, Φ_{OPT} refers to the optimal (α, β) sparsity of a cut on ρ.

In words, the spectral sweep cut of the (α, γ) eigenvector gives a provably good approximation to the sparsest (α, β) cut, as long as $\beta = \alpha + 1$ and $\gamma = \alpha + 2$. We will also show that this theorem is not possible for past definitions of eigenvectors and sparsity on probability density functions.

A.3 Past Work

Cheeger and Buser Inequalities for Graphs and Manifolds

Cheeger and Buser inequalities for graphs are the foundation of spectral graph theory [92]. These inequalities were first discovered by Alon and Millman [22] based on similar inequalities in the

manifold setting [61, 77]. These inequalities have been applied to graph partitioning, random walks, and spectral graph theory [92, 183, 193, 212, 234, 236, 270]. In the graph setting, the Buser inequality is trivial [92], while the Cheeger inequality is mathematically substantial [22].

Cheeger inequality for manifolds have been extensively used in Riemannian geometry [40, 41, 77]. Buser's inequality on manifolds is mathematically non-trivial, unlike the graph case. This inequality depends on a Ricci curvature term, and it is false if the manifold has unbounded Ricci curvature [61, 192]. This inequality has been applied to diffusion processes on Manifolds [192], machine learning [40, 135], and more [192].

For formal definitions of Cheeger and Buser inequalities for graphs and manifolds, refer to [22] and [61] respectively.

Eigenvalues, Sweep Cuts, and Isoperimetry on Probability Densities

Recently, eigenvalues and sparse cuts have been used in the probability density setting [199, 200], in connection with the Kannan-Lovasz-Simonovits conjecture. There is a Cheeger and Buser inequality in this setting, as long as strong parametric assumptions (such as log concavity) are given [200]. These inequalities use what we call $(\alpha = 1, \gamma = 1)$ eigenvectors, and $(\alpha = 1, \beta = 1)$ isoperimetric constants. We will show in our paper that any Buser inequality using $(\alpha = 1, \beta = 1, \gamma = 1)$ must fail for simple Lipschitz densities. No Cheeger-Buser inequality was previously known for probability densities without strong parametric assumptions.

In another line of work by Von Luxburg et al and Rosasco et al [243, 292], the authors use perturbation theory results to show that the second eigenvalue of a graph Laplacian generated from samples on a probability density converges to what we call the $(\alpha = 1, \gamma = 2)$-principal eigenvalue of the density. Trillos et al. [130, 280] improved the convergence rate and showed that the (extensions of the) eigenvectors of the graph Laplacian approach the eigenfunctions of the weighted Laplacian operator for a probability density.

These results show that spectral clustering algorithms like the one in [230] can be thought of as taking an iid sample from a distribution, constructing a graph Laplacian, and computing its fundamental eigenvector as an approximation for finding the eigenfunction over the original distribution. The eigenfunction that they end up approximating is the $(\alpha = 1, \gamma = 2)$ eigenfunction. The clustering algorithm then takes a sweep cut with respect to this eigenfunction.

Unfortunately, sweep cut algorithms based on the $(1, 2)$ eigenfunction can produce cuts of probability densities with bad isoperimetry properties. See Theorem G.1. The strength of our Cheeger and Buser inequalities are that they will imply new sweep cut algorithms on probability densities, with provably good isoperimetry.

Technical Contribution

The key technical contribution of our proof is proving Buser's inequality on Lipschitz probability densities via mollification [128, 263] with disks of varying radius. This chapter is the first time mollification with disks of varying radius have been used. We emphasize that the most difficult part of our paper is proving the Buser inequality.

Mollification has a long history in mathematics dating back to Sergei Sobolev's celebrated proof of the Sobolev embedding theorem [263]. It is one of the key tools in numerical analy-

sis, partial differential equations, fluid mechanics, and functional analysis [128, 206, 226, 274], and analogs of mollification have been used in computational complexity settings [107]. Informally speaking, mollification is used to create a series of smooth functions approximating a non-smooth function, by convolving the original function with a smooth function supported on a disk. Notably, an approach using convolution is used by Buser in [61] to prove the original Buser's inequality, albeit with an intricate pre-processing step on any given cut.

To prove Buser's inequality on Lipschitz probability density functions ρ, we will show that given a cut C with low (α, β)-sparsity, we can find a function u with low (α, γ)-Rayleigh quotient. We build u by starting with the indicator function I_C for cut C (which is 1 on one side of the cut and 0 on the other). Next, we mollify this function with disks of varying radii. In particular, for each point r in the domain of ρ, we spread out the point mass $I_C(r)$ over a disk of radius proportional to $\rho(r)L$, where L is the Lipschitz constant of ρ. The resulting function u obtained by 'spreading out' I_C will have low (α, γ)-Rayleigh quotient.

For all past uses of mollification, the disks on which the smooth convolving function is supported (we call this the mollification disk) have the same radius throughout the manifold. The use of a uniform radius disk is critical for most uses and proofs in mollification. Our contribution is to allow the disks to vary in radius across our density. This variation in radius allow us to deal with functions that approach 0 (and explains the importance of the density being Lipschitz). No mollification disks centered anywhere in our probability density will intersect the 0-set of the density. This overcomes significant hurdles in many results for functional analysis and PDEs, as many past significant results related to partial differential equations rely on having a positive lower bound on the density [130, 294].

Proving our Buser inequality using mollification by disks of various radius requires a fairly delicate proof with many pages of calculus. Our key technical lemma is a bound on how the l_1 norm of a mollified function when the mollification disks have various radius, which can be found in Section D.3.

B Paper Organization

In Section C, we go over example 1-D distributions that show that either Cheeger or Buser inequality must fail for past definitions of sparsity and eigenfunctions. These examples motivate our new definitions. We will prove Lemma A.6 and Lemma A.7 in this section.

We prove the Buser inequality in Section D, via a rather extensive series of calculus computations. Our proof relies on a key technical lemma, which is presented in Section D.3. The Buser inequality is by far the most difficult part of our proof.

We prove the Cheeger inequality in Section E. The proof in this section implies that the $(\alpha, \alpha + 1)$ sparsity of the $(\alpha, \alpha + 2)$ spectral sweep cut of a probability density function ρ is provably close to the $(\alpha, \alpha + 2)$ principal eigenvalue of ρ. We note that this inequality does not depend on the Lipschitz nature of the probability density function.

In Section F, we prove Theorem A.8, which shows that a $(\alpha, \alpha + 2)$ spectral sweep cut has $(\alpha, \alpha + 1)$ sparsity which provably approximates the optimal $(\alpha, \alpha + 1)$ sparsity.

In Section G, we show an example Lipschitz probability density where the $(\alpha = 1, \gamma = 2)$ spectral sweep cut has bad $(1, \beta)$ sparsity for any $\beta < 10$, and will lead to an undesirable cut

(from a clustering point of view) on this density function. This is important since the spectral clustering algorithm of Ng et al [230] is known to converge to a $(\alpha = 1, \gamma = 2)$ spectral sweep cut on the underlying probability density function, as the number of samples grows large [280].

Finally, we state conclusions and open problems in Section H.

In the appendix, we note that the Cheeger and Buser inequalities for probability densities are not easily implied by graph or manifold Cheeger-Buser. We also provide a simplified version of Cheeger's and Buser's inequality for probability densities, in the 1-dimensional case. This may make easier reading for those unfamiliar with technical multivariable mollification.

C Cheeger-Buser inequalities require carefully chosen α, β, γ

In this section, we prove Lemma A.6 and Lemma A.7. As a consequence, we show that settings of α, β, γ common in past work (see Section A.3) violates either the Cheeger or Buser inequality.

We consider two simple density functions:

1. The function $\rho_1(x) = \epsilon/2$ for $-\frac{1}{\epsilon} < x < \frac{1}{\epsilon}$, with a 1-Lipschitz dropoff to 0 at the endpoints of $[-\frac{1}{\epsilon}, \frac{1}{\epsilon}]$.
2. The function $\rho_2(x) = \min(|x| + \epsilon, -|x| + \sqrt{2})$ for $|x| < \sqrt{2}$ and $\rho_2(x) = 0$ for $|x| > \sqrt{2}$.

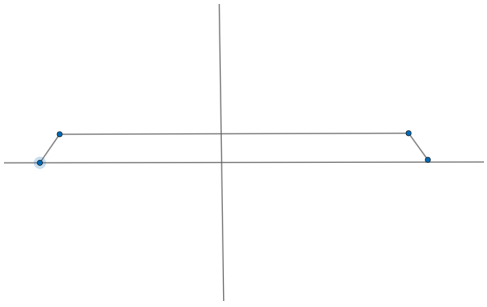

Figure 4.1: The function ρ_1, a 1-Lipschitz counterexample to Cheeger's inequality when $\alpha + \gamma > 2\beta$. The height of the function is $\epsilon/2$, and the length of the supporting interval is roughly $\frac{2}{\epsilon}$.

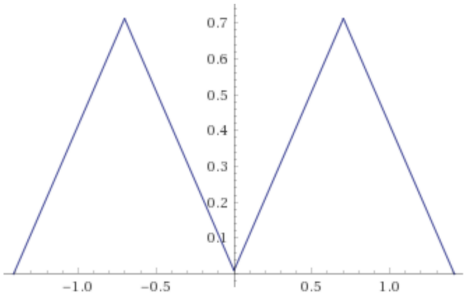

Figure 4.2: The function ρ_2, a 1-Lispchitz counterexample to Buser's inequality when $\gamma \geq 1$ and $\gamma - 1 < \beta$.

We will show that if α, β, γ satisfy $\alpha + \gamma > 2\beta$, then the Cheeger inequality will fail for ρ_1, and if $\gamma \geq 1$ and $\gamma - 1 < \beta$, then the Buser inequality will fail for ρ_2.

In example ρ_1, let $\epsilon < 0.01$. The (α, β) isoperimetric constant Φ is $O((1/\epsilon)^{\alpha-\beta-1})$, and the eigenvalue is $O((1/\epsilon)^{\alpha-\gamma-2})$. Therefore, the Cheeger inequality will fail for some ϵ if $2(\alpha - \beta - 1) > \alpha - \gamma - 2$, or $\alpha + \gamma > 2\beta$. This proves Lemma A.6.

In example ρ_2, let $\epsilon < 0.01$. The (α, β) isoperimetric constant Φ is $O(\epsilon^{\beta})$. The eigenvalue λ_2 is lower bounded by $O(\epsilon^{\gamma-1})$ when $\gamma > 1$, and $\ln(1/\epsilon)$ when $\gamma = 1$. In either case, the Buser inequality will fail if $\gamma - 1 < \beta$, proving Lemma A.7.

We show details of our eigenvalue and isoperimetry calculations in Appendix G.

D Buser Inequality for Probability Density Functions

In this section the theory of functions of bounded variation is used to justify certain formal calculations. The key step is to define the geometric quantities variationally.

Definition D.1. *For a measurable set $A \subseteq \Re^d$ and $\rho : \Re^d \to \Re_{\geq 0}$ integrable define*

$$|A|_\alpha \coloneqq \int_A \rho^\alpha(x)\, dx.$$

We define the weighted boundary area variationally [48, 238].

Definition D.2. *For $\rho : \Re^d \to \Re_{\geq 0}$ integrable and $A \subseteq \Re^d$ the weighted perimeter of A is*

$$|\partial A|_\beta \coloneqq \sup \left\{ \int_A \div(\rho^\beta(x)\phi(x))\, dx \;\middle|\; \phi \in C_c^1(\Re^d),\ \|\phi\|_\infty \leq 1 \right\}.$$

Remark D.2.1. *This definition corresponds to the intuitive definition of the boundary integral of ρ^β when ∂A is sufficiently regular. Specifically, if $A \subseteq \Re^d$ has smooth boundary and then,*

$$|\partial A|_\beta = \int_{\partial A} \rho^\beta(x)\, d\mathcal{H}^{n-1}(x),$$

where $\mathcal{H}^{n-1}$ denotes surface (Hausdorff) measure.

Definition D.3. *Let $A \subseteq \Re^d$ be a set of finite perimeter such that $|A|_\alpha, \left|\Re^d \setminus A\right|_\alpha > 0$. The isoperimetric ratio of the cut induced by A is*

$$\Phi(A) \coloneqq \frac{|\partial A|_\beta}{\min\left(|A|_\alpha, |\Re^d \setminus A|_\alpha\right)}$$

and the isoperimetric constant with weight ρ is

$$\Phi \coloneqq \inf_{A \subseteq \Re^d} \Phi(A).$$

Here, A is taken over sets of finite perimeter such that $|A|_\alpha, \left|\Re^d \setminus A\right|_\alpha > 0$.

D.1 Weighted Buser-type Inequality

We now prove our weighted Buser-type inequality Theorem A.5. We state our result in terms of general (α, β, γ).

Theorem D.4. *Let $\rho : \Re^d \to \Re_{\geq 0}$ be an L-Lipschitz function, λ_2 be a (α, γ)-principal eigenvalue, and Φ the (α, β) isoperimetric cut.*

Then:

$$\lambda_2 \leq 3 \cdot 2^{\beta+1} d \left\| \rho^{\gamma-\beta-1} \right\|_{L^\infty} \max \left(L\Phi, 2^{\beta+1} \left\| \rho^{\alpha+1-\beta} \right\|_{L^\infty} \Phi^2 \right)$$

We note that when setting $(\alpha, \beta, \gamma) = (1, 2, 3)$, the above expression simplifies into:

$$\lambda_2 \leq 24 d \max \left(L\Phi, 8\Phi^2 \right).$$

D.2 Proof Strategy: Mollification by Disks of Radius Proportional to ρ

To prove Theorem D.4, for $A \subset \Re^d$ fixed, we construct an approximation u_θ of the characteristic function, u, of A for which the numerator and denominator of the Rayleigh quotient, $R(u_\theta)$, approximate respectively the numerator and denominator of this expression. Specifically, u_θ will constructed as a mollification of u, Recall the following two equivalent definitions of a mollification. They are equivalent by the change of variables $z = x - \theta\rho(x)y$.

$$u_\theta(x) \coloneqq \int_{B(0,1)} u(x - \theta\rho(x)y)\phi(y)\, dy = \int u(z)\phi_{\theta\rho(x)}(x - z)\, dz, \quad \text{where} \quad \phi_\eta(z) = \frac{1}{\eta^d}\phi\left(\frac{z}{\eta}\right), \tag{4.1}$$

with $\theta > 0$ a parameter to be chosen and $\phi : \Re^d \to [0, \infty)$ a smooth radially symmetric function supported in the unit open ball $B(0, 1) = \{x \in \Re^d \mid |x| < 1\}$ with unit mass $\int_{\Re^d} \phi = 1$. When ρ is constant it follows from the Tonelli theorem that $\|u_\theta\|_{L^1} = \|u\|_{L^1}$; when ρ is not constant the following lemma shows that the latter still bounds the former.

D.3 Key Technical Lemma: Bounding L_1 norm of a function with the L_1 norm of its mollification

The following is our primary technical lemma, which roughly bounds the L_1 norm of a mollified function f by the L_1 norm of the original f. Here, the mollification radius is determined by a function $\delta(x)$.

Lemma D.5. *Let $\delta : \mathbb{R}^d \to \mathbb{R}$ be Lipschitz continuous with Lipschitz constant $|\nabla\delta(x)| \leq c < 1$ for almost every $x \in \mathbb{R}^d$. Let $\phi : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ be smooth, $\int_{\mathbb{R}^d} \phi = 1$, and $\operatorname{supp}(\phi) \subseteq B(0, 1)$. Then*

$$\frac{1}{1+c} \|f\|_{L^1} \leq \int_{\mathbb{R}^d} \int_{B(0,1)} |f(x - \delta(x)y)|\, \phi(y)\, dy\, dx \leq \frac{1}{1-c} \|f\|_{L^1}, \qquad f \in L^1(\mathbb{R}^d).$$

Proof. (of Lemma D.5) An application of Tonelli's theorem shows

$$\int_{\mathbb{R}^d}\int_{B(0,1)} |f(x-\delta(x)y)|\,\phi(y)\,dy\,dx = \int_{B(0,1)}\phi(y)\int_{\mathbb{R}^d}|f(x-\delta(x)y)|\;dx\,dy. \tag{4.2}$$

Fix $y \in B(0,1)$ and consider the change of variables $z = x - \delta(x)y$. The Jacobian of this mapping is $I - y\otimes\nabla\delta(x)$ which by Sylvester's determinant theorem has determinant $1 - y.\nabla\delta(x) > 0$. It follows that

$$\int_{\mathbb{R}^d}\int_{B(0,1)} |f(x-\delta(x)y)|\,\phi(y)\,dy\,dx = \int_{B(0,1)}\phi(y)\int_{\mathbb{R}^d}\frac{|f(z)|}{1-y.\nabla\delta(x)}\,dx\,dy, \tag{4.3}$$

and the lemma follows since $1 - c \leq 1 - y.\nabla\delta(x) \leq 1 + c$. □

(Here, $a.b$ denotes the dot product between a and b.)

We present the following simple corollaries, which is the primary way our proof makes use of Lemma D.5

Corollary D.5.1. *For any Lipschitz continuous function $\rho : \mathbb{R}^d \to \mathbb{R}^{\geq 0}$ with Lipschitz constant L and any θ with $0 < \theta L < 1$, we have:*

$$\frac{1}{1+\theta L}\left\|\rho^\beta\nabla u\right\|_{L^1} \leq \int_{\mathbb{R}^d}\int_{B(0,1)}\rho^\beta(x-\theta\rho(x)y)|\nabla u(x-\theta\rho(x)y)|\phi(y)\,dy\,dx \leq \frac{1}{1-\theta L}\left\|\rho^\beta\nabla u\right\|_{L^1},$$

when $\rho^\beta|\nabla u| \in L^1$.

Proof. (of Corollary D.5.1) Apply Lemma D.5 with $\delta(x) = \theta\rho(x)$, and $f(x) = \rho^\beta(x)\nabla u(x)$. □

This corollary will be used to bound the numerator of our Rayleigh quotient. Note that the expression

$$\int_{\mathbb{R}^d}\int_{B(0,1)}\rho^\beta(x-\theta\rho(x)y)|\nabla u(x-\theta\rho(x)y)|\phi(y)dydx$$

is close to $\int_{\mathbb{R}^d}\rho^\beta(x)\nabla|u_\theta(x)|dx$ when $\theta \leq \frac{1}{2L}$. This is the guiding intuition behind how Corollary D.5.1 and Lemma D.5 will be used, and will be formalized later in our proof of Theorem D.4.

We present another simple corollary whose proof is equally straightforward. This corollary will be used to bound the denominator, and is a small generalization of Corollary D.5.1. We write down both corollaries anyhow, since this will make it easier to interpret our bounds on the Rayleigh quotient.

Corollary D.5.2. *For any Lipschitz continuous function $\rho : \mathbb{R}^d \to \mathbb{R}^{\geq 0}$ with Lipschitz constant L, any $0 < t < 1$, and any θ with $0 < \theta L < 1$, we have:*

$$\frac{1}{1+\theta L}\left\|\rho^\beta\nabla u\right\|_{L^1} \leq \int_{\mathbb{R}^d}\int_{B(0,1)}\rho^\beta(x-\theta t\rho(x)y)|\nabla u(x-\theta t\rho(x)y)|\phi(y)\,dy\,dx \leq \frac{1}{1-\theta L}\left\|\rho^\beta\nabla u\right\|_{L^1}$$

Proof. (of Corollary D.5.2) Apply Lemma D.5 with $\delta(x) = \theta t\rho(x)$, and $f(x) = \rho^\beta(x)\nabla u(x)$. □

A key technical step is to observe that the above two corollaries hold when u is a function of bounded variation provided the left and right terms are interpreted as their variation. In particular, consider $A \subset \Re^d$ with finite perimeter and let u be the characteristic function of A. ($u(x) = 1$ if $x \in A$ and zero otherwise). Then there exists a sequence of functions $\{u_n\}_{n=1}^{\infty} \subset C^{\infty}(\Re^d)$ with $u_n \to u$ in L^1 for which [119]

$$|\partial A|_\beta = \lim_{n\to\infty} \int_\Omega \rho^\beta |\nabla u_n| =: \int_\Omega \rho^\beta |\nabla u|. \tag{4.4}$$

Interchanging A and $\Omega \setminus A$ if necessary, it follows that $\Phi(A)$ defined in Definition D.2 can be written as

$$\Phi(A) = \frac{\int_\Omega \rho^\beta |\nabla u|}{\int_\Omega \rho^\alpha u} = \lim_{n\to\infty} \frac{\int_\Omega \rho^\beta \, |\nabla u_n|}{\int_\Omega \rho^\alpha \, |u_n|}.$$

Now we are ready to prove our main Theorem, which is the Buser inequality for probability densities stated in Theorem D.4.

Proof. (of Theorem D.4)

Fix $A \subset \mathbb{R}^d$ with $|A|_\alpha \le |1|_\alpha/2$ and let $u(x) = \chi_A(x)$ be the characteristic function of A. Setting $\overline{u}$ to be the weighted average of u,

$$\overline{u} = \frac{\int \rho^\alpha u}{\int \rho^\alpha} = \frac{\int_A \rho^\alpha}{\int \rho^\alpha} = \frac{|A|_\alpha}{|1|_\alpha} \in [0, 1/2], \qquad \text{then} \qquad \int \rho^\alpha (u - \overline{u}) = 0,$$

and

$$\|\rho^\alpha (u - \overline{u})\|_{L^1} = \int \rho^\alpha |u - \overline{u}| = 2|A|_\alpha (1 - \overline{u}) = 2 \int \rho^\alpha |u - \overline{u}|^2. \tag{4.5}$$

Since $|A|_\alpha = \|u\|_{L^1_\alpha}$ and $1 - \overline{u} \in [0, 1/2]$ it follows that

$$(1/2) \frac{\left\|\rho^\beta \nabla u\right\|_{L^1}}{\|\rho^\alpha (u - \overline{u})\|_{L^1}} \le \Phi(A) = \frac{\left\|\rho^\beta \nabla u\right\|_{L^1}}{\|\rho^\alpha u\|_{L^1}} \le \frac{\left\|\rho^\beta \nabla u\right\|_{L^1}}{\|\rho^\alpha (u - \overline{u})\|_{L^1}}. \tag{4.6}$$

In the calculations below we omit the limiting argument with smooth approximations of u in equation (4.4) which justify formula involving ∇u. In particular, only the L^1 norm of $\rho^\beta |\nabla u|$ will appear in the estimates since this has meaning while the L^2 norm is undefined.

Next, let u_θ be the mollification of (an extension of) u given by equation (4.1). Then $u_\theta(x)$ is a local average average of u so $u_\theta(x) \ge 0$, $\|u_\theta\|_{L^\infty} \le 1$ and $\|u - u_\theta\|_{L^\infty} \le 1$. Letting L denote the Lipschitz constant of ρ, the parameter θ will to be chosen less than $1/(2L)$ so that that Lemma D.5 is applicable with constant $c = 1/2$.

The remainder of the proof constructs an upper bound on the numerator $\int_{\mathbb{R}^d} \rho^\gamma |\nabla u_\theta|^2$ of the Rayleigh quotient for $u_\theta - \overline{u}_\theta$ by $\left\|\rho^\beta \nabla u\right\|_{L^1}$ and to lower bound the denominator $\int_{\mathbb{R}^d} \rho^\alpha (u_\theta - \overline{u}_\theta)^2$ by $\|\rho^\alpha (u - \overline{u})\|_{L^1}$. The conclusion of the theorem then follows from equation (4.6).

D.4 Upper Bounding the Numerator

To bound the L^2 norm in the numerator of the Rayleigh quotient by the L^1 norm in the numerator of the expression for $\Phi(A)$ it is necessary to obtain uniform bound on $\rho(x)\nabla u_\theta(x)$.

Lemma D.6. *Let u be any function, and let u_θ be defined as in Equation 4.1. Let $\rho : \Re^d \to \Re_{\geq 0}$ be an L-Lipschitz function.*

$$\|\rho(x)\nabla u_\theta(x)\|_{L^\infty} \leq \|u\|_{L^\infty} \frac{d(2+3L)}{\theta} \tag{4.7}$$

Proof. In order to prove this lemma, we first need to get a handle on $\nabla u_\theta(x)$, which is the gradient of u after mollification by θ.

We take the the second representation of u_θ in equation (4.1) to get

$$\nabla u_\theta(x) = \int_{\mathbb{R}^d} u(z) \left\{ \frac{-d}{\theta\rho(x)} \phi_{\theta\rho}(x-z)\nabla\rho + \frac{1}{(\theta\rho(x))^{d+1}} \left(I + \nabla\rho(x) \otimes \frac{x-z}{\theta\rho(x)} \right) \nabla\phi\left(\frac{x-z}{\theta\rho(x)} \right) \right\} dz, \tag{4.8}$$

which is a consequence of the multivariable chain rule. Here, $v \otimes u$ refers to the outer product of v and u.

Multiplying by ρ gives:

$$\rho(x)\nabla u_\theta(x) = \int_{\mathbb{R}^d} u(z) \left\{ \frac{-d}{\theta} \phi_{\theta\rho(x)}(x-z)\nabla\rho(x) + \frac{1}{(\theta^{d+1}\rho(x)^d)} \left(I + \nabla\rho(x) \otimes \frac{x-z}{\theta\rho(x)} \right) \nabla\phi\left(\frac{x-z}{\theta\rho(x)} \right) \right\} dz. \tag{4.9}$$

Now, we can bound the above equation by carefully bounding each part. We note:

$$\int_{\mathbb{R}^d} \frac{1}{(\theta^{d+1}\rho(x)^d)} \nabla\phi\left(\frac{x-z}{\theta\rho(x)} \right) dz \tag{4.10}$$

$$= \int_{\mathbb{R}^d} \frac{1}{(\theta^{d+1}\rho(x)^d)} \nabla\phi\left(\frac{-z}{\theta\rho(x)} \right) dz \tag{4.11}$$

$$= \frac{1}{\theta} \int_{\mathbb{R}^d} \nabla\phi(-y) dy \tag{4.12}$$

where the last step follows by a simple change of variable. Here, we note that $\nabla\phi(y)$ is a vector, and the integral is over $\mathbb{R}^d$, which is how we eliminated $\frac{1}{(\theta\rho(x))^d}$ from the expression.

Next, we examine the term:

$$I + \nabla\rho(x) \otimes \frac{x-z}{\theta\rho(x)} \tag{4.13}$$

Here, we aim to bound the operator norm of this matrix. Here, we note that

$$|x-z| \leq \theta\rho(x)$$

when

$$\nabla\phi\left(\frac{x-z}{\theta\rho(x)} \right) \neq 0$$

and thus, when the latter equation holds, we can say:

$$\left|\frac{x-z}{\theta\rho(x)}\right| < 1.$$

Since $|\nabla\rho(x) < L|$, we now have:

$$|I + \nabla\rho(x) \otimes \frac{x-z}{\theta\rho(x)}|_2 < 3/2 \tag{4.14}$$

Combining Equation 4.14 Equation 4.10 to show:

$$\left|\int_{\mathbb{R}^d} \frac{1}{(\theta^{d+1}\phi(x)^d)} \left(I + \nabla\rho(x) \otimes \frac{x-z}{\theta\rho(x)}\right) \nabla\phi\left(\frac{x-z}{\theta\rho(x)}\right) dz\right| \tag{4.15}$$

$$\leq \frac{(1+L)}{\theta} \int_{\mathbb{R}^d} |\nabla\phi(y)|dy, \tag{4.16}$$

where $L = \|\nabla\rho\|_{L^\infty}$ is the Lipschitz constant for ρ. We note that Section D.7 shows that

$$\int_{\mathbb{R}^d} |\nabla\phi(y)dy| \leq 2d. \tag{4.17}$$

and therefore:

$$\left|\int_{\mathbb{R}^d} \frac{1}{(\theta^{d+1}\phi(x)^d)} \left(I + \nabla\rho(x) \otimes \frac{x-z}{\theta\rho(x)}\right) \nabla\phi\left(\frac{x-z}{\theta\rho(x)}\right) dz\right| \tag{4.18}$$

$$\leq \frac{2d(1+L)}{\theta} \tag{4.19}$$

Now we turn our attention to the first term, which is:

$$\int_{\mathbb{R}^d} \frac{-d}{\theta}\phi_{\theta\rho(x)}(x-z)\nabla\rho(x)dz \tag{4.20}$$

We note that

$$\int_{\mathbb{R}^d} \left|\phi_{\theta\rho(x)}(x-z)\right| dz = 1$$

by our definition of ϕ (which was defined when we defined u_θ). Combining this with $|\nabla\rho(x)| < L$, we get:

$$\int_{\mathbb{R}^d} \left|\frac{-d}{\theta}\phi_{\theta\rho(x)}(x-z)\nabla\rho(x)\right| dz \tag{4.21}$$

$$< \frac{dL}{\theta} \tag{4.22}$$

Therefore,

$$\left|\int_{\mathbb{R}^d} \frac{-d}{\theta\rho(x)}\phi_{\theta\rho}(x-z)\nabla\rho + \frac{1}{(\theta\rho(x))^{d+1}} \left(I + \nabla\rho(x) \otimes \frac{x-z}{\theta\rho(x)}\right) \nabla\phi\left(\frac{x-z}{\theta\rho(x)}\right) dz\right|$$

$$\leq \frac{d}{\theta}(L + 2(1+L))$$

$$= \frac{d(2+3L)}{\theta}. \tag{4.23}$$

where the first inequality comes from combining Equations 4.19 and 4.22.

This allows us to bound $\|\rho(x)\nabla u_\theta(x)\|_{L^\infty}$:

$$\|\rho(x)\nabla u_\theta(x)\|_{L^\infty} \tag{4.24}$$

$$= \left\| \left| \int_{\mathbb{R}^d} u(z) \left\{ \frac{-d}{\theta} \phi_{\theta\rho(x)}(x-z)\nabla\rho(x) + \frac{1}{(\theta^{d+1}\rho(x)^d)} \left(I + \nabla\rho(x) \otimes \frac{x-z}{\theta\rho(x)} \right) \nabla\phi\left(\frac{x-z}{\theta\rho(x)} \right) \right\} dz \right| \right\|_{L^\infty} \tag{4.25}$$

$$\leq \|u\|_{L^\infty} \left\| \int_{\mathbb{R}^d} \left| \frac{-d}{\theta} \phi_{\theta\rho(x)}(x-z)\nabla\rho(x) + \frac{1}{(\theta^{d+1}\rho(x)^d)} \left(I + \nabla\rho(x) \otimes \frac{x-z}{\theta\rho(x)} \right) \nabla\phi\left(\frac{x-z}{\theta\rho(x)} \right) \right| dz \right\|_{L^\infty} \tag{4.26}$$

$$\leq \|u\|_{L^\infty} \frac{d(2+3L)}{\theta} \tag{4.27}$$

where we make use of the fact that $\|ab\|_{L^\infty} < \|a\|_{L^\infty} \|b\|_{L^1}$. This completes our proof. □

Next, we want an L_1 bound on $\rho^\beta(x)\nabla u_\theta(x)$.

Lemma D.7. *Let u be any function, and let u_θ be defined as in Equation 4.1. Let $\rho : \Re^d \to \Re_{\geq 0}$ be an L-Lipschitz function, and let $\theta L < 1/2$.*

Then:

$$\left\|\rho^\beta(x)\nabla u_\theta(x)\right\|_{L^1} \leq C_\beta \left\|\rho^\beta(x)\nabla u(x)\right\|_{L^1} \tag{4.28}$$

Proof. First, we take the gradient first representation of u_θ in equation (4.1). Using the chain rule gives us an alternate form for $\nabla u_\theta(x)$:

$$\nabla u_\theta(x) = \int_{\Re^d} (I - \theta\nabla\rho \otimes y)\nabla u(x - \theta\rho y)\phi(y)\, dy, \tag{4.29}$$

so

$$\rho^\beta(x)\nabla u_\theta(x) = \int_{\Re^d} (I - \theta\nabla\rho \otimes y)\frac{\rho^\beta(x)}{\rho^\beta(x-\theta\rho y)}\rho^\beta(x - \theta\rho y)\nabla u(x - \theta\rho y)\phi(y)\, dy. \tag{4.30}$$

The ratio in the integrand is bounded using the Lipschitz assumption on ρ (and $|y| \leq 1$),

$$\frac{\rho(x)}{\rho(x-\theta\rho y)} \leq \frac{\rho(x)}{\rho(x) - L\theta\rho(x)} = \frac{1}{1-L\theta} \leq 2, \qquad \text{when } \theta < 1/(2L). \tag{4.31}$$

Note that

$$\|I - \theta\nabla\rho \otimes y\|_2 \leq 3/2 \tag{4.32}$$

where $\|M\|_2$ represents the ℓ^2 matrix norm of M. This is because $|\nabla\rho(x)| \leq L$, and $\theta L < 1/2$, and $|y| \leq 1$ every time $\phi(y) \neq 0$, and thus

$$\frac{I}{2} \preceq I - \theta\nabla\rho \otimes y \preceq \frac{3I}{2}.$$

Therefore, we can now apply Corollary D.5.1 to Equation (4.30) to show:

$$\begin{aligned}
&\left\|\rho^\beta(x)\nabla u_\theta(x)\right\|_{L^1} \\
&\leq \|I - \theta\nabla\rho(x) \otimes y\|_2 \cdot \max_x \left(\frac{\rho(x)}{\rho(x-\theta\rho y)}\right) \cdot \int_{\mathbb{R}^d} \left|\int_{\mathbb{R}^d} \rho^\beta(x-\theta\rho y)\nabla u(x-\theta\rho y)\phi(y)dy\right| \\
&\leq 3 \cdot 2^{\beta-1} \int_{\mathbb{R}^d} \rho^\beta(x-\theta\rho(x)y)\nabla u(x-\theta\rho(x)y \\
&\leq 3 \cdot 2^\beta \left\|\rho^\beta \nabla u\right\|_{L^1}, \qquad \text{when } \theta < 1/(2L).
\end{aligned}$$

here, the first inequality comes from the equation $\|abc\|_{L^1} \leq \|a\|_{L^\infty} \|b\|_{L^\infty} \|c\|_{L^1}$, the second inequality comes from Equations 4.31 and 4.32, and the third inequality comes from Corollary D.5.1 assuming $\theta L \leq 1/2$. □

Lemma D.8. *For any L-Lipschitz distribution ρ, any function u, and any θ such that $\theta L < 1/2$:*

$$\int_{\mathbb{R}^d} \rho^\gamma |\nabla u_\theta|^2 \leq C_\beta \left\|\rho^{\gamma-\beta-1}\right\|_{L^\infty} \frac{d(2+3L)}{\theta} \|u\|_{L^\infty} \left\|\rho^\beta \nabla u\right\|_{L^1}, \tag{4.33}$$

Proof. Combining the two estimates from Lemma D.6 and D.7 gives an upper bound for the Rayleigh quotient

$$\int_{\mathbb{R}^d} \rho^\gamma |\nabla u_\theta|^2 = \int_{\mathbb{R}^d} \rho^{\gamma-\beta-1}\, \rho|\nabla u_\theta|\, \rho^\beta |\nabla u_\theta| \leq 3 \cdot 2^{\beta+1} \left\|\rho^{\gamma-\beta-1}\right\|_{L^\infty} \frac{d(2+3L)}{\theta} \|u\|_{L^\infty} \left\|\rho^\beta \nabla u\right\|_{L^1}, \tag{4.34}$$

□

We note that in the case where $\gamma = \beta + 1$, and if u is a step function, the expression would simplify to:

$$\int_{\mathbb{R}^d} |\rho^\gamma |\nabla u_\theta|^2 \leq 3 \cdot 2^{\beta+1} \frac{d(2+3L)}{\theta} \|u\|_{L^\infty} \left\|\rho^\beta \nabla u\right\|_{L^1},$$

D.5 Lower Bound on the Denominator

Let $\overline{u}$ and $\overline{u}_\theta$ be the ρ^α–weighted averages of u and u_θ. For any function f, let $\|f\|_{L^2_\alpha}$ denote the L^2 norm of $\rho^\alpha f$, and let $\|f\|_{L^1}$ denote the L^1 norm of $\rho^\alpha f$ for functions f where these two quantities are well defined. Our core lemma is a bound on $\|u_\theta - \overline{u}_\theta\|_{L^2_\alpha}$ in terms of l_1 and weighted l_1 norms of ∇u and $u - \overline{u}$ respectively.

Lemma D.9. *Let ρ be an L-Lipschitz function $\rho : \Re^d \to \Re_{\geq 0}$, and let θ be such that $\theta L < 1/2$. Let u be an indicator function of a set A with finite β-perimeter. Let $\overline{u}$ be defined as $\overline{u}(x) := u(x) - \int u(y)dy$ u_θ be defined as in Equation 4.1, and $\overline{u}_\theta$ be defined as $\overline{u}_\theta(x) := u_\theta(x) - \int u_\theta(y)dy$. Then:*

$$\|u_\theta - \overline{u}_\theta\|^2_{L^2_\alpha} \geq (1/4) \|u - \overline{u}\|_{L^1_\alpha} - C(\beta)\theta \left\|\rho^{\alpha+1-\beta}\right\|_{L^\infty} \left\|\rho^\beta \nabla u\right\|_{L^1}, \qquad \textit{when } \theta < 1/(2L). \tag{4.35}$$

Note that when $\alpha + 1 = \beta$, as is true when $(\alpha, \beta, \gamma) = (1, 2, 3)$, the inequality in Lemma D.9 becomes:

$$\|u_\theta - \overline{u}_\theta\|_{L^2_\alpha}^2 \geq (1/4) \|u - \overline{u}\|_{L^1_\alpha} - C(\beta)\theta \left\|\rho^\beta \nabla u\right\|_{L^1}, \qquad \text{when } \theta < 1/(2L).$$

The estimate in Lemma D.9 will be combined with the estimate in Lemma D.8 to prove Theorem D.4 in Section D.6.

Proof. The key to this proof is to upper bound the quantity $\|u_\theta - \overline{u}_\theta\|_{L^2_\alpha}$ with the expression appearing in Corollary D.5.2. We will do so by a series of inequalities, application of the fundamental theorem of calculus, and more.

Using the property that subtracting the average from a function reduces the L^2 norm it follows that

$$\begin{aligned}
&\|u_\theta - \overline{u}_\theta\|_{L^2_\alpha} \\
&\geq \|u - \overline{u}\|_{L^2_\alpha} - \|u_\theta - u - (\overline{u}_\theta - \overline{u})\|_{L^2_\alpha} \\
&\geq \|u - \overline{u}\|_{L^2_\alpha} - \|u_\theta - u\|_{L^2_\alpha}.
\end{aligned}$$

If $a \geq b - c$ then $a^2 \geq b^2/2 - c^2$, so a lower bound for the denominator of the Rayleigh quotient

$$\begin{aligned}
&\|u_\theta - \overline{u}_\theta\|_{L^2_\alpha}^2 \\
&\geq (1/2) \|u - \overline{u}\|_{L^2_\alpha}^2 - \|u_\theta - u\|_{L^2_\alpha}^2 \\
&\geq (1/4) \|u - \overline{u}\|_{L^1_\alpha} - \|u_\theta - u\|_{L^1_\alpha},
\end{aligned} \tag{4.36}$$

where the identity $\|u - \overline{u}\|_{L^2_\alpha}^2 = \|u - \overline{u}\|_{L^1_\alpha}/2$ from Equation 4.5, and the bound $\|u_\theta - u\|_{L^\infty} \leq 1$, were used in the last step.

It remains to estimate the difference $\|u_\theta - u\|_{L^1_\alpha}$. To do this, we use the multivariable fundamental theorem of calculus to write

$$\begin{aligned}
u_\theta(x) - u(x) &= \int (u(x - \theta\rho y) - u(x))\phi(y)\, dy \\
&= \int\int_0^1 -\theta\rho(x)\nabla u(x - t\theta\rho(x)y).y\phi(y)\, dt\, dy \\
&= \int\int_0^1 \frac{-\theta\rho(x)}{\rho^\beta(x - t\theta\rho(x)y)}\rho^\beta(x - t\theta\rho(x)y)\nabla u(x - t\theta\rho(x)y).y\phi(y)\, dt\, dy,
\end{aligned}$$

where the first and second equalities came from application of the multivariable fundamental theorem of calculus, and the last equation is straightforward. This tells us that:

$$\begin{aligned}
&\rho^\alpha(x)(u_\theta\rho(x) - u(x)) \\
&= \int\int_0^1 \frac{-\theta\rho^{\alpha+1}(x)}{\rho^\beta(x - t\theta\rho y)}\rho^\beta(x - t\theta\rho(x)y)\nabla u(x - t\theta\rho(x)y).y\phi(y)\, dt\, dy. \\
&= \int\int_0^1 \frac{\rho^\beta(x)}{\rho^\beta(x - t\theta\rho(x)y)}\frac{-\theta\rho^{\alpha+1}(x)}{\rho^\beta(x)}\rho^\beta(x - t\theta\rho(x)y)\nabla u(x - t\theta\rho(x)y).y\phi(y)\, dt\, dy.
\end{aligned} \tag{4.37}$$

Equation (4.31) bounds the ratio $\rho(x)/\rho(x - t\theta\rho(x)y)$ as less than 2 when $\theta L < 1/2$, so Equation (4.37) is always less than or equal to:

$$\int\int_0^1 2^\beta \frac{-\theta\rho^{\alpha+1}(x)}{\rho^\beta(x)}\rho^\beta(x - t\theta\rho(x)y)\nabla u(x - t\theta\rho(x)y).y\phi(y)\,dy\,dt. \tag{4.38}$$

An application of Corollary D.5.2 then shows

$$\int\int_0^1 2^\beta \frac{-\theta\rho^{\alpha+1}(x)}{\rho^\beta(x)}\rho^\beta(x - t\theta\rho(x)y)\nabla u(x - t\theta\rho(x)y).y\phi(y)\,dy\,dt. \tag{4.39}$$

$$\leq \int\int_0^1 2^\beta \frac{-\theta\rho^{\alpha+1}(x)}{\rho^\beta(x)}\rho^\beta(x - t\theta\rho(x)y)\ |\nabla u(x - t\theta\rho(x)y)\phi(y)|\ dy\,dt. \tag{4.40}$$

$$\leq 2^{\beta+1}\left\|\rho^{\alpha+1-\beta}\right\|_{L^\infty}\theta\int\int_0^1 \rho^\beta(x - t\theta\rho(x)y)\ |\nabla u(x - t\theta\rho(x)y)\phi(y)|\ dy\,dt. \qquad \text{when } \theta < 1/(2L). \tag{4.41}$$

$$\leq 2^{\beta+1}\left\|\rho^{\alpha+1-\beta}\right\|_{L^\infty}\theta\left\|\rho^\beta\nabla u\right\|_{L^1} \tag{4.42}$$

where the last inequality follows from Corollary D.5.2.

Using this estimate in (4.36) gives a lower bound on the denominator of the Rayleigh quotient,

$$\|u_\theta - \overline{u}_\theta\|_{L^2_\alpha}^2 \geq (1/4)\|u - \overline{u}\|_{L^1_\alpha} - 2^{\beta+1}\theta\left\|\rho^{\alpha+1-\beta}\right\|_{L^\infty}\left\|\rho^\beta\nabla u\right\|_{L^1}, \qquad \text{when } \theta < 1/(2L). \tag{4.43}$$

as desired. □

D.6 Bounding the Rayleigh Quotient (Proof of Theorem D.4)

Combining Lemmas D.8 and Lemmas D.9 provides an upper bound for the Rayleigh quotient of $u_\theta - \overline{u}_\theta$,

$$\begin{aligned}
\lambda_2 &\leq \frac{\int_{\mathbb{R}^d}\rho^\gamma|\nabla u_\theta|^2}{\int_{\mathbb{R}^d}\rho^\alpha(u_\theta - \overline{u}_\theta)^2} \\
&\leq \frac{d\cdot 3\cdot 2^\beta}{\theta}\frac{\left\|\rho^{\gamma-\beta-1}\right\|_{L^\infty}(2+3L)\left\|\rho^\beta\nabla u\right\|_{L^1}}{\|u-\overline{u}\|_{L^1_\alpha} - 2^{\beta+1}\theta\left\|\rho^{\alpha+1-\beta}\right\|_{L^\infty}\left\|\rho^\beta\nabla u\right\|_{L^1}} \\
&\leq \frac{d\cdot 3\cdot 2^\beta}{\theta}\frac{\left\|\rho^{\gamma-\beta-1}\right\|_{L^\infty}(2+3L)}{1 - 2^{\beta+1}\theta\left\|\rho^{\alpha+1-\beta}\right\|_{L^\infty}\Phi(A)}\Phi(A).
\end{aligned}$$

Selecting $\theta = (1/2)\min\left(1/\left(2^{\beta+1}\left\|\rho^{\alpha+1-\beta}\right\|_{L^\infty}\Phi(A)\right), 1/L\right)$ shows

$$\lambda_2 \leq 2d\cdot 3\cdot 2^\beta\left\|\rho^{\gamma-\beta-1}\right\|_{L^\infty}(2+3L)\max\left(L\Phi(A), 2^{\beta+1}\left\|\rho^{\alpha+1-\beta}\right\|_{L^\infty}\Phi(A)^2\right).$$

When $\gamma = (1,2,3)$, this simplifies into:

$$\lambda_2 \leq 12(2+3L)d\max\left(L\Phi(A), 8\Phi(A)^2\right).$$

We note that, via the work shown in Section D.8, we can strengthen our inequality to:

$$\lambda_2 \leq 24d\max\left(L\Phi(A), 8\Phi(A)^2\right).$$

□

D.7 Gradient of Mollifier

Let ϕ be a standard mollifier i.e. $\phi \in C_c^\infty(\mathbb{R}^d)$ is a function from $\mathbb{R}^d \to [0,\infty)$ satisfying $\int_{\mathbb{R}^d} \phi\, dx = 1$ and $\mathrm{supp}(\phi) \subseteq B(0,1)$. We will define ϕ by its profile. Namely, let $\widehat{\phi}(r) : [0,\infty) \to [0,1]$ be a fixed monotone decreasing profile with $\widehat{\phi}(0) = 1$, $0 < \widehat{\phi}(r) < 1$ for $0 < r < 1$, and $\widehat{\phi}(r) = 0$ for $r \geq 1$. Then define $\phi : \Re^d \to \Re$ by $\phi(x) = c\widehat{\phi}(|x|)$ with $c > 0$ chosen so that $\int_{\Re^d} \phi(x)\, dx = 1$; that is,

$$1 = \int_{\Re^d} \phi(x)\, dx = c|S^{d-1}| \int_0^1 \widehat{\phi}(r) r^{d-1}\, dr \qquad \Rightarrow \qquad c = \frac{1}{|S^{d-1}| \int_0^1 \widehat{\phi}(r) r^{d-1}\, dr},$$

where $|S^{d-1}|$ is the $(d-1)$–area of the unit sphere in $\Re^d$. We claim the L_1 norm of the gradient of $\nabla\phi(x)$ is linear in d.

Lemma D.10.

$$\int_{\Re^d} |\nabla\phi(x)|\, dx \leq (d-1) \left(\frac{d 2^d}{\widehat{\phi}(1/2)} \right)^{1/(d-1)} \stackrel{d\to\infty}{\longrightarrow} 2(d-1).$$

For the classic mollifier $\widehat{\phi}(r) = \exp(-1/(1-r^2))$ *we get*

$$\int_{\Re^d} |\nabla\phi(x)|\, dx \leq 2d.$$

From the formula $\nabla\phi(x) = c\widehat{\phi}'(|x|)(x/|x|)$ we compute

$$\begin{aligned}
\int_{\Re^d} |\nabla\phi(x)|\, dx &= c|S^{d-1}| \int_0^1 |\widehat{\phi}'(r)| r^{d-1}\, dr \\
&= c|S^{d-1}| \int_0^1 -\widehat{\phi}'(r) r^{d-1}\, dr \\
&= c|S^{d-1}| \int_0^1 \widehat{\phi}(r)(d-1) r^{d-2}\, dr \\
&= (d-1) \frac{\int_0^1 \widehat{\phi}(r) r^{d-2}\, dr}{\int_0^1 \widehat{\phi}(r) r^{d-1}\, dr}.
\end{aligned}$$

To estimate the numerator use Holder's inequality: for $1 \leq s, s' \leq \infty$ with $1/s + 1/s' = 1$

$$\int fg \leq \left(\int |f|^s \right)^{1/s} \left(\int |g|^{s'} \right)^{1/s'}.$$

Set $s = (d-1)/(d-2)$ and $s' = d-1$ to get

$$\int_0^1 \widehat{\phi}(r) r^{d-2}\, dr = \int_0^1 \widehat{\phi}(r)^{1/s} r^{d-2} \times \widehat{\phi}(r)^{1/s'}\, dr \leq \left(\int_0^1 \widehat{\phi}(r) r^{d-1}\, dr \right)^{1/s} \left(\int_0^1 \widehat{\phi}(r)\, dr \right)^{1/s'}.$$

It follows that

$$\int_{\Re^d} |\nabla \phi(x)|\, dx \leq (d-1) \left(\frac{\int_0^1 \widehat{\phi}(r)\, dr}{\int_0^1 \widehat{\phi}(r) r^{d-1}\, dr} \right)^{1/(d-1)}. \tag{4.44}$$

Since $0 \leq \widehat{\phi}(r) \leq 1$ we can bound the numerator by 1, and since $\widehat{\phi}(r)$ is monotone decreasing we have $\widehat{\phi}(r) \geq \widehat{\phi}(1/2)$ on $(0, 1/2)$, so

$$\int_{\Re^d} |\nabla \phi(x)|\, dx \leq (d-1) \left(\frac{1}{\widehat{\phi}(1/2) \int_0^{1/2} r^{d-1}\, dr} \right)^{1/(d-1)} \leq (d-1) \left(\frac{d 2^d}{\widehat{\phi}(1/2)} \right)^{1/(d-1)} \stackrel{d\to\infty}{\longrightarrow} 2(d-1). \tag{4.45}$$

It will be convenient to write equation 4.45 as a simple inequality. Observer that

$$\left(\frac{d 2^d}{\widehat{\phi}(1/2)} \right)^{1/(d-1)}$$

is monotone decreasing. We now pick the classic $\widehat{\phi}(r) = \exp(-1/(1-r^2))$ we have $\widehat{\phi}(1/2) \geq 1/4$ and if $d \geq 5$ the right hand side of equation (4.45 is bounded by $2d$. If $d < 5$ explicit computations of the integrals shows the right hand side of equation (4.44 is bounded by $2d$.

D.8 Scaling

In this section we show that if one scales the density function ρ then the isoperimetric value $\Phi(A)$ and the Rayleigh quotient $R(u)$ scale nicely. More formally Let $A \subset \Omega \subseteq \Re^d$, ρ a density function over a domain Ω, and u an arbitrary differentiable function over Ω.

Consider the transformation $\widehat{x} = \ell x$ with $\ell > 0$ which maps Ω to the domain $\widehat{\Omega} = \{\ell x \mid x \in \Omega\}$. Given $u : \Omega \to \Re$, we define $\widehat{u} : \widehat{\Omega} \to \Re$ by $\widehat{u}(\widehat{x}) = u(x)$. We will future scale ρ by $\alpha \widehat{\rho}(\widehat{x}) = \ell\, \rho(x)$ where $\alpha > 0$.

Theorem D.11. *When scaling by α and ℓ then*

$$\Phi(A) = \alpha \widehat{\Phi}(\widehat{A})$$

and

$$R(u) = \alpha^2 \widehat{R}(\widehat{u}) \quad \textit{and thus} \quad \lambda_2 = \alpha^2 \widehat{\lambda}_2$$

.

We will use this scaling theorem to improve the bounds of theorem D.4.

That is, if we have a density function ρ over a domain Ω the isoperimetric number that the fundamental eigenvalue only change as a function of the scaling. Thus the optimal cut and eigenvector are unchanged by scaling up to the transformation.

If u and l are as defined above then we get the simple but basic identity. Suppose that $u : \mathbb{R} \to \mathbb{R}$ then:

$$\frac{\partial u}{\partial x} = \frac{\partial \widehat{u}}{\partial \widehat{x}} \frac{\partial \widehat{x}}{\partial x} = \frac{\partial \widehat{u}}{\partial \widehat{x}} \ell, \qquad \text{in general we get} \qquad |\nabla u(x)| = \ell |\widehat{\nabla} \widehat{u}(\widehat{x})|.$$

In the case of $\rho : \Omega \to (0, \infty)$, where $\widehat{\rho} : \widehat{\Omega} \to (0, \infty)$ is defined by $\alpha\widehat{\rho}(\widehat{x}) = \ell\,\rho(x)$ we get that

$$|\nabla\rho(x)| = \alpha|\widehat{\nabla}\widehat{\rho}(\widehat{x})|.$$

It follows that $L_{\widehat{\rho}}$ and L_ρ, the Lipschitz constants for $\widehat{\rho}$ and ρ, satisfy $L_{\widehat{\rho}} = (1/\alpha)L_\rho$.

- Since $d\widehat{x} = \ell^d\,dx$ we have
$$\int_\Omega \rho\,dx = \frac{\alpha}{\ell^{d+1}}\int_{\widehat{\Omega}} \widehat{\rho}\,d\widehat{x},$$

- If $A \subset \Omega$ and $\widehat{A} = \ell A \subset \widehat{\Omega}$, let $f_A(x) = 1$ if $x \in A$ and zero otherwise, and similarly $f_{\widehat{A}} = 1$ if $\widehat{x} \in \widehat{A}$ and zero otherwise. We next perform a set of standard integral calculations.

$$\int_\Omega \rho^2|\nabla f_A|\,dx = \int_{\widehat{\Omega}} (\frac{\alpha}{\ell})^2\widehat{\rho}^2\ell|\widehat{\nabla} f_{\widehat{A}}|\,\frac{1}{\ell^d}d\widehat{x} \tag{4.46}$$

$$= \frac{\alpha^2}{\ell^{d+1}}\int_{\widehat{\Omega}} \widehat{\rho}^2|\widehat{\nabla} f_{\widehat{A}}|\,d\widehat{x} \tag{4.47}$$

Equation 4.46 follows by making the substitutions:

$$\rho(x) = (\frac{\alpha}{\ell})\widehat{\rho}(\widehat{x}) \quad |\nabla f_A| = \ell|\widehat{\nabla} f_{\widehat{A}}| \quad dx = \frac{1}{\ell^d}d\widehat{x}$$

Observing the $f_A(x) = f_{\widehat{A}}(\widehat{x})$ we get the following identity.

$$\int_\Omega \rho f_A\,dx = \int_{\widehat{\Omega}} \frac{\alpha}{\ell}\widehat{\rho} f_{\widehat{A}}\frac{1}{\ell^d}\,d\widehat{x} = \frac{\alpha}{\ell^{d+1}}\int_{\widehat{\Omega}} \widehat{\rho} f_{\widehat{u}}\,d\widehat{x} \tag{4.48}$$

Combining equation 4.47 and equation 4.48 we get that:

$$\Phi(A) = \alpha\widehat{\Phi}(\widehat{A}) \tag{4.49}$$

- We next do a similar calculation for the Rayleigh quotient. If $u : \Omega \to \Re$ and $\widehat{u}(\widehat{x}) = u(x)$, the Rayleigh quotients can be computed as follows,

$$\int_\Omega \rho^3|\nabla u|^2\,dx = \int_{\widehat{\Omega}} (\frac{\alpha}{\ell})^3\widehat{\rho}^3\ell^2|\widehat{\nabla}\widehat{u}|^2\,\frac{1}{\ell^d}d\widehat{x} = \frac{\alpha^3}{\ell^{d+1}}\int_{\widehat{\Omega}} \widehat{\rho}^3|\widehat{\nabla}\widehat{u}|^2\,d\widehat{x}$$

$$\int_\Omega \rho u^2\,dx = \int_{\widehat{\Omega}} \frac{\alpha}{\ell}\widehat{\rho}\widehat{u}^2\frac{1}{\ell^d}\,dx = \frac{\alpha}{\ell^{d+1}}\int_{\widehat{\Omega}} \widehat{\rho}\widehat{u}^2\,dx$$

Thus

$$R(u) = \alpha^2\widehat{R}(\widehat{u})$$

.

We next use our scaling result in the $(1, 2, 3)$ case to our Buser-type bound, Theorem D.4. Theorem D.4 states that the following hold:

$$\lambda_2 \leq 24d(1 + L) \max\left(L\Phi(A), 12\Phi(A)^2\right). \tag{4.50}$$

We now make substitutions into equation 4.50 from Theorem D.11 and its proof for some parameter α to be determined.

$$\begin{aligned}
\lambda_2 = \alpha^2\widehat{\lambda}_2 &\leq \alpha^2 24d(1 + \widehat{L}) \max\left(\widehat{L}\widehat{\Phi}(\widehat{A}), 12\widehat{\Phi}(\widehat{A})^2\right) \\
&= \alpha^2 24d(1 + (L/\alpha)) \max\left((L/\alpha)(\Phi(A)/\alpha), 12(\Phi(A)/\alpha)^2\right) \\
&= 24d(1 + (L/\alpha)) \max\left(L\Phi(A), 12\Phi(A)^2\right) \\
&= 24d \max\left(L\Phi(A), 12\Phi(A)^2\right)
\end{aligned}$$

where the last line holds when taking α to infinity.

Thus we get that λ_2 only depends linear in the dimension and the Lipschitz constant:

Corollary D.11.1.

$$\lambda_2 \leq 24d \max\left(L\Phi, 12\Phi^2\right)$$

E Cheeger Inequality for Probability Density Functions

In this section, we prove the Cheeger inequality from Theorem A.5. That is a weighted Cheeger inequality in higher dimensions. This is the easier to prove than Buser's inequality, which contrasts with what happens in the graph case (the graph Buser inequality is trivial).

For a simplified proof of the Cheeger inequality for distributions in one-dimension, see Appendix I.

As we will see from simple counterexamples in Section C, the Cheeger-direction does not hold for all setting of (α, β, γ). The proof we give is requires fewer assumptions than the Buser inequality for probability densities. One, the Cheeger inequality is independent of the Lipschitz constant of ρ and two, the proof also holds when ρ is supported on a set $\Omega \subset \mathbb{R}^d$.

The proof is almost identical to the proof in one dimension and only a slight modification of standard proofs The only change in the proof is replacing the change of variables formula with a co-area formula. Let $\rho : \Omega \to \mathbb{R}_>$ be an Lipschitz density function that is (α, β, γ)-integrable over an open set $\Omega \subseteq \mathbb{R}^d$. Note a stronger hypothesis on Ω is that it is the support of ρ when $\rho : \mathbb{R}^d \to \mathbb{R}_{\leq}$.

Theorem E.1. *Let $\rho : \Omega \to \mathbb{R}_{>0}$ be a Lipschitz function. Then,*

$$\Phi^2 \leq 4 \left\| \rho^{\beta - \frac{\alpha+\gamma}{2}} \right\|_\infty^2 \lambda_2.$$

In particular, when $(\alpha, \beta, \gamma) = (1, 2, 3)$ we have

$$\Phi^2 \leq 4\lambda_2.$$

Here, Φ is the optimal (α, β)-sparsity of a cut through ρ. We note that we can say something a little stronger:

Theorem E.2. *Let $\rho : \Omega \to \mathbb{R}_{>0}$ be a Lipschitz function. Let $\Phi_{(\alpha,\beta,\gamma)}$ be the (α, β) sparsity of the (α, γ) spectral sweep cut. If $\alpha = \beta - 1 = \gamma - 2$, then:*

$$\Phi^2_{(\alpha,\beta,\gamma)} \leq 4\lambda_2$$

Proof. (of both theorems): Let $w \in W^{1,2}$, functions whose gradient is square integrable, nonzero with $\int_\Omega \rho^\alpha w \, dx = 0$. Let $v = w + a1$ where a is chosen such that $|\{v < 0\}|_\alpha = |\{v > 0\}|$. Note that

$$\begin{aligned} R(w) &= \frac{\int_\Omega \rho^\gamma \left|\nabla w\right|^2 \, dx}{\int_\Omega \rho^\alpha w^2 \, dx} \\ &\geq \frac{\int_\Omega \rho^\gamma \left|\nabla w\right|^2 \, dx}{\int_\Omega \rho^\alpha w^2 \, dx + a^2 \left|\Omega\right|_\alpha} \\ &= R(v). \end{aligned}$$

Without loss of generality, the function $u = \max(v, 0)$ satisfies $R(u) \leq R(v)$.

Let $\Omega_0 = \{v > 0\}$. Let $g = u^2$. Noting that $\nabla g = 2u\nabla u$ a.e., we can apply Cauchy-Schwarz to obtain

$$\begin{aligned} \int_{\Omega_0} \rho^\beta \left|\nabla g\right| \, dx &= 2 \int_{\Omega_0} \rho^\beta \left|u\right| \left|\nabla u\right| \, dx \\ &\leq 2 \sqrt{\int_{\Omega_0} \rho^{2\beta-\alpha} \left|\nabla u\right|^2 \, dx} \sqrt{\int_{\Omega_0} \rho^\alpha u^2 \, dx} \\ &\leq 2 \left\| \rho^{\beta - \frac{\alpha+\gamma}{2}} \right\|_\infty \sqrt{\int_{\Omega_0} \rho^\gamma \left|\nabla u\right|^2 \, dx} \sqrt{\int_{\Omega_0} \rho^\alpha u^2 \, dx}. \end{aligned}$$

Then, dividing by $\int_{\Omega_0} \rho^\alpha g \, dx$, we have

$$\frac{\int_{\Omega_0} \rho^\beta \left|\nabla g\right| \, dx}{\int_{\Omega_0} \rho^\alpha g \, dx} \leq 2 \left\| \rho^{\beta - \frac{\alpha+\gamma}{2}} \right\|_\infty \sqrt{R(w)}.$$

Let $A_t = \{g > t\}$. Then, by the weighted co-area formula,

$$\int_{\Omega_0} \rho^\beta \left|\nabla g\right| \, dx = \int_0^\infty \left|\partial A_t\right|_\beta \, dt.$$

Writing $g(x) = \int_0^{g(x)} 1 \, dt$ and applying Tonelli's theorem, we rewrite the denominator

$$\int_{\Omega_0} \rho^\alpha g \, dx = \int_0^\infty \left|A_t\right|_\alpha \, dt.$$

Thus, by averaging, there exists some t^* such that

$$\begin{aligned}\Phi &\le \Phi(A_{t^*}) \\ &\le \frac{\int_{\Omega_0} \rho^\beta \left|\nabla g\right| \, dx}{\int_{\Omega_0} \rho^\alpha g \, dx} \\ &\le 2 \left\| \rho^{\beta - \frac{\alpha+\gamma}{2}} \right\|_\infty \sqrt{R(w)}.\end{aligned}$$

Optimizing over the set $\left\{ w \in W^{1,2} \,\middle|\, w \neq 0,\ \int_\Omega \rho^\alpha w \, dx = 0 \right\}$ completes the proof. □

F Spectral Sweep Cuts have Provably Good Sparsity (proof of Theorem A.8)

Theorem E.2 tells us that

$$\Phi_{(1,2,3)}^2/4 \le \lambda_2^{(1,3)}$$

for all 1-Lipschitz ρ whose domain is on $\mathbb{R}^d$. Here, $\phi_{(1,2,3)}$ is the $(1,2)$ sparsity of the $(1,3)$-spectral sweep cut, and $\lambda_2^{(1,3)}$ is the $(1,3)$-principal eigenvalue.

Next Theorem D.4 tells us that

$$\lambda_2^{(1,3)} \le O(d\Phi_{(1,2)}),$$

where $\Phi_{(1,2)}$ is the minimal $(1,2)$-sparsity of any cut through ρ.

Therefore,

$$\Phi_{(1,2,3)}^2 \le \Phi_{(1,2)} \le \Phi_{(1,2,3)}^2,$$

where $\Phi_{(1,2)}$ is the minimum $(1,2)$-sparsity of a cut through ρ, proving Theorem A.8.

G Problems with Existing Spectral Cut Methods

In this section, we introduce a simple Lipschitz distribution where the $(\alpha = 1, \gamma = 2)$-spectral sweep cut fails to find a $(1,\beta)$ sparse cut for any $0 \le \beta < 10$. Meanwhile, the $(1,3)$-spectral sweep cut finds a desirable cut with good $(1,2)$-sparsity. We note that $(\alpha = 1, \beta > 10)$-sparse cuts are likely to find cuts where one side has extremely small probability mass, making it undesirable for machine learning.

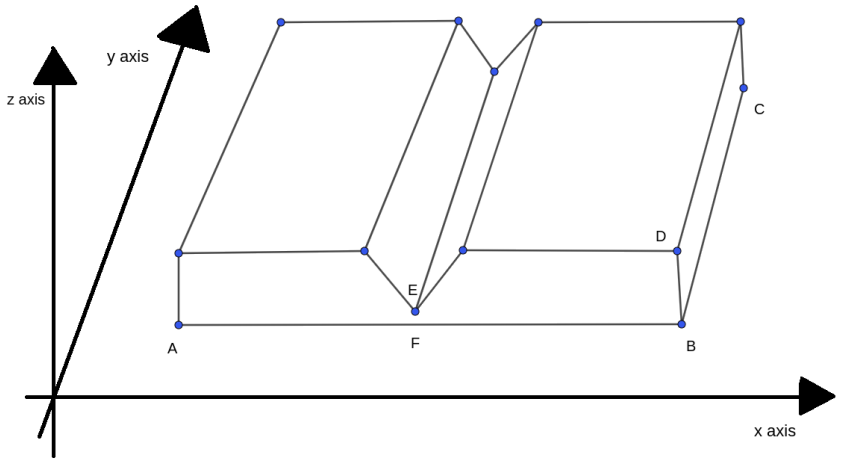

Figure 4.3: The probability density function where $\rho(x, y) = \min(\epsilon + x, \frac{1}{n})$ for arbitrary X, Y, n. Here, $\rho(x, y)$ is plotted in the z axis, and E is at point $(0, -Y, \epsilon)$. This function ρ has bad spectral sweep cuts when $\alpha = 1, \gamma = 2$.

We note that this section combined with Theorem A.8, Lemma A.6 and Lemma A.7 shows that no Cheeger and Buser inequality can hold when $\alpha = 1$ and $\gamma = 2$ for any β: this section combined with Theorem A.8 will show that the Cheeger-Buser inequalities can only hold for $\beta > 10$, while Lemma A.6 and Lemma A.7 shows that they can only hold for $\beta \leq 1$. Therefore, the Cheeger-Buser inequalities cannot hold for any β, for $\alpha = 1$ and $\gamma = 2$.

Theorem G.1. $(\alpha = 1, \gamma = 2)$***-Spectral Sweep Cut Counterexample:***

For a 1*-Lipschitz positive valued function* ρ*, let* Φ *be the sparsity of the* $(1, 3)$*-spectral sweep cut, and let* Φ_{OPT} *be the cut of optimal* $(1, \beta)$ *sparsity for any* $\beta < 10$*. There there exists a* 1*-Lipschitz density function* ρ *such that:*

$$\Phi > C \max(\Phi_{OPT}, \sqrt{\Phi_{OPT}})$$

for any constant C*.*

G.1 Our density function

We first construct our 1-Lipschitz Density function for which a $(1, 2)$ spectral cut has poor $(1, \beta)$ sparsity. Our density function has parameters X, Y, ϵ, n which we will set later.

Definition G.2. *Let* $\rho : [-X, X] \times [-Y, Y] \to \mathbb{R}$ *be a density function such that:*

$$\rho(x, y) = \min(\epsilon + x, 1/n)$$

To turn this into a 1-Lipschitz probability density function, we simply extend it to a function $\rho' : \mathbb{R}^2 \to \mathbb{R}$ where ρ' agrees with ρ on $[-X, X] \times [-Y, Y]$, and the function goes 1-Lipschitzly to 0 outside this range.

We will set $X = \sqrt{n}/10, Y = 10\sqrt{n}$, and n large, to obtain a density function where the $(1, 2)$ spectral cut has arbitrarily bad $(1, \beta)$ sparsity for all $\beta < 10$.

G.2 Proof Overview

First, we prove theorems about the zero-set of this density's $(\alpha = 1, \gamma = 2)$ eigenfunction. In particular, the zero-set of this eigenfunction must cut from the line $x = -X$ to $x = X$. It cannot cut from the line $y = -Y$ to the line $y = Y$.

We prove that any level-set of the eigenfunction can't cut from $y = -Y$ to $y = Y$. We then show that any cut that doesn't cut from $y = -Y$ to $y = Y$ has bad $(1, \beta)$ sparsity for $\beta < 10$. This completes our proof. Moreover, any cut that doesn't cut from $y = -Y$ to $y = Y$ is intuitively a poor cut of our density function, according to standard machine learning intuition.

We note the natural cut of this distribution is the straight line cut $x = 0$, which the $(1, 3)$-spectral sweep cut will find (this is an artifact of our proof, though we do not explicitly prove it here).

First, we prove a few lemmas on the zero-set of the $(1, 2)$ eigenfunction.

G.3 The Zero-set of a principal $(1, 2)$ eigenfunction is the line $y = 0$

Theorem G.3. *The Zero-set of the eigenfunction for our given density function, is the line $y = 0$.*

Lemma G.4. *Let f be any eigenfunction of our given density function, for which $f(x, y) \neq f(x, y')$ for some $x, y \neq y'$. Then*

$$\int_0^Y f(x, y)dy = 0.$$

Lemma G.5. *There exists a principal eigenfunction f_2 of our given density function, for which*

$$f_2(x, y) = f_2(-x, y) = -f_2(x, -y)$$

Proof. This follows from a (non-trivial) symmetrization argument put forward in the graph case in Guattery and Miller [147]. □

Lemma G.6. *(Nodal domains for Densities) Every principal eigenfunction f_2 of our given density function satisfies: the closure of the set $\{S = (x, y) | f_2(x, y) > 0\}$ is connected.*

Proof. This follows analogously to the proof of Fiedler's nodal domains for eigenfunctions of a graph [124]. □

Lemma G.7. *Let f be a (α, β) eigenfunction of any density function supported on a compact set $S \subset \mathbb{R}^n$ for some n. For every point in the zero-set, if any open set containing that point contains a positive element, it must also contain a negative element.*

Proof. This follows directly from the definition of eigenfunction. □

Proof. (of Theorem G.3): First, we note that there is a principal eigenfunction whose zero set contains $y = 0$, by Lemma G.5. We claim there is a principal eigenfunction for which this is the entire zero-set. This follows from Lemma G.6 and Lemma G.7. □

G.4 Any spectral sweep cut has high $(1, \beta)$ sparsity

In this section, we prove that the spectral sweep cut must have high $(1, \beta)$-sparsity for $0 < \beta < 10$, and for $\beta > 10$ the spectral sweep-cut either has high $(1, \beta)$ sparsity or else divides the probability density into two pieces, one of which has less than $\leq 1/n$ fraction of the probability mass.

Lemma G.8. *Any spectral sweep cut (of the principal $(1, 2)$ eigenfunction whose eigenvector's zero-set is the line $y = 0$) can't cut through $y = Y$ and $y = -Y$.*

Proof. This is clear. □

Lemma G.9. *Any cut that doesn't cut through both $y = Y$ and $y = -Y$ has poor $(1, \beta)$ sparsity for any $0 < \beta < 10$. For $\beta > 10$, a cut of good $(1, \beta)$ sparsity must have its smaller side contain $o_n(1)$ fraction of the mass.*

To be precise, if Φ_β is the optimal $(1, \beta)$ sparsity of the cut, and Φ is the $(1, \beta)$ sparsity induced by a cut that doesn't cut both $y = Y$ and $y = -Y$, then there is no constant C independent of n for which

$\Phi^2 < C\Phi_\beta$.

We note that Theorem G.1 follows from Lemma G.8 and G.9. Thus, it remains to show Lemma G.9.

Proof. (of Lemma G.9). We split this into two cases. Consider the side of the level set cut with smaller probability mass. The first case is when this side has at least half its probability mass outside the region $|x| < 1/n - \epsilon$. The second case is when the side has less than half its mass in this region.

In the first case, we note that we can lower bound the cut by its projection onto the x axis. A quick calculation shows that when $X = \frac{1}{10\sqrt{n}}$ and $Y = \frac{10}{\sqrt{n}}$, the $(1, \beta)$-sparsity of this cut is within a factor of 2 of the $(1, \beta)$-sparsity of the cut $y = 0$ through the uniform distribution of height $\frac{1}{n}$ supported on $[-X, X] \times [-Y, Y]$. This $(1, \beta)$ sparsity is

$$A := O(\frac{X}{n^\beta}) = O(\frac{\sqrt{n}}{n^\beta})$$

.

When ϵ is chosen to be $\frac{1}{n^2\sqrt{n}}$, then the $(1, \beta)$ sparsity of the optimal cut is the cut $x = 0$, which has $(1, \beta)$ sparsity of:

$$B := O(\frac{Y}{(n\sqrt{n})^\beta}) = O(\frac{\sqrt{n}}{(n^2\sqrt{n})^\beta}.$$

We note that this choice of ϵ is the minimum such choice such that the principal eigenvector is not constant on the Y axis.

Now we note that A^2/B goes to infinity as n gets large, if and only if

$$n^{2\beta}\sqrt{n}^\beta n/n^{2\beta}\sqrt{n}$$

goes to infinity,

or

$$\sqrt{n}(\sqrt{n}^{\beta})$$

goes to infinity. This is true for any $\beta > 0$. This proves Theorem G.1 in case 1, where at least half of the probability mass is outside the region $|x| < 1/n$.

In case 2, we consider the case when the smaller side of the cut has more than half its probability mass inside the region $|x| < 1/n - \epsilon$, which we note is a very small portion of the probability mass of the overall probability density. In this case, it turns out that we need $\beta < 10$ to give isoperimetry guarantees, since for any $\beta > 10$, it turns out that even cuts containing small probability mass are considered to have good $(1, \beta)$ sparsity, since for large β, $(1, \beta)$ sparse cuts tremendously favor small cuts, even if the smaller side has negligible probability mass.

Since at least half the mass is inside the region $|x| < 1/n$, we can assume without loss of generality that the entire probability mass of the smaller side of the cut is inside this region, by simply projecting the cut onto this region (reducing its β-perimeter while decreasing probability mass by at most a factor of 2). We can again use a symmetry argument analogous to G.5 to show that any level set of this principal eigenfunction is symmetric about the x axis (we note Lemma G.9 is slightly stronger than this as it does not assume symmetry, but for our purposes we can strictly deal with symmetric cuts, and the non-symmetric case follows through a similar argument).

Now given the cut is symmetric about the x axis, if the cut cuts through (x', y'), then it also cuts through $(-x', y')$, and we can lower bound the probability mass contained by the cut $y = y'$ with $x' \cdot \rho(x', y')$. A simple calculation using this estimate finishes the proof for us.

□

H Conclusion and Future Directions

We define a new notion of spectral sweep cuts, eigenvalues, Rayleigh quotients, and sparsity for probability densities. We present the first known Cheeger and Buser inequality on Lipschitz probability density functions, and use this to show an $(\alpha = 1, \gamma = 3)$ spectral sweep cut on a L-Lipschitz probability density function has provably low $(\alpha = 1, \gamma = 2)$-sparsity. This work is the first spectral sweep cut algorithm on non-parametric probability densities with any guarantees on the cut quality.

Further, we show that existing spectral sweep cut methods (such as those implicit in spectral clustering) compute $(1, 1)$ or $(1, 2)$ spectral sweep cuts, neither of which has any sparsity guarantees. We prove that $(1, 2)$ spectral sweep cuts, which are implicitly used in traditional spectral clustering, can lead to undesirable partitions of simple 1-Lipschitz probability densities. Meanwhile, our work showed that using $(1, 3)$ spectral sweep cuts give provably good $(1, 2)$ sparse cuts.

For future directions, we conjecture that $\beta = \alpha + 1$ and $\gamma = \alpha + 2$ is the only settings of (α, β, γ) in which both Cheeger and Buser inequalities are provable. This would be a stronger theorem than we currently have for Lemma A.6 and Lemma A.7.

In the Buser inequality, we would like to iron out the exact dimensional dependence on the dimension, d (Theorem D.4). The authors believe that this dependence can be reduced to $\sqrt{d}$. It is an open question whether *any* dimension dependence is required. In particular, the latest

version of Buser's inequality for manifolds has no dimension dependence [192]. It is an open question how to generalize their techniques into the density setting, as the Bochner formula does not easily generalize to densities.

Another open question is whether multi-way Cheeger and Buser inequalities can be proven on densities, mirroring the work on graphs [183, 193, 194, 212]. This would allow our clustering algorithms to generalize into k-way clusterings. We additionally would like to know whether one can understand **balanced cuts** on proability densities for our new definitions of sparsity. Balanced cuts in this setting may have applications to machine learning.

Finally, we would like to know whether Buser and Cheeger inequalities may exist for L-Lipschitz probability densities supported on manifolds with bounded curvature. If true, this would fully generalize the work of Cheeger and Buser on manifolds, which may lead to deeper insight into manifold theory. Moreover, it could have foundational impact: a fundamental assumption underlying modern machine learning is that most data comes from probability density supported on a manifold, and a Cheeger and Buser inequality in this setting would give provable sparsity guarantees about spectral sweep cuts in this setting.

G Calculating Eigenvalues and Isoperimetry constants for Simple Examples

Recall from Section C the definitions of ρ_1 and ρ_2. This section is devoted to computing the eigenvalues and isoperimetric constants of these densities. We note that the eigenvalue and isoperimetry computation for ρ_1 is straightforward, so we omit it. The isoperimetry constant for ρ_2 is also straightforward, as the isoperimetric cut will be at $x = 0$. The only non-trivial computation is the (α, β)-eigenvalue for ρ_2.

G.1 Notation

We will write $a \gtrsim b$ if $a \geq cb$ for some absolute constant $0 < c < \infty$. Similarly define $a \lesssim b$. We will write $a \asymp b$ if both relations hold.

G.2 A Lipschitz weight

It is clear that

$$\Phi \asymp \epsilon^{\beta}.$$

Next, we apply the Hardy-Muckenhoupt inequality [223] to estimate λ_2 for ρ_2.

We upper bound $\mathcal{H}$ as:

$$\begin{aligned}\mathcal{H} &\leq \mathbb{R}(1)M(0)\\ &\asymp \int_0^1 \frac{1}{(x+\epsilon)^\gamma}\,dx\\ &\lesssim \begin{cases}1 & \text{if } \gamma<1\\ \ln(1/\epsilon) & \text{if } \gamma=1\\ O\epsilon^{1-\gamma} & \text{if } \gamma>1.\end{cases}\end{aligned}$$

By the Hardy-Muckenhoupt inequality, we can lower bound λ_2 with the inverse of an upper bound on $\mathcal{H}$. Thus, as claimed in Section C, we can lower bound λ_2 with $\epsilon^{\gamma-1}$ when $\gamma \geq 1$.

Thus, if we want a Buser-type inequality to hold, then (α, β, γ) needs to satisfy,

$$\begin{cases}1 \lesssim \lambda_2 \lesssim \max(\Phi, \Phi^2) \asymp \epsilon^\beta & \text{if } \gamma<1\\ \frac{1}{\ln(1/\epsilon)} \lesssim \lambda_2 \lesssim \max(\Phi, \Phi^2) \asymp \epsilon^\beta & \text{if } \gamma=1\\ \epsilon^{\gamma-1} \lesssim \lambda_2 \lesssim \max(\Phi, \Phi^2) \asymp \epsilon^\beta & \text{if } \gamma>1.\end{cases}$$

By letting ϵ go to zero, it is clear that $\gamma - 1 \geq \beta$, as desired.

H Cheeger and Buser for Density Functions does not easily follow from Graph or Manifold Cheeger and Buser

H.1 Comments on Graph Cheeger-Buser

The most natural method of proving distributional Cheeger-Buser inequality using the graph Cheeger-Buser inequality is to generate a vertex and edge weighted graph approximating the distribution, and write down graph Cheeger-Buser. Then, one would generate a sequence of graphs with an increasing number of vertices. Ideally, the graph Cheeger-Buser inequality on these graphs would converge to a Cheeger-Buser inequality on the underlying distribution. This discretization approach follows a standard paradigm of approximating distributions with graphs, present in numerical methods, finite element methods, and machine learning [130, 272, 280].

Such an approach cannot work (no matter how the eigenvalues and isoperimetric cuts are defined for distributions). The easiest way to see this is to attempt to execute this strategy for a simple uniform distribution in 1 dimension, on the interval $[0, 1]$. One would naively approximate this distribution with a line graph with n vertices, with edge weights w_n and vertex weights m_n. Then one would take n to go to infinity.

If one writes down the Cheeger and Buser inequalities for graphs in this example, we get:

$$\frac{w_n}{m_n n^2} \leq \Phi_{OPT} \leq \frac{w_n}{m_n n}$$

No matter what m_n and w_n are, the ratio between the upper and lower bound is n, which diverges. Thus, either the Cheeger inequality or the Buser inequality becomes meaningless:

either the lower bound goes to 0 or the upper bound goes to ∞, or both, depending on how w_n and m_n are set.

Thus, even for the simple case of a uniform distribution on $[0, 1]$ the natural strategy for deriving probability density Cheeger/Buser from graph Cheeger/Buser fails.

H.2 Comments on Manifold Cheeger-Buser

Distributional Buser does not easily follow from an application of the manifold Buser inequality. We recall that manifold Buser only applies for manifolds with bounded Ricci curvature. The natural way to parlay manifold Buser into distributional Buser on $\mathbb{R}^d$ is to change the underlying metric tensor on $\mathbb{R}^d$ to factor in the probability density function at that point. However, the authors are unaware of any method of doing this for which one can recover a meaningful Cheeger and Buser inequality. Moreover, it is unclear how to obtain any Ricci curvature bounds when we change the metric tensor.

Most modern approaches to proving Buser's inequality for manifolds rely on the Li-Yau inequality, which in turn depends on the Bochner identity for manifolds on bounded Ricci curvature [192]. The authors are unaware of a clean Bochner-like identity for distributions. Older techniques use Almgren's minimizing currents and/or Epsilon nets [61]. For the former, we do not know of any analog for distributions. For the latter, the corresponding Buser inequality has a 2^d multiplicative dependence, which is significantly worse than our d dependence.

I A weighted Cheeger inequality in one dimension

Theorem I.1. *Let $\Omega = (a, b)$ where $-\infty < a < b < \infty$. Let $\rho : (a, b) \to \mathbb{R}_{>0}$ be Lipschitz continuous. Then,*

$$\Phi(\Omega)^2 \leq 4 \left\| \rho^{\beta - \frac{\alpha+\gamma}{2}} \right\|_\infty^2 \lambda_2(\Omega).$$

In particular, when $(\alpha, \beta, \gamma) = (1, 2, 3)$, we have

$$\Phi(\Omega)^2 \leq 4\lambda_2(\Omega).$$

Proof. Let $w \in W^{1,2}(\Omega) \cap C^\infty(\Omega)$ be a strictly decreasing function with $\int_\Omega \rho^\alpha w \, dx = 0$. Let $v = w + a1$ where a is chosen such that $|\{v < 0\}|_\alpha = |\{v > 0\}|_\alpha$. Note that

$$\begin{aligned} R(w) &= \frac{\int_\Omega \rho^\gamma (w')^2 \, dx}{\int_\Omega \rho^\alpha w^2 \, dx} \\ &\geq \frac{\int_\Omega \rho^\gamma (w')^2 \, dx}{\int_\Omega \rho^\alpha w^2 \, dx + a^2 \, |\Omega|_\alpha} \\ &= R(v). \end{aligned}$$

Let $\widehat{x} \in (a, b)$ be the unique value such that $v(\widehat{x}) = 0$. Without loss of generality, the function $u = \max(v, 0)$ satisfies $R(u) \leq R(v)$ and has $u(a) = 1$.

Let $g = u^2$. Noting that $g' = 2uu'$ a.e., we can apply Cauchy-Schwarz to obtain

$$\begin{aligned}\int_a^{\widehat{x}} \rho^\beta |g'| \, dx &= 2\int_a^{\widehat{x}} \rho^\beta |u| |u'| \, dx \\ &\leq 2\sqrt{\int_a^{\widehat{x}} \rho^{2\beta-\alpha}(u')^2 \, dx}\sqrt{\int_a^{\widehat{x}} \rho^\alpha u^2 \, dx} \\ &\leq 2\left\|\rho^{\beta-\frac{\alpha+\gamma}{2}}\right\|_\infty \sqrt{\int_a^{\widehat{x}} \rho^\gamma (u')^2 \, dx}\sqrt{\int_a^{\widehat{x}} \rho^\alpha u^2 \, dx}.\end{aligned}$$

Then, dividing by $\int_a^{\widehat{x}} \rho^\alpha g \, dx$, we have

$$\frac{\int_a^{\widehat{x}} \rho^\beta |g'| \, dx}{\int_a^{\widehat{x}} \rho^\alpha g \, dx} \leq 2\left\|\rho^{\beta-\frac{\alpha+\gamma}{2}}\right\|_\infty \sqrt{R(w)}.$$

By change of variables,

$$\int_a^{\widehat{x}} \rho^\beta |g'| \, dx = \int_0^1 \rho^\beta(g^{-1}(t)) \, dt.$$

Writing $g(x) = \int_0^{g(x)} 1 \, dt$ and applying Tonelli's theorem, we rewrite the denominator

$$\int_a^{\widehat{x}} \rho^\alpha g \, dx = \int_0^1 \left|(a, g^{-1}(t))\right|_\alpha \, dt.$$

Thus, by averaging, there exists some t^* such that,

$$\Phi(\Omega) \leq \frac{\rho^\beta(t^*)}{|(a,t^*)|_\alpha} \leq \frac{\int_a^{\widehat{x}} \rho^\beta |g'| \, dx}{\int_a^{\widehat{x}} \rho^\alpha g \, dx} \leq 2\left\|\rho^{\beta-\frac{\alpha+\gamma}{2}}\right\|_\infty \sqrt{R(w)}.$$

g □

Theorem I.2. *Let $\Omega = (a,b)$ where $-\infty < a < b < \infty$. Let $\rho : (a,b) \to \mathbb{R}_{>0}$ be Lipschitz continuous with Lipschitz constant L. Then,*

$$\lambda_2(\Omega) \leq 8 \cdot (3/2)^{\gamma/\alpha} \left\|\rho^{\gamma-1-\beta}\right\|_\infty \max\left(4\left\|\rho^{\alpha+1-\beta}\right\|_\infty \Phi^2(\Omega), \frac{\alpha}{\ln(3/2)} L\Phi(\Omega)\right).$$

In particular, when $(\alpha,\beta,\gamma) = (1,2,3)$, we have

$$\lambda_2(\Omega) \leq O\left(\max\left(\Phi^2(\Omega), L\Phi(\Omega)\right)\right).$$

Proof. Let $\widehat{x} \in (a,b)$. We will show that there exists a $u \in W^{1,2}(\Omega)$ with small Rayleigh quotient compared to $\Phi(\widehat{x})$. Let $A = (a,\widehat{x})$ and $B = (\widehat{x},b)$. Without loss of generality $|A|_\alpha \leq |B|_\alpha$ and hence $\Phi(\widehat{x}) = \frac{\rho^\beta(\widehat{x})}{|A|_\alpha}$. For notational convenience, we will write $\Phi = \Phi(\widehat{x})$ in this proof.

Let

$$u(x) = \begin{cases} |A|_\alpha & a \le x \le \widehat{x} \\ -|B|_\alpha & \widehat{x} < x \le b. \end{cases}$$

Let $\delta = \theta\rho(\widehat{x})$ where $\theta > 0$ will be picked later. Define the continuous function

$$u_\delta(x) = \begin{cases} |A|_\alpha & a \le x \le x_1 \\ \text{linear with slope } \frac{-|\Omega|_\alpha}{\delta} & x_1 \le x \le x_2 \\ -|B|_\alpha & x_2 \le x \le b \end{cases}$$

where $a \le x_1 < \widehat{x} < x_2 \le b$ are picked such that $\int_a^b \rho^\alpha u_\delta \, dx = 0$. Note $x_2 - x_1 \le \delta$.

We bound the numerator in $R(u_\delta)$ using the mean value theorem.

$$\begin{aligned} \int_a^b \rho^\gamma (u_\delta')^2 \, dx &= \frac{|\Omega|_\alpha^2}{\delta^2} \int_{x_1}^{x_2} \rho^\gamma \, dx \\ &\le \frac{|\Omega|_\alpha^2}{\delta} \rho^\gamma(\widetilde{x}) \qquad \text{for some } \widetilde{x} \in [x_1, x_2] \\ &\le |\Omega|_\alpha^2 \, \rho^{\gamma-1}(\widehat{x})(1 + L\theta)^\gamma/\theta \end{aligned}$$

In the third line we used the Lipschitz estimate $\rho(\widetilde{x}) \le \rho(\widehat{x})(1 + L\theta)$. We lower bound the denominator in $R(u_\delta)$ using the mean value theorem and the same Lipschitz estimate. We will also recall that $\Phi = \rho^\beta(\widehat{x})/\,|A|_\alpha$.

$$\begin{aligned} \int_a^b \rho^\alpha u_\delta^2 \, dx &\ge \int_a^b \rho^\alpha u^2 \, dx - \int_{x_1}^{x_2} \rho^\alpha u^2 \, dx \\ &\ge |A|_\alpha \, |B|_\alpha \, |\Omega|_\alpha - \delta \rho^\alpha(\widetilde{x}) \, |B|_\alpha^2 \qquad \text{for some } \widetilde{x} \in [x_1, x_2] \\ &\ge |A|_\alpha \, |B|_\alpha \, |\Omega|_\alpha - \rho^{\alpha+1}(\widehat{x}) \, |B|_\alpha^2 \, (1 + L\theta)^\alpha \theta \\ &\ge |\Omega|_\alpha^2 \left(|A|_\alpha / 2 - \rho^{\alpha+1}(\widehat{x})(1 + L\theta)^\alpha \theta \right) \\ &\ge |\Omega|_\alpha^2 \, |A|_\alpha \left(1/2 - \left\| \rho^{\alpha+1-\beta} \right\|_\infty \Phi(1 + L\theta)^\alpha \theta \right) \end{aligned}$$

The parameter θ will be chosen such that the estimate of the denominator is positive. We combine the two bounds above.

$$\begin{aligned} R(u_\delta) &\le \frac{|\Omega|_\alpha^2 \, \rho^{\gamma-1}(\widehat{x})(1 + L\theta)^\gamma/\theta}{|\Omega|_\alpha^2 \, |A|_\alpha \, (1/2 - \|\rho^{\alpha+1-\beta}\|_\infty \, \Phi(1 + L\theta)^\alpha \theta)} \\ &= \frac{\rho^{\gamma-1-\beta}(\widehat{x})\Phi(1 + L\theta)^\gamma/\theta}{1/2 - \|\rho^{\alpha+1-\beta}\|_\infty \, \Phi(1 + L\theta)^\alpha \theta} \\ &\le \frac{\left\|\rho^{\gamma-1-\beta}\right\|_\infty \Phi(1 + L\theta)^\gamma/\theta}{1/2 - \|\rho^{\alpha+1-\beta}\|_\infty \, \Phi(1 + L\theta)^\alpha \theta}. \end{aligned}$$

We make the following choice of $\theta > 0$,

$$\theta = \min\left(\frac{1}{4\Phi \, \|\rho^{\alpha+1-\beta}\|_\infty}, \frac{\ln(3/2)}{\alpha L} \right).$$

Then, $(1+L\theta) \leq (3/2)^{1/\alpha}$ and $\Phi\theta \leq \frac{1}{4\left\|\rho^{\alpha+1-\beta}\right\|_\infty}$. Thus,

$$\begin{aligned}
\lambda_2 &\leq R(u_\delta) \\
&\leq 8 \cdot (3/2)^{\gamma/\alpha} \left\|\rho^{\gamma-1-\beta}\right\|_\infty \frac{\Phi}{\theta} \\
&= 8 \cdot (3/2)^{\gamma/\alpha} \left\|\rho^{\gamma-1-\beta}\right\|_\infty \max\left(4\left\|\rho^{\alpha+1-\beta}\right\|_\infty \Phi^2, \frac{\alpha}{\ln(3/2)} L\Phi\right).
\end{aligned}$$

Finally, picking $\widehat{x}$ such that $\Phi(\widehat{x}) \to \Phi(\Omega)$ completes the proof. □

Remark I.2.1. *Recall the example presented in Section G.2, i.e.* $\Omega = (-1,1)$, $\rho = |x| + \epsilon$. *For the choice* $(\alpha,\beta,\gamma) = (1,1,1)$, *it was shown that* $\frac{\lambda_2(\Omega)}{\Phi(\Omega)^p}$ *diverges to infinity as* $\epsilon \to 0$ *for any* $p > 0$. *This does not contradict our Theorem I.2, which only asserts that*

$$\lambda_2(\Omega) \lesssim \frac{1}{\epsilon} \max\left(\Phi^2(\Omega), \Phi(\Omega)\right).$$

Chapter 5

Geometric Spectral Algorithms and Hardness, with Machine Learning applications

For a function $\mathsf{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$, and a set $P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ of n points, the K graph G_P of P is the complete graph on n nodes where the weight between nodes i and j is given by $\mathsf{K}(x_i, x_j)$. In this chapter, we study when efficient spectral graph theory is possible on these graphs. We investigate whether or not it is possible to solve the following problems in $n^{1+o(1)}$ time for a K-graph G_P when $d < n^{o(1)}$:

- Multiply a given vector by the adjacency matrix or Laplacian matrix of G_P
- Find a spectral sparsifier of G_P
- Solve a Laplacian system in G_P's Laplacian matrix

For each of these problems, we consider all functions of the form $\mathsf{K}(u, v) = f(\|u - v\|_2^2)$ for a function $f : \mathbb{R} \to \mathbb{R}$. We provide algorithms and comparable hardness results for many such K, including the Gaussian kernel, Neural tangent kernels, and more. For example, in dimension $d = \Omega(\log n)$, we show that there is a parameter associated with the function f for which low parameter values imply $n^{1+o(1)}$ time algorithms for all three of these problems and high parameter values imply the nonexistence of subquadratic time algorithms assuming Strong Exponential Time Hypothesis (SETH), given natural assumptions on f.

As part of our results, we also show that the exponential dependence on the dimension d in the celebrated fast multipole method of Greengard and Rokhlin cannot be improved, assuming SETH, for a broad class of functions f. To the best of our knowledge, this is the first formal limitation proven about fast multipole methods.

Contents

1 Introduction

Linear algebra has a myriad of applications throughout computer science and physics. Consider the following seemingly unrelated tasks:

1. n**-body simulation (one step)**: Given n bodies X located at points in $\mathbb{R}^d$, compute the gravitational force on each body induced by the other bodies.
2. **Spectral clustering**: Given n points X in $\mathbb{R}^d$, partition X by building a graph G on the points in X, computing the top k eigenvectors of the Laplacian matrix L_G of G for some $k \geq 1$ to embed X into $\mathbb{R}^k$, and run k-means on the resulting points.
3. **Semi-supervised learning**: Given n points X in $\mathbb{R}^d$ and a function $g : X \to \mathbb{R}$ whose values on some of X are known, extend g to the rest of X.

Each of these tasks has seen much work throughout numerical analysis, theoretical computer science, and machine learning. The first task is a celebrated application of the fast multipole method of Greengard and Rokhlin [140, 141, 142], voted one of the top ten algorithms of the twentieth century by the editors of *Computing in Science and Engineering* [110]. The second task is *spectral clustering* [207, 231], a popular algorithm for clustering data. The third task is to label a full set of data given only a small set of partial labels [69, 310, 311], which has seen increasing use in machine learning. One notable method for performing semi-supervised learning is the graph-based Laplacian regularizer method [42, 209, 309, 310].

Popular techniques for each of these problems benefit from primitives in spectral graph theory on a special class of dense graphs called *geometric graphs*. For a function $\mathsf{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and a set of points $X \subseteq \mathbb{R}^d$, the K-*graph on* X is a graph with vertex set X and edges with weight $\mathsf{K}(u, v)$ for each pair $u, v \in X$. Adjacency matrix-vector multiplication, spectral sparsification, and Laplacian system solving in geometric graphs are directly relevant to each of the above problems, respectively:

1. n**-body simulation (one step)**: For each $i \in \{1, 2, \ldots, d\}$, make a weighted graph G_i on the points in X, in which the weight of the edge between the points $u, v \in X$ in G_i is $\mathsf{K}_i(u, v) := (\frac{G_{\text{grav}} \cdot m_u \cdot m_v}{\|u-v\|_2^2})(\frac{v_i - u_i}{\|u-v\|_2})$, where G_{grav} is the gravitational constant and m_x is the mass of the point $x \in X$. Let A_i denote the weighted adjacency matrix of G_i. Then $A_i\mathbf{1}$ is the vector of ith coordinates of force vectors. In particular, gravitational force can be computed by doing $O(d)$ adjacency matrix-vector multiplications, where each adjacency matrix is that of the K_i-graph on X for some i.
2. **Spectral clustering**: Make a K graph G on X. In applications, $\mathsf{K}(u, v) = f(\|u - v\|_2^2)$, where f is often chosen to be $f(z) = e^{-z}$ [231, 290]. Instead of directly running a spectral clustering algorithm on L_G, one popular method is to construct a sparse matrix M approximating L_G and run spectral clustering on M instead [66, 83, 186]. Standard sparsification methods in the literature are heuristical, and include the widely used Nystrom method which uniformly samples rows and columns from the original matrix [87].
   If H is a spectral sparsifier of G, it has been suggested that spectral clustering with the top k eigenvectors of L_H performs just as well in practice as spectral clustering with the top k eigenvectors of L_G [66]. One justification is that since H is a spectral sparsifier of G, the eigenvalues of L_H are at most a constant factor larger than those of L_G, so cuts with similar conductance guarantees are produced. Moreover, spectral clustering using sparse

matrices like L_H is known to be faster than spectral clustering on dense matrices like L_G [66, 87, 186].

3. **Semi-supervised learning**: An important subroutine in semi-supervised learning is completion based on ℓ_2-minimization [209, 309, 311]. Specifically, given values g_v for $v \in Y$, where Y is a subset of X, find the vector $g \in \mathbb{R}^n$ (variable over $X \setminus Y$) that minimizes $\sum_{u,v \in X, u \neq v} \mathsf{K}(u,v)(g_u - g_v)^2$. The vector g can be found by solving a Laplacian system on the K-graph for X.

In the first, second, and third tasks above, a small number of calls to matrix-vector multiplication, spectral sparsification, and Laplacian system solving, respectively, were made on geometric graphs. One could solve these problems by first explicitly writing down the graph G and then using near-linear time algorithms [93, 269] to multiply, sparsify, and solve systems. However, this requires a minimum of $\Omega(n^2)$ time, as G is a dense graph.

In this chapter, we initiate a theoretical study of the geometric graphs for which efficient spectral graph theory is possible. In particular, we attempt to determine for which (a) functions K and (b) dimensions d there is a much faster, $n^{1+o(1)}$-time algorithm for each of (c) multiplication, sparsification, and Laplacian solving. Before describing our results, we elaborate on the choices of (a), (b), and (c) that we consider in this work.

We start by discussing the functions K that we consider (part (a)). Our results primarily focus on the class of functions of the form $\mathsf{K}(u,v) = f(\|u-v\|_2^2)$ for a function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ for $u, v \in \mathbb{R}^d$. Study of these functions dates back at least eighty years, to the early work of Bochner, Schoenberg, and John Von Neumann on metric embeddings into Hilbert Spaces [50, 229, 253]. These choices of K are ubiquitous in applications, like the three described above, since they naturally capture many kernel functions K from statistics and machine learning, including the Gaussian kernel ($e^{-\|u-v\|_2^2}$), the exponential kernel ($e^{-\|u-v\|_2}$), the power kernel ($\|u-v\|_2^q$) for both positive and negative q, the logarithmic kernel ($\log(\|u-v\|_2^q + c)$), and more [53, 267, 309]. See Section 1.5 below for even more popular examples. In computational geometry, many transformations of distance functions are also captured by such functions K, notably in the case when $\mathsf{K}(u,v) = \|u-v\|_2^q$ [7, 9, 90, 210].

We would also like to emphasize that many kernel functions which do not at first appear to be of the form $f(\|u-v\|_2^2)$ can be rearranged appropriately to be of this form. For instance, in Section 10 below we show that the recently popular Neural Tangent Kernel is of this form, so our results apply to it as well. That said, to emphasize that our results are very general, we will mention later how they also apply to some functions of the form $\mathsf{K}(u,v) = f(\langle u, v \rangle)$, including $\mathsf{K}(u,v) = |\langle u, v \rangle|$.

Next, we briefly elaborate on the problems that we consider (part (c)). For more details, see Section 3. The points in X are assumed to be real numbers stated with $\mathrm{polylog}(n)$ bits of precision. Our algorithms and hardness results pertain to algorithms that are allowed some degree of approximation. For an error parameter $\varepsilon > 0$, our multiplication and Laplacian system solving algorithms produce solutions with ε-additive error, and our sparsification algorithms produce a graph H for which the Laplacian quadratic form $(1 \pm \varepsilon)$-approximates that of G.

Matrix-vector multiplication, spectral sparsification, and Laplacian system solving are very natural linear algebraic problems in this setting, and have many applications beyond the three we have focused on (n-body simulation, spectral clustering, and semi-supervised learning). See

Section 1.5 below where we expand on more applications.

Finally, we discuss dependencies on the dimension d and the accuracy ε for which $n^{1+o(1)}$ algorithms are possible (part (b)). Define α, a measure of the 'diameter' of the point set and f, as

$$\alpha := \frac{\max_{u,v \in X} f(\|u-v\|_2^2)}{\min_{u,v \in X} f(\|u-v\|_2^2)} + \frac{\max_{u,v \in X} \|u-v\|_2^2}{\min_{u,v \in X} \|u-v\|_2^2}.$$

It is helpful to have the following two questions in mind when reading our results:

- (High-dimensional algorithms, e.g. $d = \Theta(\log n)$) Is there an algorithm which runs in time $\mathrm{poly}(d, \log(n\alpha/\varepsilon))n^{1+o(1)}$ for multiplication and Laplacian solving? Is there an algorithm which runs in time $\mathrm{poly}(d, \log(n\alpha))n^{1+o(1)}$ for sparsification when $\varepsilon = 1/2$?
- (Low-dimensional algorithms, e.g. $d = o(\log n)$) Is there an algorithm which runs in time $(\log(n\alpha/\varepsilon))^{O(d)}n^{1+o(1)}$ for multiplication and Laplacian solving? Is there a sparsification algorithm which runs in time $(\log(n\alpha))^{O(d)}n^{1+o(1)}$ when $\varepsilon = 1/2$?

We will see that there are many important functions K for which there are such efficient low-dimensional algorithms, but no such efficient high-dimensional algorithms. In other words, these functions K suffer from the classic 'curse of dimensionality.' At the same time, other functions K will allow for efficient low-dimensional and high-dimensional algorithms, while others won't allow for either.

We now state our results. We will give very general classifications of functions K for which our results hold, but afterwards in Section 1.4 we summarize the results for a few particular functions K of interest. The main goal of our results is as follows:

Goal: For each problem of interest (part (c)) and dimension d (part (b)), find a natural parameter $p_f > 0$ associated with the function f for which the following dichotomy holds:

- If p_f is high, then the problem cannot be solved in subquadratic time assuming SETH on points in dimension d.
- If p_f is low, then the problem of interest can be solved in almost-linear time ($n^{1+o(1)}$ time) on points in dimension d.

As we will see shortly, the two parameters p_f which will characterize the difficulties of our problems of interest in most settings are the *approximate degree* of f, and a parameter related to how *multiplicatively Lipschitz* f is. We define both of these in the next section.

1.1 High-dimensional results

We begin in this subsection by stating our results about which functions have $\mathrm{poly}(d, \log(\alpha), \log(1/\varepsilon)) \cdot n^{1+o(1)}$-time algorithms for multiplication and Laplacian solving and $\mathrm{poly}(d, \log(\alpha), 1/\varepsilon) \cdot n^{1+o(1)}$-time algorithms for sparsification. When reading these results, it is helpful to think of $d = \Theta(\log n)$, $\alpha = 2^{\mathrm{polylog}(n)}$, $\varepsilon = 1/2^{\mathrm{polylog}(n)}$ for multiplication and Laplacian solving, and $\varepsilon = 1/2$ for sparsification. With these parameter settings, $\mathrm{poly}(d)n^{1+o(1)}$ is almost-linear time, while $2^{O(d)}n^{1+o(1)}$ time is not. For results about algorithms with runtimes that are exponential in d, see Section 2.

Multiplication

In high dimensions, we give a full classification of when the matrix-vector multiplication problems are easy for kernels of the form $\mathsf{K}(u,v) = f(\|u-v\|_2^2)$ for some function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is analytic on an interval. We show that the problem can be efficiently solved only when K is very well-approximated by a simple polynomial kernel. That is, we let p_f denote the minimum degree of a polynomial that ε-additively-approximates the function f.

Theorem 1.1 (Informal version of Theorem 5.8 and Corollary 5.7.2)**.** *For any function $f : \mathbb{R}_+ \to \mathbb{R}_+$ which is analytic on an interval $(0,\delta)$ for any $\delta > 0$, and any $0 < \varepsilon < 2^{-\mathrm{polylog}(n)}$, consider the following problem: given as input $x_1, \ldots, x_n \in \mathbb{R}^d$ with $d = \Theta(\log n)$ which define a K graph G via $\mathsf{K}(x_i, x_j) = f(\|x_i - x_j\|_2^2)$, and a vector $y \in \{0,1\}^n$, compute an ε-additive-approximation to $L_G \cdot y$.*

- *If f can be ε-additively-approximated by a polynomial of degree at most $o(\log n)$, then the problem can be solved in $n^{1+o(1)}$ time.*
- *Otherwise, assuming* SETH*, the problem requires time $n^{2-o(1)}$.*

The same holds for L_G, the Laplacian matrix of G, replaced by A_G, the adjacency matrix of G.

While Theorem 1.1 yields a parameter p_f that characterizes hardness of multiplication in high dimensions, it is somewhat cumbersome to use, as it can be challenging to show that a function is far from a polynomial. We also show Theorem 5.9, which shows that if f has a single point with large $\Theta(\log n)$-th derivative, then the problem requires time $n^{2-o(1)}$ assuming SETH. The Strong Exponential Time Hypothesis (SETH) is a common assumption in fine-grained complexity regarding the difficulty of solving the Boolean satisfiability problem; see section 3.6 for more details. Theorem 1.1 informally says that assuming SETH, the curse of dimensionality is inherent in performing adjacency matrix-vector multiplication. In particular, we directly apply this result to the n-body problem discussed at the beginning:

Corollary 1.1.1. *Assuming* SETH*, in dimension $d = \Theta(\log n)$ one step of the n-body problem requires time $n^{2-o(1)}$.*

The fast multipole method of Greengard and Rokhlin [140, 142] solves one step of this n-body problem in time $(\log(n/\varepsilon))^{O(d)} n^{1+o(1)}$. Our Corollary 1.1.1 shows that assuming SETH, such an exponential dependence on d is required and cannot be improved. To the best of our knowledge, this is the first time such a formal limitation on fast multipole methods has been proved. This hardness result also applies to fast multipole methods for other popular kernels, like the Gaussian kernel $\mathsf{K}(u,v) = \exp(-\|u-v\|_2^2)$, as well.

Sparsification

We next show that sparsification can be performed in almost-linear time in high dimensions for kernels that are "multiplicatively Lipschitz" functions of the ℓ_2-distance. We say $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is (C, L)-*multiplicatively Lipschitz* for $C > 1, L > 1$ if for all $x \in \mathbb{R}_{\geq 0}$ and all $\rho \in (1/C, C)$,

$$C^{-L} f(x) \leq f(\rho x) \leq C^L f(x).$$

Here are some popular functions that are helpful to think about in the context of our results:

1. $f(z) = z^L$ for any positive or negative constant L. This function is $(C, |L|)$-multiplicatively Lipschitz for any $C > 1$.

2. $f(z) = e^{-z}$. This function is not (C, L)-multiplicatively Lipschitz for any $L > 1$ and $C > 1$. We call this the *exponential function*.

3. The piecewise function $f(z) = e^{-z}$ for $z \leq L$ and $f(z) = e^{-L}$ for $z > L$. This function is $(C, O(L))$-multiplicatively Lipschitz for any $C > 1$. We call this a *piecewise exponential function*.

4. The piecewise function $f(z) = 1$ for $z \leq k$ and $f(z) = 0$ for $z > k$, where $k \in \mathbb{R}_{\geq 0}$. This function is not (C, L)-multiplicatively Lipschitz for any $C > 1$ or $L > 1$. This is a *threshold function*.

We show that multiplicatively Lipschitz functions can be sparsified in $n^{1+o(1)}\mathrm{poly}(d)$ time:

Theorem 1.2 (Informal version of Theorem 6.3)**.** *For any function f such that f is $(2, L)$-multiplicatively Lipschitz, building a $(1 \pm \varepsilon)$-spectral sparsifier of the K-graph on n points in $\mathbb{R}^d$ where $\mathsf{K}(u, v) = f(\|u - v\|_2^2)$, with $O(n \log n/\varepsilon^2)$ edges, can be done in time*

$$O(nd\sqrt{L \log n}) + n \log n \cdot 2^{O(\sqrt{L \log n})} \cdot (\log \alpha)/\varepsilon^2.$$

This Theorem applies even when $d = \Omega(\log n)$. When L is constant, the running time simplifies to $O(nd\sqrt{\log n} + n^{1+o(1)} \log \alpha/\varepsilon^2)$. This covers the case when $f(x)$ is any rational function with non-negative coefficients, like $f(z) = z^L$ or $f(z) = z^{-L}$.

It may seem more natural to instead define L-multiplicatively Lipschitz functions, without the parameter C, as functions with $\rho^{-L} f(x) \leq f(\rho x) \leq \rho^L f(x)$ for all ρ and x. Indeed, an L-multiplicatively Lipschitz function is also (C, L)-multiplicative Lipschitz for any $C > 1$, so our results show that efficient sparsification is possible for such functions. However, the parameter C is necessary to characterize when efficient sparsification is possible. Indeed, as in Theorem 1.2 above, it is sufficient for f to be (C, L)-multiplicative Lipschitz for a C that is bounded away from 1. To complement this result, we also show a lower bound for sparsification for any function f which is not (C, L)-multiplicatively Lipschitz for any L and sufficiently large C:

Theorem 1.3 (Informal version of Theorem 8.3)**.** *Consider an $L > 1$. There is some sufficiently large value $C_L > 1$ depending on L such that for any decreasing function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is not (C_L, L)-multiplicatively Lipschitz, no $O(n2^{L^{.48}})$-time algorithm for constructing an $O(1)$-spectral sparsifier of the K-graph of a set of n points in $O(\log n)$ dimensions exists assuming SETH, where $\mathsf{K}(x, y) = f(\|x - y\|_2^2)$.*

For example, when $L = \Theta(\log^{2+\delta} n)$ for some constant $\delta > 0$, Theorem 1.3 shows that there is a C for which, whenever f is not (C, L)-multiplicatively Lipschitz, the sparsification problem cannot be solved in time $n^{1+o(1)}$ assuming SETH.

Bounding C in terms of L above is important. For example, if C is small enough that $C^L = 2$, then f could be close to constant. Such K-graphs are easy to sparsify by uniformly sampling edges, so one cannot hope to show hardness for such functions.

Theorem 1.3 shows that geometric graphs for threshold functions, the exponential function, and the Gaussian kernel do not have efficient sparsification algorithms. Furthermore, this hardness result essentially completes the story of which *decreasing* functions can be sparsified in high dimensions, modulo a gap of $L^{.48}$ versus L in the exponent. The tractability landscape is likely much more complicated for non-decreasing functions. That said, many of the kernels used in practice, like the Gaussian kernel, are decreasing functions of distance, so our dichotomy applies to them.

We also show that our techniques for sparsification extend beyond kernels that are functions of ℓ_2 norms; specifically $\mathsf{K}(u,v) = |\langle u, v\rangle|$:

Lemma 1.4 (Informal version of Lemma 7.1)**.** *The $\mathsf{K}(u,v) = |\langle u, v\rangle|$-graph on n points in $\mathbb{R}^d$ can be ε-approximately sparsified in $n^{1+o(1)}\mathrm{poly}(d)/\varepsilon^2$ time.*

Laplacian solving

Laplacian system solving has a similar tractability landscape to that of adjacency matrix multiplication. We prove the following algorithmic result for solving Laplacian systems:

Theorem 1.5 (Informal version of Corollary 5.7.1 and Proposition 3.8)**.** *There is an algorithm that takes $n^{1+o(1)}\mathrm{poly}(d, \log(n\alpha/\varepsilon))$ time to ε-approximately solve Laplacian systems on n-vertex K-graphs, where $\mathsf{K}(u,v) = f(\|u-v\|_2^2)$ for some (nonnegative) polynomial f.*[1]

We show that this theorem is nearly tight via two hardness results. The first applies to multiplicatively Lipschitz kernels, while the second applies to kernels that are not multiplicatively Lipschitz. The second hardness result only works for kernels that are decreasing functions of ℓ_2 distance. We now state our first hardness result:

Corollary 1.5.1. *Consider a function f that is $(2, o(\log n))$-multiplicatively Lipschitz for which f cannot be $(\varepsilon = 2^{-\mathrm{poly}(\log n)})$-approximated by a polynomial of degree at most $o(\log n)$. Then, assuming SETH, there is no $n^{1+o(1)}\mathrm{poly}(d, \log(\alpha n/\varepsilon))$-time algorithm for ε-approximately solving Laplacian systems in the K-graph on n points, where $\mathsf{K}(u,v) = f(\|u-v\|_2^2)$.*

In Section 4, we will see, using an iterative refinement approach, that if a K graph can be efficiently sparsified, then there is an efficient Laplacian multiplier for K graphs if and only if there is an efficient Laplacian system solver for K graphs. Corollary 1.5.1 then follows using this connection: it describes functions which we have shown have efficient sparsifiers but not efficient multipliers.

Corollary 1.5.1, which is the first of our two hardness results in this setting, applies to slowly-growing functions that do not have low-degree polynomial approximations, like $f(z) = 1/(1+z)$. Next, we state our second hardness result:

Theorem 1.6 (Informal version of Theorem 8.7)**.** *Consider an $L > 1$. There is some sufficiently large value $C_L > 1$ depending on L such that for any decreasing function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is not (C_L, L)-multiplicatively Lipschitz, no $O(n2^{L^{.48}} \log \alpha)$-time algorithm exists for solving Laplacian systems $2^{-\mathrm{poly}(\log n)}$ approximately in the K-graph of a set of n points in $O(\log n)$ dimensions assuming SETH, where $\mathsf{K}(u,v) = f(\|u-v\|_2^2)$.*

This yields a quadratic time hardness result when $L = \Omega(\log^2 n)$. By comparison, the first hardness result, Corollary 1.5.1, only applied for $L = o(\log n)$. In particular, this shows that for non-Lipschitz functions like the Gaussian kernel, the problem of solving Laplacian systems and, in particular, doing semi-supervised learning, cannot be done in almost-linear time assuming SETH.

[1] f is a nonnegative function if $f(x) \geq 0$ for all $x \geq 0$.

1.2 Our Techniques

Multiplication

Our goal in matrix-vector multiplication is, given points $P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ and a vector $y \in \mathbb{R}^n$, to compute a $(1 \pm \varepsilon)$-approximation to the vector $L_G \cdot y$ where L_G is the Laplacian matrix of the K graph on P, for $\varepsilon = n^{-\Omega(1)}$ (see Definition 4.1for the precise error guarantees on ε). We call this the K Laplacian Evaluation (KLapE) problem. A related problem, in which the Laplacian matrix L_G is replaced by the adjacency matrix A_G, is the K Adjacency Evaluation (KAdjE) problem.

We begin by showing a simple, generic equivalence between KLapE and KAdjE for any K: an algorithm for either one can be used as a black box to design an algorithm for the other with only negligible blowups to the running time and error. It thus suffices to design algorithms and prove lower bounds for KAdjE.

Algorithmic Techniques We use two primary algorithmic tools: the Fast Multipole Method (FMM), and a 'kernel method' for approximating A_G by a low-rank matrix.

FMM is an algorithmic technique for computing aggregate interactions between n bodies which has applications in many different areas of science. Indeed, when the interactions between bodies is described by our function K, then the problem solved by FMM coincides with our KAdjE problem.

Most past work on FMM either considers the low-dimensional case, in which d is a small constant, or else the low-error case, in which ε is a constant. Thus, much of the literature does not consider the simultaneous running time dependence of FMM on ε and d. In order to solve KAdjE, we need to consider the high-dimensional, high-error case. We thus give a clean mathematical overview and detailed analysis of the running time of FMM in Section 9, following the seminal work of Greengard and Strain [144], which may be of independent interest.

As discussed in section 1.1 above, the running time of FMM depends exponentially on d, and so it is most useful in the low-dimensional setting. Our main algorithmic tool in high dimensions is a low-rank approximation technique: we show that when $f(x)$ can be approximated by a sufficiently low-degree polynomial (e.g. any degree $o(\log n)$ suffices in dimension $\Theta(\log n)$), then we can quickly find a low-rank approximation of the adjacency matrix A_G, and use this to efficiently multiply by a vector. Although this seems fairly simple, in Theorem 1.1 we show it is optimal: when such a low-rank approximation is not possible in high dimensions, then SETH implies that $n^{2-o(1)}$ time is required for KAdjE.

The simplest way to show that $f(x)$ can be approximated by a low-degree polynomial is by truncating its Taylor series. In fact, the FMM *also* requires that a truncation of the Taylor series of f gives a good approximation to f. By comparison, the FMM puts more lenient restrictions on what degree the series must be truncated to in low dimensions, but in exchange adds other constraints on f, including that f must be monotone. See Section 9.3 and Corollary 5.7.2 for more details.

Lower Bound Techniques We now sketch the proof of Theorem 1.1, our lower bound for KAdjE for many functions K in high enough dimensions (typically $d = \Omega(\log n)$), assuming

SETH. Although SETH is a hardness hypothesis about the Boolean satisfiability problem, a number of recent results [17, 81, 244, 245] have showed that it implies hardness for a variety of *nearest neighbor search* problems. Our lower bound approach is hence to show that KAdjE is useful for solving nearest neighbor search problems.

The high-level idea is as follows. Suppose we are given as input points $X = \{x_1, \ldots, x_n\} \subset \{0,1\}^d$, and our goal is to find the closest pair of them. For each $\ell \in \{1, 2, \ldots, d\}$, let c_ℓ denote the number of pairs of distinct points $x_i, x_j \in X$ with distance $\|x_i - x_j\|_2^2 = \ell$. Using an algorithm for KAdjE for our function $\mathsf{K}(x, y) = f(\|x - y\|_2^2)$, we can estimate

$$1^\top A_G 1 = \sum_{i \neq j} \mathsf{K}(x_i, x_j) = \sum_{\ell=1}^{d} c_\ell \cdot f(\ell).$$

Similarly, for any nonnegative reals $a, b \geq 0$, we can take an appropriate affine transformation of X so that an algorithm for KAdjE can estimate

$$\sum_{\ell=1}^{d} c_\ell \cdot f(a \cdot \ell + b). \tag{5.1}$$

Suppose we pick real values $a_1, \ldots, a_d, b_1, \ldots, b_d \geq 0$ and define the $d \times d$ matrix M by $M[i, \ell] = f(a_i \cdot \ell + b_i)$. By estimating the sum (5.1) for each pair (a_i, b_i), we get an estimate of the matrix-vector product Mc, where $c \in \mathbb{R}^d$ is the vector of the c_ℓ values. We show that if M has a large enough determinant relative to the magnitudes of its entries, then one can recover an estimate of c itself from this, and hence solve the nearest neighbor problem.

The main tool we need for this approach is a way to pick $a_1, \ldots, a_d, b_1, \ldots, b_d$ for a function f which cannot be approximated by a low degree polynomial so that M has large determinant. We do this by decomposing $\det(M)$ in terms of the derivatives of f using the Cauchy-Binet formula, and then noting that if f cannot be approximated by a polynomial, then many of the contributions in this sum must be large. The specifics of this construction are quite technical; see section 5.4 for the details.

Comparison with Previous Lower Bound Techniques Prior work (e.g. [70], [36], [37]) has shown SETH-based fine-grained complexity results for matrix-related computations. For instance, [36] showed hardness results for exact algorithms for many machine-learning related tasks, like kernel PCA and gradient computation in training neural networks, while [70] and [37] showed hardness results for kernel density estimation. In all of this work, the authors are only able to show hardness for a limited set of kernels. For example, [36] shows hardness for kernel PCA only for Gaussian kernels. These limitations arise from the technique used. To show hardness, [36] exploits the fact that the Gaussian kernel decays rapidly to obtain a gap between the completeness and soundness cases in approximate nearest neighbors, just as we do for functions f like $f(x) = (1/x)^{\Omega(\log n)}$. The hardness results of [70] and [37] employ a similar idea.

As discussed in *Lower Bound Techniques*, we circumvent these limitations by showing that applying the multiplication algorithm for one kernel a small number of times and linearly combining the results is enough to solve Hamming closest pair. This idea is enough to give a nearly

tight characterization of the analytic kernels for which subquadratic-time multiplication is possible in dimension $d = \Theta(\log n)$. As a result, by combining with reductions similar to those from past work, our lower bound also applies to a variety of similar problems, including kernel PCA, for a much broader set of kernels than previously known; see Section 5.7 below for the details.

Our lower bound is also interesting when compared with the Online Matrix-Vector Multiplication (OMV) Conjecture of Henzinger et al. [154]. In the OMV problem, one is given an $n \times n$ matrix M to preprocess, then afterwards one is given a stream $v_1, \ldots, v_n$ of length-n vectors, and for each v_i, one must output $M \times v_i$ before being given v_{i+1}. The OMV Conjecture posits that one cannot solve this problem in total time $n^{3-\Omega(1)}$ for a general matrix M. At first glance, our lower bound may seem to have implications for the OMV Conjecture: For some kernels K, our lower bound shows that for an input set of points P and corresponding adjacency matrix A_G, and input vector v_i, there is no algorithm running in time $n^{2-\Omega(1)}$ for multiplying $A_G \times v_i$, so perhaps multiplying by n vectors cannot be done in time $n^{3-\Omega(1)}$. However, this is not necessarily the case, since the OMV problem allows $O(n^{2.99})$ time for preprocessing A_G, which our lower bound does not incorporate. More broadly, the matrices A_G which we study, which have very concise descriptions compared to general matrices, are likely not the best candidates for proving the OMV Conjecture. That said, perhaps our results can lead to a form of the OMV Conjecture for geometric graphs with concise descriptions.

Sparsification

Algorithmic techniques Our algorithm for constructing high-dimensional sparsifiers for $\mathsf{K}(x, y) = f(\|x - y\|_2^2)$, when f is a $(2, L)$ multiplicatively Lipschitz function, involves using three classic ideas: the Johnson Lindenstrauss lemma of random projection [162, 169], the notion of well-separated pair decomposition from Callahan and Kosaraju [63, 65], and spectral sparsification via oversampling [181, 269]. Combining these techniques carefully gives us the bounds in Theorem 1.2.

To overcome the 'curse of dimensionality', we use the Lindenstrauss lemma to project onto $\sqrt{L \log n}$ dimensions. This preserves all pairs distance, with a distortion of at most $2^{O(\sqrt{\log n/L})}$. Then, using a $1/2$-well-separated pair decomposition partitions the set of projected distances into bicliques, such that each biclique has edges that are no more than $2^{O(\sqrt{\log n/L})}$ larger than the smallest edge in the biclique. This ratio will upper bound the maximum leverage score of an edge in this biclique in the original K-graph. Each biclique in the set of projected distances has a one-to-one correspondence to a biclique in the original K-graph. Thus to sparsify our K-graph, we sparsify each biclique in the K-graph by uniform sampling, and take the union of the resulting sparsified bicliques. Due to the $(2, L)$-Lipschitz nature of our function, it is guaranteed that the longest edge in any biclique (measured using $\mathsf{K}(x, y)$) is at most $2^{O(\sqrt{L \log n})}$. This upper bounds the maximum leverage score of an edge in this biclique with respect to the K-graph, which then can be used to upper bound the number of edges we need to sample from each biclique via uniform sampling. We take the union of these sampled edges over all bicliques, which gives our results for high-dimensional sparsification summarized in Theorem 1.2. Details can be found in the proof of Theorem 6.3 in Section 6. When L is constant, we get almost linear time sparsification algorithms.

For low dimensional sparsification, we skip the Johnson Lindenstrauss step, and use a $(1 + 1/L)$-well separated pair decomposition. This gives us a nearly linear time algorithm for sparsifying (C, L) multiplicative Lipschitz functions, when $(2L)^{O(d)}$ is small, which covers the case when d is constant and $L = n^{o(1)}$. See Theorem 6.8 for details.

For $\mathsf{K}(u, v) = |\langle u, v\rangle|$, our sparsification algorithm is quite different from the multiplicative Lipschitz setting. In particular, the fact that $\mathsf{K}(u, v) = 0$ on a large family of pairs $u, v \in \mathbb{R}^d$ presents challenges. Luckily, though, this kernel does have some nice structure. For simplicity, just consider defining the K-graph on a set of unit vectors. The weight of any edge in this graph is at most 1 by Cauchy-Schwarz. The key structural property of this graph is that for every set S with $|S| > d + 1$, there is a pair $u, v \in S$ for which the u-v edge has weight at least $\Omega(1/d)$. In other words, the unweighted graph consisting of edges with weight between $\Omega(1/d)$ and 1 does not have independent sets with size greater than $d + 1$. It turns out that all such graphs are dense (see Proposition 7.6)and that all dense graphs have an expander subgraph consisting of a large fraction of the vertices (see Proposition 7.7). Thus, if this expander could be found in $O(n)$ time, we could partition the graph into expander clusters, sparsify the expanders via uniform sampling, and sparsify the edges between expanders via uniform sampling. It is unclear to us how to identify this expander efficiently, so we instead identify clusters with effective resistance diameter $O(\mathrm{poly}(d \log n)/n)$. This can be done via uniform sampling and Johnson-Lindenstrauss [269]. As part of the proof, we prove a novel Markov-style lower bound on the probability that effective resistances deviate too low in a randomly sampled graph, which may be of independent interest (see Lemma 7.10).

Lower bound techniques To prove lower bounds on sparsification for decreasing functions that are not (C_L, L)-multiplicatively Lipschitz, we reduce from exact bichromatic nearest neighbors on two sets of points A and B. In high dimensions, nearest neighbors is hard even for Hamming distance [244], so we may assume that $A, B \subseteq \{0, 1\}^d$. In low dimensions, we may assume that the coordinates of points in A and B consist of integers on at most $O(\log n)$ bits. In both cases, the set of possible distances between points in A and B is discrete. We take advantage of the discrete nature of these distance sets to prove a lower bound. In particular, C_L is set so that C_L is the smallest ratio between any two possible distances between points in A and B. To see this in more detail, see Lemma 8.4.

Let x_f be a point at which the function f is not (C_L, L)-multiplicatively Lipschitz and suppose that we want to solve the decision problem of determining whether or not $\min_{a\in A, b\in B} \|a - b\|_2 \leq k$. We can do this using sparsification by scaling the points in A and B by a factor of k/x_f, sparsifying the K-graph on the resulting points, and thresholding based on the total weight of the resulting A-B cut. If there is a pair with distance at most k, there is an edge crossing the cut with weight at least $f(x_f)$ because f is a decreasing function. Therefore, the sparsifier has total weight at least $f(x_f)/(1 + \varepsilon) = f(x_f)/2$ crossing the A-B cut by the cut sparsification approximation guarantee. If there is not a pair with distance at most k, no edges crossing the cut with weight larger than $f(C_L x_f) \leq C_L^{-L} f(x_f) \leq (1/n^{10}) \cdot f(x_f)$ by choice of C_L. Therefore, the total weight of the A-B cut is at most $(1/n^8) \cdot f(x_f)$, which means that it is at most $((1 + \varepsilon)/n^8) \cdot f(x_f) < f(x_f)/4$ in the sparsifier. In particular, thresholding correctly solves the decision problem and one sparsification is enough to solve bichromatic nearest neighbors.

1.3 Brief summary of our results in terms of p_f

Before proceeding to the body of the chapter, we summarize our results. Recall that we consider three linear-algebraic problems in this chapter, along with two different dimension settings (low and high). This gives six different settings to consider. We now define p_f in each of these settings. In all high-dimensional settings, we have found a definition of p_f that characterizes the complexity of the problem. In some low-dimensional settings, we do not know of a suitable definition for p_f and leave this as an open problem. For simplicity, we focus here only on decreasing functions f, although all of our algorithms, and most of our hardness results, hold for more general functions as well.

Dimension	**Multiplication**	**Sparsification**	**Solving**
$d = \text{poly}(\log n)$	f_1	f_1, f_2	f_1
$c^{\log^* n} < d < O(\log^{1-\delta} n)$ for $\delta > 0$	f_1, f_2, f_3	f_1, f_2, f_3	f_1, f_2, f_3

Table 5.1: Functions among f_1, f_2, f_3, f_4 that have almost-linear time algorithms

1. Adjacency matrix-vector multiplication
   (a) High dimensions: p_f is the minimum degree of any polynomial that $1/2^{\text{poly}(\log n)}$-additively approximates f. $p_f > \Omega(\log n)$ implies subquadratic-time hardness (Theorem 1.1 part 2), while $p_f < o(\log n)$ implies an almost-linear time algorithm (Theorem 1.1, part 1).
   (b) Low dimensions: Not completely understood. The fast multipole method yields an almost-linear time algorithm for some functions, like the Gaussian kernel (Theorem 2.1), but functions exist that are hard in low dimensions (Proposition 5.22).
2. Sparsification. In both settings, p_f is the minimum value for which f is (C, p_f)-multiplicatively Lipschitz, where $C = 1 + 1/p_f^c$ for some constant $c > 0$ independent of f.
   (a) High dimensions: If $p_f > \Omega(\log^2 n)$ and f is nonincreasing, then no subquadratic time algorithm exists (Theorem 1.3). If $p_f < o(\log n)$, then an almost-linear time algorithm for sparsification exists (Theorem 1.2).
   (b) Low dimensions: There is some constant $t > 1$ such that if $p_f > \Omega(n^t)$ and f is nonincreasing, then no subquadratic time algorithm exists (Theorem 2.5).If $p_f < n^{o(1/d)}$, then there is a subquadratic time algorithm (Theorem 2.4).
3. Laplacian solving.
   (a) High dimensions: p_f is the maximum of the p_f values in the *Adjacency matrix-vector multiplication* and *Sparsification* settings, with hardness occurring for decreasing functions f if $p_f > \Omega(\log^2 n)$ (Corollary 1.5.1 combined with Theorem 1.6) and an algorithm existing when $p_f < o(\log n)$ (Theorem 1.5).
   (b) Low dimensions: Not completely understood, as in the low-dimensional multiplication setting. As in the sparsification setting, we are able to show that there is a constant t such that if f is nonincreasing and $p_f > \Omega(n^t)$ where p_f is defined as in the *Sparsifiction* setting, then no subquadratic time algorithm exists (Theorem 8.6).

Many of our results are not tight for two reasons: (a) some of the hardness results only apply to decreasing functions, and (b) there are gaps in p_f values between the upper and lower

bounds. However, neither of these concerns are important in most applications, as (a) weight often decreases as a function of distance and (b) p_f values for natural functions are often either very low or very high. For example, $p_f > \Omega(\text{polylog}(n))$ for all problems for the Gaussian kernel ($f(x) = e^{-x}$), while $p_f = O(1)$ for sparsification and $p_f > \Omega(\text{polylog}(n))$ for multiplication for the gravitational potential ($f(x) = 1/x$). Resolving the gap may also be difficult, as for intermediate values of p_f, the true best running time is likely an intermediate running time of $n^{1+c+o(1)}$ for some constant $0 < c < 1$. Nailing down and proving such a lower bound seems beyond the current techniques in fine-grained complexity.

1.4 Summary of our Results on Examples

To understand our results better, we illustrate how they apply to some examples. For each of the functions f_i given below, make the K-graph, where $\mathsf{K}_i(u, v) = f_i(\|u - v\|_2^2)$:

1. $f_1(z) = z^k$ for a positive integer constant k.
2. $f_2(z) = z^c$ for a negative constant or a positive non-integer constant c.
3. $f_3(z) = e^{-z}$ (the Gaussian kernel).
4. $f_4(z) = 1$ if $z \leq \theta$ and $f_4(z) = 0$ if $z > \theta$ for some parameter $\theta > 0$ (the threshold kernel).

In Table 5.1, we summarize for which of the above functions there are efficient algorithms and for which we have hardness results. There are six regimes, corresponding to three problems (multiplication, sparsification, and solving) and two dimension regimes ($d = \text{poly}(\log n)$ and $d = c^{\log^* n}$). A function is placed in a table cell if an almost-linear time algorithm exists, where runtimes are $n^{1+o(1)}(\log(\alpha n/\varepsilon))^t$ in the case of multiplication and system solving and $n^{1+o(1)}(\log(\alpha n))^t/\varepsilon^2$ in the case of sparsification for some $t \leq O(\log^{1-\delta} n)$ for some $\delta > 0$. Moreover, for each of these functions f_1, f_2, f_3, and f_4, if it does not appear in a table cell, then we show a lower bound that no subquadratic time algorithm exists in that regime assuming SETH.

1.5 Other Related Work

Linear Program Solvers Linear Program is a fundamental problem in convex optimization. There is a long list of work focused on designing fast algorithms for linear program [56, 57, 95, 101, 167, 173, 176, 197, 198, 201, 264, 266, 282, 283]. For the dense input matrix, the state-of-the-art algorithm [167] takes $n^{\max\{\omega, 2+1/18\}} \log(1/\varepsilon)$ time, ω is the exponent of matrix multiplication [19]. The solver can run faster when matrix A has some structures, e.g. Laplacian matrix.

Laplacian System Solvers It is well understood that a Laplacian linear system can be solved in time $\widetilde{O}(m \log(1/\varepsilon))$, where m is the number of edges in the graph generating the Laplacian [93, 175, 181, 182, 187, 188, 196, 271]. This algorithm is very efficient when the graph is sparse. However, in our setting where the K graph is dense but succinctly describable by only n points in $\mathbb{R}^d$, we aim for much faster algorithms.

Algorithms for Kernel Density Function Approximation A recent line of work by Charikar et al. [37, 70] also studies the algorithmic KDE problem. They show, among other things, that kernel density functions for "smooth" kernels K can be estimated in time which depends only polynomially on the dimension d, but which depends polynomially on the error ε. We are unfortunately unable to use their algorithms in our setting, where we need to solve KAdjE with $\varepsilon = n^{-\Omega(1)}$, and the algorithms of Charikar et al. do not run in subquadratic time. We instead design and make use of algorithms whose running times have only polylogarithmic dependences on ε, but often have exponential dependences on d.

Kernel Functions Kernel functions are useful functions in data analysis, with applications in physics, machine learning, and computational biology [267]. There are many kernels studied and applied in the literature; we list here most of the popular examples.

The following kernels are of the form $\mathsf{K}(x, y) = f(\|x - y\|_2^2)$, which we study in this chapter: the Gaussian kernel [231, 241], exponential kernel, Laplace kernel [241], rational quadratic kernel, multiquadric kernel [39], inverse multiquadric kernel [218, 221], circular kernel [55], spherical kernel, power kernel [126], log kernel [39, 218], Cauchy kernel [241], and generalized T-Student kernel [52].

For these next kernels, it is straightforward that their corresponding graphs have low-rank adjacency matrices, and so efficient linear algebra is possible using the Woodbury Identity (see Section 3.4 below): the linear kernel [157, 222, 256, 260] and the polynomial kernel [44, 68, 97, 132].

Finally, the following relatively popular kernels are not of the form we directly study in this chapter, and we leave extending our results to them as an important open problem: the Hyperbolic tangent (Sigmoid) kernel [53, 155, 165, 184, 248, 306, 307, 308], spline kernel [148, 281], B-spline kernel [149, 219], Chi-Square kernel [288], and the histogram intersection kernel and generalized histogram intersection [54]. More interestingly, our result also can be applied to Neural Tangent Kernel [164], which plays a crucial role in the recent work about convergence of neural network training [12, 13, 58, 111, 166, 195, 203, 265]. For more details, we refer the readers to Section 10.

Acknowledgements The authors would like to thank Lijie Chen for helpful suggestions in the hardness section and explanation of his papers. The authors would like to thank Sanjeev Arora, Simon Du, and Jason Lee for useful discussions about the neural tangent kernel.

2 Summary of Low Dimensional Results

In the results we've discussed so far, we show that in high-dimensional settings, the curse of dimensionality applies to a wide variety of functions that are relevant in applications, including the Gaussian kernel and inverse polynomial kernels. Luckily, in many settings, the points supplied as input are very low-dimensional. In the classic n-body problem, for example, the input points are 3-dimensional. In this subsection, we discuss our results pertaining to whether algorithms with runtimes exponential in d exist; such algorithms can still be efficient in low dimensions $d = o(\log n)$.

2.1 Multiplication

The prior work on the fast multipole method [140, 141, 142] yields algorithms with runtime $(\log(n/\varepsilon))^{O(d)} n^{1+o(1)}$ for ε-approximate adjacency matrix-vector multiplication for a number of functions K, including when $\mathsf{K}(u,v) = \frac{1}{\|u-v\|_2^c}$ for a constant c and when $\mathsf{K}(u,v) = e^{-\|u-v\|_2^2}$. In order to explain what functions K the fast multipole methods work well for, and to clarify dependencies on d in the literature, we give a complete exposition of how the fast multipole method of [144] works on the Gaussian kernel:

Theorem 2.1 (fast Gaussian transform [144], exposition in Section 9)**.** *Let $\mathsf{K}(x,y) = \exp(-\|x-y\|_2^2)$. Given a set of points $P \subset \mathbb{R}^d$ with $|P| = n$. Let G denote the K-graph. For any vector $u \in \mathbb{R}^d$, for accuracy parameter ε, there is an algorithm that runs in $n \log^{O(d)}(\|u\|_1/\varepsilon)$ time to approximate $A_G \cdot u$ within ε additive error.*

The fast multipole method is fairly general, and so similar algorithms also exist for a number of other functions K; see Section 9.3 for further discussion. Unlike in the high-dimensional case, we do not have a characterization of the functions for which almost-linear time algorithms exist in near-constant dimension. We leave this as an open problem. Nonetheless, we are able to show lower bounds, even in barely super-constant dimension $d = \exp(\log^*(n))$[2], on adjacency matrix-vector multiplication for kernels that are not multiplicatively Lipschitz:

Theorem 2.2 (Informal version of Proposition 5.22)**.** *For some constant $c > 1$, any function f that is not (C,L)-multiplicatively Lipschitz for any constants $C > 1, L > 1$ does not have an $n^{1+o(1)}$ time adjacency matrix-vector multiplication (up to $2^{-\mathrm{poly}(\log n)}$ additive error) algorithm in $c^{\log^* n}$ dimensions assuming SETH when $\mathsf{K}(u,v) = f(\|u-v\|_2^2)$.*

This includes threshold functions, but does not include piecewise exponential functions. Piecewise exponential functions do have efficient adjacency multiplication algorithms by Theorem 9.13.

To illustrate the complexity of the adjacency matrix-vector multiplication problem in low dimensions, we are also able to show hardness for the function $\mathsf{K}(u,v) = |\langle u,v\rangle|$ in nearly constant dimensions. By comparison, we are able to sparsify for this function K, even in very high $d = n^{o(1)}$ dimensions (in Theorem 1.4 above).

Theorem 2.3 (Informal version of Corollary 5.21.1)**.** *For some constant $c > 1$, assuming SETH, adjacency matrix-vector multiplication (up to $2^{-\mathrm{poly}(\log n)}$ additive error) in $c^{\log^* n}$ dimensions cannot be done in subquadratic time in dimension $d = \exp(\log^*(n))$ when $\mathsf{K}(u,v) = |\langle u,v\rangle|$.*

2.2 Sparsification

We are able to give a characterization of the decreasing functions for which sparsification is possible in near-constant dimension. We show that a polynomial dependence on the multiplicative Lipschitz constant is allowed, unlike in the high-dimensional setting:

Theorem 2.4 (Informal version of Theorem 6.8)**.** *Let f be a $(1+1/L, L)$-multiplicatively Lipschitz function and let $\mathsf{K}(u,v) = f(\|u-v\|_2^2)$. Then an $(1\pm\varepsilon)$-spectral sparsifier for the K-graph on n points can be found in $n^{1+o(1)} L^{O(d)} (\log \alpha)/\varepsilon^2$ time.*

[2]Here, $\log^*(n)$ denotes the *very* slowly growing iterated logarithm of n.

Thus, geometric graphs for piecewise exponential functions with $L = n^{o(1)}$ can be sparsified in almost-linear time when d is constant, unlike in the case when $d = \Omega(\log n)$. In particular, spectral clustering can be done in $O(kn^{1+o(1)})$ time for k clusters in low dimensions. Unfortunately, not all geometric graphs can be sparsified, even in nearly constant dimensions:

Theorem 2.5 (Informal version of Theorem 8.2)**.** *There are constants $c' \in (0,1), c > 1$ and a value C_L given $L > 1$ for which any decreasing function f that is not (C_L, L)-multiplicatively Lipschitz does not have an $O(nL^{c'})$ time sparsification algorithm for K-graphs on $c^{\log^* n}$ dimensional points, where $\mathsf{K}(u,v) = f(\|u-v\|_2^2)$.*

This theorem shows, in particular, that geometric graphs of threshold functions are not sparsifiable in subquadratic time even for low-dimensional pointsets. These two theorems together nearly classify the decreasing functions for which efficient sparsification is possible, up to the exponent on L.

2.3 Laplacian solving

As in the case of multiplication, we are unable to characterize the functions for which solving Laplacian systems can be done in almost-linear time in low dimensions. That said, we still have results for many functions K, including most kernel functions of interest in applications. We prove most of these using the aforementioned connection from Section 4: if a K graph can be efficiently sparsified, then there is an efficient Laplacian multiplier for K graphs if and only if there is an efficient Laplacian system solver for K graphs.

For the kernels $\mathsf{K}(u,v) = 1/\|u-v\|_2^c$ for constants c and the piecewise exponential kernel, we have almost-linear time algorithms in low dimensions by Theorems 6.3 and 6.8 respectively. Furthermore, the fast multipole method yields almost-linear time algorithms for multiplication. Therefore, there are almost-linear time algorithm for solving Laplacian systems in geometric graphs for these kernels.

A similar approach also yields hardness results. Theorem 1.4 above implies that an almost-linear time algorithm for solving Laplacian systems on K-graphs for $\mathsf{K}(u,v) = |\langle u, v\rangle|$ yields an almost-linear time algorithm for K-adjacency multiplication. However, no such algorithm exists assuming SETH by Theorem 2.3 above. Therefore, SETH implies that no almost-linear time algorithm for solving Laplacian systems in this kernel can exist.

We directly (i.e. without using a sparsifier algorithm) show an additional hardness result for solving Laplacian systems for kernels that are not multiplicatively Lipschitz, like threshold functions of ℓ_2-distance:

Theorem 2.6 (Informal version of Theorem 8.6)**.** *Consider an $L > 1$. There is some sufficiently large value $C_L > 1$ depending on L such that for any decreasing function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is not (C_L, L)-multiplicatively Lipschitz, no $O(nL^{c'} \log \alpha)$-time algorithm exists for solving Laplacian systems $2^{-\mathrm{poly}(\log n)}$ approximately in the K-graph of a set of n points in $c^{\log^* n}$ dimensions for some constants $c > 1, c' \in (0,1)$ assuming SETH, where $\mathsf{K}(u,v) = f(\|u-v\|_2^2)$.*

3 Preliminaries

Our results build off of algorithms and hardness results from many different areas of theoretical computer science. We begin by defining the relevant notation and describing the important past work.

3.1 Notation

For an $n \in \mathbb{N}_+$, let $[n]$ denote the set $\{1, 2, \cdots, n\}$.

For any function f, we write $\widetilde{O}(f)$ to denote $f \cdot \log^{O(1)}(f)$. In addition to $O(\cdot)$ notation, for two functions f, g, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leq Cg$ (resp. $\geq$) for an absolute constant C. We use $f \eqsim g$ to mean $cf \leq g \leq Cf$ for constants c, C.

For a matrix A, we use $\|A\|_2$ to denote the spectral norm of A. Let $A^\top$ denote the transpose of A. Let $A^\dagger$ denote the Moore-Penrose pseudoinverse of A. Let A^{-1} denote the inverse of a full rank square matrix.

We say matrix A is positive semi-definite (PSD) if $A = A^\top$ and $x^\top Ax \geq 0$ for all $x \in \mathbb{R}^n$. We use $\preceq, \succeq$ to denote the semidefinite ordering, e.g. $A \succeq 0$ denotes that A is PSD, and $A \succeq B$ means $A - B \succeq 0$. We say matrix A is positive definite (PD) if $A = A^\top$ and $x^\top Ax > 0$ for all $x \in \mathbb{R}^n - \{0\}$. $A \succ B$ means $A - B$ is PD.

For a vector v, we denote $\|v\|_p$ as the standard ℓ_p norm. For a vector v and PSD matrix A, we let $\|v\|_A = (v^\top Av)^{1/2}$.

The iterated logarithm $\log^* : \mathbb{R} \to \mathbb{Z}$ is given by

$$\log^*(n) = \begin{cases} 0, & \text{if } n \leq 1; \\ 1 + \log^*(\log n), & \text{otherwise.} \end{cases}$$

We use G_{grav} to denote the Gravitational constant.

We define α slightly differently in different sections. Note that both are less than the value of α used in Theorem 1.2:

Table 5.2

Notation	Meaning	Location
α	$\frac{\max_{i,j} f(\|x_i - x_j\|_2^2)}{\min_{i,j} f(\|x_i - x_j\|_2^2)}$	Section 4
α	$\frac{\max_{i,j} \|x_i - x_j\|_2}{\min_{i,j} \|x_i - x_j\|_2}$	Section 6

3.2 Graph and Laplacian Notation

Let $G = (V, E, w)$ be a connected weighted undirected graph with n vertices and m edges and edge weights $w_e > 0$. We say $r_e = 1/w_e$ is the resistance of edge e. If we give a direction to the edges of G arbitrarily, we can write its Laplacian as $L_G = B^\top WB$, where $W \in \mathbb{R}^{m \times m}$ is the

diagonal matrix $W(e,e) = w_e$ and $B \in \mathbb{R}^{m \times n}$ is the signed edge-vertex incidence matrix and can be defined in the following way

$$B(e,v) = \begin{cases} 1, & \text{if } v \text{ is } e\text{'s head;} \\ -1, & \text{if } v \text{ is } e\text{'s tail;} \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

A useful notion related to Laplacian matrices is the effective resistance of a pair of nodes:

Definition 3.1 (Effective resistance)**.** *The effective resistance of a pair of vertices $u, v \in V_G$ is defined as*

$$\texttt{Reff}_G(u,v) = b_{u,v}^\top L^\dagger b_{u,v}$$

where $b_{u,v} \in \mathbb{R}^{|V_G|}$ is an all zero vector except for entries of 1 at u and -1 at v.

Using effective resistance, we can define leverage score

Definition 3.2 (Leverage score)**.** *The leverage score of an edge $e = (u,v) \in E_G$ is defined as*

$$l_e = w_e \cdot \texttt{Reff}_G(u,v).$$

We define a useful notation called electrical flow

Definition 3.3 (Electrical flow)**.** *Let $B \in \mathbb{R}^{m \times n}$ be defined as Eq.* (5.2)*, for a given demand vector $d \in \mathbb{R}^n$, we define electrical flow $f \in \mathbb{R}^m$ as follows:*

$$f = \arg \min_{f : B^\top f = d} \sum_{e \in E_G} f_e^2 / w_e.$$

We let $d(i)$ denote the degree of vertex i. For any set $S \subseteq V$, we define volume of S: $\mu(S) = \sum_{i \in S} d(i)$. It is obvious that $\mu(V) = 2|E|$. For any two sets $S, T \subseteq V$, let $E(S,T)$ be the set of edges connecting a vertex in S with a vertex in T. We call $\Phi(S)$ to be the conductance of a set of vertices S, and can be formally defined as

$$\Phi(S) = \frac{|E(S, V \setminus S)|}{\min(\mu(S), \mu(V \setminus S))}.$$

We define the notation conductance, which is standard in the literature of graph partitioning and graph clustering [23, 24, 170, 211, 235, 271, 312].

Definition 3.4 (Conductance)**.** *The conductance of a graph G is defined as follows:*

$$\Phi_G = \min_{S \subset V} \Phi(S).$$

Lemma 3.5 ([11, 271])**.** *A graph G with minimum conductance Φ_G has the property that for every pair of vertices u, v,*

$$\texttt{Reff}_G(u,v) \leq O\left(\left(\frac{1}{c_u} + \frac{1}{c_v}\right) \cdot \frac{1}{\Phi_G^2}\right)$$

where c_u is the sum of the weights of edges incident with u. Furthermore, for every pair of vertices u, v,

$$\texttt{Reff}_G(u,v) \geq \max(1/c_u, 1/c_v)$$

For a function $\mathsf{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, the K-graph on a set of points $X \subseteq \mathbb{R}^d$ is the graph with vertex set X and edge weights $\mathsf{K}(u,v)$ for $u, v \in X$. For a function $f\mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, the f-graph on a set of points X is defined to be the K graph on X for $\mathsf{K}(u,v) = f(\|u - v\|_2)$.

3.3 Spectral Sparsification via Random Sampling

Here, we state some well known results on spectral sparsification via random sampling, from previous works. The theorems below are essential for our results on sparsifying geometric graphs quickly.

Theorem 3.6 (Oversampling [182]). *Consider a graph $G = (V, E)$ with edge weights $w_e > 0$ and probabilities $p_e \in (0, 1]$ assigned to each edge and parameters $\delta \in (0, 1), \varepsilon \in (0, 1)$. Generate a reweighted subgraph H of G with q edges, with each edge e sampled with probability p_e/t and added to H with weight $w_e t/(p_e q)$, where $t = \sum_{e \in E} p_e$. If*

1. *$q \geq C \cdot \varepsilon^{-2} \cdot t \log t \cdot \log(1/\delta)$, where $C > 1$ is a sufficiently large constant*
2. *$p_e \geq w_e \cdot \mathtt{Reff}_G(u, v)$ for all edges $e = \{u, v\}$ in G*

then $(1 - \varepsilon)L_G \preceq L_H \preceq (1 + \varepsilon)L_G$ with probability at least $1 - \delta$.

Algorithm 1

```
procedure OVERSAMPLING(G, w, p, ε, δ)                    ▷ Theorem 3.6
    t ← Σ_{e∈E} p_e
    q ← C · ε^{-2} · t log t · log(1/δ)
    Initialize H to be an empty graph
    for i = 1 → q do
        Sample one e ∈ E with probability p_e/t
        Add that edge with weight w_e t/(p_e q) to graph H
    end for
    return H
end procedure
```

Theorem 3.7 ([269] effective resistance data structure). *There is a $\widetilde{O}(m(\log \alpha)/\varepsilon^2)$ time algorithm which on input $\varepsilon > 0$ and $G = (V, E, w)$ with $\alpha = w_{\max}/w_{\min}$ computes a $(24 \log n/\varepsilon^2) \times n$ matrix $\widetilde{Z}$ such that with probability at least $1 - 1/n$,*

$$(1 - \varepsilon)\mathtt{Reff}_G(u, v) \leq \|\widetilde{Z} b_{uv}\|_2^2 \leq (1 + \varepsilon)\mathtt{Reff}_G(u, v)$$

for every pair of vertices $u, v \in V$.

The following is an immediate corollary of Theorems 3.6 and 3.7:

Corollary 3.7.1 ([269]). *There is a $\widetilde{O}(m(\log \alpha)/\varepsilon^2)$ time algorithm which on input $\varepsilon > 0$ and $G = (V, E, w)$ with $\alpha = w_{\max}/w_{\min}$, produces an $(1 \pm \varepsilon)$-approximate sparsifier for G.*

3.4 Woodbury Identity

Proposition 3.8 ([299, 300]). *The Woodbury matrix identity is*

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

where A, U, C and V all denote matrices of the correct (conformable) sizes: For integers n and k, A is $n \times n$, U is $n \times k$, C is $k \times k$ and V is $k \times n$.

The Woodbury identity is useful for solving linear systems in a matrix M which can be written as the sum of a diagonal matrix A and a low-rank matrix UV for $k \ll n$ (setting $C = I$).

3.5 Tail Bounds

We will use several well-known tail bounds from probability theory.

Theorem 3.9 (Chernoff Bounds [85]). *Let $X = \sum_{i=1}^{n} X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$, and all X_i are independent. Let $\mu = \mathbf{E}[X] = \sum_{i=1}^{n} p_i$. Then*
1. $\Pr[X \geq (1+\delta)\mu] \leq \exp(-\delta^2\mu/3)$, $\forall \delta > 0$;
2. $\Pr[X \leq (1-\delta)\mu] \leq \exp(-\delta^2\mu/2)$, $\forall 0 < \delta < 1$.

Theorem 3.10 (Hoeffding bound [156]). *Let $X_1, \cdots, X_n$ denote n independent bounded variables in $[a_i, b_i]$. Let $X = \sum_{i=1}^{n} X_i$, then we have*

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

3.6 Fine-Grained Hypotheses

Strong Exponential Time Hypothesis Impagliazzo and Paturi [159] introduced the Strong Exponential Time Hypothesis (SETH) to address the complexity of CNF-SAT. Although it was originally stated only for deterministic algorithms, it is now common to extend SETH to randomized algorithms as well.

Hypothesis 3.11 (Strong Exponential Time Hypothesis (SETH)). *For every $\varepsilon > 0$ there exists an integer $k \geq 3$ such that CNF-SAT on formulas with clause size at most k (the so called k-SAT problem) and n variables cannot be solved in $O(2^{(1-\varepsilon)n})$ time even by a randomized algorithm.*

Orthogonal Vectors Conjecture The Orthogonal Vectors (OV) problem asks: given n vectors $x_1, \cdots, x_n \in \{0,1\}^d$, are there i, j such that $\langle v_i, v_j \rangle = 0$ (where the inner product is taken over $\mathbb{Z}$)? It is easy to see that $O(n^2 d)$ time suffices for solving OV, and slightly subquadratic-time algorithms are known in the case of small d [4, 67]. It is conjectured that there is no OV algorithm running in $n^{1.99}$ time when $d = \omega(\log n)$.

Conjecture 3.12 (Orthogonal Vectors Conjecture (OVC) [3, 296]). *For every $\varepsilon > 0$, there is a $c \geq 1$ such that OV cannot be solved in $n^{2-\varepsilon}$ time on instances with $d = c \log n$.*

In particular, it is known that SETH implies OVC [296]. SETH and OVC are the most common hardness assumption in fine-grained complexity theory, and they are known to imply tight lower bounds for a number of algorithmic problems throughout computer science. See, for instance, the survey [297] for more background.

3.7 Dimensionality Reduction

We make use of the following binary version of the Johnson-Lindenstrauss lemma due to Achlioptas [5]:

Theorem 3.13 ([5, 169]). *Given fixed vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ and $\varepsilon > 0$, let $Q \in \mathbb{R}^{k \times d}$ be a random $\pm 1/\sqrt{k}$ matrix (i.e. independent Bernoulli entries) with $k \geq 24(\log n)/\varepsilon^2$. Then with probability at least $1 - 1/n$,*

$$(1-\varepsilon)\|v_i - v_j\|^2 \leq \|Qv_i - Qv_j\|^2 \leq (1+\varepsilon)\|v_i - v_j\|^2$$

for all pairs $i, j \in [n]$.

We will also use the following variant of Johnson Lindenstrauss for Euclidean space, for random projections onto $o(\log n)$ dimensions:

Lemma 3.14 (Ultralow Dimensional Projection [103, 169], see Theorem 8.2 in [258] for example). *For $k = o(\log n)$, with high probability the maximum distortion in pairwise distance obtained from projecting n points into k dimensions (with appropriate scaling) is at most $n^{O(1/k)}$.*

3.8 Nearest Neighbor Search

Our results will make use of a number of prior results, both algorithms and lower bounds, for nearest neighbor search problems.

Nearest Neighbor Search Data Structures

Problem 1 ([25, 242] data-structure ANN). *Given an n-point dataset P in $\mathbb{R}^d$ with $d = n^{o(1)}$, the goal is to preprocess it to answer the following queries. Given a query point $q \in \mathbb{R}^d$ such that there exists a data point within ℓ_p distance r from q, return a data point within ℓ_p distance cr from q.*

Theorem 3.15 ([26]). *There exists a data structure that returns a $2c$-approximation to the nearest neighbor distance in ℓ_2^d with preprocessing time and space $O_c(n^{1+1/c^2+o_c(1)} + nd)$ and query time $O_c(dn^{1/c^2+o_c(1)})$*

Hardness for Approximate Hamming Nearest Neighbor Search We provide the definition of the Approximate Nearest Neighbor search problem

Problem 2 (monochromatic ANN). *The monochromatic Approximate Nearest Neighbor (ANN) problem is defined as : given a set of n points $x_1, \cdots, x_n \in \mathbb{R}^d$ with $n^{o(1)}$, the goal is to compute α-approximation of $\min_{i \neq j} \mathbf{d}(x_i, x_j)$.*

Theorem 3.16 ([245]). *Let $\mathbf{d}(x, y)$ be ℓ_p distance. Assuming SETH, for every $\delta > 0$, there is a $\varepsilon > 0$ such that the monochromatic $(1+\varepsilon)$-ANN problem for dimension $d = \Omega(\log n)$ requires time $n^{1.5-\delta}$.*

Problem 3 (bichromatic ANN). *Let $\mathbf{d}(\cdot, \cdot)$ denote the some distance function. Let $\alpha > 1$ denote some approximation factor. The bichromatic Approximate Nearest Neighbor (ANN) problem is defined as: given two sets A, B of vectors $\mathbb{R}^d$, the goal is to compute α-approximation of $\min_{a \in A, b \in B} \mathbf{d}(a, b)$.*

Theorem 3.17 ([244]). *Let $\mathbf{d}(x, y)$ be any of Euclidean, Manhattan, Hamming($\|x - y\|_0$), and edit distance. Assuming SETH, for every $\delta > 0$, there is a $\varepsilon > 0$ such that the bichromatic $(1+\varepsilon)$-ANN problem for dimension $d = \Omega(\log n)$ requires time $n^{2-\delta}$.*

By comparison, the best known algorithm for $d = \Omega(\log n)$ for each of these distance measures other than edit distance runs in time about $dn + n^{2-\Omega(\varepsilon^{1/3}/\log(1/\varepsilon))}$ [20].

Hardness for $\mathbb{Z}$-Max-IP

Problem 4. *For $n, d \in \mathbb{N}$, the $\mathbb{Z}$-MaxIP problem for dimension d asks: given two sets A, B of vectors from $\mathbb{Z}^d$, compute*

$$\max_{a \in A, b \in B} \langle a, b \rangle.$$

$\mathbb{Z}$-MaxIP is known to be hard even when the dimension d is barely superconstant:

Theorem 3.18 (Theorem 1.14 in [81])**.** *Assuming SETH (or OVC), there is a constant c such that any exact algorithm for $\mathbb{Z}$-MaxIP for $d = c^{\log^* n}$ dimensions requires $n^{2-o(1)}$ time, with vectors of $O(\log n)$-bit entries.*

It is believed that $\mathbb{Z}$-MaxIP cannot be solved in truly subquadratic time even in constant dimension [81]. Even for $d = 3$, the best known algorithm runs in $O(n^{4/3})$ time and has not been improved for decades:

Theorem 3.19 ([8, 220, 304])**.** *$\mathbb{Z}$-MaxIP for $d = 3$ can be solved in time $O(n^{4/3})$. For general d, it can be solved in $n^{2-\Theta(1/d)}$.*

The closely related problem of ℓ_2-nearest neighbor search is also hard in barely superconstant dimension:

Theorem 3.20 (Theorem 1.16 in [81])**.** *Assuming SETH (or OVC), there is a constant c such that any exact algorithm for bichromatic ℓ_2-closest pair for $d = c^{\log^* n}$ dimensions requires $n^{2-o(1)}$ time, with vectors of $c_0 \log n$-bit entries for some constants $c > 1$ and $c_0 > 1$.*

3.9 Geometric Laplacian System

Building off of a long line of work on Laplacian system solving [93, 175, 181, 182, 271], we study the problem of solving geometric Laplacian systems:

Problem 5. *Let $\mathsf{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Given a set of points $x_1, \cdots, x_n \in \mathbb{R}^d$, a vector $b \in \mathbb{R}^n$ and accuracy parameter ε. Let graph G denote the graph that has n vertices and each edge(i, j)'s weight is $\mathsf{K}(x_i, x_j)$. Let L_G denote the Laplacian matrix of graph G. The goal is to output a vector $u \in \mathbb{R}^n$ such that*

$$\|u - L_G^\dagger b\|_{L_G} \leq \varepsilon \|L_G^\dagger b\|_{L_G}$$

where $L_G^\dagger$ denotes the pseudo-inverse of L_G and matrix norm is defined as $\|c\|_A = \sqrt{c^\top A c}$.

4 Equivalence of Matrix-Vector Multiplication and Solving Linear Systems

In this section, we show that for linear systems with sparse preconditioners, approximately solving them is equivalent to approximate matrix multiplication. We begin by formalizing our notion of approximation.

Definition 4.1. *Given a matrix M and a vector x, we say that b is an ε-*approximate multiplication of Mx *if*

$$(b - Mx)^\top M^\dagger (b - Mx) \leq \varepsilon \cdot x^\top M x.$$

Given a vector d, we say that a vector y is an ε-approximate solution to $My = d$ *if y is an ε-approximate multiplication of $M^\dagger d$.*

Before stating the desired reductions, we state a folklore fact about the Laplacian norm:

Proposition 4.2 (property of Laplacian norm)**.** *Consider a w-weighted n-vertex connected graph G and let $w_{\min} = \min_{u,v \in G} w_{uv}$, $w_{\max} = \max_{u,v \in G} w_{uv}$, and $\alpha = w_{\max}/w_{\min}$. Then, for any vector $x \in \mathbb{R}^n$ that is orthogonal to the all ones vector,*

$$\frac{w_{\min}}{2n^4\alpha^2}\|x\|_\infty^2 \leq \|x\|_{L_G}^2 \leq n^2 w_{\max}\|x\|_\infty^2$$

and for any vector $b \in \mathbb{R}^n$ orthogonal to all ones,

$$\frac{1}{n^2 w_{\max}}\|b\|_\infty^2 \leq \|b\|_{L_G^\dagger}^2 \leq \frac{2n^4\alpha^2}{w_{\min}}\|b\|_\infty^2$$

Proof. **Lower Bound for L_G**: Let $\lambda_{\min}$ and $\lambda_{\max}$ denote the minimum nonzero and maximum eigenvalues of $D_G^{-1/2} L_G D_G^{-1/2}$ respectively, where D_G is the diagonal matrix of vertex degrees. Since G is connected, all cuts have conductance at least $w_{\min}/(n^2 w_{\max}) = 1/(n^2\alpha)$. Therefore, by Cheeger's Inequality [78], $\lambda_{\min} \geq 1/(2n^4\alpha^2)$. It follows that,

$$\begin{aligned}\|x\|_{L_G}^2 &= x^\top L_G x \\ &\geq (x^\top D_G x)/(2n^4\alpha^2) \\ &\geq \|x\|_\infty^2 w_{\min}/(2n^4\alpha^2)\end{aligned}$$

as desired.

Upper bound for L_G: $\lambda_{\max} \leq 1$. Therefore,

$$\|x\|_{L_G}^2 \leq x^\top D_G x \leq n^2 w_{\max}\|x\|_\infty^2$$

as desired.

Lower bound for $L_G^\dagger$:

$$\|b\|_{L_G^\dagger}^2 \geq (1/\lambda_{\max})\|b\|_{D_G^{-1}}^2 \geq 1/(n^2 w_{\max})\|b\|_\infty^2$$

Upper bound for $L_G^\dagger$:

$$\|b\|_{L_G^\dagger}^2 \leq (1/\lambda_{\min})\|b\|_{D_G^{-1}}^2 \leq (2n^4\alpha^2/w_{\min})\|b\|_\infty^2$$

□

Proposition 4.2 implies the following equivalent definition of ε-approximate multiplication:

Corollary 4.2.1. *Let G be a connected n-vertex graph with edge weights $\{w_e\}_{e \in E(G)}$ with $w_{\min} = \min_{e \in E(G)} w_e$, $w_{\max} = \max_{e \in E(G)} w_e$, and $\alpha = w_{\max}/w_{\min}$ and consider any vectors $b, x \in \mathbb{R}^n$. If*

$$\|b - L_G x\|_\infty \leq \varepsilon w_{\max}\|x\|_\infty$$

,$b^\top \mathbf{1} = 0$, and $x^\top \mathbf{1} = 0$, then b is an $2n^3\alpha^2\varepsilon$-approximate multiplication of $L_G x$.

Proof. Since $b^\top \mathbf{1} = 0$, $(b - L_G x)^\top \mathbf{1} = 0$ also. By the upper bound for $L_G^\dagger$-norms in Proposition 4.2,

$$\|b - L_G x\|_{L_G^\dagger}^2 \leq \frac{2n^4\alpha^2}{w_{\min}} \|b - L_G x\|_\infty^2 \leq 2n^4\alpha^4\varepsilon^2 w_{\min} \|x\|_\infty^2$$

Since $x^\top \mathbf{1} = 0$, x has both nonnegative and nonpositive coordinates. Therefore, since G is connected, there exists vertices a, b in G for which $\{a, b\}$ is an edge and for which $|x_a - x_b| \geq \|x\|_\infty / n$. Therefore,

$$x^\top L_G x \geq w_{ab}(x_a - x_b)^2 \geq (w_{\min}/n^2)\|x\|_\infty^2$$

Substitution shows that

$$\|b - L_G x\|_{L_G^\dagger}^2 \leq 2n^4\alpha^4\varepsilon^2 (n^2 x^\top L_G x)$$

This is the desired result by definition of ε-approximate multiplication. □

Corollary 4.2.2. *Let G be a connected n-vertex graph with edge weights $\{w_e\}_{e \in E(G)}$ with $w_{\min} = \min_{e \in E(G)} w_e$, $w_{\max} = \max_{e \in E(G)} w_e$, and $\alpha = w_{\max}/w_{\min}$ and consider any vectors $b, x \in \mathbb{R}^n$. If b is an $\varepsilon/(2n^3\alpha^2)$-approximate multiplication of $L_G x$ and $b^\top \mathbf{1} = 0$, then*

$$\|b - L_G x\|_\infty \leq \varepsilon w_{\min} \|x\|_\infty$$

Proof. Since $b^\top \mathbf{1} = 0$, $(b - L_G x)^\top \mathbf{1} = 0$ as well. By the lower bound for $L_G^\dagger$-norms in Proposition 4.2 and the fact that b is an approximate multiplication for $L_G x$,

$$\|b - L_G x\|_\infty^2 \leq n^2 \alpha w_{\min} \|b - L_G x\|_{L_G^\dagger}^2 \leq \frac{\varepsilon^2 w_{\min}}{4n^4\alpha^3} x^\top L_G x.$$

Notice that

$$x^\top L_G x = \sum_{\{a,b\} \in E(G)} w_{ab}(x_a - x_b)^2 \leq n^2 w_{\max}(4\|x\|_\infty^2).$$

Therefore, by substitution,

$$\|b - L_G x\|_\infty^2 \leq (\frac{\varepsilon^2 w_{\min}}{4n^4\alpha^3})(n^2 \alpha w_{\min}(4\|x\|_\infty^2)) \leq w_{\min}^2 \varepsilon^2 \|x\|_\infty^2.$$

Taking square roots gives the desired result. □

4.1 Solving Linear Systems Implies Matrix-Vector Multiplication

Lemma 4.3. *Consider an n-vertex w-weighted graph G, let $w_{\min} = \min_{e \in G} w_e$, $w_{\max} = \max_{e \in G} w_e$, $\alpha = w_{\max}/w_{\min}$, and H be a known graph for which*

$$(1 - 1/900)L_G \preceq L_H \preceq (1 + 1/900)L_G.$$

Suppose that H has at most Z edges and suppose that there is a $\mathcal{T}(n,\delta)$-time algorithm SOLVEG(b,δ) *that, when given a vector $b \in \mathbb{R}^n$ and $\delta \in (0,1)$, returns a vector $x \in \mathbb{R}^n$ with*

$$\|x - L_G^\dagger b\|_{L_G} \leq \delta \cdot \|L_G^\dagger b\|_{L_G}.$$

Then, given a vector $x \in \mathbb{R}^n$ and an $\varepsilon \in (0,1)$, there is a

$$\widetilde{O}((Z + \mathcal{T}(n, \varepsilon/(n^2\alpha))) \log(Zn\alpha/\varepsilon))$$

-time algorithm MULTIPLYG(x,ε) *(Algorithm 2) that returns a vector $b \in \mathbb{R}^n$ for which*

$$\|b - L_G x\|_\infty \leq \varepsilon \cdot w_{\min} \cdot \|x\|_\infty.$$

The algorithm MULTIPLYG (Algorithm 2) uses standard preconditioned iterative refinement. It is applied in the opposite from the usual way. Instead of using iterative refinement to solve a linear system given matrix-vector multiplication, we use iterative refinement to do matrix-vector multiplication given access to a linear system solver.

Algorithm 2 MULTIPLYG and MULTIPLYGADDITIVE

procedure MULTIPLYG(x,ε) ▷ Lemma 4.3, Theorem 4.4
  Given: $x \in \mathbb{R}^n$, $\varepsilon \in (0,1)$, the sparsifier H for G, and a system solver for G
  Returns: an approximation b to $L_G x$
  return MULTIPLYGADDITIVE$(x, \varepsilon\|x\|_\infty/(\log^2(\alpha n/\varepsilon)))$
end procedure
procedure MULTIPLYGADDITIVE(x,τ)
  if $\|x\|_\infty \leq \tau/(n^{10}\alpha^5)$ **then**
    return 0
  end if
  $x_{\text{main}} \leftarrow$ SOLVEG$(L_H x, \tau\sqrt{w_{\min}}/(n^{10}\alpha^5))$
  $x_{\text{res}} \leftarrow x - x_{\text{main}}$
  $b_{\text{res}} \leftarrow$ MULTIPLYGADDITIVE(x_{res}, τ)
  return $L_H x + b_{\text{res}}$
end procedure

Proof. In this proof, assume that $\mathbf{1}^\top x = 0$. If this is not the case, then shifting x so that it is orthogonal to $\mathbf{1}$ only decreases its ℓ_2-norm, which means that the ℓ_∞ norm only increases by a factor of $\sqrt{n}$, so the error only increases by $O(\log n)$ additional solves.

Reduction in residual and iteration bound: First, we show that

$$\|x_{\text{res}}\|_{L_G} \leq (1/14)\|x\|_{L_G}$$

Since $(I - L_H L_G^\dagger) L_G (I - L_G^\dagger L_H) \preceq 3(1/900) L_G$,

$$\begin{aligned}
\|x_{\text{res}}\|_{L_G} &= \|x - x_{\text{main}}\|_{L_G} \\
&\leq \|x - L_G^\dagger L_H x\|_{L_G} + \|L_G^\dagger L_H x - x_{\text{main}}\|_{L_G} \\
&\leq (1/15)\|x\|_{L_G} + (1/10000)\|L_G^\dagger L_H x\|_{L_G} \\
&\leq (1/14)\|x\|_{L_G}
\end{aligned}$$

Let x_{final} be the lowest element of the call stack and let k be the number of recursive calls to MULTIPLYGADDITIVE (Algorithm 2). By Proposition 4.2,

$$\|x\|_{L_G} \leq \|x\|_\infty \sqrt{w_{\max}} \cdot n \quad \text{and} \quad \|x_{\text{final}}\|_{L_G} \geq \|x_{\text{final}}\|_\infty \sqrt{w_{\min}}/(2n^2\alpha).$$

By definition of x_{final},

$$\|x_{\text{final}}\|_\infty \geq \frac{\tau}{(14n^{10}\alpha^5)} = \frac{\varepsilon\sqrt{w_{\min}}\|x\|_\infty}{28n^{12}\alpha^5 \log(\alpha n/\varepsilon)}.$$

Therefore,

$$k \leq \log_{14}(\|x\|_{L_G}/\|x_{\text{final}}\|_{L_G}) \leq \log^2(\alpha n/\varepsilon)$$

as desired.

Error: We start by bounding error in the $L_G^\dagger$ norm. Let b be the output of MULTIPLYGADDITIVE(x, τ) (Algorithm 2). We bound the desired error recursively:

$$\begin{aligned}
\|b - L_G x\|_{L_G^\dagger} &= \|L_H x + b_{\text{res}} - L_G x\|_{L_G^\dagger} \\
&= \|b_{\text{res}} - L_G(x - L_G^\dagger L_H x)\|_{L_G^\dagger} \\
&= \|b_{\text{res}} - L_G(x - x_{\text{main}}) - L_G(x_{\text{main}} - L_G^\dagger L_H x)\|_{L_G^\dagger} \\
&\leq \|b_{\text{res}} - L_G(x - x_{\text{main}})\|_{L_G^\dagger} + \|L_G(x_{\text{main}} - L_G^\dagger L_H x)\|_{L_G^\dagger} \\
&= \|b_{\text{res}} - L_G x_{\text{res}}\|_{L_G^\dagger} + \|x_{\text{main}} - L_G^\dagger L_H x\|_{L_G} \\
&\leq \|b_{\text{res}} - L_G x_{\text{res}}\|_{L_G^\dagger} + \sqrt{w_{\min}}\tau/(n^{10}\alpha^5)
\end{aligned}$$

Because $0 = $ MULTIPLYGADDITIVE(x_{final}, τ) (Algorithm 2),

$$\begin{aligned}
\|b - L_G x\|_{L_G^\dagger} &\leq \|x_{\text{final}}\|_{L_G} + k\sqrt{w_{\min}}\tau/(n^{10}\alpha^5) \\
&\leq n\sqrt{w_{\max}}\|x_{\text{final}}\|_\infty + k\sqrt{w_{\min}}\tau/(n^{10}\alpha^5) \\
&\leq \sqrt{w_{\min}}\tau/(n^8\alpha^4) \\
&\leq \varepsilon\sqrt{w_{\min}}\|x\|_\infty/(n^8\alpha^4) && \text{by } \tau \leq \varepsilon\|x\|_\infty
\end{aligned}$$

By Proposition 4.2 applied to $L_G^\dagger$, $\|b - L_G x\|_{L_G^\dagger} \geq \|b - L_G x\|_\infty / (n\sqrt{w_{\max}})$. Therefore,

$$\begin{aligned}
\|b - L_G x\|_\infty &\leq n\sqrt{w_{\max}} \|b - L_G x\|_{L_G^\dagger} \\
&\leq n\sqrt{w_{\max}} \varepsilon \sqrt{w_{\min}} \frac{1}{n^8 \alpha^4} \|x\|_\infty \\
&\leq \varepsilon w_{\min} \|x\|_\infty \cdot \frac{1}{n^7 \alpha^{3.5}} && \text{by } \alpha = w_{\max}/w_{\min} \\
&\leq \varepsilon w_{\min} \|x\|_\infty
\end{aligned}$$

as desired.

Runtime: There is one call to SOLVEG and one multiplication by L_H per call to MULTIPLYGADDITIVE (Algorithm 2). Each multiplication by L_H takes $O(Z)$ time. As we have shown, there are only $k \leq O(\log(n\alpha/\varepsilon))$ calls to MULTIPLYGADDITIVE (Algorithm 2). Therefore, we are done. □

4.2 Matrix-Vector Multiplication Implies Solving Linear Systems

The converse is well-known to be true [182, 271]:

Lemma 4.4 ([271]). *Consider an n-vertex w-weighted graph G, let $w_{\min} = \min_{e \in G} w_e$, $w_{\max} = \max_{e \in G} w_e$, $\alpha = w_{\max}/w_{\min}$, and H be a known graph with at most Z edges for which*

$$(1 - 1/900) L_G \preceq L_H \preceq (1 + 1/900) L_G.$$

Suppose that, given an $\varepsilon \in (0, 1)$, there is a $\mathcal{T}(n, \varepsilon)$-time algorithm MULTIPLYG(x, ε) *(Algorithm 2) that, given a vector $x \in \mathbb{R}^n$, returns a vector $b \in \mathbb{R}^n$ for which*

$$\|b - L_G x\|_\infty \leq \varepsilon \cdot w_{\min} \cdot \|x\|_\infty.$$

Then, there is an algorithm SOLVEG(b, δ) *that, when given a vector $b \in \mathbb{R}^n$ and $\delta \in (0, 1)$, returns a vector $x \in \mathbb{R}^n$ with*

$$\|x - L_G^\dagger b\|_{L_G} \leq \delta \cdot \|L_G^\dagger b\|_{L_G}.$$

in

$$\widetilde{O}(Z + \mathcal{T}(n, \delta/(n^4 \alpha^2))) \log(Zn\alpha/\delta)$$

time.

4.3 Lower bound for high-dimensional linear system solving

We have shown in this section that if a K graph can be efficiently sparsified, then there is an efficient Laplacian multiplier for K graphs if and only if ther is an efficient Laplacian system solver for K graphs. Here we give one example of how this connection can be used to prove *lower bounds* for Laplacian system solving:

Corollary 4.4.1 (Restatement of Corollary 1.5.1)**.** *Consider a function f that is $(2, o(\log n))$-multiplicatively Lipschitz for which f cannot be ε-approximated by a polynomial of degree at most $o(\log n)$. Then, assuming* SETH*, there is no $\mathrm{poly}(d\log(\alpha n/\varepsilon))n^{1+o(1)}$-time algorithm for ε-approximately solving Laplacian systems in the* K*-graph on n points, where $\mathsf{K}(u,v) = f(\|u-v\|_2^2)$.*

Proof. By Theorem 6.3, there is a $\mathrm{poly}(d\log(\alpha))n^{1+o(1)}$-time algorithm for sparsifying the K-graph on n points. Since sparsification is efficient, Lemma 4.3 implies that the existence of a $\mathrm{poly}(d\log(\alpha n/\varepsilon))n^{1+o(1)}$-time Laplacian solver yields access to a $\mathrm{poly}(d\log(\alpha n/\varepsilon))n^{1+o(1)}$-time Laplacian multiplier. The existence of this multiplier contradicts Theorem 5.8 assuming SETH, as desired. □

5 Matrix-Vector Multiplication

Recall the adjacency and Laplacian matrices of a K graph: For any function $\mathsf{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, and any set $P = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$ of n points, define the matrix $A_{\mathsf{K},P} \in \mathbb{R}^{n\times n}$ by

$$A_{\mathsf{K},P}[i,j] = \begin{cases} \mathsf{K}(x_i,x_j), & \text{if } i \neq j; \\ 0, & \text{if } i = j. \end{cases}$$

Similarly, define the matrix $L_{\mathsf{K},P} \in \mathbb{R}^{n\times n}$ by

$$L_{\mathsf{K},P}[i,j] = \begin{cases} -\mathsf{K}(x_i,x_j), & \text{if } i \neq j; \\ \sum_{a\in[n]\setminus\{i\}} \mathsf{K}(x_i,x_a), & \text{if } i = j. \end{cases}$$

$A_{\mathsf{K},P}$ and $L_{\mathsf{K},P}$ are the adjacency matrix and Laplacian matrix, respectively, of the complete weighted graph on n nodes where the weight between node i and node j is $\mathsf{K}(x_i,x_j)$.

In this section, we study the algorithmic problem of computing the linear transformations defined by these matrices:

Problem 6 (K Adjacency Evaluation)**.** *For a given function $\mathsf{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, the* K *Adjacency Evaluation (*KAdjE*) problem asks: Given as input a set $P = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$ with $|P| = n$ and a vector $y \in \mathbb{R}^n$, compute a vector $b \in \mathbb{R}^n$ such that $\|b - A_{\mathsf{K},P} \cdot y\|_\infty \leq \varepsilon \cdot w_{\max} \cdot \|y\|_\infty$.*

Problem 7 (K Laplacian Evaluation)**.** *For a given function $\mathsf{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, the* K *Laplacian Evaluation (*KLapE*) problem asks: Given as input a set $P = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$ with $|P| = n$ and a vector $y \in \mathbb{R}^n$, compute a vector $b \in \mathbb{R}^n$ such that $\|b - L_{\mathsf{K},P} \cdot y\|_\infty \leq \varepsilon \cdot w_{\max} \cdot \|y\|_\infty$.*

We make a few important notes about these problems:

- In both of the above, problems, $w_{\max} := \max_{u,v\in P} |\mathsf{K}(u,v)|$.
- We assume $\varepsilon = 2^{-\mathrm{polylog} n}$ when it is omitted in the above problems. As discussed in Section 4, this is small enough error so that we can apply such an algorithm for K Laplacian Evaluation to solve Laplacian systems, and furthermore, if we can prove hardness for any such ε, it implies hardness for solving Laplacian systems.
- Note, by Corollary 4.2.1, that when $\varepsilon = 2^{-\mathrm{polylog} n}$, the result of KAdjE is an ε-approximate multiplication of $A_{\mathsf{K},P} \cdot y$ (see Definition 4.1), and the result of KLapE is an ε-approximate multiplication of $L_{\mathsf{K},P} \cdot y$.

- We will also sometimes discuss the f KAdjE and f KLapE problems for a single-input function $f : \mathbb{R} \to \mathbb{R}$. In this case, we implicitly pick $\mathsf{K}(u, v) = f(\|u - v\|_2^2)$.

Suppose the function K can be evaluated in time T (in this chapter we've been assuming $T = \widetilde{O}(1)$). Then, both the KAdjE and KLapE problems can be solved in $O(Tn^2)$ time, by computing all n^2 entries of the matrix and then doing a straightforward matrix-vector multiplication. However, since the input size to the problem is only $O(nd)$ real numbers, we can hope for much faster algorithms when $d = o(n)$. In particular, we will aim for $n^{1+o(1)}$ time algorithms when $d = n^{o(1)}$.

For some functions K, like $\mathsf{K}(x, y) = \|x - y\|_2^2$, we will show that a running time of $n^{1+o(1)}$ is possible for all $d = n^{o(1)}$. For others, like $\mathsf{K}(x, y) = 1/\|x - y\|_2^2$ and $\mathsf{K}(x, y) = \exp(-\|x - y\|_2^2)$, we will show that such an algorithm is only possible when $d \ll \log(n)$. More precisely, for these K:

1. When $d = O(\log(n)/\log\log(n))$, we give an algorithm running in time $n^{1+o(1)}$, and
2. For $d = \Omega(\log n)$, we prove a conditional lower bound showing that $n^{2-o(1)}$ time is necessary.

Finally, for some functions like $\mathsf{K}(x, y) = |\langle x, y\rangle|$, we will show a conditional lower bound showing that $\Omega(n^{2-\delta})$ time is required even when $d = 2^{\Omega(\log^* n)}$ is just barely super-constant.

In fact, assuming SETH, we will characterize the functions f for which the KAdjE and KLapE problems can be efficiently solved in high dimensions $d = \Omega(\log n)$ in terms of the *approximate degree* of f (see subsection 5.2 below). The answer is more complicated in low dimensions $d = o(\log n)$, and for some functions f we make use of the Fast Multipole Method to design efficient algorithms (in fact, we will see that the Fast Multipole Method solves a problem equivalent to our KAdjE problem).

5.1 Equivalence between Adjacency and Laplacian Evaluation

Although our goal in this section is to study the K Laplacian Evaluation problem, it will make the details easier to instead look at the K Adjacency Evaluation problem. Here we show that any running time achievable for one of the two problems can also be achieved for the other (up to a $\log n$ factor), and so it will be sufficient in the rest of this section to only give algorithms and lower bounds for the K Adjacency Evaluation problem.

Proposition 5.1. *Suppose the* KAdjE *(Problem 6) can be solved in $\mathcal{T}(n, d, \varepsilon)$ time. Then, the* KLapE *(Problem 7) can be solved in $O(\mathcal{T}(n, d, \varepsilon/2))$ time.*

Proof. We use the K Adjacency Evaluation algorithm with error $\varepsilon/2$ twice, to compute $s := A_{\mathsf{K},P} \cdot y$ and $g := A_{\mathsf{K},P} \cdot \vec{1}$, where $\vec{1}$ is the all-1s vector of length n. We then output the vector $z \in \mathbb{R}^n$ given by $z_i = g_i \cdot y_i - s_i$, which can be computed in $O(n) = O(\mathcal{T}(n, d, \varepsilon/2))$ time. □

Proposition 5.2. *Suppose the* KLapE *(Problem 7) can be solved in $\mathcal{T}(n, d, \varepsilon)$ time, and that $\mathcal{T}$ satisfies $\mathcal{T}(n_1 + n_2, d, \varepsilon) \geq \mathcal{T}(n_1, d, \varepsilon) + \mathcal{T}(n_2, d, \varepsilon)$ for all n_1, n_2, d, ε. Then, the* KAdjE *(Problem 6) can be solved in $O(\mathcal{T}(n \log n, d, 0.5\varepsilon/\log n))$ time.*

Proof. We will show that the K Adjacency Evaluation problem can be solved in

$$\sum_{i=0}^{\log n} O(2^i \cdot \mathcal{T}(n/2^i, d, 0.5\varepsilon/\log n))$$

time, and then apply the superadditive identity for T to get the final running time. For a fixed d, we proceed by strong induction on n, and assume the K Adjacency Evaluation problem can be solved in this running time for all smaller values of n.

Let $a', a'' \in \mathbb{R}^n$ be the vectors given by $a'_i = y_i$ and $a''_i = 0$ when $i \in [n/2]$, and $a'_i = 0$ and $a''_i = y_i$ when $i > n/2$.

We first compute $z' \in \mathbb{R}^n$ and $z'' \in \mathbb{R}^n$ as follows:

$$z' := L_{\mathsf{K},P} \cdot a' \quad \text{and} \quad z'' := L_{\mathsf{K},P} \cdot a''$$

in $O(\mathcal{T}(n, d, 0.5\varepsilon/\log n))$ time.

Next, let $y', y'' \in \mathbb{R}^{n/2}$ be the vectors given by $y'_i = y_i$ and $y''_i = y_{n/2+i}$ for all $i \in [n/2]$, and let $P', P'' \subseteq \mathbb{R}^d$ be given by $P' = \{x_1, \ldots, x_{n/2}\}$ and $P'' = \{x_{n/2+1}, \ldots, x_n\}$.

We recursively compute $r', r'' \in \mathbb{R}^{n/2}$ given by

$$r' = L_{\mathsf{K},P'} \cdot y' \quad \text{and} \quad r'' = L_{\mathsf{K},P''} \cdot y''.$$

Finally, we can output the vector $z \in \mathbb{R}^n$ given by $z_i = z''_i + r'_i$ and $z_{n/2+i} = z'_{n/2+i} + r''_i$ for all $i \in [n/2]$. Each of the two recursive calls took time

$$\sum_{i=1}^{\log_2(n)} O(2^{i-1} \cdot \mathcal{T}(n/2^i, d, 0.5\varepsilon/\log n)),$$

and our two initial calls took time $O(\mathcal{T}(n, d, 0.5\varepsilon/\log n))$, leading to the desired running time. Each output entry is ultimately the sum of at most $2\log n$ terms from calls to the given algorithm, and hence has error ε (since we perform all recursive calls with error $0.5\varepsilon/\log n$ and the additive error guarantees in recursive calls can only be more stringent). □

Remark 5.2.1. *In both Proposition 5.1 and Proposition 5.2, if the input to the* KLapE *(resp.* KAdjE*) problem is a* $\{0,1\}$ *vector, then we only apply the given* KAdjE *(*KLapE*) algorithm on* $\{0,1\}$ *vectors. Hence, the two problems are equivalent even in the special case where the input vector* y *must be a* $\{0,1\}$ *vector.*

5.2 Approximate Degree

We will see in this section that the key property of a function $f : \mathbb{R} \to \mathbb{R}$ for determining whether f KAdjE is easy or hard is how well it can be approximated by a low-degree polynomial.

Definition 5.3. *For a positive integer* k *and a positive real number* $\varepsilon > 0$*, we say a function* $f : [0,1] \to \mathbb{R}$ *is* ε-close to a polynomial of degree k *if there is a polynomial* $p : [0,1] \to \mathbb{R}$ *of degree at most* k *such that, for every* $x \in [0,1]$*, we have* $|f(x) - p(x)| \leq \varepsilon$*.*

The Stone-Weierstrass theorem says that, for any $\varepsilon > 0$, and any continuous function f which is bounded on $[0, 1]$, there is a positive integer k such that f is ε-close to a polynomial of degree k. That said, k can be quite large for some natural and important continuous functions f. For some examples:

Example 5.4. *For the function $f(x) = 1/(1 + x)$, we have $f(x) = \sum_{\ell=0}^{\infty}(-1)^\ell x^\ell$ for all $x \in [0, 1)$. Truncating this series to degree $O(\log(1/\varepsilon))$ gives a ε approximation on the interval $[0, 1/2]$. The following proposition shows that this is optimal up to constant factors.*

Proposition 5.5. *Any polynomial $p(x)$ such that $|p(x) - 1/(1 + x)| \leq \varepsilon$ for all $x \in [0, 1/2]$ has degree at least $\Omega(\log(1/\varepsilon))$.*

Proof. For such a polynomial $p(x)$, define $q(x) := 1 - x \cdot p(x - 1)$. Thus, the polynomial q has the two properties that $|q(x)| < \varepsilon$ for all $x \in [1, 3/2]$, and $q(0) = 1$. By standard properties of the Chebyshev polynomials (see e.g. [246, Proposition 2.4]), the polynomial q with those two properties of minimum degree is an appropriately scaled and shifted Chebyshev polynomial, which requires degree $\Omega(\log(1/\varepsilon))$. □

Example 5.6. *For the function $f(x) = e^{-x}$, we have $f(x) = \sum_{\ell=0}^{\infty}(-1)^\ell x^\ell/\ell!$ for all $x \in \mathbb{R}_+$. Truncating this series to degree $O(\log(1/\varepsilon)/\log\log(1/\varepsilon))$ gives a ε approximation on any interval $[0, a]$ for constant $a > 0$. Such a dependence is believed to be optimal, and is known to be optimal if we must approximate $f(x)$ on the slightly larger interval $[0, \log^2(1/\varepsilon)/\log^2\log(1/\varepsilon)]$ [246, Section 5].*

In both of the above settings, for error $\varepsilon = n^{-\Omega(\log^4 n)}$, the function f is only ε-close to a polynomial of degree $\omega(\log n)$. We will see in Theorem 5.8 below that this implies that, for each of these functions f, the ε-approximate f KAdjE problem in dimension $d = \Omega(\log n)$ requires time $n^{2-o(1)}$ assuming SETH.

5.3 'Kernel Method' Algorithms

Lemma 5.7. *For any integer $q \geq 0$, let $\mathsf{K}(u, v) = (\|u - v\|_2^2)^q$. The* KAdjE *problem (Problem 6) can be solved exactly (with 0 error) in time $\widetilde{O}(n \cdot \binom{2d+2q-1}{2q})$.*

Proof. The function

$$\mathsf{K}(u, v) = \left(\sum_{i=1}^{d}(u_i - v_i)^2\right)^q$$

is a homogeneous polynomial of degree $2q$ in the variables $u_1, \ldots, u_d, v_1, \ldots, v_d$. Let

$$V = \{u_1, \ldots, u_d, v_1, \ldots, v_d\},$$

and let T be the set of functions $t : V \to \{0, 1, 2, \ldots 2q\}$ such that $\sum_{v \in V} t(v) = 2q$.

We can count that

$$|T| = \binom{2d + 2q - 1}{2q}.$$

Hence, there are coefficients $c_t \in \mathbb{R}$ for each $t \in T$ such that

$$\mathsf{K}(u, v) = \sum_{t \in T} c_t \cdot \prod_{v \in V} v^{t(v)}. \tag{5.3}$$

Let $V_u = \{u_1, \ldots, u_d\}$ and $V_v = V \setminus V_u$. Define $\phi_u : \mathbb{R}^d \to \mathbb{R}^{|T|}$ by, for $t \in T$,

$$\phi_u(u_1, \ldots, u_d)_t = c_t \cdot \prod_{u_i \in V_u} {u_i}^{t(u_i)}.$$

Similarly define $\phi_v : \mathbb{R}^d \to \mathbb{R}^{|T|}$ by, for $t \in T$,

$$\phi_v(v_1, \ldots, v_d)_t = \prod_{v_i \in V_v} {v_i}^{t(v_i)}.$$

It follows from (5.3) that, for all $u, v \in \mathbb{R}^d$, we have $\mathsf{K}(u, v) = \langle \phi_u(u), \phi_v(v) \rangle$.

Our algorithm thus constructs the matrix $M_u \in \mathbb{R}^{n \times |T|}$ whose rows are the vectors $\phi_u(x_i)$ for $i \in [n]$, and the matrix $M_v \in \mathbb{R}^{|T| \times n}$ whose columns are the vectors $\phi_v(x_i)$ for $i \in [n]$. Then, on input $y \in \mathbb{R}^n$, it computes $y' := M_v \cdot y \in \mathbb{R}^{|T|}$ in $\widetilde{O}(n \cdot |T|)$ time, then $z := M_u \cdot y' \in \mathbb{R}^n$, again in $\widetilde{O}(n \cdot |T|)$ time, and outputs z. Since $M_u \cdot M_v = A_{\mathsf{K},\{x_1,\ldots,x_n\}}$, it follows that the vector we output is the desired $z = A_{\mathsf{K},\{x_1,\ldots,x_n\}} \cdot y$. □

Remark 5.7.1. *The running time in Lemma 5.7 can be improved to $\widetilde{O}(n \cdot \binom{d+q-1}{q})$ with more careful work, by noting that each monomial has either 'x-degree' or 'y-degree' at most d, but we omit this here since the difference is negligible for our parameters of interest.*

Corollary 5.7.1. *Let q, d be positive integers which may be functions of n, such that $\binom{2(d+q)}{2q} < n^{o(1)}$. For example:*

- *when $d = o(\log n / \log\log n)$ and $q \leq \mathrm{poly}(\log n)$, or*
- *when $d = o(\log n)$ and $q \leq O(\log n)$, or*
- *when $d = \Theta(\log n)$ and $q < o(\log n)$.*

If $f : \mathbb{R} \to \mathbb{R}$ is a polynomial of degree at most q, and we define $\mathsf{K}(u, v) := f(\|u - v\|_2^2)$, then the KAdjE *problem in dimension d can be solved exactly in $n^{1+o(1)}$ time.*

Proof. This follows by applying Lemma 5.7 separately to each monomial of f, and summing the results.

When $d = o(\log n)$ and $q \leq O(\log n)$, then we can write $d = \frac{1}{a(n)} \log n$ for some $a(n) =$

$\omega(1)$, and $q = b(n) \cdot \log n$ for some $b(n) = O(1)$. It follows that

$$\begin{aligned}
\binom{2(d+q)}{2q} &= \binom{2(d+q)}{2d} \\
&= \binom{O(q)}{O(d)} && \text{by } d = O(q) \\
&\leq O(q/d)^{O(d)} \\
&= 2^{O(d \log(q/d))} \\
&= 2^{O(\frac{\log(ab)}{a}) \cdot \log n} && \text{by } d = \frac{\log n}{a}, q = b \log n \\
&\leq 2^{O(\frac{\log(a)}{a}) \cdot \log n} && \text{by } b = O(1) \\
&< n^{o(1)}. && \text{by } a = \omega(1)
\end{aligned}$$

The other cases are similar. □

Corollary 5.7.2. *Suppose $f : \mathbb{R} \to \mathbb{R}$ is ε/n-close to a polynomial of degree q (Definition 5.3), where q, d are positive integers such that $\binom{2(d+q)}{2q} < n^{o(1)}$ (such as the parameter setting examples in Corollary 5.7.1), and define $\mathsf{K}(u, v) := f(\|u - v\|_2^2)$. Then, the* KAdjE *problem in dimension d can be solved ε-approximately in $n^{1+o(1)}$ time.*

Proof. Apply Corollary 5.7.1 for the degree q approximation of f. □

5.4 Lower Bound in High Dimensions

We now prove that in the high dimensional setting, where $d = \Theta(\log n)$, the algorithm from Corollary 5.7.2 is essentially tight. In that algorithm, we showed that (recalling Definition 5.3) functions f which are ε-close to a polynomial of degree $o(\log n)$ have efficient algorithms; here we show a lower bound if f is not ε-close to a polynomial of degree $O(\log n)$.

Theorem 5.8. *Let $f : [0, 1] \to \mathbb{R}$ be an analytic function on $[0, 1]$ and let $\kappa : \mathbb{N} \to [0, 1]$ be a nonincreasing function. Suppose that, for infinitely many positive integers k, f is not $\kappa(k)$-close to a polynomial of degree k.*

Then, assuming SETH*, the* KAdjE *problem for $\mathsf{K}(x, y) = f(\|x - y\|_2^2)$ in dimension d and error $(\kappa(d + 1))^{O(d^4)}$ on $n = 1.01^d$ points requires time $n^{2-o(1)}$.*

This theorem will be a corollary of another result, which is simpler to use in proving lower bounds:

Theorem 5.9. *Let $f : [0, 1] \to \mathbb{R}$ be an analytic function on $[0, 1]$ and let $\kappa : \mathbb{N} \to [0, 1]$ be a nonincreasing function. Suppose that, for infinitely many positive integers k, there exists an $x_k \in [0, 1]$ for which $|f^{(k+1)}(x_k)| > \kappa(k)$.*

Then, assuming SETH*, the* KAdjE *problem for $\mathsf{K}(x, y) = f(\|x - y\|_2^2)$ in dimension d and error $(\kappa(d + 1))^{O(d^4)}$ on $n = 1.01^d$ points requires time $n^{2-o(1)}$.*

To better understand Theorem 5.8 in the context of our dichotomy, think about the following example:

Example 5.10. *Consider a function f that is exactly $\kappa(k) = 2^{-k^3}$-far from the closest polynomial of degree k for every $k \in \mathbb{N}$. By Corollary 5.7.2, there is an $d^{\log^{1/3} n} n < n^{1+o(1)}$-time algorithm for $1/poly(n)$-approximate adjacency matrix multiplication on an n-vertex f-graph when $d = \Theta(logn)$. In fact, there is an algorithm even for $2^{-o(\log^3 n)}$-error that takes $n^{1+o(1)}$. However, by Theorem 5.8, there is no $n^{2-o(1)}$-time algorithm for $2^{-d^3 \cdot d^4} = 2^{-\Theta(\log^7 n)}$-approximate multiplication.*

We now give a more concrete version of the proof outline described in the introduction. To prove Theorem 5.8 given Theorem 5.9, it suffices to show that for any function that is far from a degree k polynomial, there exists a point with high $(k+1)$-th derivative (Lemma 5.13). To prove Theorem 5.9, we start by showing that there exists an interval (not just a single point) with high $(k+1)$-th derivative (Lemma 5.14). This is done by integrating over the $(k+2)$-nd derivative, exploiting the fact that it is bounded for analytic functions (Proposition 5.12). Then, we further improve this derivative lower bound by showing that there is an interval on which *all* i-th derivatives for $i \leq k+1$ are bounded from below (Lemma 5.15). This is done by induction, deriving a bound for i-th derivatives by integrating over the $(i+1)$-th derivative. The lower bound on the $(i+1)$-th derivative ensures that it can only be close to 0 at a small interval around one point, so picking an interval far from that point suffices for the inductive step.

Up to this point, we have argued that there must be an interval $I \subset [0,1]$ on which all of f's $\leq (k+1)$-derivatives are large in absolute value (Lemma 5.15). We exploit this property to solve an exact bichromatic nearest neighbors problem in Hamming distance in $d = \Theta(\log n)$ dimensions (Lemma 5.19). Since even approximate nearest neighbors cannot be solved in $n^{2-\delta}$-time for $\delta > 0$ assuming SETH (Theorem 3.17), this suffices. To solve Hamming nearest neighbors a a pair of sets S and T with $|S| = |T| = n$, we set up $d+1$ different f-graph adjacency matrix multiplication problems. In problem i, we scale the points in S and T by a factor of ζi for some $\zeta 0$ and translate them by $c \in [0,1]$ so that they are in the interval I. Then, with one adjacency multiplication, one can evaluate an expression Z_i, where $Z_i = \sum_{x \in S, y \in T} f(c + i^2\zeta^2 \|x-y\|_2^2)$. For each distance $i \in [d]$, let $u_j = |\{x \in S, y \in T : \|x-y\|_2^2 = j\}|$. To solve bichromatic nearest neighbors, it suffices to compute all of the u_js. This can be done by setting up a linear system in the u_js, where there is one equation for each Z_i. The matrix for this linear system has high determinant because f has high $\leq (k+1)$-th derivatives on I (Lemma 5.17). Cramer's Rule can be used to bound the error in our estimate of the u_js that comes from the error in the multiplication oracle (Lemma 5.18). Therefore, $O(d)$ calls to a multiplication oracle suffices for computing the number of pairs of vertices in $S \times T$ that are at each distance value. Returning the minimum distance i for which $u_i > 0$ solves bichromatic nearest neighbors, as desired.

In Lemma 5.17, we will make use of the Cauchy-Binet formula:

Lemma 5.11 (Cauchy-Binet formula for infinite matrices)**.** *Let k be a positive integer, and for functions $A : [k] \times \mathbb{N} \to \mathbb{R}$ and $B : \mathbb{N} \times [k] \to \mathbb{R}$, define the matrix $C \in \mathbb{R}^{k \times k}$ by, for $i, j \in [k]$,*

$$C_{ij} := \sum_{\ell=0}^{\infty} A_{i\ell} \cdot B_{\ell j},$$

and suppose that the sum defining C_{ij} converges absolutely for all i, j. Then,

$$\det(C) = \sum_{1 \leq \ell_1 < \ell_2 < \cdots < \ell_k} \det(A[\ell_1, \ell_2, \cdots, \ell_k]) \cdot \det(B[\ell_1, \ell_2, \cdots, \ell_k]),$$

where $A[\ell_1, \ell_2, \cdots, \ell_k] \in \mathbb{R}^{k\times k}$ denotes the matrix whose i, j entry is given by $A_{i\ell_j}$ and $B[\ell_1, \ell_2, \cdots, \ell_k]$ denotes the matrix whose i, j entry is given by $B_{\ell_i j}$.

In this section, we exploit the following property of analytic functions:

Proposition 5.12. *Consider a function $f : [0, 1] \to \mathbb{R}$ that is analytic. Then, there is a constant $B > 0$ depending on f such that for all $k \geq 0$ and all $x \in [0, 1]$, $|f^{(k)}(x)| < B4^k k!$*

Proof. We first show that, for any $x \in [0, 1]$, $|f^{(k)}(x)| < B_x 2^k k!$ for some constant B_x depending on x. Write f's Taylor expansion around x:

$$f(y) = \sum_{i=0}^{\infty} f^{(i)}(x)(y - x)^i/i!.$$

Let $y_0 = \arg\max_{a\in\{0,1\}} |a - x|$. Note that $|y_0 - x| \geq 1/2$. Since $f(y_0)$ is a convergent series, there exists a constant N_x dependent on x such that for all $i > N_x$, the absolute value of the i-th term of the series for $f(y_0)$ is at most 1/2. Therefore, for all $i > N_x$, $|f^{(i)}(x)| < 2(2^i)i!$. For all $i \leq N_x$, $f^{(i)}(x)$ is a constant depending on x, so we are done with this part.

Next, we show that $|f^{(k)}(x)| < B4^k k!$ for all $x \in [0, 1]$ and some constant B depending only on f. Let $x_0 \in \{i/8\}_{i=0}^{8}$ be the point that minimizes $|x - x_0|$. Note that $|x - x_0| < 1/16$. Taylor expand f around x_0:

$$f(x) = \sum_{i=0}^{\infty} f^{(i)}(x_0)(x - x_0)^i/i!.$$

Take derivatives for some $k \geq 0$ and use the triangle inequality:

$$|f^{(k)}(x)| \leq \sum_{i=0}^{\infty} |f^{(i+k)}(x_0)||x - x_0|^i/i!$$

By the first part, $|f^{(i+k)}(x_0)| \leq B_{x_0} 2^{i+k}(i + k)!$, so

$$|f^{(k)}(x)| \leq \sum_{i=0}^{\infty} B_{x_0} 2^{i+k}((i + k)!/i!)(1/16)^{(i+k)}$$

Note that $(i + k)!/i! \leq (2i)^k$ for $i > k$ (we are done for $i \leq k$). There is some constant C for which $\sum_{i=0}^{\infty} i^k 4^{-i} = C$, so letting $B = B_{x_0} C$ suffices, as desired. □

We now move on to proving the main results of this section, which consists of several steps.

Lemma 5.13 (Step 1: high $(k + 1)$-derivative)**.** *Let $f : [0, 1] \to \mathbb{R}$ be an analytic function on $[0, 1]$ and let $\kappa : \mathbb{N} \to [0, 1]$ be a nonincreasing function. Suppose that, for some positive integer k, f is not $\kappa(k)$-close to a polynomial of degree k. Then, there exists some $x \in [0, 1]$ for which $|f^{(k+1)}(x)| > \kappa(k)$.*

Proof. Define the function $g : [0, 1] \to \mathbb{R}$ by $g(x) = f(x) - \sum_{\ell=0}^{k} \frac{f^{(\ell)}(0)\cdot x^\ell}{\ell!}$. We claim that, for each $i \in \{0, 1, \ldots, k + 1\}$, there is an $x_i \in [0, 1]$ such that $|g^{(i)}(x_i)| > \kappa(k)$. Since $g^{(k+1)}(x) =$

$f^{(k+1)}(x)$, plugging in $i = k+1$ into this will imply our desired result. We prove this by induction on i.

For the base case $i = 0$: note that g is the difference between f and a polynomial of degree k, and so by our assumption that f is not $\kappa(k)$-close to a polynomial of degree k, there must be an $x_0 \in [0,1]$ such that $|g(x_0)| > \kappa(k)$.

For the inductive step, consider an integer $i \in [k+1]$. Notice that $g^{(i-1)}(0) = 0$ by definition of g since $i \leq k+1$. By the inductive hypothesis, there is an $x_{i-1} \in [0,1]$ with $|g^{(i-1)}(x_{i-1})| > \kappa(k)$. Hence, by the mean value theorem, there must be an $x_i \in [0, x_{i-1}]$ such that

$$|g^{(i)}(x_i)| \geq |g^{(i-1)}(x_{i-1}) - g^{(i-1)}(0)|/x_{i-1} \geq |g^{(i-1)}(x_{i-1})| > \kappa(k),$$

as desired. □

Proof of Theorem 5.8 given Theorem 5.9. For each of the infinitely many ks for which f is $\kappa(k)$-far from a degree k polynomial, Lemma 5.13 implies the existance of an x_k for which $|f^{(k+1)}(x_k)| > \kappa(x_k)$. Thus, f satisfies the input condition of Theorem 5.9, so applying Theorem 5.9 proves Theorem 5.8 as desired. □

Now, we focus on Theorem 5.9:

Lemma 5.14 (Step 2: high $(k+1)$-derivative on interval)**.** *Let $f : [0,1] \to \mathbb{R}$ be an analytic function on $[0,1]$ and let $\kappa : \mathbb{N} \to [0,1]$ be a nonincreasing function. There is a constant $B > 0$ depending only on f such that the following holds.*

Suppose that, for some sufficiently large positive integer k, there is an $x \in [0,1]$ for which $|f^{(k+1)}(x)| > \kappa(k)$. Then, there exists an interval $[a,b] \subseteq [0,1]$ with the property that both $b - a \geq \kappa(k)/(32Bk4^k \cdot k!)$ and, for all $y \in [a,b]$, $|f^{(k+1)}(y)| > \kappa(k)/2$.

Proof. Since f is analytic on $[0,1]$, Proposition 5.12 applies and it follows that there is a constant $B > 0$ dependent on f such that for every $y \in [0,1]$ and every nonnegative integer m, we have $|f^{(m)}(y)| \leq Bm4^m \cdot m!$. In particular, for all $y \in [0,1]$, we have $|f^{(k+2)}(y)| \leq 16Bk4^k \cdot k!$. Let $\delta = \kappa(k)/(32Bk4^k \cdot k!)$, then let $a = \max\{0, x - \delta\}$, and $b = \min\{1, x + \delta\}$. We have $b - a \geq \delta = \kappa(k)/(32Bk4^k \cdot k!)$, since when k is large enough, we get that $\delta < 1/2$, so we cannot have both $a = 0$ and $b = 1$. Meanwhile, for any $y \in [a,b]$, we have as desired that

$$\begin{aligned} |f^{(k+1)}(y)| &\geq |f^{(k+1)}(x)| - |x - y| \cdot \sup_{y' \in [a,b]} |f^{(k+2)}(y')| \\ &> \kappa(k) - \delta \cdot 16Bk4^k \cdot k! \\ &= \kappa(k)/2. \end{aligned}$$

□

Lemma 5.15 (Step 3: high lower derivatives on subintervals)**.** *Let $f : [0,1] \to \mathbb{R}$ be an analytic function on $[0,1]$ and let $\kappa : \mathbb{N} \to [0,1]$ be a nonincreasing function. There is a constant $B > 1$ depending only on f such that the following holds.*

Suppose that, for some sufficiently large positive integer k, there exists an $x \in [0,1]$ for which $|f^{(k+1)}(x)| > \kappa(k)$. Then, there exists an interval $[c,d] \subseteq [0,1]$ with the property that both $d - c > \kappa(k)/(128Bk4^{2k+1} \cdot k!)$ and, for all $y \in [c,d]$ and all $i \leq k+1$, $|f^{(i)}(y)| > \left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+2-i}$.

Proof. We will prove that, for all integers $0 \leq i \leq k+1$, there is an interval $[c_i, d_i] \subseteq [0,1]$ such that $d_i - c_i > \kappa(k)/(32Bk4^{2k+1-i} \cdot k!)$, and for all integers $i \leq i' \leq k+1$ and all $y \in [c_i, d_i]$ we have $|f^{(i')}(y)| > \left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+2-i'}$. Plugging in $i = 0$ gives the desired statement. We will prove this by induction on i, from $i = k+1$ to $i = 0$. The base case $i = k+1$ is given (with slightly better parameters) by Lemma 5.14.

For the inductive step, suppose the statement is true for $i+1$. We will pick $[c_i, d_i]$ to be a subinterval of $[c_{i+1}, d_{i+1}]$, so the inductive hypothesis says that for every $y \in [c_i, d_i]$ and every integer $i < i' \leq k+1$ we have $|f^{(i')}(y)| > \left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+2-i'}$. It thus remains to show that we can further pick c_i and d_i such that $d_i - c_i \geq \frac{1}{4}(d_{i+1} - c_{i+1})$ and $|f^{(i)}(y)| > \left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+2-i}$ for all $y \in [c_i, d_i]$.

Recall that $|f^{(i+1)}(y)| > \left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+1-i}$ for all $y \in [c_{i+1}, d_{i+1}]$. Since $f^{(i+1)}$ is continuous, we must have

- either $f^{(i+1)}(y) > \left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+1-i}$ for all such y,
- or $\quad -f^{(i+1)}(y) > \left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+1-i}$ for all such y.

Let us assume we are in the first case; the second case is nearly identical. Let $\delta = (d_{i+1} - c_{i+1})/4$, and consider the four subintervals

$$[c_{i+1}, c_{i+1} + \delta], \quad [c_{i+1} + \delta, c_{i+1} + 2\delta], \quad [c_{i+1} + 2\delta, c_{i+1} + 3\delta], \quad \text{and} \quad [c_{i+1} + 3\delta, c_{i+1} + 4\delta].$$

Since $f^{(i+1)}(y) > \left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+1-i}$ for all y in each of those intervals, we know that for each of the intervals, letting c' denote its left endpoint and d' denote its right endpoint, we have

$$\begin{aligned} f^{(i)}(d') - f^{(i)}(c') &\geq \delta \cdot \left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+1-i} \\ &\geq \left[\kappa(k)/(32Bk4^{2k+1-i} \cdot k!)\right] \cdot \left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+1-i}/4 \\ &\geq \left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+2-i}. \end{aligned}$$

In particular, $f^{(i)}$ is increasing on the interval $[c_{i+1}, d_{i+1}]$, and if we look at the five points $y = c_{i+1} + a \cdot \delta$ for $a \in \{0,1,2,3,4\}$ which form the endpoints of our four subintervals, $f^{(i)}$ increases by more than $\left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+2-i}$ from each to the next. It follows by a simple case analysis (on where in our interval $f^{(i)}$ has a root) that there must be one of our four subintervals with $|f^{(i)}(y)| > \left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+2-i}$ for all y in the subinterval. We can pick that subinterval as desired. □

To simplify notation in the rest of the proof, we will let $\rho(k) = \left[\kappa(k)^2/(64Bk4^{2k+1} \cdot k!)\right]^{k+2}$. We now use these properties of f to reason about a certain matrix connected to f that can be used to count the number of pairs of points at each distance.

Definition 5.16 (Counting matrix)**.** *For a function $f : \mathbb{R} \to \mathbb{R}$, an integer $k \geq 1$, and a function $\rho : \mathbb{N} \to \mathbb{R}_{>0}$, let $[c, d]$ be the interval from Lemma 5.15. Define the* counting matrix *be the $k \times k$ matrix M for which*

$$M_{ij} = f(c + (\rho(k)/(B(200k)^k))^{10k} \cdot i \cdot j/k^2).$$

Lemma 5.17 (Step 4: determinant lower bound for functions f that are far from polynomials using Cauchy-Binet)**.** *Let $f : [0,1] \to \mathbb{R}$ be an analytic function on $[0,1]$, let $\kappa : \mathbb{N} \to [0,1]$ be a nonincreasing function, and let $\rho(\ell) = \left[\kappa(\ell)^2/(64B\ell 4^{2\ell+1} \cdot \ell!)\right]^{\ell+2}$ (as discussed before). Let k be an integer for which there exists an $x \in [0,1]$ with $|f^{(k+1)}(x)| > \kappa(k)$.*

Let M be the counting matrix (Definition 5.16) for f, k, and ρ. Then

$$|\det(M)| > (\rho(k))^k (\rho(k)/(Bk^2(200k)^k))^{10k^3}.$$

Proof. Since f is analytic on $[c,d]$, we can Taylor expand it around c:

$$f(x) = \sum_{\ell=0}^{\infty} \frac{f^{(\ell)}(c)}{\ell!}(x-c)^\ell$$

Let $\delta = (\rho(k)/(B(200k)^k))^{10k}/k^2$. Note that for all values of $i, j \in [k]$, the input to f in M_{ij} is in the interval $[c,d]$ by the lower bound on $d-c$ in Lemma 5.15. In particular, for all $i, j \in [k]$,

$$M_{ij} = \sum_{\ell=0}^{\infty} \frac{f^{(\ell)}(c)}{\ell!}\delta^\ell i^\ell j^\ell$$

Define two infinite matrices $A : [k] \times \mathbb{Z}_{\geq 0} \to \mathbb{R}$ and $C : \mathbb{Z}_{\geq 0} \times [k] \to \mathbb{R}$ as follows:

$$A_{i\ell} = \frac{f^{(\ell)}(c)}{\ell!}\delta^\ell i^\ell$$

$$C_{\ell j} = j^\ell$$

for all $i, j \in [k]$ and $\ell \in \mathbb{Z}_{\geq 0}$. Then

$$M_{ij} = \sum_{\ell=0}^{\infty} A_{i\ell} C_{\ell j}$$

for all $i, j \in [k]$ and converges, so we may apply Lemma 5.11. By Lemma 5.11,

$$\det(M) = \sum_{0 \leq \ell_1 < \ell_2 < \ldots < \ell_k} \det(A[\ell_1, \ell_2, \ldots, \ell_k]) \det(C[\ell_1, \ell_2, \ldots, \ell_k])$$

To lower bound $|\det(M)|$, we

(1) lower bound the contribution of the term for the tuple $(\ell_1, \ell_2, \ldots, \ell_k) = (0, 1, \ldots, k-1)$,
(2) upper bound the contribution of every other term,
(3) show that the lower bound dominates the sum.

We start with part (1). Let D and P be $k \times k$ matrices, with D diagonal, $D_{\ell\ell} = \frac{f^{(\ell)}(c)\delta^\ell}{\ell!}$, and $P_{i\ell} = i^\ell$ for all $\ell \in \{0, 1, \ldots, k\}$ and $i \in [k]$. Then $A[0, 1, \ldots, k-1] = PD$, which means that

$$\det(A[0, 1, \ldots, k-1]) = \det(P) \cdot \det(D)$$

P and $C[0,1,\ldots,k-1]$ are Vandermonde matrices, so their determinants has a closed form and, in particular, have the property that $|\det(P)| \geq 1$ and $|\det(C[0,1,\ldots,k-1])| \geq 1$. By Lemma 5.15, $|D_{\ell\ell}| > \delta^\ell \rho(\ell) \geq \delta^\ell \rho(k)$ for all $\ell \in \{0,1\ldots,k-1\}$. Therefore,

$$\begin{aligned}|\det(A[0,1,\ldots,k-1])| \cdot |\det(C[0,1,\ldots,k-1])| &> \delta^{1+2+\ldots+(k-1)}\rho(k)^k \\ &= \delta^{\binom{k}{2}}\rho(k)^k.\end{aligned}$$

This completes part (1). Next, we do part (2). Consider a k-tuple $0 \leq \ell_1 < \ell_2 < \ldots < \ell_k$ and a permutation $\sigma : [k] \to [k]$. By Proposition 5.12, there is some constant $B > 0$ depending on f for which $|f^{(\ell)}(c)| \leq B10^\ell(\ell!) \leq B(10\ell)^\ell$ for all ℓ. Therefore,

$$\left|\prod_{i=1}^k A_{i\ell_{\sigma(i)}}\right| \leq B^k(10\delta k)^{\sum_{i=1}^k \ell_i}$$

We also get that

$$\left|\prod_{j=1}^k C_{\ell_{\sigma(j)}j}\right| \leq k^{\sum_{j=1}^k \ell_j}$$

Summing over all $k!$ permutations σ yields an upper bound on the determinants of the blocks of A and C, excluding the top block:

$$\begin{aligned}&\sum_{0\leq\ell_1<\ell_2<\ldots<\ell_k,\ell_k\neq k-1} |\det(A[\ell_1,\ell_2,\ldots,\ell_k])||\det(C[\ell_1,\ell_2,\ldots,\ell_k])| \\ \leq &\sum_{0\leq\ell_1<\ell_2<\ldots<\ell_k,\ell_k\neq k-1} (k!)^2B^k(10\delta k^2)^{\sum_{i=1}^k \ell_i} \\ \leq &\sum_{\tau=1+2+\ldots+(k-3)+(k-2)+k}^{\infty} \tau^k(k!)^2B^k(10\delta k^2)^\tau \\ \leq &\, 2\tau_0^k(k!)^2B^k(10\delta k^2)^{\tau_0},\end{aligned}$$

where $\tau_0 = \binom{k}{2} + 1$. This completes part (2). Now, we do part (3). By Lemma 5.11,

$$\begin{aligned}|\det(M)| \geq &|\det(A[0,1,\ldots,k-1])||\det(C[0,1,\ldots,k-1])| \\ &- \sum_{0\leq\ell_1<\ell_2<\ldots<\ell_k,\ell_k\neq k-1} |\det(A[\ell_1,\ldots,\ell_k])||\det(C[\ell_1,\ldots,\ell_k])|\end{aligned}$$

Plugging in the part (1) lower bound and the part (2) upper bound yields

$$\begin{aligned}|\det(M)| &\geq \delta^{\tau_0-1}\rho(k)^k - 2\tau_0^k(k!)^2B^k(10\delta k^2)^{\tau_0} \\ &= \delta^{\tau_0-1}\left(\rho(k)^k - 2\delta\tau_0^k(k!)^2B^k(10k^2)^{\tau_0}\right) \\ &> \delta^{\tau_0-1}\rho(k)^k/2 \\ &> \rho(k)^k(\rho(k)/(Bk^2(200k)^k))^{10k^3}\end{aligned}$$

as desired. □

Lemma 5.18 (Step 5: Cramer's rule-based bound on error in linear system)**.** *Let M be an invertible k by k matrix with $|M_{ij}| \leq B$ for all $i, j \in [k]$. Let b be a k-dimensional vector for which $|b_i| \leq \varepsilon$ for all $i \in [k]$. Then, $\|M^{-1}b\|_\infty \leq \varepsilon k! B^k / |\det(M)|$.*

Proof. Cramer's rule says that, for each $i \in [k]$, the entry i of the vector $M^{-1}b$ is given by

$$(M^{-1}b)_i = \frac{\det(M_i)}{\det(M)},$$

where M_i is the matrix which one gets by replacing column i of M by b. Let us upper-bound $|\det(M_i)|$. We are given that each entry of M_i in column i has magnitude at most ε, and each entry in every other column has magnitude at most B. Hence, for any permutation $\sigma \in S_k$ on $[k]$, we have

$$\left| \prod_{j=1}^{k} (M_i)_{j,\sigma(j)} \right| \leq \varepsilon \cdot B^{k-1},$$

and so

$$|\det(M_i)| \leq \sum_{\sigma \in S_k} \left| \prod_{j=1}^{k} (M_i)_{j,\sigma(j)} \right| \leq \varepsilon \cdot B^{k-1} \cdot k!.$$

It follows from Cramer's rule that $|(M^{-1}b)_i| \leq \varepsilon \cdot B^{k-1} \cdot k! / |\det(M)|$, as desired. □

Lemma 5.19 (Step 6: reduction)**.** *Let $f : [0,1] \to \mathbb{R}$ be an analytic function on $[0,1]$, let $\kappa : \mathbb{N} \to [0,1]$ be a nonincreasing function, and let $\rho(\ell) = \left[\kappa(\ell)^2/(64B\ell 4^{2\ell+1} \cdot \ell!)\right]^{\ell+2}$ for any $\ell \in \mathbb{N}$ (as discussed before). Suppose that, for infinitely many positive integers k, there exists an x_k for which $|f^{(k+1)}(x)| > \kappa(k)$.*

Suppose that there is an algorithm for ε-approximate matrix-vector multiplication by an $n \times n$ f-matrix for points in $[0,1]^d$ in $T(n, d, \varepsilon)$ time. Then, there is a

$$n \cdot \mathrm{poly}(d + \log(1/\kappa(d+1))) + O(d \cdot T(2n, d+1, (\kappa(d+1))^{O(d^4)}))$$

time algorithm for exact bichromatic Hamming nearest neighbors on n-point sets in dimension d.

Proof. Let $x_1, \ldots, x_n \in \{0,1\}^d$ be the input to the Hamming nearest neighbors problem, so our goal is to compute $\min_{1 \leq i < j \leq n} \|x_i - x_j\|$. Let $t \in \mathbb{Z}_{\geq 0}^{d+1}$ be the 0-indexed vector of nonnegative integers, where $t_\ell := |\{1 \leq i < j \leq n \mid \|x_i - x_j\|_2^2 = \ell\}|$ counts the number of pairs of input points with distance ℓ. Our goal will be to recover the vector t, from which we can recover the answer to the Hamming nearest neighbors problem by returning the smallest index where t is nonzero.

Let $k = d + 1$ and let $\varepsilon = (\kappa(k))^{\alpha \cdot k^4}$ for a constant $\alpha > 0$ to be picked later. In order to recover t, we will make $d + 1$ calls to our given algorithm. For $\ell \in \{0, 1, \ldots, d\}$, the goal of call ℓ is to compute a value u_ℓ which is an approximation, with additive error at most ε, of entry ℓ of the vector Mt, where M is the counting matrix defined above.

Let us explain why this is sufficient to recover t. Suppose we have computed this vector u. We claim that if we compute $M^{-1}u$, and round each entry to the nearest integer, the result is the vector t. Indeed, by Lemma 5.18, each entry of $M^{-1}u$ differs from the corresponding entry of t by at most an additive $\varepsilon \cdot B^{k-1} \cdot k!/|\det(M)|$, where the constant B is from Proposition 5.12 (since $|f(z)| \leq B$ for all $z \in [0,1]$). Substituting our lower bound on $|\det(M)|$ from Lemma 5.17, we see this additive error is at most $1/3$ as long as we've picked a sufficiently large constant $\alpha > 0$. Thus, rounding each entry to the nearest integer recovers t, as desired.

It remains to show how to compute u_ℓ, an approximation with additive error at most ε of entry ℓ of the vector Mt. In other words, we need to approximate the sum

$$\sum_{p=0}^{d} M_{\ell,p} \cdot t_p = \sum_{1 \leq i < j \leq k} f(c + (\rho(k)/(B(200k)^k))^{10k} \cdot \ell \cdot \|x_i - x_j\|_2^2/k^2).$$

To do this, we will pick points $y_1, \ldots, y_n, z_1, \ldots, z_n \in [0,1]^{d+1}$ such that

$$\|y_i - z_j\|_2^2 = c + (\rho(k)/(B(200k)^k))^{10k} \cdot \ell \cdot \|x_i - x_j\|_2^2/k^2$$

for all $i, j \in [n]$, and apply our given algorithm with error ε to these points. For $i \in [n]$, let $x_i' = (\rho(k)/(B(200k)^k))^{5k} \cdot \sqrt{\ell} \cdot x_i/k$ be a rescaling of x_i. We pick y_i to equal x_i' in the first d entries and 0 in the last entry, and z_i to equal x_i' in the first d entries and $\sqrt{c}$ in the last entry. These points have the desired distances, completing the proof. □

Step 7: proof of Theorem 5.9. If the KAdjE problem for $\mathsf{K}(x,y) = f(\|x-y\|_2^2)$ in dimension d and error $(\kappa(d+1))^{O(d^4)}$ on $n = 1.01^d$ points could be solved in time time $n^{2-\delta}$ for any constant $\delta > 0$, then one could immediately substitute this into Lemma 5.19 to refute SETH in light of Theorem 3.17. □

5.5 Lower Bounds in Low Dimensions

The landscape of algorithms available in low dimensions $d = o(\log n)$ is a fair bit more complex. The Fast Multipole Method allows us to solve KAdjE for a number of functions f, including $\mathsf{K}(x,y) = \exp(-\|x-y\|_2^2)$ and $\mathsf{K}(x,y) = 1/\|x-y\|_2^2$, for which we have a lower bound in high dimensions. Classifying when these multipole methods apply to a function f seems quite difficult, as researchers have introduced more and more tools to expand the class of applicable functions. See Section 9, below, in which we give a much more detailed overview of these methods.

That said, in this subsection, we prove lower bounds for a number of functions K of interest. We show that for a number of functions K with applications to geometry and statistics, the KAdjE problem seems to become hard even in dimension $d = 3$ (see the end of this subsection for a list of such K).

We begin with the function $\mathsf{K}(x,y) = |\langle x, y \rangle|$. Here, the K Adjacency Evaluation problem becomes hard even for very small d. We give an $n^{1+o(1)}$ time algorithm only for $d \leq 2$. For $d = 3$ we show that such an algorithm would lead to a breakthrough in algorithms for the $\mathbb{Z}$-MaxIP problem, and for the only slightly super-constant $d = 2^{\Omega(\log^* n)}$, we show that a $n^{2-\varepsilon}$ time algorithm would refute SETH.

Lemma 5.20. *For the function* $\mathsf{K}(x,y) = |\langle x, y\rangle|$*, the* KAdjE *problem (Problem 6) can be solved exactly in time* $n^{1+o(1)}$ *when* $d = 2$*.*

Proof. Given as input $x_1, \ldots, x_n \in \mathbb{R}^2$ and $y \in \mathbb{R}^n$, our goal is to compute $z \in \mathbb{R}^n$ given by $z_i := \sum_{j \neq i} |\langle x_i, x_j\rangle| \cdot y_j$. We will first compute $z'_i := \sum_j |\langle x_i, x_j\rangle| \cdot y_j$, and then subtract $|\langle x_i, x_i\rangle| \cdot y_i$ for each i to get z_i.

We first sort the input vectors by their polar coordinate angle, and relabel so that $x_1, \ldots, x_n$ are in sorted order. Let ϕ_i be the polar coordinate angle of x_i. We will maintain two vectors $x^+, x^- \in \mathbb{R}^2$. We will 'sweep' an angle θ from 0 to 2π, and maintain that x^+ is the sum of the $y_i \cdot x_i$ with $\langle x_i, \theta\rangle > 0$, and x^- is the sum of the other $y_i \cdot x_i$. Initially let $\theta = 0$ and let x^+ be the sum of the $y_i \cdot x_i$ with $\phi_i \in [-\pi/2, \pi/2)$, and x^- be the sum of the remaining $y_i \cdot x_i$. As we sweep, whenever θ is in the direction of an x_i we can set $z'_i = \langle x_i, x^+ - x^-\rangle$. Whenever θ is orthogonal to an x_i, we swap $y_i \cdot x_i$ from one of x^+ or x^- to the other. Over the whole sweep, each point is swapped at most twice, so the total running time is indeed $n^{1+o(1)}$. □

Lemma 5.21. *For the function* $\mathsf{K}(x,y) = |\langle x, y\rangle|$*, if the* KAdjE *problem (Problem 6) with error* $1/n^{\omega(1)}$ *can be solved in time* $\mathcal{T}(n,d)$*, and* $n > d+1$*, then* $\mathbb{Z}$-MaxIP *(Problem 4) with* d*-dimensional vectors of integer entries of bit length* $O(\log n)$ *can be solved in time* $O(\mathcal{T}(n,d)\log^2 n + nd)$*.*

Proof. Let $x_1, \ldots, x_n \in \mathbb{Z}^d$ be the input vectors. Let $M \leq n^{O(1)}$ be the maximum magnitude of any entry of any input vector. Thus, $\max_{i \neq j}\langle x_i, x_j\rangle$ is an integer in the range $[-dM^2, dM^2]$. We will binary search for the answer in this interval. The total number of binary search steps will be $O(\log(dM^2)) \leq O(\log n)$.

We now show how to do each binary search step. Suppose we are testing whether the answer is $\leq a$, i.e. testing whether $\langle x_i, x_j\rangle \leq a$ for all $i \neq j$. Let $S_1, \ldots, S_{\log n} \subseteq \{1, \ldots, n\}$ be subsets such that for each $i \neq j$, there is a k such that $|S_k \cap \{i, j\}| = 1$. For each $k \in \{1, \ldots, \log n\}$ we will show how to test whether there are i, j with $|S_k \cap \{i, j\}| = 1$ such that $\langle x_i, x_j\rangle \leq a$, which will complete the binary search step.

Define the vectors $x'_1, \ldots, x'_n \in \mathbb{Z}^{d+1}$ by $(x'_i)_j = (x_i)_j$ for $j \leq d$, and $(x'_i)_{d+1} = a - 1$ if $i \in P_k$, and $(x'_i)_{d+1} = -1$ if $i \notin P_k$. Hence, for i, j with $|S_k \cap \{i, j\}| = 1$ we have $\langle x'_i, x'_j\rangle = \langle x_i, x_j\rangle - a + 1$, and so our goal is to test whether there are any such i, j with $\langle x'_i, x'_j\rangle > 0$.

Let $v_k \in \{0,1\}^n$ be the vector with $(v_k)_i = 1$ when $i \in P_k$ and $(v_k)_i = 0$ when $i \notin P_k$. Use the given algorithm to vector v_k, we can compute a $(a \pm n^{-\omega(1)})$ approximation to

$$s_1 := \sum_{i \in P_k} \sum_{j \notin P_k} |\langle x'_i, x'_j\rangle|$$

in time $O(\mathcal{T}(n,d))$. Similarly, using the fact that the corresponding matrix has rank d by definition, we can exactly compute

$$s_2 := \sum_{i \in P_k} \sum_{j \notin P_k} \langle x'_i, x'_j\rangle$$

in time $O(nd)$. Our goal is to determine whether $s_1 = -s_2$. Since each s_1 and s_2 is a polynomially-bounded integer, and we have a superpolynomially low error approximation to each, we can determine this as desired. □

Combining Lemma 5.21 with Theorem 3.18 we get:

Corollary 5.21.1. *Assuming* SETH*, there is a constant c such that for the function $\mathsf{K}(x,y) = |\langle x,y\rangle|$, the* KAdjE *problem (Problem 6) with error $1/n^{\omega(1)}$ and dimension $d = c^{\log^* n}$ vectors of $O(\log n)$ bit entries requires time $n^{2-o(1)}$.*

Similarly, combining with Theorem 3.19 we get:

Corollary 5.21.2. *For the function $\mathsf{K}(x,y) = |\langle x,y\rangle|$, if the* KAdjE *problem (Problem 6) with error $1/n^{\omega(1)}$ and dimension $d = 3$ vectors of $O(\log n)$ bit entries can be solved in time $n^{4/3-O(1)}$, then we would get a faster-than-known algorithm for $\mathbb{Z}$-*MaxIP *(Problem 4) in dimension $d = 3$.*

The same proof, but using Theorem 3.20 instead of Theorem 3.18, can also show hardness of thresholds of distance functions:

Corollary 5.21.3. *Corollary 5.21.1 also holds for the function $\mathsf{K}(x,y) = \mathrm{TH}(\|x-y\|_2^2)$, where* TH *is any threshold function (i.e. $\mathrm{TH}(z) = 1$ for $z \geq \theta$ and $\mathrm{TH}(z) = 0$ otherwise, for some $\theta \in \mathbb{R}_{>0}$).*

Using essentially the same proof as for Lemma 8.4 in Section 8, we can further extend Corollary 5.21.3 to any non-Lipschitz functions f:

Proposition 5.22. *Suppose $f : \mathbb{R}_+ \to \mathbb{R}$ is any function which is not (C,L)-multiplicatively Lipschitz for any constants $C, L \geq 1$. Then, assuming* SETH*, the* KAdjE *problem (Problem 6) with error $1/n^{\omega(1)}$ and dimension $d = c^{\log^* n}$ requires time $n^{2-o(1)}$.*

5.6 Hardness of the n-Body Problem

We now prove Corollary 1.1.1 from the Introduction, showing that our hardness results for the KAdjE problem (Problem 6) also imply hardness for the n-body problem.

Corollary 5.22.1 (Restatement of Corollary 1.1.1)**.** *Assuming* SETH*, there is no*

$$\mathrm{poly}(d, \log(\alpha)) \cdot n^{1+o(1)}$$

-time algorithm for one step of the n-body problem.

Proof. We reduce from the K graph Laplacian multiplication problem, where $\mathsf{K}(u,v) = f(\|u-v\|_2^2)$ and $f(z) = \frac{1}{(1+z)^{3/2}}$. A K graph Laplacian multiplication instance consists of the K graph G on a set of n points $X \subseteq \mathbb{R}^d$ and a vector $y \in \{0,1\}^n$ for which we wish to compute $L_G y$. Think of y as vector with coordinates in the set X. Compute this multiplication using an n-body problem as follows:

1. For each $b \in \{0,1\}$, let $X_b = \{x \in X \mid y_x = b\}$. Let $Z \subseteq \mathbb{R}^{d+1}$ be the set of all $(x,0)$ for $x \in X_0$ and $(x,1)$ for $x \in X_1$.
2. Solve the one-step n-body problem on Z with unit masses. Let $z \in \mathbb{R}^n$ be the vector of the $(d+1)$-th coordinate of these forces, with forces negated for coordinates $x \in X_0$.
3. Return $-z/G_{\text{grav}}$ (Note that G_{grav} is the Gravitational constant)

We now show that $z = L_G y$. For $x \in X_b$, $(L_G y)_x = (-1)^{1-b} \sum_{x' \in X_{1-b}} \mathsf{K}(x,x')$. We now check that z_x is equal to this by going through pairs $\{x, x'\}$ individually. Note that the $(d+1)$-th

coordinate of the force between (x, b) and (x', b) is 0. The $(d+1)$-th coordinate of the force exerted by $(x', 1)$ on $(x, 0)$ is

$$\frac{G_{\text{grav}}}{\|(x,0)-(x',1)\|_2^2} \cdot \frac{(1-0)}{\|(x,0)-(x',1)\|_2} = G_{\text{grav}} \cdot \mathsf{K}(x, x')$$

Negating this gives the force exerted by $(x', 0)$ on $(x, 1)$. All of these contributions agree with the corresponding contributions to the sum $(L_G y)_x$, so $-z/G_{\text{grav}} = L_G \cdot y$ as desired.

The runtime of this reduction is $O(n)$ plus the runtime of the n-body problem. However, Theorem 1.1 shows that no almost-linear time algorithm exists for K-Laplacian multiplication, since f is not approximable by a polynomial with degree less than $\Theta(\log n)$. Therefore, no almost-linear time algorithm exists for n-body either assuming SETH, as desired. □

5.7 Hardness of Kernel PCA

For any function $\mathsf{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, and any set $P = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$ of n points, define the matrix $K_{\mathsf{K},P} \in \mathbb{R}^{n \times n}$ by

$$K_{\mathsf{K},P}[i, j] = \mathsf{K}(x_i, x_j)$$

$A_{\mathsf{K},P}$ and $K_{\mathsf{K},P}$ differ only on their diagonal entries, so a $n^{1+o(1)}$ time algorithm for multiplying by one can be easily converted into such an algorithm for the other. Kernel PCA studies

Problem 8 (K PCA)**.** *For a given function $\mathsf{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, the K PCA problem asks: Given as input a set $P = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$ with $|P| = n$, output the n eigenvalues of the matrix $(I_n - J_n) \times K_{\mathsf{K},P} \times (I_n - J_n)$, where J_n is the $n \times n$ matrix whose entries are all $1/n$. In ε-approximate K PCA, we want to return a $(1 \pm \varepsilon)$-multiplicative approximation to each eigenvalue.*

We can now show a general hardness result for K PCA:

Theorem 5.23 (Approximate)**.** *For every function $f : \mathbb{R} \to \mathbb{R}$ which is equal to a Taylor expansion $f(x) = \sum_{i=0}^{\infty} c_i x^i$ on an interval $(0, 1)$, if f is not ε-approximated by a polynomial of degree $O(\log n)$ (Definition 5.3) on an interval $(0, 1)$ for $\varepsilon = 2^{-\log^4 n}$, then, assuming SETH, the ε-approximate K PCA problem (Problem 6) in dimension $d = O(\log n)$ requires time $n^{2-o(1)}$.*

Proof Sketch. Theorem 5.23 follows almost directly from Theorem 5.8 when combined with the reduction from [37, Section 5]. The idea is as follows: Suppose we are able to estimate the n eigenvalues of $(I_n - J_n) \times K_{\mathsf{K},P} \times (I_n - J_n)$. Then, in particular, we can estimate their sum, which is equal to:

$$\operatorname{tr}((I_n - J_n) \times K_{\mathsf{K},P} \times (I_n - J_n)) = \operatorname{tr}(K_{\mathsf{K},P} \times (I_n - J_n)^2) = \operatorname{tr}(K_{\mathsf{K},P} \times (I_n - J_n)) = \operatorname{tr}(K_{\mathsf{K},P}) - S(K_{\mathsf{K},P})/n,$$

where $S(K_{\mathsf{K},P})$ denotes the sum of the entries of $K_{\mathsf{K},P}$. We can compute $\operatorname{tr}(K_{\mathsf{K},P})$ exactly in time $n^{1+o(1)}$, so we are able to get an approximation to $S(K_{\mathsf{K},P})$. However, in the proof of Theorem 5.8, we showed hardness for approximating $S(K_{\mathsf{K},P})$, which concludes our proof sketch. □

6 Sparsifying Multiplicatively Lipschitz Functions in Almost Linear Time

Given n points, let α denote

$$\alpha = \frac{\max_{u,v \in P}(\|u-v\|_2^2)}{\min_{u,v \in P}(\|u-v\|_2^2)}.$$

In this section, we give an algorithm to compute sparsifiers for a large class of kernels K in almost linear time in nd, with logarithmic dependency on α and $1/\varepsilon^2$ dependence on ε. When $d = \log n$, our algorithm runs in almost linear time in n. To formally state our main theorem, we define multiplicatively Lipschitz functions:

Definition 6.1. *Let $C \geq 1$ and $L \geq 1$. A function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is (C, L)-multiplicatively Lipschitz iff for all $c \in [1/C, C]$, we have:*

$$\frac{1}{C^L} < \frac{f(cx)}{f(x)} < C^L, \forall x \in \mathbb{R}_{\geq 0}.$$

Examples: Any polynomial with non-negative coefficients and maximum degree q is $(1+\varepsilon, q)$ multiplicatively Lipschitz for any $\varepsilon > 0$. The function $f(x) = 1$ when $x < 1$ and $f(x) = 2$ when $x \geq 1$ is $(2, 1)$ multiplicatively Lipschitz.

The following lemma is a simple consequence of our definition of multiplicatively Lipschitz functions:

Lemma 6.2. *Let $C \geq 1$ and $L > 0$. Any function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is (C, L)-multiplicatively Lipschitz satisfies for all $c \in (0, 1/C) \cup [C, +\infty)$:*

$$\frac{1}{c^{2L}} < \frac{f(cx)}{f(x)} < c^{2L}.$$

We now state the core theorem of this section:

Theorem 6.3. *Let $C \geq 1$ and $L \geq 1$. Consider any (C, L)-multiplicatively Lipschitz function $f : \mathbb{R} \to \mathbb{R}$, and let $\mathsf{K}(x, y) = f(\|x-y\|_2^2)$. Let P be a set of n points in $\mathbb{R}^d$. For any $k \in [\Omega(1), O(\log n)]$ such that $C = n^{O(1/k)}$, there exists an algorithm* SPARSIFY-K-GRAPH$(P, n, d, \mathsf{K}, k, L)$ *(Algorithm 3) that runs in time:*

$$O(ndk) + \varepsilon^{-2} \cdot n^{1+O(L/k)} 2^{O(k)} \log n \cdot \log \alpha$$

and outputs an ε-spectral sparsifier H of the K graph with $|E_H| = O(n \log n/\varepsilon^2)$.

We give a corollary of this theorem.

Corollary 6.3.1. *Consider a K graph where $\mathsf{K}(x, y) = f(\|x-y\|_2^2)$, and f is $(2, L)$-multiplicatively Lipschitz. Let G denote the K graph from n points in $\mathbb{R}^d$. There is an algorithm that takes in time*

$$O(nd\sqrt{L \log n}) + \varepsilon^{-2} \cdot n \cdot 2^{O(\sqrt{L \log n} \log \log n)} \cdot \log \alpha$$

and outputs an ε-spectral sparsifier H of G with $|E_H| = O(n \log n/\varepsilon^2)$. If $L = o(\log n/(\log \log n)^2)$, this runs in time

$$o(nd \log n) + \varepsilon^{-2} n^{1+o(1)} \log \alpha.$$

Proof. Set $k = \sqrt{L \log n}$, and the corollary follows from Theorem 6.3. □

This implies that if f is a polynomial with non-negative coefficients, then sparsifiers of the corresponding K-graph can be found in almost linear time. The same result holds if f is the reciprocal of a polynomial with non-negative coefficients.

We will need a few geometric preliminaries in order to present our core algorithm of this section, SPARSIFY-K-GRAPH.

Definition 6.4 (ε-well separated pair)**.** *Given two sets of points A and B. We say A, B is an ε-well separated pair if the diameter of A_i and B_i are at most ε times the distance between A_i and B_i.*

Definition 6.5 (ε-well separated pair decomposition [65])**.** *An ε-well separated pair decomposition (ε-WSPD) of a given point set P is a family of pairs $\mathcal{F} = \{(A_1, B_1), \ldots (A_s, B_s)\}$ with $A_i, B_i \subset P$ such that:*

- *$\forall i \in [s]$, A_i, B_i are ε-well separated pair (Definition 6.4)*
- *For any pair $p, q \in P$, there is a unique $i \in [s]$ such that $p \in A_i$ and $q \in B_i$*

A famous theorem of Callahan and Kosaraju [65] states:

Theorem 6.6 (Callahan and Kosaraju [65])**.** *Given any point set $P \subset \mathbb{R}^d$ and $0 \leq \varepsilon \leq 9/10$, an ε-WSPD (Definition 6.5) of size $O(n/\varepsilon^d)$ can be found in $2^{O(d)} \cdot (n \log n + n/\varepsilon^d)$ time. Moreover, each vertex participates in at most $2^{O(d)} \cdot \log \alpha$ ε-well separated pairs.*

Well-separated pairs can be interpreted as complete bipartite graphs on the vertex set, or **bicliques**. The biclique associated with a well-separated pair is the bipartite graph connecting all vertices on one side of the pair to another.

This concludes our definitions on well-separated pairs. We now give names to some algorithms in past work, which will be used in our algorithm SPARSIFY-k-GRAPH. We define the algorithm GENERATEWSPD(P, ε) to output an ε-WSPD (Definition 6.5) of P. We define the algorithm RANDOMPROJECT(P, k) to be a random projection of P onto k dimensions.

Let BICLIQUE(K, P, A, B) be the complete biclique on the K-graph of P with one side of the biclique having verticese corresponding to points in A, and the other side having vertices corresponding to points in B. We store this biclique implicitly as (A, B) rather than as a collection of edges.

Let RANDSAMPLE(G, s) be an algorithm uniformly at random sampling $O(s)$ edges from G, where the big O is the same constant as the big O in the $n^{O(1/k)}$ from Lemma 3.14.

Let SPECTRALSPARSIFY(G, ε) be any nearly linear time spectral sparsification algorithm that outputs a $(1 + \varepsilon)$ spectral sparsifier with $O(n \log n/\varepsilon^2)$ edges, such as that in Theorem 3.7 from [269].

The rest of this section is devoted to proving Theorem 6.3.

6.1 High Dimensional Sparsification

We are nearly ready to prove Theorem 6.3. We start with a Lemma:

Lemma 6.7 ((C, L) multiplicative Lipschitz functions don't distort a graph's edge weights much)**.** *Consider a complete graph G, and a complete graph G', where vertices of G are identified with vertices of G' (which induces an identification between edges). Let $K \geq 1$. Suppose each edge*

Algorithm 3

procedure SPARSIFY-K-GRAPH($P, n, d, \mathsf{K}, k, L, \varepsilon$) ▷ Theorem 6.3
  Input: A point set P with n points in dimension d, a kernel function $\mathsf{K}(x, y) = f(\|x - y\|_2^2)$, an integer variable $\Omega(1) \leq k \leq O(\log n)$, and a variable L, and error ε.
  Output: A candidate sparsifier of the K-graph on P.
  $P' \leftarrow$ RANDOMPROJECT(P, k)
  $H \leftarrow$ empty graph with n vertices.
  $\{(A'_1, B'_1), \ldots (A'_t, B'_t)\} \leftarrow$ GENERATEWSPD($P', n, d, 1/2$) ▷ $t = n \cdot 2^d$
  for $i = 1 \to t$ **do**
    Find (A_i, B_i) corresponding to (A'_i, B'_i), where $A_i, B_i \subset P$.
    $Q \leftarrow$ BICLIQUE(K, P, A_i, B_i).
    $s \leftarrow \varepsilon^{-2} n^{O(L/k)} (|A_i| + |B_i|) \log(|A_i| + |B_i|)$
    $\overline{Q} \leftarrow$ RANDSAMPLE(Q, s)
    $\overline{Q} \leftarrow \overline{Q}$ with each edge scaled by $|A_i||B_i|/s$.
    $H \leftarrow H + \overline{Q}$
  end for
  $H \leftarrow$ SPECTRALSPARSIFY(H, ε) ▷ Corollary 3.7.1
  Return H
end procedure
procedure GENERATEWSPD(P, n, d, ε) ▷ Theorem 6.6
  ... ▷ See details in [65]
end procedure
procedure RANDSAMPLE(G, s)
  Sample $O(s)$ edges from G and generate a new graph $\overline{G}$
  return $\overline{G}$
end procedure
procedure RANDOMPROJECT(P, d, k)
  $P' \leftarrow \emptyset$
  Choose a JL matrix $S \in \mathbb{R}^{k \times d}$
  for $x \in P$ **do**
    $x' \leftarrow S \cdot x$
    $P' \leftarrow P' \cup x'$
  end for
  return P'
end procedure

in G satisfies:

$$\frac{1}{K} \cdot w_{G'}(e) \leq w_G(e) \leq K \cdot w_{G'}(e)$$

If f is a (C, L) multiplicative Lipschitz function, and $f(G)$ refers to the graph G where f is

applied to each edge length, and $C < K$ then:

$$\frac{1}{K^{2L}} \cdot w_{G'}(e) \leq w_{f(G)}(e) \leq K^{2L} \cdot w_{G'}(e).$$

Proof. This follows from Lemma 6.2. □

Proof. (of Theorem 6.3): The Algorithm SPARSIFY-K-GRAPH starts by performing a random projection of point set P into k dimensions. Call the new point set P'. This runs in time $O(ndk)$, and incurs distortion $n^{O(1/k)}$, as seen in Lemma 3.14. Next, our algorithm performs a $1/2$-WSPD on P'. As seen in Theorem 6.6, this runs in time

$$O(n \log n + n2^{O(k)})$$

We view each well-separated pair (A'_i, B'_i) on P' as a biclique, where the edge length between any two points in P' corresponds to the edge length between those two points in the original K-graph. By the guarantees of Theorem 6.6, the longest edge divided by the shortest edge between two sides of a well-separated pair in P' is at most 2. Thus, the longest edge divided by the shortest edge within any induced bipartite graph on the K-graph is $2 \cdot n^{O(1/k)}$, by Lemma 6.7.

For each such biclique, the leverage score for each edge is overestimated by

$$n^{O(L/k)} \cdot (|A'_i| + |B'_i|)/(|A'_i||B'_i|).$$

This comes from first applying Lemma 6.7 to upper bound the ratio of the longest edge in a biclique divided by the shortest edge. This ratio comes out to be $2n^{O(L/k)}$. Now recall the definition of leverage score on graphs as $w_e R_e$, where R_e is the effective resistance assuming conductances of w_e on the graph, and w_e is the edge weight. Here, w_e is upper bounded by the longest edge length, and R_e is upper bounded by the leverage score of a biclique supported on the same edges, where all edges lengths are equal to the shortest edge length (this is an underestimate of effective resistance due to Rayleigh monotonicity, see [91] for details). Therefore, a leverage score overestimate of the graph can be obtained by $n^{O(L/k)} \cdot (|A'_i| + |B'_i|)/(|A'_i||B'_i|)$, as claimed. The union of these graphs is a spectral sparsifier of our K-graph.

Finally, our algorithm samples $n^{O(L/k)}(|A'_i|+|B'_i|)\log(|A'_i|+|B'_i|)$ edges uniformly at random from each biclique, scaling each sampled edge's weight so that the expected value of the sampled graph is equal to the original biclique. Each vertex participates in at most $\log \alpha 2^{O(k)}$ bicliques (see Theorem 6.6). Thus, this uniform sampling procedures' run time is bounded above by

$$n^{1+O(L/k)}2^{O(k)} \log n \cdot \log \alpha.$$

Finally, our algorithm runs a sparsification algorithm on our graph after uniform sampling, which gets the edge count of the final graph down to $O(n \log n/\varepsilon^2)$. This completes our proof of Theorem 6.3. □

6.2 Low Dimensional Sparsification

We now present a result on sparsification in low dimensions, when d is assumed to be small or constant.

Theorem 6.8. *Let $L \geq 1$. Consider a K-graph with n vertices arising from a point set in d dimensions, and let α be the ratio of the maximum Euclidean distance to the minimum Euclidean distance in the point set . Let f be a $(1 + 1/L, L)$ multiplicatively Lipschitz function. Then an ε spectral sparsifier of the K-graph can be found in time*

$$\varepsilon^{-2} \cdot n \cdot \log n \cdot \log \alpha \cdot (2L)^{O(d)}$$

Proof. We roughly follow the proof of Theorem 6.3, except without the projection onto low dimensions. Now, on the d dimensional data, we create a $1/L$-WSPD. This takes time

$$O(n \log n) + n \cdot (2L)^{O(d)}.$$

Since f is $(1 + 1/L, L)$-multiplicatively Lipschitz, it follows that within each biclique of the K-graph induced by the WSPD (Definition 6.5), the maximum edge length divided by the minimum edge length is bounded above by $(1 + 1/L)^{O(L)} = O(1)$. Therefore, performing scaled and reweighted uniform sampling on each biclique takes $O(s \log s)$ time if there are s vertices in the biclique, and gives a sparsified biclique with $O(s \log s)$ edges.

Taking the union of this number over all bicliques gives an algorithm that runs in time

$$\varepsilon^{-2} \cdot n \cdot \log n \cdot \log \alpha \cdot (2L)^{O(d)}$$

as desired. □

7 Sparsifiers for $|\langle x, y\rangle|$

In this section, we construct sparsifiers for Kernels of the form $|\langle x, y\rangle|$.

Lemma 7.1 (sparsification algorithm for inner product kernel)**.** *Given a set of vectors $X \subseteq \mathbb{R}^d$ with $|X| = n$ and accuracy parameter $\varepsilon \in (0, 1/2)$, there is an algorithm that runs in $\varepsilon^{-2} n \cdot \mathrm{poly}(d, \log n)$ time, and outputs a graph H that satisfies both of the following properties with probability at least $1 - 1/\mathrm{poly}(n)$:*

1. $(1 - \varepsilon)L_G \preceq L_H \preceq (1 + \varepsilon)L_G$;
2. $|E(H)| \leq \varepsilon^{-2} \cdot n \cdot \mathrm{poly}(d, \log n)$.

where G is the K-graph on X, where $\mathsf{K}(x, y) = |\langle x, y\rangle|$.

Throughout this section, we specify various values C_i. For each subscript i, it is the case that $1 \leq C_i \leq \mathrm{poly}(d, \log n)$.

7.1 Existence of large expanders in inner product graphs

We start by showing that certain graphs that are related to unweighted versions of K-weighted graphs contain large expanders:

Definition 7.2 (k-dependent graphs)**.** *For a positive integer $k > 1$, call an unweighted graph G* k-dependent *if no independent set with size at least $k + 1$ exists in G.*

We start by observing that inner product graphs are $(d + 1)$-dependent.

Definition 7.3 (inner product graphs)**.** *For a set of points $X \subseteq \mathbb{R}^d$, the* unweighted inner product graph for X *is a graph G with vertex set X and unweighted edges $\{u, v\} \in E(G)$ if and only if $|\langle u, v\rangle| \geq \frac{1}{d+1}\|u\|_2\|v\|_2$. The* weighted inner product graph for X *is a complete graph G with edge weights w_e for which $w_{uv} = |\langle u, v\rangle|$.*

We now show that these graphs are k-dependent:

Lemma 7.4. *Suppose $M \in \mathbb{R}^{n\times n}$ such that $M[i, i] = 1$ for all $i \in [n]$, and $|M[i, j]| < 1/n$ for all $i \neq j$. Then, M has full rank.*

Proof. Assume to the contrary that M does not have full rank. Thus, there are values $c_1, \ldots, c_{n-1} \in \mathbb{R}$ such that for all $i \in [n]$ we have $\sum_{j=1}^{n-1} c_j M[i, j] = M[i, n]$.

First, note that there must be a j with $|c_j| \geq 1 + 1/n$. Otherwise, we would have

$$1 = |M[n, n]| = \left|\sum_{j=1}^{n-1} c_j M[n, j]\right| < \frac{1}{n}\sum_{j=1}^{n-1} |c_j| < \frac{n-1}{n}(1 + 1/n) < 1.$$

Assume without loss of generality that $|c_1| \geq |c_j|$ for all $j \in \{2, 3, \ldots, n-1\}$, so in particular $|c_1| \geq 1 + 1/n$. Letting $c_n = -1$, this means that $\sum_{j=1}^{n} c_j M[1, j] = 0$, and so $M[1, 1] = -\sum_{j=2}^{n}(c_j/c_1)M[1, j]$. Thus,

$$1 = |M[1, 1]| = \left|\sum_{j=2}^{n} \frac{c_j}{c_1} M[1, j]\right| \leq \sum_{j=2}^{n} \left|\frac{c_j}{c_1} M[1, j]\right| < \sum_{j=2}^{n} |M[1, j]| < (n-1) \cdot \frac{1}{n} < 1,$$

a contradiction as desired. □

Proposition 7.5. *The unweighted inner product graph for $X \subseteq \mathbb{R}^d$ is $(d + 1)$-dependent.*

Proof. For an independent set S in the unweighted inner product graph G for X, define an $S \times S$ matrix M with $M[i, j] = \langle s_i, s_j\rangle$ where $S = \{s_1, s_2, \ldots, s_{|S|}\}$. Then Lemma 7.4 coupled with the definition for edge presence in G shows that M is full rank. However, M is a rank d matrix because it is the matrix of inner products for dimension d vectors. Therefore, $d \geq |S|$, so no independent set has size greater than d. □

Next, we show that k-dependent graphs are dense:

Proposition 7.6 (dependent graph is dense)**.** *Any k-dependent graph G has at least $n^2/(2k^2)$ edges.*

Proof. Consider any $k+1$-tuple of vertices in G. There are $\binom{n}{k+1}$ such $k+1$-tuples. By definition of k-dependence, there must be some edge with endpoints in any $k + 1$-tuple. The number of k-tuples that any given edge can be a part of is at most $\binom{n-2}{k-1}$. Therefore, the number of edges in the graph is at least

$$\frac{\binom{n}{k+1}}{\binom{n-2}{k-1}} \geq \frac{n(n-1)}{(k+1)k} \geq \frac{n^2}{2k^2}$$

as desired. □

Next, we argue that unweighted inner product graphs have large expanders:

Proposition 7.7 (every dense graph has a large expander)**.** *Consider an unweighted graph G with n vertices and at least n^2/c edges for some $c > 1$. Then, there exists a set $S \subseteq V(G)$ with the following properties:*

1. *(Size) $|S| \geq n/(40c)$*
2. *(Expander) $\Phi_{G[S]} \geq 1/(100c \log n)$*
3. *(Degree) The degree of each vertex in S within $G[S]$ is at least $n/(10000c)$*

Proof. We start by partitioning the graph as follows:

1. Initialize $\mathcal{F} = \{V(G)\}$
2. While there exists a set $U \in \mathcal{F}$ with (a) a partition $U = U_1 \cup U_2$ with U_1 cut having conductance $\leq 1/(100c \log n)$ **or** (b) a vertex u with degree less than $n/(10000c)$ in $G[U]$
   (a) If (a), replace U in $\mathcal{F}$ with U_1 and U_2
   (b) Else if (b), replace U in $\mathcal{F}$ with $U \setminus \{u\}$ and $\{u\}$

We now argue that when this procedure stops,

$$\sum_{U \in \mathcal{F}} |E(G[U])| \geq n^2/(2c)$$

To prove this, think of each splitting of U into U_1 and U_2 as deleting the edges in $E(U_1, U_2)$ from G. Design a charging scheme that assigns deleted edges due to (a) steps to edges of G as follows. Let c_e denote the charge assigned to an edge e and initialize each charge to 0. When U is split into U_1 and U_2, let U_1 denote the set with $|E(U_1)| \leq |E(U_2)|$. When U is split, increase the charge c_e for each $e \in E(U_1) \cup E(U_1, U_2)$ by $|E(U_1, U_2)|/|E(U_1) \cup E(U_1, U_2)|$.

We now bound the charge assigned to each edge at the end of the algorithm. By construction, $\sum_{e \in E(G)} c_e$ is the number of edges deleted over the course of the algorithm of type (a). Each edge is assigned charge at most $\log |E(G)| \leq 2 \log n$ times, because $|E(U_1)| \leq \frac{|E(U_1)|+|E(U_2)|}{2} \leq \frac{|E(U)|}{2}$ when charge is assigned to edges in U_1. Furthermore, the amount of charge assigned is the conductance of the cut deleted, which is at most $1/(100c \log n)$. Therefore,

$$c_e \leq \frac{2 \log n}{100c \log n} = \frac{1}{50c}$$

for all edges in G, which means that the total number of edges deleted of type (a) was at most $n^2/(100c)$. Each type (b) deletion reduces the number of edges in G by at most $n/(10000c)$, so the total number of type (b) edge deletions is at most $n^2/(10000c)$. Therefore, the total number of edges remaining is at least

$$\frac{n^2}{c} - \frac{n^2}{100c} - \frac{n^2}{10000c} > \frac{n^2}{2c},$$

as desired.

By the stopping condition of the algorithm, each connected component of G after edge deletions is a graph with all cuts having conductance at least $1/(100c \log n)$ and all vertices having

degree at least $n/(10000c)$. Next, we show that some set in $\mathcal{F}$ has at least $n/(40c)$ vertices. If this is not the case, then

$$\begin{aligned}
\sum_{U \in \mathcal{F}} |E(G[U])| &\leq \sum_{U \in \mathcal{F}} |U|^2 \\
&\leq \sum_{U \in \mathcal{F}} \frac{n}{40c} |U| && \text{by assuming all } |U| \leq n/(40c) \\
&\leq \frac{n^2}{40c} && \text{by } |U| \leq n \\
&< \frac{n^2}{2c}
\end{aligned}$$

which leads to a contradiction. Therefore, there must be some connected component with at least $n/(40c)$ vertices. Let S be this component. By definition S satisfies the *Size* guarantee. By the stopping condition for $\mathcal{F}$, S satisfies the other two guarantees as well, as desired. □

Proposition 7.7 does not immediately lead to an efficient algorithm for finding S. Instead, we give an algorithm for finding a weaker but sufficient object:

Proposition 7.8 (algorithm for finding sets with low effective resistance diameter)**.** *For any set of vectors $X \subseteq \mathbb{R}^d$ with $n = |X|$ and unweighted inner product graph G for X, there is an algorithm* LOWDIAMSET(X) *that runs in time*

$$\text{poly}(d, \log n)n$$

and returns a set Q that has the following properties with probability at least $1 - 1/\text{poly}(n)$:

1. *(Size) $|Q| \geq n/C_1$, where $C_1 = 320d^2$*
2. *(Low effective resistance diameter) For any pair of vertices $u, v \in Q$, $\texttt{Reff}_G(u, v) \leq \frac{C_2}{n}$, where $C_2 = (10d \log n)^{10}$*

We prove this proposition in Section 7.2.

7.2 Efficient algorithm for finding sets with low effective resistance diameter

In this section, we prove Proposition 7.8. We implement LOWDIAMSET by picking a random vertex v and checking to see whether or not it belongs to a set with low enough effective resistance diameter. We know that such a set exists by Proposition 7.7 and the fact that dense expander graphs have low effective resistance diameter. One could check that v lies in this set in $\widetilde{O}(n^2)$ time by using the effective resistance data structure of [269]. Verifying that v is in such a set in $\text{poly}(d, \log n)n$ time is challenging given that G is dense. We instead use the following data structure, which is implemented by uniformly sampling sparse subgraphs of G and using [269] on those subgraphs:

Proposition 7.9 (Sampling-based effective resistance data structure)**.** *Consider a set $X \subseteq \mathbb{R}^d$, let $n = |X|$, let G be the unweighted inner product graph for X, and let $S \subseteq X$ be a set with the following properties:*

1. *(Size)* $|S| \geq n/C_{3a}$*, where* $C_{3a} = 40 \cdot 8d$
2. *(Expander)* $\Phi_{G[S]} \geq 1/C_{3b}$*, where* $C_{3b} = 800d \log n$
3. *(Degree) The degree of each vertex in* S *within* $G[S]$ *is at least* n/C_{3c}*, where* $C_{3c} = 10000 \cdot 8d$*.*

There is a data structure that, when given a pair of query points $u, v \in X$*, outputs a value* $\textsc{ReffQuery}(u, v) \in \mathbb{R}_{\geq 0}$ *that satisfies the following properties with probability at least* $1 - 1/\mathrm{poly}(n)$*:*

1. *(Upper bound) For any pair* $u, v \in S$*,* $\textsc{ReffQuery}(u, v) \leq C_4 \mathtt{Reff}_G(u, v)$*, where* $C_4 = 10$
2. *(Lower bound) For any pair* $u, v \in X$*,* $\textsc{ReffQuery}(u, v) \geq \mathtt{Reff}_G(u, v)/2$*.*

The preprocessing method $\textsc{ReffPreproc}(X)$ *takes* $\mathrm{poly}(d, \log n)n$ *time and* ReffQuery *takes* $\mathrm{poly}(d, \log n)$ *time.*

We now implement this data structure. ReffPreproc uniformly samples $O(\log n)$ subgraphs of G and builds an effective resistance data structure for each one using [269]. ReffQuery queries each data structure and returns the maximum:

---

procedure $\textsc{ReffPreproc}(X)$
  Input: $X \subseteq \mathbb{R}^d$ with unweighted inner product graph G
  $C_5 \leftarrow 1000 \log n$
  $C_6 \leftarrow 80000CC_{3a}^2C_{3b}^2C_{3c}^2$, where C is the constant in Theorem 3.6
  for i from 1 through C_5 **do**
    $H_i \leftarrow$ uniformly random subgraph of G; sampled by picking $C_6 n$ uniformly random pairs $(u, v) \in X \times X$ and adding the $e = \{u, v\}$ edge to H_i if and only if $\{u, v\} \in E(G)$ (that is when $|\langle u, v \rangle| \geq \frac{1}{d+1}\|u\|_2\|v\|_2$).
    For each edge $e \in E(H_i)$, let $w_e = \frac{|E(G)|}{|E(H_i)|}$.
    $f_i \in \mathbb{R}^{X \times X} \leftarrow$ approximation to $\mathtt{Reff}_{H_i}(u, v)$ given by the data structure of Theorem 3.7 for $\varepsilon = 1/6$.
  end for
end procedure
procedure $\textsc{ReffQuery}(u, v)$
  Input: A pair of points $u, v \in X$
  Output: An estimate for the u-v effective resistance in G
  return $\max_{i=1}^{C_5} f_i(u, v)$
end procedure

Bounding the runtime of these two routines is fairly straightforward. We now outline how we prove the approximation guarantee for ReffQuery. To obtain the upper bound, we use Theorem 3.6 to show that H_i contains a sparsifier for $G[S]$, so effective resistances are preserved within S. To obtain the lower bound, we use the following novel Markov-style bound on effective resistances:

Lemma 7.10. *Let* G *be a* w*-weighted graph with vertex set* X *and assign numbers* $p_e \in [0, 1]$ *to each edge. Sample a reweighted subgraph* H *of* G *by independently and identically selecting*

q edges, with an edge chosen with probability proportional to p_e and added to H with weight $tw_e/(p_e q)$, where $t = \sum_{e \in G} p_e$. Fix a pair of vertices u, v. Then for any $\kappa > 1$,

$$\Pr\left[\mathtt{Reff}_H(u,v) \le \mathtt{Reff}_G(u,v)/\kappa\right] \le 1/\kappa.$$

Proof. For two vertices u, v, define the (folklore) *effective conductance* between u and v in the graph I to be

$$\mathtt{Ceff}_I(u,v) := \min_{q \in \mathbb{R}^n : q_u = 0, q_v = 1} \sum_{\text{edges } \{x,y\} \in I} w_{xy}(q_x - q_y)^2.$$

It is well-known that $\mathtt{Ceff}_I(u,v) = 1/\mathtt{Reff}_I(u,v)$.

Let

$$q^* = \arg \min_{q \in \mathbb{R}^n : q_u = 0, q_v = 1} \sum_{\{x,y\} \in E(G)} w_{xy}(q_x - q_y)^2.$$

Using q^* as a feasible solution in the $\mathtt{Ceff}_H$ optimization problem shows that

$$\begin{aligned}
\mathbf{E}[\mathtt{Ceff}_H(u,v)] &\le \mathbf{E}\left[\sum_{\{x,y\} \in E(G)} w^H_{xy}(q^*_x - q^*_y)^2\right] \\
&= \sum_{\{x,y\} \in E(G)} \mathbf{E}[w^H_{xy}](q^*_x - q^*_y)^2 \\
&= \sum_{\{x,y\} \in E(G)} \frac{p_{xy}}{t} \sum_{i=1}^{q} \left(\frac{t w^G_{xy}}{q p_{xy}}\right)(q^*_x - q^*_y)^2 \\
&= \sum_{\{x,y\} \in E(G)} w^G_{xy}(q^*_x - q^*_y)^2 \\
&= \mathtt{Ceff}_G(u,v)
\end{aligned}$$

where w^I_e denotes the weight of the edge e in the graph I.

Finally, we have

$$\begin{aligned}
\Pr[\mathtt{Reff}_H(u,v) < \mathtt{Reff}_G(u,v)/\kappa] &= \Pr[\mathtt{Ceff}_H(u,v) > \kappa \cdot \mathtt{Ceff}_G(u,v)] \\
&\le 1/\kappa,
\end{aligned}$$

where the first step follows from $\mathtt{Reff}_G(u,v) = 1/\mathtt{Ceff}_G(u,v)$, and the last step follows from Markov's inequality. □

We now prove Proposition 7.9:

Proof of Proposition 7.9. **Runtime.** We start with preprocessing. Sampling $C_6 n$ pairs and checking if each pair satisfies the edge presence condition for G takes $O(dC_6 n) = \mathrm{poly}(d, \log n)n$ time. Preprocessing for the function f_i takes $\mathrm{poly}(d, \log n)n$ time by the preprocessing guarantee of Theorem 3.7. Since there are $C_5 \le \mathrm{poly}(d, \log n)$ different is, the total preprocessing time is $\mathrm{poly}(d, \log n)n$, as desired.

Next, we reason about query time. This follows immediately from the query time bound of Theorem 3.7, along the the fact that $C_5 \leq \text{poly}(d, \log n)$.

Upper bound. Let $p_e^{\text{upper}} = C_7/n$ for each edge $e \in E(G[S])$, where $C_7 = 2C_{3a}C_{3b}^2$. We apply Theorem 3.6 to argue that $H_i[S]$ is a sparsifier for $G[S]$ for each i. By choice of C_7 and the *Size* condition on $|S|$, $p_{u,v}^{\text{upper}} \geq \frac{2}{\Phi^2_{G[S]}|S|}$. By Lemma 3.5 and the *Expander* condition on $G[S]$, $\frac{2}{\Phi^2_{G[S]}|S|} \geq \texttt{Reff}_{G[S]}(u,v)$ for each pair $u, v \in S$. Therefore, the second condition of Theorem 3.6 is satisfied by the probabilities p_e. Let $q = 10000C(n^2/(C_{3a}C_{3c}))\log(n^2/(C_{3a}C_{3c}))(C_7/n)$, where C is the constant in Theorem 3.6. This value of q satisfies the first condition of Theorem 3.6.

To apply Theorem 3.6, we just need to show that $H_i[S]$ has at least q edges with probability at least $1 - 1/\text{poly}(n)$. First, note that for uniformly random $u, v \in X$

$$\begin{aligned}\Pr_{u,v\in X}[\{u,v\} \in E(G[S])] &= \Pr[\{u,v\} \in E(G[S]), u, v \in S] \\ &= \Pr[\{u,v\} \in E(G[S])|u, v \in S]\Pr[u \in S]\Pr[v \in S] \\ &\geq \frac{1}{2C_{3c}}\frac{1}{C_{3a}^2}\end{aligned}$$

by the *Degree* and *Size* conditions on S. Since at least $C_6 n$ pairs in $X \times X$ are chosen, $\mathbf{E}[|E(H_i[S])|] \geq C_6 n(\frac{1}{2C_{3c}})(\frac{1}{C_{3a}^2}) \geq 2q$. By Chernoff bounds, this means that $|E(H_i[S])| \geq q$ with probability at least $1 - 1/\text{poly}(n)$. Therefore, Theorem 3.6 applies and shows that $H_i[S]$ with edge weights $|E(G[S])|/|E(H_i[S])|$ is a $(1/6)$-sparsifier for $G[S]$ with probability at least $1 - 1/\text{poly}(n)$. By Chernoff bounds, $|E(G[S])|/|E(H_i[S])| \geq |E(G)|/(2|E(H_i)|)$, so $\texttt{Reff}_{H_i[S]}(u,v) \leq \texttt{Reff}_{G[S]}(u,v)(1 + 1/6)2 \leq C_4\texttt{Reff}_{G[S]}(u,v)$ for all $u, v \in S$ by the sparsification accuracy guarantee. Since this holds for each i, the maximum over all i satisfies the guarantee as well, as desired.

Lower bound. Let $p_e^{\text{lower}} = 1/|E(G)|$ for each edge e. Note that $t = 1$, so with $q = |E(H_i)|$, all edges in the sampled graph should have weight $tw_e/(p_e q) = |E(G)|/|E(H_i)|$. Therefore, by Lemma 7.10, for a pair $u, v \in X$

$$\Pr[\texttt{Reff}_{H_i}(u,v) \leq 4\texttt{Reff}_G(u,v)/5] \leq \frac{4}{5}$$

for each i. Since the H_is are chosen independently and identically,

$$\Pr[\max_i \texttt{Reff}_{H_i}(u,v) \leq 4\texttt{Reff}_G(u,v)/5] \leq \left(\frac{4}{5}\right)^{C_5} \leq \frac{1}{n^{100}}$$

Union bounding over all pairs shows that

$$\textsc{ReffQuery}(u,v) > (6/7)(4/5)\texttt{Reff}_G(u,v) > \texttt{Reff}_G(u,v)/2$$

for all $u, v \in X$ with probability at least $1 - 1/n^{98} = 1 - 1/\text{poly}(n)$, as desired. □

We now describe the algorithm LOWDIAMSET. This algorithm simply picks random vertices v and queries the effective resistance data structure to check that v is in a set with the desired properties:

```
1: procedure LOWDIAMSET(X)
2:     REFFPREPROC(X)
3:     while true do
4:         Pick a uniformly random vertex v from X
5:         Q_v ← {u ∈ X : REFFQUERY(u, v) ≤ C_2/(2n)}
6:         Return Q_v if |Q_v| ≥ n/C_1
7:     end while
8: end procedure
```

We now prove that this algorithm suffices:

Proof of Proposition 7.8. **Size and Low effective resistance diameter.** Follows immediately from the return condition and the fact that for every $u \in Q_v$ for the returned Q_v,

$$\texttt{Reff}_G(u, v) \leq 2\text{REFFQUERY}(u, v) \leq C_2/n$$

by the *Lower bound* guarantee of Proposition 7.9.

Runtime. We start by showing that the while loop terminates after $\text{poly}(d, \log n)$ iterations with probability at least $1 - 1/\text{poly}(n)$. By Proposition 7.5, G is $(d+1)$-dependent. By Proposition 7.6, G has at least $n^2/(8d^2)$ edges. By Proposition 7.7, G has a set of vertices S for which $|S| \geq n/(320d^2)$, $\Phi_{G[S]} \geq 1/(800d^2 \log n)$, and for which the minimum degree of $G[S]$ is at least $n/(80000d^2)$. By the first condition on S,

$$\Pr[v \in S] \geq 1/(320d^2)$$

so the algorithm picks a v in S with probability at least $1-1/\text{poly}(n)$ after at most $32000d^2 \log n \leq \text{poly}(d, \log n)$ iterations. By Lemma 3.5 applied to S,

$$\texttt{Reff}_G(x, y) \leq \texttt{Reff}_{G[S]}(x, y) \leq \left(\frac{1}{d_S(x)} + \frac{1}{d_S(y)}\right) \frac{1}{\Phi^2_{G[S]}}$$

for any $x, y \in S$, where $d_S(w)$ denotes the degree of the vertex w in $G[S]$. By the third property of S, $d_S(x) \geq n/(80000d^2)$ and $d_S(y) \geq n/(80000d^2)$. By this and the second property of S,

$$\texttt{Reff}_G(x, y) \leq 102400000000d^6(\log^2 n)/n \leq C_2/(2C_4 n)$$

for any $x, y \in S$. S satisfies the conditions required of Proposition 7.9 by choice of the values C_{3a}, C_{3b}, and C_{3c}. Therefore, by the *Upper bound* guarantee of Proposition 7.9,

$$\text{REFFQUERY}(u, v) \leq C_2/(2n)$$

for every $u \in S$ if $v \in S$. Since $|S| \geq n/C_1$ by choice of C_1, the return statement returns Q_v with probability at least $1 - 1/\text{poly}(n)$ when $v \in S$. Therefore, the algorithm returns a set Q_v with probability at least $1 - 1/\text{poly}(n)$ after at most $\text{poly}(d, \log n)$ iterations.

Each iteration consists of $O(n)$ REFFQUERY calls and $O(n)$ additional work. Therefore, the total work done by the while loop is $\mathrm{poly}(d, \log n)n$ with probability at least $1 - 1/\mathrm{poly}(n)$. REFFPREPROC(X) takes $\mathrm{poly}(d, \log n)n$ by Proposition 7.9. Thus, the total runtime is $\mathrm{poly}(d, \log n)n$, as desired. □

7.3 Using low-effective-resistance clusters to sparsify the unweighted IP graph

In this section, we prove the following result:

Proposition 7.11 (Unweighted inner product sparsification)**.** *There is a* $\mathrm{poly}(d, \log n)n/\varepsilon^2$*-time algorithm for constructing an* ε*-sparsifier with* $O(n/\varepsilon^2)$ *for the unweighted inner product graph of a set of* n *points* $X \subseteq \mathbb{R}^d$*.*

To sparsify an unweighted inner product graph, it suffices to apply Proposition 7.8 repeatedly to partition the graph into $\mathrm{poly}(d, \log n)$ clusters, each with low effective resistance diameter. We can use this structure to get a good bound on the leverage scores of edges between clusters:

Proposition 7.12 (bound leverage scores of edges between clusters)**.** *For a* w*-weighted graph* G *with vertex set* X *and* $n = |X|$*, let* $S_1, S_2 \subseteq X$ *be two sets of vertices, let* $R_1 = \max_{u,v \in S_1} \mathtt{Reff}_G(u, v)$*, and let* $R_2 = \max_{u,v \in S_2} \mathtt{Reff}_G(u, v)$*. Then, for any* $u \in S_1$ *and* $v \in S_2$*,*

$$\mathtt{Reff}_G(u, v) \leq 3R_1 + 3R_2 + \frac{3}{\sum_{x \in S_1, y \in S_2} w_{xy}}.$$

where $w_{xx} = \infty$ *for all* $x \in X$

Proof. Let $\chi \in \mathbb{R}^n$ be the vector with $\chi_u = 1$, $\chi_v = -1$, and $\chi_x = 0$ for all $x \in X$ with $x \neq u$ and $x \neq v$. For each vertex $x \in S_1$, let $s_x = \sum_{y \in S_2} w_{xy}$. For each vertex $y \in S_2$, let $s_y = \sum_{x \in S_1} w_{xy}$. Let $\tau = \sum_{x \in S_1} s_x = \sum_{y \in S_2} s_y = \sum_{x \in S_1, y \in S_2} w_{xy}$. Write χ as a sum of three vectors $d^{(1)}, d^{(12)}, d^{(2)} \in \mathbb{R}^n$ as follows:

$$d_x^{(1)} = \begin{cases} 1 - \frac{s_x}{\tau} & x = u \\ -\frac{s_x}{\tau} & x \in S_1 \setminus \{u\} \\ 0 & \text{otherwise} \end{cases}$$

$$d_x^{(2)} = \begin{cases} \frac{s_x}{\tau} - 1 & x = v \\ \frac{s_x}{\tau} & x \in S_2 \setminus \{v\} \\ 0 & \text{otherwise} \end{cases}$$

$$d_x^{(12)} = \begin{cases} \frac{s_x}{\tau} & x \in S_1 \\ -\frac{s_x}{\tau} & x \in S_2 \\ 0 & \text{otherwise} \end{cases}$$

Notice that $d^{(1)} + d^{(2)} + d^{(12)} = \chi$. Furthermore, notice that $d^{(1)} = \sum_{x \in S_1} p_x \chi^{(ux)}$ and $d^{(2)} = \sum_{y \in S_2} q_y \chi^{(yv)}$, where $\chi^{(ab)}$ is the signed indicator vector of the edge from a to b and $\sum_{x \in S_1} p_x =$

1, $\sum_{y \in S_2} q_y = 1$, and $p_x \geq 0$ and $q_y \geq 0$ for all $x \in S_1, y \in S_2$. The function $f(d) = d^\top L^\dagger d$ is convex, so by Jensen's Inequality,

$$\begin{aligned}(d^{(1)})^\top L_G^\dagger (d^{(1)}) &\leq \sum_{x \in S_1} p_x (\chi^{(ux)})^\top L_G^\dagger \chi^{(ux)} \\ &\leq \sum_{x \in S_1} p_x R_1 \\ &\leq R_1\end{aligned}$$

and $(d^{(2)})^\top L_G^\dagger (d^{(2)}) \leq R_2$. Let $f \in \mathbb{R}^{|E(G)|}$ be the vector with $f_{xy} = \frac{w_{xy}}{\tau}$ for all $x \in S_1, y \in S_2$ and let $f_e = 0$ for all other $e \in E(G)$. By definition of the s_us, f is a feasible flow for the electrical flow optimization problem for the demand vector $d^{(12)}$. Therefore,

$$\begin{aligned}(d^{(12)})^\top L_G^\dagger d^{(12)} &\leq \sum_{x \in S_1, y \in S_2} \frac{f_{xy}^2}{w_{xy}} \\ &= \sum_{x \in S_1, y \in S_2} \frac{w_{xy}}{\tau^2} \\ &= \frac{1}{\tau}\end{aligned}$$

so we can upper bound $\texttt{Reff}_G(u, v)$ in the following way

$$\begin{aligned}\texttt{Reff}_G(u, v) &= (d^{(1)} + d^{(2)} + d^{(12)})^\top L_G^\dagger (d^{(1)} + d^{(2)} + d^{(12)}) \\ &\leq 3(d^{(1)})^\top L_G^\dagger d^{(1)} + 3(d^{(2)})^\top L_G^\dagger d^{(2)} + 3(d^{(12)})^\top L_G^\dagger d^{(12)} \\ &\leq 3R_1 + 3R_2 + 3/\tau\end{aligned}$$

as desired. □

7.4 Sampling data structure

In this section, we give a data structure for efficiently sampling pairs of points $(u, v) \in \mathbb{R}^d \times \mathbb{R}^d$ with probability proportional to a constant-factor approximation of $|\langle u, v \rangle|$:

Lemma 7.13. *Given a pair of sets $S_1, S_2 \subseteq \mathbb{R}^d$, there is a data structure that can be constructed in $\widetilde{O}(d|S_1| + |S_2|)$ time that, in $\text{poly}(d \log(|S_1| + |S_2|))$ time per sample, independently samples pairs $u \in S_1, v \in S_2$ with probability p_{uv}, where*

$$\frac{1}{2} \frac{|\langle u, v \rangle|}{\sum_{a \in S_1, b \in S_2} |\langle a, b \rangle|} \leq p_{uv} \leq 2 \frac{|\langle u, v \rangle|}{\sum_{a \in S_1, b \in S_2} |\langle a, b \rangle|}$$

Furthermore, it is possible to query the probability p_{uv} in $\text{poly}(d \log(|S_1| + |S_2|))$ time.

To produce this data structure, we use the following algorithm for sketching ℓ_1-norms:

Theorem 7.14 (Theorem 3 in [160])**.** *An efficiently computable, $\text{poly}(\log d, 1/\varepsilon)$-space linear sketch exists for the ℓ_1 norm. That is, given a $d \in \mathbb{Z}_{\geq 1}$, $\delta \in (0, 1)$, and $\varepsilon \in (0, 1)$, there is a matrix $C = \textsc{SketchMatrix}(d, \delta, \varepsilon) \in \mathbb{R}^{\ell \times d}$ and an algorithm $\textsc{RecoverNorm}(s, d, \delta, \varepsilon)$ with the following properties:*

1. *(Approximation) For any vector $v \in \mathbb{R}^d$, with probability at least $1-\delta$ over the randomness of* SKETCHMATRIX*, the value $r = \text{RECOVERNORM}(Cv, d, \delta, \varepsilon)$ is as follows:*

$$(1-\varepsilon)\|v\|_1 \le r \le (1+\varepsilon)\|v\|_1$$

2. *$\ell = (c/\varepsilon^2)\log(1/\delta)$ for some constant $c > 1$*
3. *(Runtime)* SKETCHMATRIX *and* RECOVERNORM *take $\widetilde{O}(\ell d)$ and $\text{poly}(\ell)$ time respectively.*

We use this sketching algorithm to obtain the desired sampling algorithm in the following subroutine:

Corollary 7.14.1. *Given a set $S \subseteq \mathbb{R}^d$, an $\varepsilon \in (0,1)$, and a $\delta \in (0,1)$, there exists a data structure which, when given a query point $u \in \mathbb{R}^d$, returns a $(1\pm\varepsilon)$-multiplicative approximation to $\sum_{v\in S} |\langle u, v\rangle|$ with probability at least $1-\delta$. This data structure can be computed in $O(\ell d|S|)$ preprocessing time and takes $\text{poly}(\ell d)$ time per query, where $\ell = O(\varepsilon^{-2}\log(1/\delta))$.*

Proof. Let $n = |S|$ and $C = \text{SKETCHMATRIX}(n, \delta, \varepsilon)$. We will show that the following algorithm returns the desired estimate with probability at least $1-\delta$:

1. Preprocessing:
   (a) Index the rows of C by integers between 1 and ℓ. Index columns of C by points $v \in S$.
   (b) Compute the vector $x^{(i)} = \sum_{v\in S} C_{iv} v$ for each $i \in \ell$, where $\ell = (c/\varepsilon^2)\log(1/\delta)$ for the constant c in Theorem 7.14.
2. Given a query point $u \in \mathbb{R}^d$,
   (a) Let $y \in \mathbb{R}^\ell$ be a vector with $y_i = \langle u, x^{(i)}\rangle$ for each $i \in [\ell]$.
   (b) Return $\text{RECOVERNORM}(y, n, \delta, \varepsilon)$

Approximation. Let $w \in \mathbb{R}^n$ be the vector with $w_v = \langle u, v\rangle$ for each $v \in S$. It suffices to show that the number that a query returns is a $(1\pm\varepsilon)$-approximation to $\|w\|_1$. By definition, $y_i = \sum_{v\in S} C_{iv} w_v$ for all $i \in [\ell]$ and $v \in S$, so $y = Cw$. Therefore, by the *Approximation* guarantee of Theorem 7.14, $\text{RECOVERNORM}(y, n, \delta, \varepsilon)$ returns a $(1\pm\varepsilon)$-approximation to $\|w\|_1$ with probability at least $1-\delta$, as desired.

Preprocessing time. Computing the matrix C takes $\widetilde{O}(n\ell)$ time by the *Runtime* guarantee of Theorem 7.14. Computing the vectors $x^{(i)}$ for all $i \in [\ell]$ takes $O(d\ell n)$ time. This is all of the preprocessing steps, so the total runtime is $\widetilde{O}(d\ell n)$, as desired.

Query time. Each query consists of ℓ inner products of d dimensioal vectors and one call to RECOVERNORM, for a total of $O(\ell d + \text{poly}(\ell))$ work, as desired. □

We use this corollary to obtain a sampling algorithm as follows, where $n = |S_1| + |S_2|$:

1. Preprocessing:
   (a) Use the Corollary 7.14.1 data structure to $(1\pm 1/(100\log n))$-approximate $\sum_{v\in S_2} |\langle u, v\rangle|$ for each $u \in S_1$. Let t_u be this estimate for each $u \in S_1$. (one preprocess for $S \leftarrow S_2$, $|S_1|$ queries).

(b) Form a balanced binary tree $\mathcal{T}$ of subsets of S_2, with S_2 at the root, the elements of S_2 at the leaves, and the property that for every parent-child pair (P, C), $|C| \leq 2|P|/3$.

(c) For every node S in the binary tree, construct a $(1 \pm 1/(100 \log n))$-approximate data structure for S.

2. Sampling query:

(a) Sample a point $u \in S_1$ with probability $t_u/(\sum_{a \in S_1} t_a)$.

(b) Initialize $S \leftarrow S_2$. While $|S| > 1$,

i. Let P_1 and P_2 denote the two children of S in $\mathcal{T}$

ii. Let s_1 and s_2 be the $(1 \pm 1/(100 \log n))$-approximations to $\sum_{v \in P_1} |\langle u, v \rangle|$ and $\sum_{v \in P_2} |\langle u, v \rangle|$ respectively obtained from the data structure for S computed during preprocessing.

iii. Reset S to P_1 with probability $s_1/(s_1 + s_2)$; otherwise reset S to P_2.

(c) Return the single element in S.

3. p_{uv} query:

(a) Return the product of the $O(\log n)$ probabilities attached to ancestor nodes of the node $\{v\}$ in $\mathcal{T}$ for u, obtained from the preprocessing step (as during the sampling query)

$s_i/(s_1 + s_2)$ is a $(1 \pm 1/(100 \log n))^2$-approximation to $\Pr[v \in P_i | v \in S]$ for each $i \in \{1, 2\}$. A $v \in S_2$ is sampled with probability proportional to the product of these conditional probabilities for the ancestors, for which p_{uv} is a $(1 + 1/(100 \log n))^{2 \log n + 1} \leq 2$-approximation. The total preprocessing time is proportional to the size of all sets in the tree, which is at most $O(|S_2| \log |S_2|)$. The total query time is also polylog$(|S_1| + |S_2|)\text{poly}(d)$ due to the logarithmic depth of the query binary tree.

We now expand on this intuition to prove Lemma 7.13:

Proof of Lemma 7.13. We show that the algorithm given just before this proof satisfies this lemma:

Probability guarantee. Consider a pair $u \in S_1$, $v \in S_2$. Let $A_0 = S_2, A_1, \ldots A_{k-1}, A_k = \{v\}$ denote the sequence of ancestor sets of the singleton set $\{v\}$ in $\mathcal{T}$. For each $i \in [k]$, let B_i be the child of A_{i-1} besides A_i (unique because $\mathcal{T}$ is binary). For a node X of $\mathcal{T}$, let s_X be the $(1 \pm 1/(100 \log n))$-approximation to $\sum_{v \in X} |\langle u, v \rangle|$ used by the algorithm. The sampling probability p_{uv} is the following product of probabilities:

$$p_{uv} = \frac{t_u}{\sum_{a \in S_1} t_a} \prod_{i=1}^{k} \frac{s_{A_i}}{s_{A_i} + s_{B_i}}$$

$s_{A_i} + s_{B_i}$ is a $(1 \pm 1/(100 \log n))$-approximation to $\sum_{v \in A_{i-1}} |\langle u, v \rangle|$ by the approximation guarantee of Corollary 7.14.1. s_{A_i} is a $(1 \pm 1/(100 \log n))$-approximation to $\sum_{v \in A_i} |\langle u, v \rangle|$ by the approximation guarantee of Corollary 7.14.1. By these guarantees and the approximation guarantees for the t_as for $a \in S_1$, p_{uv} is a $(1 \pm 1/(100 \log n))^{2k+2} \leq (1 \pm 1/2)$-approximation to

$$\frac{\sum_{b\in S_2}|\langle u,b\rangle|}{\sum_{a\in S_1,b\in S_2}|\langle a,b\rangle|}\prod_{i=1}^{k}\frac{\sum_{b\in A_i}|\langle u,b\rangle|}{\sum_{b\in A_{i-1}}|\langle u,b\rangle|}=\frac{|\langle u,v\rangle|}{\sum_{a\in S_1,b\in S_2}|\langle a,b\rangle|}$$

as desired, since $k\le\log n$.

Preprocessing time. Let $\delta=1/n^{1000}$ and $\ell=(c/\varepsilon^2)\log(1/\delta)$, where c is the constant from Theorem 7.14. Approximating all t_us takes $\widetilde{O}(\ell d|S_2|)+|S_1|\mathrm{poly}(\ell d)=n\mathrm{poly}(\ell d)$ time by Corollary 7.14.1. Preprocessing the data structures for each node $S\in\mathcal{T}$ takes $\sum_{S\in\mathcal{T}}O(\ell d|S|)$ time in total. Each member of S_2 is in at most $O(\log n)$ sets in $\mathcal{T}$, since $\mathcal{T}$ is a balanced binary tree. Therefore, $\sum_{S\in\mathcal{T}}O(\ell d|S|)\le\widetilde{O}(n\ell d)$, so the total preprocessing time is $\widetilde{O}(n\ell d)$, as desired.

Query time. $O(\log n)$ of the data structures attached to sets in $\mathcal{T}$ are queried per sample query, for a total of $O(\log n)\mathrm{poly}(\ell d)$ time by Corollary 7.14.1, as desired.

□

We use this data structure via a simple reduction to implement the following data structure, which suffices for our applications:

Proposition 7.15. *Given a family $\mathcal{G}$ of sets in $\mathbb{R}^d$, a collection of positive real numbers $\{\gamma_S\}_{S\in\mathcal{G}}$, and a set $S_2\subseteq\mathbb{R}^d$, there is a data structure that can be constructed in $\widetilde{O}(d(|S_2|+\sum_{S\in\mathcal{G}}|S|))$ time that, in $\mathrm{poly}(d\log(|S_2|+\sum_{S\in\mathcal{G}}|S|))$ time per sample, independently samples pairs $u\in S$ for some $S\in\mathcal{G}$, $v\in S_2$ with probability p_{uv}, where*

$$\frac{\gamma_S|\langle u,v\rangle|}{(\sum_{A\in\mathcal{G}}\gamma_A)\sum_{a\in A,b\in S_2}|\langle a,b\rangle|}$$

is a 2-approximation to p_{uv}. Furthermore, it is possible to query the number p_{uv} in time $\mathrm{poly}(d\log(|S_2|+\sum_{S\in\mathcal{G}}|S|))$ time.

Proof. Define a new set $S_1\subseteq\mathbb{R}^{d+\log n}$ as follows, where $n=|S_2|+\sum_{S\in\mathcal{G}}|S|$:

$$S_1=\cup_{S\in\mathcal{G}}\{f_S(u)\forall u\in S\}$$

where the function $f_S:\mathbb{R}^d\to\mathbb{R}^d$ is defined as follows for any set $S\in\mathcal{G}$:

$$f_S(u)=\frac{\gamma_S(u,\mathrm{id}(u))}{\sum_{a\in S,b\in S_2}|\langle a,b\rangle|}$$

where $\mathrm{id}:\cup_{S\in\mathcal{G}}S\to\{0,1\}^{\log n}$ is a function that outputs a unique ID for each element of $\cup_{S\in\mathcal{G}}S$. Define $f(u)=f_S(u)$ for the unique S containing u (Without loss of generality assume that exactly one S contains u). Let $S_2'=\{(x,0^{\log n})\forall x\in S_2\}$. Construct the data structure $\mathcal{D}$ from Lemma 7.13 on the pair of sets S_1,S_2'. Now, sample a pair of d-dimensional vectors as follows:

1. Sample:
   (a) Sample a pair $(x,(y,0^{\log n}))$ from $\mathcal{D}$, where $y\in S_2$ and $x\in\mathbb{R}^{d+\log n}$.
   (b) Since the function id outputs values that are not proportional to one other, the function f is injective.
   (c) Return the pair $(f^{-1}(x),y)$.

(d) (For the proof, let $w = f^{-1}(x)$ and let S be the unique set for which $w \in S$)

This data structure has preprocessing time $\widetilde{O}(d(|S_2'| + |S_1|)) = \widetilde{O}(d(|S_2| + \sum_{S \in \mathcal{G}} |S|))$ and sample query time $\mathrm{poly}(d \log n)$ by Lemma 7.13, as desired. Therefore, we just need to show that it samples a pair (w, y) with the desired probability. By the probability guarantee for Lemma 7.13 and the injectivity of the mapping f combined over all $S \in \mathcal{G}$, p_{wy} is 2-approximated by

$$\frac{|\langle x, (y, 0^{\log n})\rangle|}{\sum_{a \in S_1, b \in S_2'} |\langle a, b\rangle|} = \frac{\gamma_S |\langle w, y\rangle|}{\sum_{A \in \mathcal{G}} \sum_{p \in A, q \in S_2} |\langle p, q\rangle|}$$

as desired. □

7.5 Weighted IP graph sparsification

In this section, we use the tools developed in the previous sections to sparsify weighted inner product graphs. To modularize the exposition, we define a partition of the edge set of a weighted inner product graph:

Definition 7.16. *For a w-weighted graph G and three functions on pairs of vertex sets ζ, κ, δ, a collection of vertex set family-vertex set pairs $\mathcal{F}$ is called a (ζ, κ, δ)-*cover *for G iff the following property holds:*

1. *(Coverage) For any $e = \{u, v\} \in E(G)$, there exists a pair $(\mathcal{G}, S_1) \in \mathcal{F}$ and an $S_0 \in \mathcal{G}$ for which $u \in S_0, v \in S_1$ or $u \in S_1, v \in S_0$, and*

$$\mathtt{Reff}_G(u, v) \leq \frac{\delta(S_0, S_1)}{w_{uv}} + \frac{\kappa(S_0, S_1)}{\max_{x \in S_0, y \in S_1} w_{xy}} + \frac{\zeta(S_0, S_1)}{\sum_{x \in S_0, y \in S_1} w_{xy}}$$

*A (ζ, κ, δ)-cover is said to be s-*sparse *if $\sum_{(\mathcal{G}, S_1) \in \mathcal{F}} \sum_{S_0 \in \mathcal{G}} ((\delta(S_0, S_1) + \kappa(S_0, S_1))|S_0||S_1| + \zeta(S_0, S_1)) \leq s$. A (ζ, κ)-cover is said to be w-*efficient *if $\sum_{(\mathcal{G}, S_1) \in \mathcal{F}} \left(|S_1| + \sum_{S_0 \in \mathcal{G}} |S_0|\right) \leq w$. When $\delta = 0$, we simplify notation to refer to (ζ, κ)-covers instead.*

Given a (ζ, κ)-cover for a weighted or unweighted inner product graph, one can sparsify it using Theorem 3.6 and the sampling data structure from Proposition 7.15:

Proposition 7.17. *Given a set $X \subseteq \mathbb{R}^d$ with $n = |X|$ and an s-sparse w-efficient (ζ, κ, δ)-cover $\mathcal{F}$ for the weighted inner product graph G on X, and $\varepsilon, \delta \in (0, 1)$, there is an*

$$\mathrm{poly}(d, \log n, \log s, \log w, \log(1/\delta))(s + w + n)/\varepsilon^4$$

time algorithm for constructing an $(1 \pm \varepsilon)$-sparsifier for G with $O(n \log n/\varepsilon^2)$ edges with probability at least $1 - \delta$.

Proof. **Filling in algorithm details (the bolded parts)**. We start by filling in the details in the algorithm

OVERSAMPLINGWITHCOVER. First, we define r_{uv} for each pair of distinct $u, v \in X$. $\{u, v\}$ is a weighted edge in G with weight w_{uv}. Define

Algorithm 4

procedure OVERSAMPLINGWITHCOVER($X, \mathcal{F}, \varepsilon, \delta$)
    (for analysis only: define r_{uv} for each pair $u, v \in X$ as in proof)
    Construct the Proposition 7.15 data structure $\mathcal{D}_{(\mathcal{G},S_1)}$ with $\gamma_S = \zeta(S, S_1)$ for each $S \in \mathcal{G}$
    $t \leftarrow \sum_{u,v \in X} r_{uv}$
    $q \leftarrow C \cdot \varepsilon^{-2} \cdot t \log t \cdot \log(1/\delta)$
    Initialize H to be an empty graph
    for $i = 1 \to q$ **do**
        Sample one $e = \{u, v\} \in X \times X$ with probability r_e/t **by sampling $\{u, v\}$ uniformly or from some data structure $\mathcal{D}_{(\mathcal{G},S_1)}$ (see proof for details)**
        Add that edge with weight $w_e t/(r_e q)$ to graph H (note: r_e can be computed in $\mathrm{poly}(d, \log w)$ by the p_{uv} query time in Prop 7.15)
    end for
    return Spielman-Srivastava [269] applied to H
end procedure

$$r_{uv} = 2 \sum_{(\mathcal{G},S_1) \in \mathcal{F}} \sum_{S_0 \in \mathcal{G}: u \in S_0, v \in S_1 \text{ or } v \in S_0, u \in S_1} \left(\delta(S_0, S_1) + \kappa(S_0, S_1) + \left(\sum_{A \in \mathcal{G}} \zeta(A, S_1) \right) p_{uv}^{(\mathcal{G},S_1)} \right)$$

where $p_{uv}^{(\mathcal{G},S_1)}$ is the probability p_{uv} defined for the data structure $\mathcal{D}^{(\mathcal{G},S_1)}$ in Proposition 7.15. Next, we fully describe how to sample pairs $\{u, v\}$ with probability proportional to r_{uv}. Notice that t can be computed in $O(w)$ time because

$$\begin{aligned} t &= 2 \sum_{u,v \in X} r_{uv} \\ &= 2 \sum_{(\mathcal{G},S_1) \in \mathcal{F}} \sum_{S_0 \in \mathcal{G}} \sum_{u \in S_0, v \in S_1} \left(\delta(S_0, S_1) + \kappa(S_0, S_1) + \left(\sum_{A \in \mathcal{G}} \zeta(A, S_1) \right) p_{uv}^{(\mathcal{G},S_1)} \right) \\ &= 2 \sum_{(\mathcal{G},S_1) \in \mathcal{F}} \left(\left(\sum_{A \in \mathcal{G}} \zeta(A, S_1) \right) + \sum_{S_0 \in \mathcal{G}} |S_0||S_1| (\delta(S_0, S_1) + \kappa(S_0, S_1)) \right) \end{aligned}$$

can be computed in $O(w)$ time. Sample a pair $\{u, v\}$ with probability equal to r_{uv}/t as follows:

1. Sample a pair:
    (a) Sample a Bernoulli $b \sim \text{Bernoulli}\left(\frac{1}{t} \sum_{(\mathcal{G},S_1) \in \mathcal{F}} \left(\sum_{A \in \mathcal{G}} \zeta(A, S_1) \right) \right)$.
    (b) If $b = 1$
        i. Sample a pair $(\mathcal{G}, S_1) \in \mathcal{F}$ with probability proportional to $\sum_{A \in \mathcal{G}} \zeta(A, S_1)$.

ii. Sample the pair (u, v) using the data structure $\mathcal{D}^{(\mathcal{G},S_1)}$.

(c) Else

i. Sample a pair $(\mathcal{G}, S_1) \in \mathcal{F}$ with probability proportional to $\sum_{S_0 \in \mathcal{G}} |S_0||S_1|(\delta(S_0, S_1) + \kappa(S_0, S_1))$.

ii. Sample an $S_0 \in \mathcal{G}$ with probability proportional to $|S_0|(\delta(S_0, S_1) + \kappa(S_0, S_1))$.

iii. Sample $(u, v) \in S_0 \times S_1$ uniformly.

All sums in the above sampling procedure can be precomputed in $\text{poly}(d)w$ time. After doing this precomputation, each sample from the above procedure takes $\text{poly}(d, \log n, \log w)$ time by to Proposition 7.15 for the last step in the if statement and uniform sampling from $[0, 1]$ with intervals otherwise.

Sparsifier correctness. By the *Coverage* guarantee of $\mathcal{F}$ and the approximation guarantee for the p_{uv}s in Proposition 7.15, $w_{uv}\texttt{Reff}_G(u, v) \leq r_{uv}$ for all $u, v \in X$. Therefore, Theorem 3.6 applies and shows that the graph H returned is a $(1 \pm \varepsilon)$-sparsifier for G with probability at least $1 - \delta$. Spielman-Srivastava only worsens the approximation guarantee by a $(1 + \varepsilon)$ factor, as desired.

Number of edges in H. It suffices to bound q. In turn, it suffices to bound t. Recall from above that

$$\begin{aligned} t &= 2 \sum_{(\mathcal{G},S_1)\in\mathcal{F}} \left(\left(\sum_{A \in \mathcal{G}} \zeta(A, S_1) \right) + \sum_{S_0 \in \mathcal{G}} |S_0||S_1|(\delta(S_0, S_1) + \kappa(S_0, S_1)) \right) \\ &= 2 \sum_{(\mathcal{G},S_1)\in\mathcal{F}} \left(\sum_{S_0 \in \mathcal{G}} (|S_0||S_1|(\delta(S_0, S_1) + \kappa(S_0, S_1)) + \zeta(S_0, S_1)) \right) \\ &\leq 2s \end{aligned}$$

since $\mathcal{F}$ is s-sparse. Therefore, $q \leq \text{poly}(d, \log s, \log 1/\delta)s/\varepsilon^2$.

Runtime. We start by bounding the runtime to produce H. Constructing the data structure $\mathcal{D}_{(\mathcal{G},S_1)}$ takes $\text{poly}(d, \log n, \log w)(|S_1| + \sum_{S_0 \in \mathcal{G}} |S_0|)$ time by Proposition 7.15. Therefore, the total time to construct all data structures is at most $\text{poly}(d, \log n, \log w)w$. Computing t and q, as discussed above, takes $O(w)$ time. $q \leq \text{poly}(d, \log s, \log 1/\delta)s/\varepsilon^2$ as discussed above and each iteration of the for loop takes $\text{poly}(d, \log n, \log w)$ time by the query complexity bounds of Proposition 7.15. Therefore, the total time required to produce H is at most $\text{poly}(d, \log n, \log s, \log w, \log 1/\delta)(s + w)/\varepsilon^2$. Running Spielman-Srivastava requires an additional $\text{poly}(\log s, \log n)(s/\varepsilon^2 + n)/\varepsilon^2$ time, for a total of $\text{poly}(d, \log n, \log s, \log w, \log 1/\delta)(s + w)/\varepsilon^4$ time, as desired. □

Therefore, to sparsify weighted inner product graphs, it suffices to construct an $\text{poly}(d, \log n)n$-sparse, $\text{poly}(d, \log n)n$-efficient (ζ, κ)-cover. We break up this construction into a sequence of steps:

(ζ, κ)-cover for unweighted IP graphs

We start by constructing covers for unweighted inner product graphs. The algorithm repeatedly peels off sets constructed using LOWDIAMSET and returns all pairs of such sets. The sparsity of the cover is bounded due to Proposition 7.12. The efficiency of the cover is bounded thanks to a $\mathrm{poly}(d, \log n)$ bound on the number of while loop iterations, which in turn follows from the *Size* guarantee of Proposition 7.9.

Proposition 7.18 (Cover for unweighted graphs)**.** *Given a set $X \subseteq \mathbb{R}^d$ with $|X| = n$, there is an $\mathrm{poly}(d, \log n)$-time algorithm* UNWEIGHTEDCOVER(X) *that, with probability at least $1 - 1/\mathrm{poly}(n)$, produces an $\mathrm{poly}(d, \log n)n$-sparse $\mathrm{poly}(d, \log n)n$-efficient (ζ, κ)-cover for the unweighted inner product graph G on X.*

Algorithm 5

procedure UNWEIGHTEDCOVER(X)
  Input: $X \subseteq \mathbb{R}^d$
  Output: An sparse, efficient (ζ, κ) cover for the unweighted inner product graph G on X
  $\mathcal{U} \leftarrow \emptyset$
  $Y \leftarrow X$
  while $Y \neq \emptyset$ **do** ▷ Finding expanders
    Let $Q \leftarrow$ LOWDIAMSET(Y)
    Add the set Q to $\mathcal{U}$
    Remove the vertices Q from Y
  end while
  return $\{(\mathcal{U}, S) : \forall S \in \mathcal{U}\}$
end procedure

Proof. Let $\mathcal{F} =$ UNWEIGHTEDCOVER(X) and define the functions ζ, κ as follows: $\zeta(S_0, S_1) = 3$ and $\kappa(S_0, S_1) = C_2 \left(\frac{3}{|S_0|} + \frac{3}{|S_1|}\right)$ for any pair of sets $S_0, S_1 \subseteq X$. Recall that C_2 is defined in the statement of Proposition 7.8.

Number of while loop iterations. We start by showing that there are at most $\mathrm{poly}(d, \log n)$ while loop iterations with probability at least $1 - 1/\mathrm{poly}(n)$. By the *Size* guarantee of Proposition 7.8, when LOWDIAMSET succeeds (which happens with probability $1 - 1/\mathrm{poly}(n)$), Y decreases in size by a factor of at least $1 - 1/p(d, \log n)$ for some fixed constant degree polynomial p. Therefore, after $p(d, \log n) \log n = \mathrm{poly}(d, \log n)$ iterations, Y will be empty, as desired. Therefore, $|\mathcal{U}| \leq \mathrm{poly}(d, \log n)$.

Runtime. The runtime follows immediately from the bound on the number of while loop iterations and the runtime bound on LOWDIAMSET from Proposition 7.8.

Coverage. For each $S \in \mathcal{U}$, $\max_{u,v \in S} \mathtt{Reff}_G(u, v) \leq \frac{C_2}{|S|}$ by the *Low effective resistance diameter* guarantee of Proposition 7.8. Plugging this into Proposition 7.12 immediately shows that $\mathcal{F}$ is a (ζ, κ)-cover for G.

Sparsity bound. We bound the desired quantity directly using the fact that $|\mathcal{U}| \leq \mathrm{poly}(d, \log n)$:

$$\sum_{(\mathcal{U},S_1)\in\mathcal{F}}\sum_{S_0\in\mathcal{U}}(\kappa(S_0,S_1)|S_0||S_1|+\zeta(S_0,S_1)) \le \text{poly}(d,\log n)\sum_{(\mathcal{U},S_1)\in\mathcal{F}}\sum_{S_0\in\mathcal{U}}(|S_0|+|S_1|+1)$$
$$\le \text{poly}(d,\log n)n$$

as desired. □

Efficiency bound. Follows immediately from the bound on $|\mathcal{U}|$.

(ζ,κ)-cover for weighted IP graphs on bounded-norm vectors

Given a covers for unweighted inner product graphs, it is easy to construct covers for weighted inner product graphs on bounded norm vectors simply by removing edge weights and producing the cover. Edge weights only differ by a factor of $O(d)$ in these two graphs, so effective resistances also differ by at most that amount. Note that the following algorithm also works for vectors with norms between z and $2z$ for any real number z.

Proposition 7.19 (Weighted bounded norm cover)**.** *Given a set $X \subseteq \mathbb{R}^d$ with $1 \le \|u\|_2 \le 2$ for all $u \in X$ and $|X| = n$, there is an $n\text{poly}(d,\log n)$ time algorithm* BOUNDEDCOVER(X) *that produces a $\text{poly}(d,\log n)$-sparse, $\text{poly}(d,\log n)$-efficient (ζ,κ)-cover for the weighted inner product graph G on X.*

Proof. Let G_0 be the unweighted inner product graph on X. Let G_1 be the weighted graph G with all edges that are not in G_0 deleted. Let $\mathcal{F}$ be the (ζ_0,κ_0)-cover given by Proposition 7.18 for G_0. Let this cover be the output of BOUNDEDCOVER(X). It suffices to show that $\mathcal{F}$ is a (ζ,κ)-cover for G, where $\zeta = 8d\zeta_0$ and $\kappa = 8d\kappa_0$. By Rayleigh monotonicity,

$$\texttt{Reff}_G(u,v) \le \texttt{Reff}_{G_1}(u,v)$$

for all $u,v \in X$. Let w_e denote the weight of the edge e in G_1. For all edges e in G_1, $\frac{1}{d+1} \le w_e \le 4$ by the norm condition on X. Therefore, for all $u,v \in X$,

$$\texttt{Reff}_{G_1}(u,v) \le (d+1)\texttt{Reff}_{G_0}(u,v)$$

By the *Coverage* guarantee on $\mathcal{F}$, there exists a pair $(\mathcal{G},S_1)$ and an $S_0 \in \mathcal{G}$ for which $u \in S_0, v \in S_1$ or $v \in S_0, u \in S_1$ and

$$\texttt{Reff}_{G_1}(u,v) \le (d+1)\left(\kappa_0(S_0,S_1) + \frac{\zeta_0(S_0,S_1)}{|S_0||S_1|}\right)$$

By the upper bound on the edge weights for G_1,

$$\texttt{Reff}_{G_1}(u,v) \le 4(d+1)\left(\frac{\kappa_0(S_0,S_1)}{\max_{x\in S_0,y\in S_1} w_{xy}} + \frac{\zeta_0(S_0,S_1)}{\sum_{x\in S_0,y\in S_1} w_{xy}}\right)$$

Since $4(d+1) \le 8d$, $\mathcal{F}$ is a (ζ,κ)-cover for G as well, as desired. □

(ζ, κ)-cover for weighted IP graphs on vectors with norms in the set $[1,2] \cup [z,2z]$ for any $z > 1$

By the previous subsection, it suffices to cover the pairs (u, v) for which $\|u\|_2 \in [1,2]$ and $\|v\|_2 \in [z, 2z]$. This can be done by clustering using LOWDIAMSET on the $[z,2z]$-norm vectors. For each cluster S_1, let $\mathcal{G} = \{\{u\} : \forall u \in X \text{ with } \|u\|_2 \in [1,2]\}$. This cover is sparse because of the fact that the clusters have low effective resistance diameter. It is efficient because of the small number of clusters.

Proposition 7.20 (Two-scale cover)**.** *Given a set $X \subseteq \mathbb{R}^d$ for which $|X| = n$ and $\|u\|_2 \in [1,2] \cup [z,2z]$ for all $u \in X$, there is a $\text{poly}(d, \log n)n$-time algorithm* TWOBOUNDEDCOVER(X) *that produces an $\text{poly}(d, \log n)n$-sparse, $\text{poly}(d, \log n)$-efficient (ζ, κ)-cover for the weighted inner product graph G on X.*

Algorithm 6

procedure TWOBOUNDEDCOVER(X)
  Input: $X \subseteq \mathbb{R}^d$, where $\|u\|_2 \in [1,2] \cup [z,2z]$ for all $u \in X$
  Output: An sparse, efficient (ζ, κ) cover for the weighted inner product graph G on X
  $X_{\text{low}} \leftarrow \{u \in X : \|u\|_2 \in [1,2]\}$
  $X_{\text{high}} \leftarrow \{u \in X : \|u\|_2 \in [z,2z]\}$
  $\mathcal{U} \leftarrow \emptyset$
  $Y \leftarrow X_{\text{high}}$
  while $Y \neq \emptyset$ **do**
    Let $Q \leftarrow$ LOWDIAMSET(Y)
    Add the set Q to $\mathcal{U}$
    Remove the vertices Q from Y
  end while
  return $\{(\{\{u\} \forall u \in X_{\text{low}}\}, S_1) : \forall S_1 \in \mathcal{U}\}$
  $\cup$BOUNDEDCOVER$(X_{\text{low}}) \cup$ BOUNDEDCOVER(X_{high})
end procedure

Proof. Suppose that the BOUNDEDCOVERs returned for X_{low} and X_{high} are $(\zeta_{\text{low}}, \kappa_{\text{low}})$ and $(\zeta_{\text{high}}, \kappa_{\text{high}})$-covers respectively. Recall the value $C_2 \leq \text{poly}(d, \log n)$ from the statement of Proposition 7.8. Let $w_{uv} = |\langle u, v \rangle|$ denote the weight of the u-v edge in G. Let $\mathcal{F} =$ TWOBOUNDEDCOVER(X) and define the functions ζ, κ as follows:

$$\zeta(S_0, S_1) = \begin{cases} \zeta_{\text{low}}(S_0, S_1) & \text{if } S_0, S_1 \subseteq X_{\text{low}} \\ \zeta_{\text{high}}(S_0, S_1) & \text{if } S_0, S_1 \subseteq X_{\text{high}} \\ 3 & \text{if } S_0 \subseteq X_{\text{low}} \text{ and } S_1 \subseteq X_{\text{high}} \\ \infty & \text{otherwise} \end{cases}$$

$$\kappa(S_0, S_1) = \begin{cases} \kappa_{\text{low}}(S_0, S_1) & \text{if } S_0, S_1 \subseteq X_{\text{low}} \\ \kappa_{\text{high}}(S_0, S_1) & \text{if } S_0, S_1 \subseteq X_{\text{high}} \\ \frac{24dC_2}{|S_1|} & \text{if } S_0 \subseteq X_{\text{low}} \text{ and } S_1 \subseteq X_{\text{high}} \\ \infty & \text{otherwise} \end{cases}$$

Number of while loop iterations. A $\text{poly}(d, \log n)$-round bound follows from the *Size* bound of Proposition 7.8. For more details, see the same part of the proof of Proposition 7.18, which used the exact same algorithm for producing $\mathcal{U}$.

Runtime. Follows immediately from the runtime bounds of LOWDIAMSET, BOUNDEDCOVER, and the number of while loop iterations.

Coverage. Consider a pair $u, v \in X$. We break the analysis up into cases:

Case 1: $u, v \in X_{\text{low}}$. In this case, the *Coverage* property of BOUNDEDCOVER(X_{low}) implies that the pair (u, v) is covered in $\mathcal{F}$ by Rayleigh monotonicity (since G_{low} is a subgraph of G, where G_{low} is the weighted inner product graph for X_{low}).

Case 2: $u, v \in X_{\text{high}}$. In this case, the *Coverage* property of BOUNDEDCOVER(X_{high}) implies that the pair (u, v) is covered in $\mathcal{F}$ by Rayleigh monotonicity.

Case 3: $u \in X_{\text{low}}$ and $v \in X_{\text{high}}$. Since $\mathcal{U}$ is a partition of X_{high}, there is a unique pair $(\mathcal{G}, S_1) \in \mathcal{F}$ for which $\{u\} \in \mathcal{G}$ and $v \in S_1$. Let H denote the unweighted inner product graph on X_{high}. By Rayleigh monotonicity, the fact that $\frac{z^2}{2d} \leq \frac{z^2}{d+1} \leq w_{xy}$ for all $\{x, y\} \in E(H)$, and the *Low effective resistance diameter* guarantee of Proposition 7.8,

$$\begin{aligned} \texttt{Reff}_G(x, y) &\leq \texttt{Reff}_{G_{\text{high}}}(x, y) \\ &\leq \frac{2d\texttt{Reff}_H(x, y)}{z^2} \\ &\leq \frac{2dC_2}{z^2|S_1|} \end{aligned}$$

for any $x, y \in S_1$. Since $z > 1$, $w_{xy} \leq 4z^2$ for all $x, y \in X$. Therefore,

$$\texttt{Reff}_G(x, y) \leq \frac{8dC_2}{\max_{p \in X_{\text{low}}, q \in S_1} w_{pq}}$$

for any $x, y \in S_1$. Therefore, Proposition 7.12 implies the desired *Coverage* bound in this case.

Sparsity bound. We use the fact that $|\mathcal{U}| \leq \text{poly}(d, \log n)$ along with sparsity bounds for BOUNDEDCOVER from Proposition 7.19 to bound the sparsity of $\mathcal{F}$ as follows:

$$\begin{aligned}\sum_{(\mathcal{G},S_1)\in\mathcal{F}}\sum_{S_0\in\mathcal{G}}(\kappa(S_0,S_1)|S_0||S_1|+\zeta(S_0,S_1)) &= \text{Sparsity}(\textsc{BoundedCover}(X_{\text{low}}))\\ &\quad+\text{Sparsity}(\textsc{BoundedCover}(X_{\text{high}}))\\ &\quad+\sum_{u\in X_{\text{low}},S_1\in\mathcal{U}}(\kappa(\{u\},S_1)|S_1|+\zeta(\{u\},S_1))\\ &\le \text{poly}(d,\log n)n+|X_{\text{low}}||\mathcal{U}|(24dC_2+3)\\ &\le \text{poly}(d,\log n)n\end{aligned}$$

as desired.

Efficiency bound. We use the efficiency bounds of Proposition 7.19 along with the bound on $|\mathcal{U}|$:

$$\begin{aligned}\sum_{(\mathcal{G},S_1)\in\mathcal{F}}\left(|S_1|+\sum_{S_0\in\mathcal{G}}|S_0|\right) &= \text{Efficiency}(\textsc{BoundedCover}(X_{\text{low}}))\\ &\quad+\text{Efficiency}(\textsc{BoundedCover}(X_{\text{high}}))\\ &\quad+\sum_{S_1\in\mathcal{U}}(|S_1|+|X_{\text{low}}|)\\ &\le \text{poly}(d,\log n)n+|X_{\text{high}}|+|\mathcal{U}||X_{\text{low}}|\\ &\le \text{poly}(d,\log n)n\end{aligned}$$

as desired. □

(ζ,κ)-cover for weighted IP graphs with polylogarithmic dependence on norm

We now apply the subroutine from the previous subsection to produce a cover for weighted inner product graphs on vectors with arbitrary norms. However, we allow the sparsity and efficiency of the cover to depend on the ratio τ between the maximum and minimum norm of points in X. To obtain this cover, we bucket vectors by norm and call $\textsc{TwoBoundedCover}$ on all pairs of buckets.

Proposition 7.21 (Log-dependence cover)**.** *Given a set of vectors $X\subseteq\mathbb{R}^d$ with $\tau=\frac{\max_{x\in X}\|x\|_2}{\min_{x\in X}\|x\|_2}$ and $n=|X|$, there is a $\text{poly}(d,\log n,\log\tau)n$-time algorithm $\textsc{LogCover}(X)$ that produces a $\text{poly}(d,\log n,\log\tau)n$-sparse, $\text{poly}(d,\log n,\log\tau)n$-efficient (ζ,κ)-cover for the weighted inner product graph G on X.*

Proof. **Coverage**. For any edge $\{u,v\}\in E(G)$, there exists a pair $i,j\in\{0,1,\dots,\log\tau\}$ for which $u,v\in X_i\cup X_j$. Therefore, the *Coverage* property for $\textsc{TwoBoundedCover}(X_i\cup X_j)$ (which is part of $\mathcal{F}$) implies that the pair $\{u,v\}$ is covered by $\mathcal{F}$.

Algorithm 7

procedure LOGCOVER(X)
  Input: $X \subseteq \mathbb{R}^d$
  Output: An sparse, efficient (ζ, κ) cover for the weighted inner product graph G on X
  $d_{\min} \leftarrow \min_{x \in X} \|x\|_2$, $d_{\max} \leftarrow \max_{x \in X} \|x\|_2$
  for $i \in \{0, 1, \ldots, \log \tau\}$ **do**
    $X_i \leftarrow \{x \in X : \|x\|_2 \in [2^i d_{\min}, 2^{i+1} d_{\min})\}$
  end for
  $\mathcal{F} = \emptyset$
  for each pair $i, j \in \{0, 1, \ldots, \log \tau\}$ **do**
    Add TWOBOUNDEDCOVER($X_i \cup X_j$) to $\mathcal{F}$
  end for
  return $\mathcal{F}$
end procedure

Runtime, efficiency, and sparsity. Efficiency and sparsity of $\mathcal{F}$ are at most the sum of the efficiencies and sparsities respectively of the constituent TWOBOUNDEDCOVERs, each of which are at most $\text{poly}(d, \log n)n$ by Proposition 7.20. There are $O(\log^2 \tau)$ such covers in $\mathcal{F}$, so the efficiency and sparsity of $\mathcal{F}$ is at most $\text{poly}(d, \log n) \log^2 \tau n \leq \text{poly}(d, \log n, \log \tau)n$, as desired. Runtime is also bounded due to the fact that there are at most $\log^2 \tau$ for loop iterations. □

Desired (ζ, κ, δ)-cover

Now, we obtain a cover for all X with sparsity, efficiency, and runtime $\text{poly}(d, \log n)$. To do this, we break up pairs to cover $\{u, v\} \in X \times X$ into two types. Without loss of generality, suppose that $\|u\|_2 \leq \|v\|_2$. The first type consists of pairs for which $\|v\|_2 \leq (dn)^{1000}\|u\|_2$. These pairs are covered using several LOGCOVERs. The total efficiency, sparsity, and runtime required for these covers is $\text{poly}(d, \log n)n$ due to the fact that each vector is in at most $\text{poly}(d, \log n)$ of these covers.

The second type consists of all other pairs, i.e. those with $\|v\|_2 > (dn)^{1000}\|u\|_2$. For these pairs, we take care of them via a clustering argument. We cluster all vectors in X into $d + 1$ clusters in a greedy fashion. Specifically, we sort vectors in decreasing order by norm and create a new cluster for a vector $x \in X$ if $|\langle x, y \rangle| < \frac{1}{d+1}\|x\|_2\|y\|_2$ for the first vector y in each cluster. Otherwise, we assign x to an arbitrary cluster for which $|\langle x, y \rangle| \geq \frac{1}{d+1}\|x\|_2\|y\|_2$ for first cluster vector y. We then cover the pair $\{u, v\}$ using the pair of sets $(\{u\}, C)$, where C is the cluster containing v. To argue that this satisfies the *Coverage* property, we exploit the norm condition on the pair $\{u, v\}$. To bound efficiency, sparsity, and runtime, it suffices to bound the number of clusters, which is at most $d + 1$ by Proposition 7.5.

In order to define this algorithm, we use the notion of an *interval family*, which is exactly the same as the one-dimensional interval tree from computational geometry.

Definition 7.22. *For a set X and a function $f : X \to \mathbb{R}$, define the* interval family *for X, denoted $\mathcal{X} = \text{INTERVALFAMILY}(X)$, to be a family of sets produced recursively by initializing $\mathcal{X} = \{X\}$ and repeatedly taking an element $S \in \mathcal{X}$, splitting it evenly into two subsets S_0 and*

S_1 for which $\max_{x \in S_0} f(x) \leq \min_{x \in S_1} f(x)$, and adding S_0 and S_1 to $\mathcal{X}$ until $\mathcal{X}$ contains all singleton subsets of X.

$\mathcal{X}$ has the property that for any set $S \subseteq X$ consisting of all $x \in X$ for which $a \leq f(x) \leq b$ for two $a, b \in \mathbb{R}$, S is the disjoint union of $O(\log |X|)$ sets in $\mathcal{X}$. Furthermore, each element in X is in at most $O(\log |X|)$ sets in $\mathcal{X}$.

Proposition 7.23 (Desired cover)**.** *Given a set $X \subseteq \mathbb{R}^d$ with $n = |X|$, there is a $\mathrm{poly}(d, \log n)n$-time algorithm* DESIREDCOVER(X) *that produces a $\mathrm{poly}(d, \log n)n$-sparse, $\mathrm{poly}(d, \log n)n$-efficient (ζ, κ, δ)-cover for the weighted inner product graph G on X.*

Algorithm 8

procedure DESIREDCOVER(X)
  Input: $X \subseteq \mathbb{R}^d$
  Output: An sparse, efficient (ζ, κ, δ)-cover for the weighted inner product graph G on X, where ζ, κ, and δ are defined in the proof of Proposition 7.23.
  $\mathcal{F} \leftarrow \emptyset$
  $\xi \leftarrow (dn)^{1000}$
  $d_{\min} \leftarrow \min_{x \in X} \|x\|_2$, $d_{\max} \leftarrow \max_{x \in X} \|x\|_2$
  ▷ Cover nearby norm pairs
  for $i \in \{0, 1, \ldots, \log(d_{\max}/d_{\min}) - \lceil \log \xi \rceil$ **do**
    $X_i \leftarrow \{x \in X : \|x\|_2 \in [d_{\min} 2^i, d_{\min} 2^{i+1})$
    Add LOGCOVER$(X_i \cup X_{i+1} \cup \ldots \cup X_{i+\lceil \log \xi \rceil})$ to $\mathcal{F}$
  end for
  ▷ Create approximate basis for spread pairs
  $B \leftarrow \emptyset$
  for $x \in X$ in decreasing order by $\|x\|_2$ **do**
    if there does not exist $y \in B$ for which $|\langle x, y \rangle| \geq \|x\|_2 \|y\|_2/(d+1)$ **then**
      Add x to B and initialize a cluster $C_x = \{x\}$
    else
      Add x to C_y for an arbitrary choice of y satisfying the condition
    end if
  end for
  ▷ Cover the spread pairs
  for $w \in B$ **do**
    $\mathcal{C}_w \leftarrow$ INTERVALFAMILY(C_w)
    for each set $C \in \mathcal{C}_w$ **do**
      $\mathcal{G}_C \leftarrow$ the family of all singletons of $x \in X$ for which the disjoint union of $O(\log n)$ sets for $\{y \in C_w : \|y\|_2 \geq \xi \|u\|_2\}$ obtained from $\mathcal{C}_w$ contains C
      Add $(\mathcal{G}_C, C)$ to $\mathcal{F}$
    end for
  end for
  return $\mathcal{F}$
end procedure

Proof. We start by defining the functions ζ, κ, and δ. Let ζ_i and κ_i denote the functions for which LOGCOVER(Y_i) is a (ζ_i, κ_i)-cover for the weighted inner product graph on Y_i, where $Y_i = X_i \cup X_{i+1} \cup \ldots \cup X_{i+\lceil \log \xi \rceil}$ for all $i \leq \log(d_{\max}/d_{\min}) - \lceil \log \xi \rceil$. Let

$$\zeta(S_0, S_1) = \begin{cases} \sum_{i: S_0, S_1 \subseteq Y_i, \zeta_i(S_0,S_1) \neq \infty} \zeta_i(S_0, S_1) & \text{if there exists } i \text{ for which } S_0, S_1 \subseteq Y_i \\ 2 & \text{if } S_1 \in \mathcal{C}_y \text{ for some } y \in B \text{ and } S_0 = \{x\} \text{ for some } x \\ & \text{with } \|x\|_2 \leq \min_{a \in C} \|a\|_2/\xi \\ \infty & \text{otherwise} \end{cases}$$

$$\kappa(S_0, S_1) = \begin{cases} \sum_{i: S_0, S_1 \subseteq Y_i, \kappa_i(S_0,S_1) \neq \infty} \kappa_i(S_0, S_1) & \text{if there exists } i \text{ for which } S_0, S_1 \subseteq Y_i \\ 0 & \text{if } S_1 \in \mathcal{C}_y \text{ for some } y \in B \text{ and } S_0 = \{x\} \text{ for some } x \\ & \text{with } \|x\|_2 \leq \min_{a \in C} \|a\|_2/\xi \\ \infty & \text{otherwise} \end{cases}$$

$$\delta(S_0, S_1) = \begin{cases} 0 & \text{if there exists } i \text{ for which } S_0, S_1 \subseteq Y_i \\ \frac{1}{|S_1|} & \text{if } S_1 \in \mathcal{C}_y \text{ for some } y \in B \text{ and } S_0 = \{x\} \text{ for some } x \\ & \text{with } \|x\|_2 \leq \min_{a \in C} \|a\|_2/\xi \\ \infty & \text{otherwise} \end{cases}$$

Before proving that the required guarantees are satisfied, we bound some important quantities.

Bound on w_{yw} in terms of w_{uy} for $y \in C_w$ if $\|y\|_2 \geq \xi \|u\|_2$. By definition of C_w, $w_{yw} \geq \frac{1}{d+1}\|y\|_2\|w\|_2$ for any $y \in C_w$. w was the first member added to C_w, so $\|w\|_2 \geq \|y\|_2$. By the norm assumption on y, $\|y\|_2 \geq \xi\|u\|_2$. By Cauchy-Schwarz, $\|y\|_2\|u\|_2 \geq |\langle y, u\rangle| = w_{uy}$. Therefore,

$$w_{yw} \geq \frac{\xi}{d+1} w_{uy}$$

Bound on $\texttt{Reff}_G(u, w)$ for $w \in B$. We start by bounding the effective resistance between $u \in X$ and any $w \in B$ for which $\|w\|_2 > \xi\|u\|_2$. Recall that $w_{xy} = |\langle x, y\rangle|$ for any $x, y \in X$. Consider any $C \in \mathcal{C}_w$ for which $\min_{a \in C} \|a\|_2 > \xi\|u\|_2$. We show that

$$\texttt{Reff}_G(u, w) \leq \frac{2}{\sum_{y \in C} w_{uy}}$$

Consider all 2-edge paths of the form u-y-w for $y \in C$. By assumption on C, $\|y\|_2 \geq \xi\|u\|_2$ for any $y \in C$. Therefore, the bound on w_{yw} applies:

$$w_{yw} \geq \frac{\xi}{d+1} w_{uy}$$

for any $y \in C$. By series-parallel reductions, the u-w effective resistance is at most

$$\begin{aligned}\texttt{Reff}_G(u,w) &\leq \frac{1}{\sum_{y\in C}\frac{1}{1/w_{uy}+1/w_{yw}}}\\ &\leq \frac{1}{\sum_{y\in C}\frac{1}{(1+(d+1)/\xi)/w_{uy}}}\\ &\leq \frac{2}{\sum_{y\in C} w_{uy}}\end{aligned}$$

as desired.

Bound on $\texttt{Reff}_G(u,y)$ for $y \in C$. Any $y \in C$ has the property that $\|y\|_2 \geq \xi\|u\|_2$. Therefore, for $y \in C$, $\texttt{Reff}_G(y,w) \leq \frac{d+1}{\xi w_{uy}} \leq \frac{1}{|C|w_{uy}}$. By the triangle inequality for effective resistance,

$$\texttt{Reff}_G(u,y) \leq \texttt{Reff}_G(u,w) + \texttt{Reff}_G(w,y) \leq \frac{1}{|C|w_{uy}} + \frac{2}{\sum_{a\in C} w_{ua}}$$

Coverage. For any pair $\{u,v\}$ for which there exists i with $u,v \in Y_i$, $\{u,v\}$ is still covered by $\mathcal{F}$ by the *Coverage* property of LOGCOVER(Y_i). Therefore, we may assume that this is not the case. Without loss of generality, suppose that $\|v\|_2 \geq \|u\|_2$. Then, by assumption, $\|v\|_2 \geq \xi\|u\|_2$. By definition of the C_ws, there exists a $w \in B$ for which $v \in C_w$. By the first property of interval families, the set $\{x \in C_w : \|x\|_2 \geq \xi\|u\|_2\}$ is the disjoint union of $O(\log n)$ sets in $\mathcal{C}_w$. Let C be the unique set among these for which $v \in C$. By our effective resistance bound,

$$\begin{aligned}\texttt{Reff}_G(u,v) &\leq \frac{1}{|C|w_{uv}} + \frac{2}{\sum_{x\in C} w_{ux}}\\ &= \frac{\delta(\{u\},C)}{w_{uv}} + \frac{\kappa(\{u\},C)}{\max_{x\in C} w_{ux}} + \frac{\zeta(\{u\},C)}{\sum_{x\in C} w_{ux}}\end{aligned}$$

so the coverage property for the pair $\{u,v\}$ is satisfied within $\mathcal{F}$ by the pair $(\mathcal{G}_C, C)$, as desired.

Efficiency. The efficiency of $\mathcal{F}$ is at most the efficiency of the LOGCOVERs and the remaining part for spread pairs. We start with the LOGCOVERs. By Proposition 7.21,

$$\begin{aligned}\sum_i \text{Efficiency}(\text{LOGCOVER}(Y_i)) &\leq \sum_i \text{poly}(d, \log|Y_i|, \log\xi)|Y_i|\\ &\leq \sum_i \text{poly}(d,\log n)|Y_i|\\ &\leq (\log\xi+1)\text{poly}(d,\log n)\sum_i |X_i|\\ &\leq \text{poly}(d,\log n)n\end{aligned}$$

Therefore, we just need to bound the efficiency of the remainder of $\mathcal{F}$. The efficiency of $\mathcal{F}$ is at most

$$\begin{aligned}
\text{Efficiency}(\mathcal{F}) &= \sum_{(\mathcal{G},S_1)\in\mathcal{F}} \sum_{S_0\in\mathcal{G}} ((\delta(S_0,S_1)+\kappa(S_0,S_1))|S_0||S_1|+\zeta(S_0,S_1)) \\
&= \sum_i \text{Efficiency}(\textsc{LogCover}(Y_i)) \\
&+ \sum_{w\in B}\sum_{C\in\mathcal{C}_w}\sum_{\{u\}\in\mathcal{G}_C} (\delta(\{u\},C)|C|+\zeta(\{u\},C)) \\
&\le \text{poly}(d,\log n)n + \sum_{w\in B}\sum_{C\in\mathcal{C}_w} 3|\mathcal{G}_C|
\end{aligned}$$

By the first property of interval families, each $x \in X$ is present as a singleton in at most $O(\log n)$ $\mathcal{G}_C$s for C that are a subset of a given C_w. Therefore,

$$\text{Efficiency}(\mathcal{F}) \le \text{poly}(d,\log n)n + \sum_{w\in B} O(\log n)n$$

By Proposition 7.5, $|B| \le d+1$. Therefore, $\text{Efficiency}(\mathcal{F}) \le \text{poly}(d,\log n)n$, as desired.

Sparsity. By Proposition 7.21,

$$\sum_i \text{Sparsity}(\textsc{LogCover}(Y_i)) \le \sum_i \text{poly}(d,\log n,\log\xi)|Y_i| \le \text{poly}(d,\log n)n$$

Therefore, we may focus on the remaining part for spread pairs. In particular,

$$\begin{aligned}
\text{Sparsity}(\mathcal{F}) &= \sum_{(\mathcal{G},S_1)\in\mathcal{F}} (|S_1| + \sum_{S_0\in\mathcal{G}} |S_0|) \\
&\le \text{poly}(d,\log n)n \\
&+ \sum_{w\in B}\sum_{C\in\mathcal{C}_w} (|C| + |\mathcal{G}_C|) \\
&\le \text{poly}(d,\log n)n + \sum_{w\in B}\sum_{C\in\mathcal{C}_w} |C|
\end{aligned}$$

where the last inequality follows from the first property of interval families. By the second property of interval families, each element of C_w is in at most $O(\log n)$ sets in $\mathcal{C}_w$, so $\sum_{C\in\mathcal{C}_w}|C| \le O(\log n)|C_w|$. Since $|B| \le d+1$, $\text{Sparsity}(\mathcal{F}) \le \text{poly}(d,\log n)n$, as desired.

Runtime. The first for loop takes $\sum_i \text{poly}(d,\log n)|Y_i| \le \text{poly}(d,\log n)n$ by Proposition 7.21. The second for loop takes $O(d|B|n) \le \text{poly}(d,\log n)n$ by the bound on $|B|$. The third for loop takes $\text{poly}(d,\log n)n$ by the runtime for IntervalFamily and the two properties of interval families. Therefore, the total runtime is $\text{poly}(d,\log n)n$, as desired.

□

Proof of Lemma 7.1

Proof of Lemma 7.1. Follows immediately from constructing the cover $\mathcal{F}$ given by Proposition 7.23 and plugging that into Proposition 7.17. □

8 Hardness of Sparsifying and Solving Non-Multiplicatively-Lipschitz Laplacians

We now define some terms to state our hardness results:

Definition 8.1. *For a decreasing function f that is not (C, L)-multiplicatively Lipschitz, there exists a point x for which $f(Cx) \leq C^{-L} f(x)$. Let x_0 denote one such point. A set of real numbers $S \subseteq \mathbb{R}_{\geq 0}$ is called* ρ-discrete *for some $\rho > 1$ if for any pair $a, b \in S$ with $b > a$, $b \leq \rho a$. A set of points $X \subseteq \mathbb{R}^d$ is called* ρ-spaced *for some $\rho > 1$ if there is some ρ-discrete set $S \subseteq \mathbb{R}_{\geq 0}$ with the property that for any pair $x, y \in X$, $\|x - y\|_2 \in S$. S is called the* distance set *for X.*

Dim.	Thm.	d	$g(p)$	ρ	Time
Low	8.2	$c^{\log^* n}$	p	$1 + 16\log(10(L^{1/(4c_0)}))/L$	$O(nL^{1/(8c_0)})$
High	8.3	$\log n$	e^p	$1 + 2\log(10(2^{L^{0.48}}))/L$	$O(n2^{L^{.48}})$

Table 5.3: Sparsification Hardness

In this section, we show the following two hardness results:

Theorem 8.2 (Low-dimensional sparsification hardness)**.** *Consider a decreasing function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is not (ρ, L)-multiplicatively lipschitz for some $L > 1$, where c and c_0 are the constants given in Theorem 3.20 and $\rho = 1 + 2\log(10L^{1/(4c_0)})/L$. There is no algorithm that, given a set of n points X in $d = c^{\log^* n}$ dimensions, returns a sparsifier of the f-graph for X in less than $O(nL^{1/(8c_0)})$ time assuming* SETH.

Theorem 8.3 (High-dimensional sparsification hardness)**.** *Consider a decreasing function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is not (ρ, L)-multiplicatively Lipschitz for some $L > 1$, where $\rho = 1 + 2\log(10(2^{L^{0.48}}))/L$. There is no algorithm that, given a set of n points X in $d = O(\log n)$ dimensions, returns a sparsifier of the f-graph for X in less than $O(n2^{L^{.48}})$ time assuming* SETH.

Both of these results follow from the following reduction:

Lemma 8.4. *Consider a decreasing function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is not (ρ, L)-multiplicatively Lipschitz for some $L > 1$, where $\rho = 1 + 2\log(10n)/L$ and $n > 1$. Suppose that there is an algorithm $\mathcal{A}$ that, when given a set of n points $X \subseteq \mathbb{R}^d$, returns a 2-approximate sparsifier for the f-graph of X with $O(n)$ edges in $\mathcal{T}(n, L, d)$ time. Then, there is an algorithm (Algorithm 9) that, given two sets $A, B \subseteq \mathbb{R}^d$ for which $A \cup B$ is ρ-spaced with distance set S, $k \in S$, and $|A \cup B| = n$, returns whether or not $\min_{a \in A, b \in B} \|a - b\|_2 \leq k$ in*

$$O(\mathcal{T}(|A \cup B|, L, d) + |A \cup B|)$$

time.

The reduction described starts by scaling the points in $A \cup B$ by a factor of x_0/k to obtain $\widetilde{A}$ and $\widetilde{B}$ respectively. Then, it sparsifies the f-graph for $\widetilde{A} \cup \widetilde{B}$. Finally, it computes the weight of the edges in the $\widetilde{A}$-$\widetilde{B}$ cut. Because f is not multiplicatively Lipschitz and the distance set for $\widetilde{A} \cup \widetilde{B}$ is spaced, thresholding suffices for solving the $A \times B$ nearest neighbor problem.

Proof of Lemma 8.4. Consider the following algorithm, BICHROMATICNEARESTNEIGHBOR (Algorithm 9), given below:

Algorithm 9

procedure BICHROMATICNEARESTNEIGHBOR(A, B, k) ▷ Lemma 8.4
  Given: $A, B \subset \mathbb{R}^d$ with the property that $A \cup B$ is ρ-spaced, where $\rho = 1 + 2(\log(10n))/L$, and $k \in S$
  Returns: whether there are $a \in A, b \in B$ for which $\|a - b\|_2 \leq k$
  $\widetilde{A} \leftarrow \{a \cdot \sqrt{x_0}/k \mid \forall a \in A\}$ ▷ $\widetilde{A} \subset \mathbb{R}^d$
  $\widetilde{B} \leftarrow \{b \cdot \sqrt{x_0}/k \mid \forall b \in B\}$ ▷ $\widetilde{B} \subset \mathbb{R}^d$
  $H \leftarrow \mathcal{A}(\widetilde{A} \cup \widetilde{B})$ ▷ H is a 2-approximate sparsifier for the f-graph G of $\widetilde{A} \cup \widetilde{B}$
  if the total weight of edges between $\widetilde{A}$ and $\widetilde{B}$ in H is at least $f(x_0)/2$ **then**
    return true
  else
    return false
  end if
end procedure

We start by bounding the runtime of this algorithm. Constructing $\widetilde{A}$ and $\widetilde{B}$ and calculating the total weight of edges between $\widetilde{A}$ and $\widetilde{B}$ takes $O(n)$ time since H has $O(n)$ edges. Since the sparsification algorithm is only called once, the total runtime is therefore $\mathcal{T}(n, L, d) + O(n)$, as desired. For the rest of the proof, we may therefore focus on correctness.

First, suppose that $\min_{a \in A, b \in B} \|a - b\|_2 \leq k$. There exists a pair of points $\widetilde{a} \in \widetilde{A}, \widetilde{b} \in \widetilde{B}$ with $\|\widetilde{a} - \widetilde{b}\|_2 \leq \sqrt{x_0}$. Since f is a decreasing function, the edge between $\widetilde{a}$ and $\widetilde{b}$ in G has weight at least $f(x_0)$, which means that the total weight of edges in the $\widetilde{A}$-$\widetilde{B}$ cut in G is at least $f(x_0)$. Since H is a 2-approximate sparsifier for G, the total weight of edges in the $\widetilde{A}$-$\widetilde{B}$ cut is at least $f(x_0)/2$. This means that true is returned, as desired.

Next, suppose that $\min_{a \in A, b \in B} \|a - b\|_2 > k$. Since $A \cup B$ is ρ-spaced with distance set S and $k \in S$, $\|a - b\|_2 \geq \rho \cdot k$ for all $a \in A$ and $b \in B$. Therefore, $\|\widetilde{a} - \widetilde{b}\|_2 \geq \rho \cdot \sqrt{x_0}$ for all $\widetilde{a} \in \widetilde{A}$ and $\widetilde{b} \in \widetilde{B}$.

Since f is decreasing and not (ρ, L)-multiplicatively Lipschitz, the weight of any edge between $\widetilde{A}$ and $\widetilde{B}$ in G is at most

$$f(\rho \cdot x_0) \leq f(x_0)/(100n^2).$$

The total weight of edges between $\widetilde{A}$ and $\widetilde{B}$ is therefore at most

$$n^2 \cdot (f(x_0)/(100n^2)) < f(x_0)/8.$$

Since H is a 2-approximate sparsifier for G, the total weight between C and D is at most $f(x_0)/4 < f(x_0)/2$, so the algorithm returns false, as desired. □

We now prove the theorems:

Proof of Theorem 8.2. Consider an instance of ℓ_2-bichromatic closest pair for $n = L^{1/(4c_0)}$ and $d = c^{\log^* n}$, where c is the constant given in the dimension bound of Theorem 3.20 and c_0 is such that integers have bit length $c_0 \log n$ in Theorem 3.20. This consists of two sets of points $A, B \subseteq \mathbb{R}^d$ with $|A \cup B| = n$ for which we wish to compute $\min_{a \in A, b \in B} \|a - b\|_2$. By Theorem 3.20, the coordinates of points in A are also $c_0 \log n$ bit integers. Therefore, the set S of possible ℓ_2 distances between points in A and B is a set of square roots of integers with $\log d + c_0 \log n \leq 2c_0 \log n$ bits. Therefore, S is a ρ-discrete (recall $\rho = 1 + (2\log(10n))/L$), since $1 + 1/n^{2c_0} > 1 + (2\log(10n))/L$. Furthermore, note that f is not (ρ, L)-multiplicatively Lipschitz.

We now describe an algorithm for solving ℓ_2-closest pair on $A \times B$. Use binary search on the values in S to compute the minimum distance between points in A, B. For each query point $k \in S$, by Lemma 8.4, there is a

$$\mathcal{T}(n, L, d) = O(\mathcal{T}(L^{1/(4c_0)}, L, c^{\log^* L}) + L^{1/(4c_0)})$$

-time algorithm for determining whether or not the closest pair has distance at most k. Therefore, there is a

$$O(\log |S| \cdot (\mathcal{T}(L^{1/(4c_0)}, L, c^{\log^* L}) + L^{1/4c_0})) = \widetilde{O}(L^{1/(4c_0)} L^{1/(8c_0)}) < O(n^{3/2})$$

time algorithm for solving ℓ_2-closest pair on pairs of sets with n points. But this is impossible given SETH by Theorem 3.20, a contradiction. This completes the result. □

Proof of Theorem 8.3. Consider an instance of bichromatic Hamming nearest neighbor search for $n = 2^{L^{0.49}}$ and $d = c_1 \log n$ for the constant c_1 in the dimension bound in Theorem 3.17. This consists of two sets of points $A, B \subseteq \mathbb{R}^d$ with $|A \cup B| = n$ for which we wish to compute $\min_{a \in A, b \in B} \|a - b\|_2$. The coordinates of points in A and B are 0-1. Therefore, the set S of possible ℓ_2 distances between points in A and B is the set of square roots of integers between 0 and $c_1 \log n$, which differ by a factor of at least $1 + 1/(2c_1 \log n) > \rho$ (recall $\rho = 1 + (2\log(10n))/L$). Therefore, $A \cup B$ is ρ-spaced. Note that f is also no (ρ, L)-multiplicatively Lipschitz by definition.

We now give an algorithm for solving ℓ_2-closest pair on $A \times B$. Use binary search on S. For each query $k \in S$, Lemma 8.4 implies that one can check if there is a pair with distance at most k in

$$\begin{aligned}
\mathcal{T}(n, L, d) &= O(\mathcal{T}(2^{L^{.49}}, L, c_1 L^{.49}) + 2^{L^{.49}}) \\
&\leq O(2^{L^{.49} + L^{.48}}) \\
&= n^{1 + o(1)}
\end{aligned}$$

time on pairs of sets with n points. But this is impossible given SETH by Theorem 3.17. This completes the result. □

Next, we prove hardness results for solving Laplacian systems. In these hardness results, we insist that kernels are bounded:

Definition 8.5. *Call a function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ $g(p)$-*bounded *for a function $g : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ iff for any pair $a, b > 0$ with $b > a$, $f(b) \geq f(a) \cdot g(b/a)$. Call a set $S \subseteq \mathbb{R}_{\geq 0}$ γ-*bounded *iff $\max_{s \in S} s \leq \gamma \min_{s \in S, s \neq 0} s$. A set of points $X \subseteq \mathbb{R}^d$ is called γ-*boxed *iff the set of distances between points in X is γ-bounded.*

Dim.	Thm.	d	$g(p)$	ρ	Time	ε
Low	8.6	$c^{\log^* n}$	p	$1 + 16\log(10(L^{1/(4c_0)}))/L$	$n\log(g(\gamma))L^{1/(64c_0)}$	$1/(g(\gamma)^3 2^{\mathrm{poly}(\log n)})$
High	8.7	$\log n$	e^p	$1 + 2\log(10(2^{L^{0.48}}))/L$	$n\log(g(\gamma))2^{L^{.48}}$	$1/(g(\gamma)^3 2^{\mathrm{poly}(\log n)})$

Table 5.4: Linear System Hardness

We show the following results:

Theorem 8.6 (Partial low-dimensional linear system hardness)**.** *Consider a decreasing $g(p) = p$-bounded function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is not (ρ, L)-multiplicatively Lipschitz for some $L > 1$, where c and c_0 are the constants given in Theorem 3.20 and $\rho = 1 + 16\log(10(L^{1/(32c_0)}))/L$. Assuming* SETH*, there is no algorithm that, given a γ-boxed set of n points X in $d = c^{\log^* n}$ dimensions with f-graph G and a vector $b \in \mathbb{R}^n$, returns a $\varepsilon = 1/(g(\gamma)^3 2^{\mathrm{poly}(\log n)})$-approximate solution $x \in \mathbb{R}^n$ to the geometric Laplacian system $L_G x = b$ in less than $O(n\log(g(\gamma))L^{1/(64c_0)})$ time.*

Theorem 8.7 (Partial high-dimensional linear system hardness)**.** *Consider a decreasing $g(p) = e^p$-bounded function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is not (ρ, L)-multiplicatively Lipschitz for some $L > 1$, where $\rho = 1 + 16\log(10(2^{L^{0.48}}))/L$. Assuming* SETH*, there is no algorithm that, given a γ-boxed set of n points X in $d = O(\log n)$ dimensions with f-graph G and a vector $b \in \mathbb{R}^n$, returns a $\varepsilon = 1/(g(\gamma)^3 2^{\mathrm{poly}(\log n)})$-approximate solution $x \in \mathbb{R}^n$ to the geometric Laplacian system $L_G x = b$ in less than $O(n\log(g(\gamma))2^{L^{.48}})$ time assuming* SETH*.*

To prove these theorems, we use the following reduction from bichromatic nearest neighbors:

Lemma 8.8. *Consider a decreasing $g(p)$-bounded function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is not (ρ, L)-multiplicatively Lipschitz for some $L > 1$, where $\rho = 1 + 16(\log(10n))/L$ and $n > 1$. Suppose that there is an algorithm $\mathcal{A}$ that, given a γ-boxed set of n points X in d dimensions with f-graph G and a vector $b \in \mathbb{R}^n$, returns an $\varepsilon = 1/(g(\gamma)^3 2^{\mathrm{poly}(\log n)})$-approximate solution $x \in \mathbb{R}^n$ to the geometric Laplacian system $L_G x = b$ in $\mathcal{T}(n, L, g(\gamma), d)$ time. Then, there is an algorithm that, given two sets $A, B \subseteq \mathbb{R}^d$ for which $A \cup B$ is ρ-spaced with $|S|^{O(1)}$-bounded distance set S, $k \in S$, and $|A \cup B| = n$, returns whether or not $\min_{a \in A, b \in B} \|a - b\|_2 \leq k$ in*

$$O((\log n)\mathcal{T}(n, L, g(|S|^{O(1)}), d) + |A \cup B|)$$

time.

This reduction works in a similar way to the effective resistance data structure of Spielman and Srivastava [269], but with minor differences due to the fact that their data structure requires multiplication by incidence matrix of the graph, which in our case is dense. Our reduction uses Johnson-Lindenstrauss to embed the points

$$v_s = L^\dagger b_s$$

for vertices s in the graph G into $O(\log n)$ dimensions in a way that distorts the distances

$$b_{st}^\top (L^\dagger)^2 b_{st}$$

for vertices s, t in G by a factor of at most 2. After computing this embedding, we build an $O(\log n)$-approximate nearest neighbor data structure on the resulting points. This allows us to determine whether or not a vertex in A has a high-weight edge in G to B in almost-constant time. After looping through all of the edges in A in total time $n^{1+o(1)}$, we determine whether or not there are any high-weight edges between A and B in G, allowing us to answer the bichromatic nearest neighbors decision problem.

We start by proving a result that links norms of v_s to effective resistances:

Proposition 8.9. *In an n-vertex graph G with vertices s and t,*

$$0.5 \cdot (\texttt{Reff}_G(s,t))^2 \leq \|v_s - v_t\|_2^2 \leq n \cdot (\texttt{Reff}_G(s,t))^2$$

Proof. **Lower bound.** Let $x = v_t - v_s = L_G^\dagger b_{st} \in \mathbb{R}^n$. By definition,

$$\begin{aligned}
\|v_s - v_t\|_2^2 &= \|x\|_2^2 \\
&\geq (x_s)^2 + (x_t)^2 \\
&= (x_s)^2 + (x_s - b_{st}^\top L_G^\dagger b_{st})^2 \\
&\geq (b_{st}^\top L_G^\dagger b_{st})^2/2,
\end{aligned}$$

where third step follows from $x_t = x_s - b_{st}^\top L_G^\dagger b_{st}$.

Thus we complete the proof of the lower bound.

Upper bound. Next, we prove the upper bound. The maximum and minimum coordinates of x are x_t and x_s respectively. By definition of the pseudoinverse, image$(L_G^\dagger)$ = image(L_G). Therefore, $\mathbf{1}^\top x = 0$, $x_s \leq 0$, and $x_t \geq 0$. $x_s \leq 0$ implies that for all $i \in [n]$,

$$x_i \leq x_s + b_{st}^\top L_G^\dagger b_{st} \leq b_{st}^\top L_G^\dagger b_{st}.$$

$x_t \geq 0$ implies that for all $i \in [n]$,

$$x_i \geq x_t - b_{st}^\top L_G^\dagger b_{st} \geq -b_{st}^\top L_G^\dagger b_{st}.$$

Therefore, $|x_i| \leq b_{st}^\top L_G^\dagger b_{st} = \texttt{Reff}_G(s,t)$ for all $i \in [n]$. Summing across $i \in [n]$ yields the desired upper bound. □

Furthermore, the minimum effective resistance of an edge across a cut is related to the maximum weight edge across the cut:

Proposition 8.10. *In an m-edge graph G with vertex set S,*

$$\min_{s \in S, t \notin S} \texttt{Reff}_G(s,t) \leq \min_{e \in \partial S} (1/w_e) \leq m \min_{s \in S, t \notin S} \texttt{Reff}_G(s,t).$$

Proof. **Lower bound.** The lower bound on $\min_e 1/w_e$ follows immediately from the fact that for any edge $e = \{s, t\}$, $\texttt{Reff}_G(s, t) \leq r_e$.

Upper bound. For the upper bound, recall that

$$\begin{aligned}
\texttt{Reff}_G(s,t) &= \min_{f \in \mathbb{R}^m : B^\top f = b_{st}} \sum_{e \in E(G)} f_e^2 / w_e \\
&\geq \min_{f \in \mathbb{R}^m : B^\top f = b_{st}} \sum_{e \in \partial S} f_e^2 / w_e \\
&\geq \left(\min_{f \in \mathbb{R}^m : B^\top f = b_{st}} \sum_{e \in \partial S} f_e^2 \right) \left(\min_{e \in \partial S} 1/w_e \right) \\
&\geq \frac{1}{|\partial S|} \left(\min_{f \in \mathbb{R}^m : B^\top f = b_{st}} \sum_{e \in \partial S} |f_e| \right)^2 \left(\min_{e \in \partial S} 1/w_e \right)
\end{aligned}$$

where the first step follows from definition of effective resistance, the third step follows from taking w out, and the last step follows from Cauchy-Schwarz.

Since $s \in S$ and $t \notin S$, $\sum_{e \in \partial S} |f_e| \geq 1$. Therefore,

$$\texttt{Reff}_G(s,t) \geq \frac{1}{m} \min_{e \in \partial S} 1/w_e$$

completing the upper bound. □

Proposition 8.11. *Consider an n-vertex connected graph G with edge weights $\{w_e\}_{e \in G}$, two matrices $Z, \widetilde{Z} \in \mathbb{R}^{k \times n}$ with rows $\{z_i\}_{i=1}^k$ and $\{\widetilde{z}_i\}_{i=1}^k$ respectively for $k \leq n$, and $\varepsilon \in (1/n, 1)$. Suppose that both of the following properties hold:*

1. *$\|z_i - \widetilde{z}_i\|_{L_G} \leq 0.01 n^{-12} w_{\min}^2 w_{\max}^{-2} \|z_i\|_{L_G}$ for all $i \in [k]$, where $w_{\min}$ and $w_{\max}$ are the minimum and maximum weights of edges in G respectively*
2. *$(1 - \varepsilon/10) \cdot \|L_G^\dagger b_{st}\|_2 \leq \|Z b_{st}\|_2 \leq (1 + \varepsilon/10) \cdot \|L_G^\dagger b_{st}\|$ for any vertices s, t in G*

Then for any vertices s, t in G,

$$(1 - \varepsilon) \cdot \|L_G^\dagger b_{st}\|_2 \leq \|\widetilde{Z} b_{st}\|_2 \leq (1 + \varepsilon) \cdot \|L_G^\dagger b_{st}\|_2.$$

Proof. We start by bounding

$$((z_i - \widetilde{z}_i)^\top b_{st})^2$$

for each $i \in [k]$. Since G is connected, there is a path from s to t consisting of edges $e_1, e_2, \ldots, e_\ell$ in that order, where $\ell \leq n$. By the Cauchy-Schwarz inequality,

$$\begin{aligned}
((z_i - \widetilde{z}_i)^\top b_{st})^2 &\leq n \sum_{j=1}^{\ell} ((z_i - \widetilde{z}_i)^\top b_{e_j})^2 \\
&\leq (n / w_{\min}) \cdot \|z_i - \widetilde{z}_i\|_{L_G}^2 \\
&\leq 0.01 n^{-23} w_{\min}^3 w_{\max}^{-4} \cdot \|z_i\|_{L_G}^2
\end{aligned}$$

where the last step from property 1 in proposition statement.

By the upper bound on $\|Zb_{s't'}\|_2$ for any vertices s', t' in G, $(z_i^\top b_e)^2 \leq (1+\varepsilon)^2 b_e^\top (L_G^\dagger)^2 b_e$ for all edges e in G, so

$$\begin{aligned}
0.01n^{-23}w_{\min}^3 w_{\max}^{-4} \cdot \|z_i\|_{L_G}^2 &= 0.01n^{-23}w_{\min}^3 w_{\max}^{-4} \cdot \sum_{e\in E(G)} w_e((z_i)^\top b_e)^2 \\
&\leq 0.01n^{-23}w_{\min}^3 w_{\max}^{-3} \cdot \sum_{e\in E(G)} ((z_i)^\top b_e)^2 \\
&\leq 0.01n^{-23}w_{\min}^3 w_{\max}^{-3} \cdot \sum_{e\in E(G)} (1+\varepsilon)^2 b_e^\top (L_G^\dagger)^2 b_e \\
&\leq 0.01n^{-21}w_{\min}^3 w_{\max}^{-3}(1+\varepsilon)^2 \max_{e\in E(G)} (b_e^\top (L_G^\dagger)^2 b_e) \\
&\leq 0.04n^{-21}w_{\min}^3 w_{\max}^{-3} \max_{e\in E(G)} (b_e^\top (L_G^\dagger)^2 b_e).
\end{aligned}$$

where the first step follows from $\|z_i\|_{L_G}^2 = \sum_{e\in E} z_i^\top w_e b_e b_e^\top z_i$, the second step follows from $w_{\max} = \max_{e\in G} w_e$, and the third step follows from $(z_i^\top b_e)^2 \leq (1+\varepsilon)^2 b_e^\top (L_G^\dagger) b_e$, the forth step follows from summation has at most n^2 terms, the last step follows from $(1+\varepsilon)^2 \leq 4$, $\forall \varepsilon \in (0,1)$.

$L_G^\dagger b_e$ is a vector that is maximized and minimized at the endpoints of e. Furthermore, $\mathbf{1} \in$ kernel$(L_G^\dagger)$ by definition of the pseudoinverse. Thus, we know $L_G^\dagger b_e$ has both positive and negative coordinates and that $\|L_G^\dagger b_e\|_\infty \leq \max_{i\neq j} |(L_G^\dagger b_e)_i - (L_G^\dagger b_e)_j| \leq b_e^\top L_G^\dagger b_e$.

We have

$$\begin{aligned}
&0.04n^{-21}w_{\min}^3 w_{\max}^{-3} \cdot \max_{e\in E(G)} b_e^\top (L_G^\dagger)^2 b_e \\
\leq\ &0.04n^{-20}w_{\min}^3 w_{\max}^{-3} \cdot \max_{e\in E(G)} (b_e^\top L_G^\dagger b_e)^2 \\
\leq\ &0.4n^{-20}w_{\min} w_{\max}^{-3} \\
\leq\ &0.4n^{-16}w_{\min} w_{\max}^{-1} \cdot (b_{st}^\top L_G^\dagger b_{st})^2 \\
\leq\ &0.8n^{-16}w_{\min} w_{\max}^{-1} \cdot \|L_G^\dagger b_{st}\|_2^2
\end{aligned}$$

where the first step follows from $\max_{e\in E(G)} b_e^\top (L_G^\dagger)^2 b_e = \max_{e\in E} \|L_G^\dagger b_e\|_2^2 \leq n \max_{e\in E} \|L_G^\dagger b_e\|_\infty^2 \leq n \max_{e\in E(G)} (b_e^\top L_G^\dagger b_e)^2$, the second step follows from $\max_e (b_e^\top L_G^\dagger b_e)^2 \leq w_{\min}^{-2}$ (the lower bound in Proposition 8.10), and the third step follows from $w_{\max}^{-2} \leq n^4 (b_{st} L_G^\dagger b_{st})^2$ (the upper bound in Proposition 8.10), and the last step follows from $(b_{st}^\top L_G^\dagger b_{st})^2 \leq 2\|L_G^\dagger b_{st}\|_2^2$ (Since $L_G^\dagger b_{st}$ is maximized and minimized at t and s respectively).

Combining these inequalities shows that

$$((z_i - \widetilde{z}_i)^\top b_{st})^2 \leq 0.8n^{-16}w_{\min} w_{\max}^{-1} \cdot \|L_G^\dagger b_{st}\|_2^2$$

Summing over all i and using the fact that $k \le n$, $\varepsilon > 1/n$, and $w_{\min} \le w_{\max}$ shows that

$$\begin{aligned}\|(Z - \widetilde{Z})b_{st}\|_2^2 &= \sum_{i=1}^{k}((z_i - \widetilde{z}_i)^\top b_{st})^2 \\ &\le 0.8kn^{-16}w_{\min}w_{\max}^{-1} \cdot \|L_G^\dagger b_{st}\|_2^2 \\ &\le 0.8\varepsilon^2 n^{-13}w_{\min}w_{\max}^{-1} \cdot \|L_G^\dagger b_{st}\|_2^2 \\ &\le 0.8\varepsilon^2 n^{-13} \cdot \|L_G^\dagger b_{st}\|_2^2\end{aligned}$$

Combining this with the given upper and lower bounds on $\|Zb_{st}\|_2$ using the triangle inequality yields the desired result. □

Proof of Lemma 8.8. Consider the following algorithm BICHROMATICNEARESTNEIGHBOR, given below:

Algorithm 10

procedure BICHROMATICNEARESTNEIGHBORPROJ(A, B, k) ▷ Lemma 8.8
  Given: $A, B \in \mathbb{R}^d$ with the property that $A \cup B$ is ρ-spaced with $|S|^{O(1)}$-bounded distance set , where $\rho = 1 + 16(\log(10n))/L$, and $k \in S$
  Returns: whether there are $a \in A, b \in B$ for which $\|a - b\|_2 \le k$
  $C \leftarrow \{a \cdot \sqrt{x_0}/k \mid \forall a \in A\}$
  $D \leftarrow \{b \cdot \sqrt{x_0}/k \mid \forall b \in B\}$
  $\ell \leftarrow 200 \log n$
  Construct matrix $P \in \mathbb{R}^{\ell \times d}$, where each entry is $1/\sqrt{\ell}$ with prob $1/2$ and $-1/\sqrt{\ell}$ with prob $1/2$
  $\widetilde{Z}_{i,*} \leftarrow \mathcal{A}(C \cup D, P_{i,*})$ for each $i \in [\ell]$ ▷ $P_{i,*}$, $\widetilde{Z}_{i,*}$ denotes row i of $P \in \mathbb{R}^{\ell \times d}$, $\widetilde{Z} \in \mathbb{R}^{\ell \times n}$
  $\widehat{C} \leftarrow \{\widetilde{Z} \cdot b_c \mid \forall c \in C\}$ ▷ $b_c \in \mathbb{R}^n$ denotes the indicator vector of the vertex c, i.e. $(b_c)_c = 1$ and $(b_c)_i = 0$ for all $i \ne c$
  $\widehat{D} \leftarrow \{\widetilde{Z} \cdot b_c \mid \forall c \in D\}$
  $t \leftarrow (\log n)$-approximation to closest $\widehat{C}$-$\widehat{D}$ ℓ_2-distance using Theorem 3.15
  if $t \le 3\sqrt{n}(\log n)/f(x_0)$ **then**
    return true
  else
    return false
  end if
end procedure

First, we bound the runtime of BICHROMATICNEARESTNEIGHBORPROJ (Algorithm 10). Computing the sets C, D and the matrix P trivially takes $\widetilde{O}(n)$ time. Computing the matrix $\widetilde{Z}$ takes

$$O(\log n)\mathcal{T}(n, L, g(|S|^{O(1)}), d)$$

time since the point set $C \cup D$ is $|S|^{O(1)}$-boxed. Computing $\widehat{C}$ and $\widehat{D}$ takes $\widetilde{O}(n)$ time, as computing $\widehat{Z}b_i$ takes $O(\log n)$ time for each $i \in C \cup D$ since b_i is supported on just one vertex.

Computing t takes $n^{1+o(1)}$ time by Theorem 3.15. In particular, one computes t by preprocessing a $O(\log n)$-approximate nearest neighbors data structure on $\widehat{D}$ (takes $n^{1+o(1)}$ time), queries the data structure on all points in $\widehat{C}$ (takes $n(n^{o(1)}) = n^{1+o(1)}$ time), and returns the minimum of all of the queries. The subsequent if statement takes constant time. Therefore, the reduction takes

$$O((\log n) \cdot \mathcal{T}(n, L, |S|^{O(1)}, d) + n^{1+o(1)}$$

time overall, as desired.

Next, suppose that there exists $a \in A$ and $b \in B$ for which $\|a - b\|_2 \leq k$. We show that the reduction returns true with probability at least $1 - 1/n$. Let G denote the f-graph on $C \cup D$. By definition of C and D and the fact that f is decreasing, there exists a pair of points in C and D with edge weight at least $f(x_0)$.

By the lower bound on Proposition 8.10,

$$\exists p \in C, q \in D \text{ s.t. } \texttt{Reff}_G(p, q) \leq 1/f(x_0).$$

By the upper bound of Proposition 8.9,

$$b_{pq}^\top (L_G^\dagger)^2 b_{pq} \leq n \cdot (\texttt{Reff}_G(p, q))^2 \leq n/f(x_0)^2.$$

Let $Z \in \mathbb{R}^{\ell \times n}$ be defined as $Z = P \cdot L_G^\dagger$. By Theorem 3.13 with $\varepsilon = 1/2$ applied to the collection of vectors $\{L_G^\dagger b_s\}_{s \in C \cup D}$ and projection matrix P,

$$\frac{1}{2}\|L_G^\dagger b_{st}\|_2 \leq \|Z b_{st}\|_2 \leq \frac{3}{2}\|L_G^\dagger b_{st}\|_2$$

for all pairs $s, t \in C \cup D$ with high probability. Therefore, the second input guarantee of Proposition 8.11 is satisfied with high probability. Furthermore, for each $i \in [\ell]$, $\widetilde{z}_i$, the ith row of $\widetilde{Z}$, satisfies the first input guarantee by the output error guarantee of the algorithm $\mathcal{A}$. Therefore, Proposition 8.11 applies and shows that

$$\|\widetilde{Z} \cdot b_{pq}\|_2 \leq \frac{9}{4}\|L_G^\dagger \cdot b_{pq}\|_2 \leq 3\sqrt{n}/f(x_0).$$

This means that there exists of vectors $a \in \widehat{C}, b \in \widehat{D}$ with

$$\|a - b\|_2 \leq 3\sqrt{n}/f(x_0).$$

By the approximation guarantee of the nearest neighbors data structure, $t \leq (3\sqrt{n}/f(x_0)) \log n$ and the reduction returns true with probability at least $1 - 1/n$, as desired.

Next, suppose that there do not exist $a \in A$ and $b \in B$ for which $\|a - b\|_2 \leq k$. We show that the reduction returns false with probability at least $1 - 1/n$. Since $k \in S$ and $A \cup B$ is ρ-spaced, $\|a - b\|_2 \geq \rho \cdot k$ for all $a \in A$ and $b \in B$. Therefore, all edges between C and D in the f-graph G for $C \cup D$ have weight at most $f(\rho x_0) \leq \frac{f(x_0)}{100n^{16}}$ since f is not (ρ, L)-multiplicatively Lipschitz.

By the upper bound of Proposition 8.10,

$$\texttt{Reff}_G(s, t) \geq \frac{1}{n^2 f(\rho x_0)} \geq \frac{100n^{14}}{f(x_0)}$$

for any pair of vertices $s \in C, t \in D$.

By the lower bound of Proposition 8.9,

$$b_{st}^\top (L_G^\dagger)^2 b_{st} \geq (\texttt{Reff}_G(s,t))^2/2 \geq (\frac{50n^{14}}{f(x_0)})^2$$

for all $s \in C, t \in D$.

Recall from the discussion of the $\|a - b\| \leq k$ case that Theorem 3.13 and Proposition 8.11 apply. Therefore, with probability at least $1 - 1/n$, by the lower bound of Proposition 8.11,

$$\|\widetilde{Z} b_{st}\|_2 \geq \frac{1}{4}\|L_G^\dagger b_{st}\|_2 \geq \frac{12n^{14}}{f(x_f)}$$

for any $s \in C, t \in D$.

Therefore, by the approximate nearest neighbors guarantee,

$$t \geq \frac{12n^{14}}{(\log n) \cdot f(x_0)} > \frac{3\sqrt{n}(\log n)}{f(x_0)},$$

so the algorithm returns false with probability at least $1 - 1/n$, as desired. □

We now prove the theorems:

Proof of Theorem 8.6. Consider an instance of bichromatic ℓ_2-closest pair for $n = L^{1/(32c_0)}$ and $d = c^{\log^* n}$, where c is the constant given in the dimension bound of Theorem 3.20 and c_0 is such that integers have bit length $c_0 \log n$ in Theorem 3.20. This consists of two sets of points $A, B \subseteq \mathbb{R}^d$ with $|A \cup B| = n$ for which we wish to compute $\min_{a \in A, b \in B} \|a - b\|_2$. By Theorem 3.20, the coordinates of points in A are also $c_0 \log n$ bit integers. Therefore, the set S of possible ℓ_2 distances between points in A and B is a set of square roots of integers with $\log d + c_0 \log n \leq 2c_0 \log n$ bits. Therefore, S is a ρ-discrete (recall $\rho = 1 + (16\log(10n))/L$ since $1 + 1/n^{2c_0} > 1 + (16 \log n)/L$. Furthermore, note that f is not (ρ, L)-multiplicatively Lipschitz by assumption.

We now describe an algorithm for solving ℓ_2-closest pair on $A \times B$. Use binary search on the values in S to compute the minimum distance between points in A, B. S consists of integers between 1 and n^{c_0} (pairs with distance 0 can be found in linear time), so $\gamma \leq n^{c_0}$. For each query point $k \in S$, by Lemma 8.8, there is a

$$O((\log n) \cdot \mathcal{T}(n, L, g(\gamma), d) + n) = O((\log n) \cdot \mathcal{T}(L^{1/(32c_0)}, L, L^{1/32}, c^{\log^* L}) + L^{1/(32c_0)})$$

-time algorithm for determining whether or not the closest pair has distance at most k. Therefore, there is a

$$O((\log n)\mathcal{T}(L^{1/(32c_0)}, L, L^{1/32}, c^{\log^* L}) + L^{1/(32c_0)}) = \widetilde{O}(L^{1/(32c_0)} L^{1/(64c_0)} \log(L^{1/32})) < \widetilde{O}(n^{3/2})$$

time algorithm for solving ℓ_2-closest pair on pairs of sets with n points. But this is impossible given SETH by Theorem 3.20, a contradiction. This completes the result. □

Proof of Theorem 8.7. Consider an instance of bichromatic Hamming nearest neighbor search for $n = 2^{L^{0.49}}$ and $d = c_1 \log n$ for the constant c_1 in the dimension bound in Theorem 3.17. This consists of two sets of points $A, B \subseteq \mathbb{R}^d$ with $|A \cup B| = n$ for which we wish to compute $\min_{a \in A, b \in B} \|a - b\|_2$. The coordinates of points in A and B are 0-1. Therefore, the set S of possible ℓ_2 distances between points in A and B is the set of square roots of integers between 0 and $c_1 \log n$, which differ by a factor of at least $1 + 1/(16 c_1 \log n) > \rho$ (recall $\rho = 1 + (16 \log(10n))/L$). Therefore, $A \cup B$ is ρ-spaced. Note that f is not (ρ, L)-multiplicatively Lipschitz by assumption. Furthermore, $A \cup B$ is $\sqrt{c_1 \log n} \leq L^{.25}$-boxed.

We now give an algorithm for solving ℓ_2-closest pair on $A \times B$. Use binary search on S. For each query $k \in S$, Lemma 8.8 implies that one can check if there is a pair with distance at most k in

$$\begin{aligned}\mathcal{T}(n, L, d) &= O(\mathcal{T}(2^{L^{.49}}, L, e^{L^{.25}}, 2^{c_1 L^{.49}}) + 2^{L^{.49}}) \\ &\leq O(2^{L^{.49} + L^{.48}}) \\ &= n^{1+o(1)}\end{aligned}$$

time on pairs of sets with n points. But this is impossible given SETH by Theorem 3.17. This completes the result. □

9 Fast Multipole Method

The fast multipole method (FMM) was described as one of the top-10 most important algorithms of the 20th century [110]. It is a numerical technique that was developed to speed up calculations of long-range forces in the n-body problem in physics. In 1987, FMM was first introduced by Greengard and Rokhlin [140], based on the multipole expansion of the vector Helmholtz equation. By treating the interactions between far-away basis functions using the FMM, the corresponding matrix elements do not need to be explicitly computed or stored. This is technique allows us to improve the naive $O(n^2)$ matrix-vector multiplication time to $o(n^2)$.

Since Greengard and Rokhlin invented FMM, the topic has attracted researchers from many different fields, including physics, math, and computer science [39, 102, 116, 137, 138, 139, 140, 141, 142, 143, 144, 218, 302, 303].

We first give a quick overview of the high-level ideas of FMM in Section 9.1. In Section 9.2, we provide a complete description and proof of correctness for the fast Gaussian transform, where the kernel function is the Gaussian kernel. Although a number of researchers have used FMM in the past, most of the previous papers about FMM either focus on the low-dimensional or low-error cases. We therefore focus on the superconstant-error, high dimensional case, and carefully analyze the joint dependence on ε and d. We believe that our presentation of the original proof in Section 9.2 is thus of independent interest to the community. In Section 9.4, we give the analogous results for other kernel functions used in this paper.

9.1 Overview

We begin with a description of high-level ideas of the Fast Multipole Method (FMM). Let $\mathsf{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ denote a kernel function. The inputs to the FMM are N sources $s_1, s_2, \cdots, s_N \in$

$\mathbb{R}^d$ and M targets $t_1, t_2, \cdots, t_M$. For each $i \in [N]$, the source s_i has a strength q_i. Suppose all sources are in a 'box' $\mathcal{B}$ and all the targets are in a 'box' $\mathcal{C}$. The goal is to evaluate

$$u_j = \sum_{i=1}^{N} \mathsf{K}(s_i, t_j) q_i, \quad \forall j \in [M]$$

Intuitively, if K has some nice property (e.g. smooth), we can hope to approximate K in the following sense

$$\mathsf{K}(s, t) \approx \sum_{p=0}^{P-1} B_p(s) \cdot C_p(t), \quad s \in \mathcal{B}, t \in \mathcal{C}$$

where P is a small positive integer, usually called the *interaction rank* in the literature.

Now, we can construct u_i in two steps:

$$v_p = \sum_{i \in \mathcal{B}} B_p(s_i) q_i, \quad \forall p = 0, 1, \cdots, P-1,$$

and

$$\widetilde{u}_j = \sum_{p=0}^{P-1} C_p(t_j) v_p, \quad \forall i \in [M].$$

Intuitively, as long as $\mathcal{B}$ and $\mathcal{C}$ are well-separated, then $\widetilde{u}_j$ is very good estimation to u_j even for small P, i.e., $|\widetilde{u}_j - u_j| < \varepsilon$.

Recall that, at the beginning of this section, we assumed that all the sources are in the the same box $\mathcal{B}$ and $\mathcal{C}$. This is not true in general. To deal with this, we can discretize the continuous space into a batch of boxes $\mathcal{B}_1, \mathcal{B}_2, \cdots$ and $\mathcal{C}_1, \mathcal{C}_2, \cdots$. For a box $\mathcal{B}_{l_1}$ and a box $\mathcal{C}_{l_2}$, if they are very far apart, then the interaction between points within them is small, and we can ignore it. If the two boxes are close, then we deal wit them efficiently by truncating the high order expansion terms in K (only keeping the first $\log^{O(d)}(1/\varepsilon)$). For each box, we will see that the number of nearby relevant boxes is at most $\log^{O(d)}(1/\varepsilon)$.

9.2 $\mathsf{K}(x, y) = \exp(-\|x - y\|_2^2)$, Fast Gaussian transform

Given N vectors $s_1, \cdots s_N \in \mathbb{R}^d$, M vectors $t_1, \cdots, t_M \in \mathbb{R}^d$ and a strength vector $q \in \mathbb{R}^n$, Greengard and Strain [144] provided a fast algorithm for evaluating discrete Gauss transform

$$G(t_i) = \sum_{j=1}^{N} q_j e^{-\|t_i - s_j\|^2/\delta}$$

for $i \in [M]$ in $O(M + N)$ time. In this section, we re-prove the algorithm described in [144], and determine the exact dependences on ε and d in the running time.

By shifting the origin and rescaling δ, we can assume that the sources s_j and targets t_i all lie in the unit box $\mathcal{B}_0 = [0, 1]^d$.

Let t and s lie in d-dimensional Euclidean space $\mathbb{R}^d$, and consider the Gaussian

$$e^{-\|t-s\|_2^2} = e^{-\sum_{i=1}^d (t_i - s_i)^2}$$

We begin with some definitions.

Definition 9.1 (one-dimensional Hermite polynomial)**.** *The Hermite polynomials $\widetilde{h}_n : \mathbb{R} \to \mathbb{R}$ is defined as follows*

$$\widetilde{h}_n(t) = (-1)^n e^{t^2} \frac{\mathrm{d}^n}{\mathrm{d}t} e^{-t^2}$$

Definition 9.2 (one-dimensional Hermite function)**.** *The Hermite functions $h_n : \mathbb{R} \to \mathbb{R}$ is defined as follows*

$$h_n(t) = e^{-t^2} \widetilde{h}_n(t)$$

We use the following Fact to simplify $e^{-(t-s)^2/\delta}$.

Fact 9.3. *For $s_0 \in \mathbb{R}$ and $\delta > 0$, we have*

$$e^{-(t-s)^2/\delta} = \sum_{n=0}^{\infty} \frac{1}{n!} \cdot \left(\frac{s - s_0}{\sqrt{\delta}} \right)^n \cdot h_n \left(\frac{t - s_0}{\sqrt{\delta}} \right)$$

and

$$e^{-(t-s)^2/\delta} = e^{-(t-s_0)^2/\delta} \sum_{n=0}^{\infty} \frac{1}{n!} \cdot \left(\frac{s - s_0}{\sqrt{\delta}} \right)^n \cdot \widetilde{h}_n \left(\frac{t - s_0}{\sqrt{\delta}} \right).$$

Proof.

$$\begin{aligned} e^{-(t-s)^2/\delta} &= e^{-(t-s_0-(s-s_0))^2/\delta} \\ &= \sum_{n=0}^{\infty} \frac{1}{n!} \left(\frac{s - s_0}{\sqrt{\delta}} \right)^n h_n \left(\frac{t - s_0}{\sqrt{\delta}} \right) \\ &= e^{-(t-s_0)^2/\delta} \sum_{n=0}^{\infty} \frac{1}{n!} \left(\frac{s - s_0}{\sqrt{\delta}} \right)^n \widetilde{h}_n \left(\frac{t - s_0}{\sqrt{\delta}} \right). \end{aligned}$$

□

Using Cramer's inequality, we have the following standard bound.

Lemma 9.4. *For any constant $K \leq 1.09$, we have*

$$|\widetilde{h}_n(t)| \leq K \cdot 2^{n/2} \cdot \sqrt{n!} \cdot e^{t^2/2}$$

and

$$|h_n(t)| \leq K \cdot 2^{n/2} \cdot \sqrt{n!} \cdot e^{-t^2/2}.$$

Next, we will extend the above definitions and observations to the high dimensional case. To simplify the discussion, we define multi-index notation. A multi-index $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_d)$ is a d-tuple of nonnegative integers, playing the role of a multi-dimensional index. For any multi-index $\alpha \in \mathbb{R}^d$ and any $t \in \mathbb{R}^t$, we write

$$\alpha! = \prod_{i=1}^{d} (\alpha_i!), \quad t^\alpha = \prod_{i=1}^{d} t_i^{\alpha_i}, \quad D^\alpha = \partial_1^{\alpha_1} \partial_2^{\alpha_2} \cdots \partial_d^{\alpha_d}.$$

where ∂_i is the differentiatial operator with respect to the i-th coordinate in $\mathbb{R}^d$. For integer p, we say $\alpha \geq p$ if $\alpha_i \geq p, \forall i \in [d]$.

We can now define:

Definition 9.5 (multi-dimensional Hermite polynomial)**.** *We define function $\widetilde{H}_\alpha : \mathbb{R}^d \to \mathbb{R}$ as follows:*

$$\widetilde{H}_\alpha(t) = \prod_{i=1}^{d} \widetilde{h}_{\alpha_i}(t_i).$$

Definition 9.6 (multi-dimensional Hermite function)**.** *We define function $H_\alpha : \mathbb{R}^d \to \mathbb{R}$ as follows:*

$$H_\alpha(t) = \prod_{i=1}^{d} h_{\alpha_i}(t_i).$$

It is easy to see that $H_\alpha(t) = e^{-\|t\|_2^2} \cdot \widetilde{H}_\alpha(t)$

The Hermite expansion of a Gaussian in $\mathbb{R}^d$ is

$$e^{-\|t-s\|_2^2} = \sum_{\alpha \geq 0} \frac{(t-s_0)^\alpha}{\alpha!} h_\alpha(s - s_0). \tag{5.4}$$

Cramer's inequality generalizes to

Lemma 9.7 (Cramer's inequality)**.** *Let $K < (1.09)^d$, then*

$$|\widetilde{H}_\alpha(t)| \leq K \cdot e^{\|t\|_2^2/2} \cdot 2^{\|\alpha\|_1/2} \cdot \sqrt{\alpha!}$$

and

$$|H_\alpha(t)| \leq K \cdot e^{-\|t\|_2^2/2} \cdot 2^{\|\alpha\|_1/2} \cdot \sqrt{\alpha!}$$

The Taylor series of H_α is

$$H_\alpha(t) = \sum_{\beta \geq 0} \frac{(t-t_0)^\beta}{\beta!} (-1)^{\|\beta\|_1} H_{\alpha+\beta}(t_0) \tag{5.5}$$

Estimation

We first give a definition

Definition 9.8. *Let $\mathcal{B}$ denote a box with center $s_{\mathcal{B}}$ and side length $r\sqrt{2\delta}$ with $r < 1$. If source s_j is in box $\mathcal{B}$, we say $j \in \mathcal{B}$. Then the Gaussian evaluation from the sources in box $\mathcal{B}$ is,*

$$G(t) = \sum_{j \in \mathcal{B}} q_j \cdot e^{-\|t-s_j\|_2^2/\delta}.$$

The Hermite expansion of $G(t)$ is

$$G(t) = \sum_{\alpha \geq 0} A_\alpha \cdot h_\alpha \left(\frac{t - s_{\mathcal{B}}}{\sqrt{\delta}} \right), \tag{5.6}$$

where the coefficients A_α are defined by

$$A_\alpha = \frac{1}{\alpha!} \sum_{j \in \mathcal{B}} q_j \cdot \left(\frac{s_j - s_{\mathcal{B}}}{\sqrt{\delta}} \right)^\alpha \tag{5.7}$$

The rest of this section will present a batch of Lemmas that bound the error of the function truncated at certain degree of Taylor and Hermite expansion.

Lemma 9.9. *Let p denote an integer, let $\mathrm{Err}_H(p)$ denote the error after truncating the series $G(t)$ (as defined in Def. 9.8) after p^d terms, i.e.,*

$$\mathrm{Err}_H(p) = \sum_{\alpha \geq p} A_\alpha \cdot H_\alpha \left(\frac{t - s_{\mathcal{B}}}{\sqrt{\delta}} \right).$$

Then we have

$$|\mathrm{Err}_H(p)| \leq K \cdot \sum_{j \in \mathcal{B}} |q_j| \cdot \left(\frac{1}{p!} \right)^{d/2} \cdot \left(\frac{r^{p+1}}{1-r} \right)^d,$$

where $K = (1.09)^d$.

Proof. Using Eq. (5.4) to expand each Gaussian (see Definition 9.8) in the

$$G(t) = \sum_{j \in \mathcal{B}} q_j \cdot e^{-\|t-s_j\|_2^2/\delta}$$

into a Hermite series about $s_{\mathcal{B}}$

$$\sum_{\alpha \geq 0} \left(\frac{1}{\alpha!} \sum_{j \in \mathcal{B}} q_j \cdot \left(\frac{s_j - s_{\mathcal{B}}}{\sqrt{\delta}} \right)^\alpha \right) H_\alpha \left(\frac{t - s_{\mathcal{B}}}{\sqrt{\delta}} \right)$$

and swap the summation over α and j to obtain

$$\sum_{j \in \mathcal{B}} q_j \sum_{\alpha \geq 0} \frac{1}{\alpha!} \cdot \left(\frac{s_j - s_{\mathcal{B}}}{\sqrt{\delta}} \right)^\alpha \cdot H_\alpha \left(\frac{t - s_{\mathcal{B}}}{\sqrt{\delta}} \right)$$

The truncation error bound follows from Cramer's inequality (Lemma 9.7) and the formula for the tail of a geometric series.

□

The next Lemma shows how to convert a Hermite expansion about $s_{\mathcal{B}}$ into a Taylor expansion about $t_{\mathcal{C}}$. The Taylor series converges rapidly in a box of side length $r\sqrt{2\delta}$ about $t_{\mathcal{C}}$, where $r < 1$.

Lemma 9.10. *The Hermite expansion of $G(t)$ is*

$$G(t) = \sum_{\alpha \geq 0} A_\alpha \cdot H_\alpha \left(\frac{t - s_{\mathcal{B}}}{\sqrt{\delta}} \right)$$

has the following Taylor expansion, at an arbitrary point t_0 :

$$G(t) = \sum_{\beta \geq 0} B_\beta \left(\frac{t - t_0}{\sqrt{\delta}} \right)^\beta . \tag{5.8}$$

where the coefficients B_β are defined as

$$B_\beta = \frac{(-1)^{|\beta|}}{\beta!} \sum_{\alpha \geq 0} A_\alpha \cdot H_{\alpha+\beta} \left(\frac{s_{\mathcal{B}} - t_0}{\sqrt{\delta}} \right) . \tag{5.9}$$

Let $\mathrm{Err}_T(p)$ denote the error by truncating the Taylor series after p^d terms, in the box $\mathcal{C}$ with center $t_{\mathcal{C}}$ and side length $r\sqrt{2\delta}$, i.e.,

$$\mathrm{Err}_T(p) = \sum_{\beta \geq p} B_\beta \left(\frac{t - t_{\mathcal{C}}}{\sqrt{\delta}} \right)^\beta$$

Then

$$|\mathrm{Err}_T(p)| \leq K \cdot Q_B \cdot \left(\frac{1}{p!} \right)^{d/2} \left(\frac{r^{p+1}}{1 - r} \right)^d .$$

Proof. Each Hermite function in Eq. (5.6) can be expanded into a Taylor series by means of Eq. (5.5). The expansion in Eq. (5.8) is obtained by swapping the order of summation.

The truncation error bound can be proved as follows. Using Eq. (5.7) for A_α, we can rewrite B_β:

$$\begin{aligned}
B_\beta &= \frac{(-1)^{|\beta|}}{\beta!} \sum_{\alpha \geq 0} A_\alpha H_{\alpha+\beta} \left(\frac{s_{\mathcal{B}} - t_{\mathcal{C}}}{\sqrt{\delta}} \right) \\
&= \frac{(-1)^{|\beta|}}{\beta!} \sum_{\alpha \geq 0} \left(\frac{1}{\alpha!} \sum_{j \in \mathcal{B}} q_j \left(\frac{s_j - s_{\mathcal{B}}}{\sqrt{\delta}} \right)^\alpha \right) H_{\alpha+\beta} \left(\frac{s_{\mathcal{B}} - t_{\mathcal{C}}}{\sqrt{\delta}} \right) \\
&= \frac{(-1)^{|\beta|}}{\beta!} \sum_{j \in \mathcal{B}} q_j \sum_{\alpha \geq 0} \frac{1}{\alpha!} \left(\frac{s_j - s_{\mathcal{B}}}{\sqrt{\delta}} \right)^\alpha \cdot H_{\alpha+\beta} \left(\frac{s_{\mathcal{B}} - t_{\mathcal{C}}}{\sqrt{\delta}} \right)
\end{aligned}$$

By Eq. (5.5), the inner sum is the Taylor expansion of $H_\beta((s_j - t_{\mathcal{C}})/\sqrt{\delta})$. Thus

$$B_\beta = \frac{(-1)^{\|\beta\|_1}}{\beta!} \sum_{j \in \mathcal{B}} q_j \cdot H_\beta \left(\frac{s_j - t_{\mathcal{C}}}{\sqrt{\delta}} \right)$$

and Cramer's inequality implies

$$|B_\beta| \le \frac{1}{\beta!} K \cdot Q_B 2^{\|\beta\|_1/2} \sqrt{\beta!} \le K Q_B \frac{2^{\|\beta\|_1/2}}{\sqrt{\beta!}}$$

The truncation error follows from summation the tail of a geometric series. □

For the purpose of designing our algorithm, we'd like to make a variant of Lemma 9.10 in which the Hermite series is truncated before converting it to a Taylor series. This means that in addition to truncating the Taylor series itself, we are also truncating the finite sum formula in Eq. (5.9) for the coefficients.

Lemma 9.11. *Let $G(t)$ be defined as Def 9.8. For an integer p, let $G_p(t)$ denote the Hermite expansion of $G(t)$ truncated at p,*

$$G_p(t) = \sum_{\alpha \le p} A_\alpha H_\alpha \left(\frac{t - s_{\mathcal{B}}}{\sqrt{\delta}} \right).$$

The function $G_p(t)$ has the following Taylor expansion about an arbitrary point t_0:

$$G_p(t) = \sum_{\beta \ge 0} C_\beta \cdot \left(\frac{t - t_0}{\sqrt{\delta}} \right)^\beta,$$

where the the coefficients C_β are defined as

$$C_\beta = \frac{(-1)^{\|\beta\|_1}}{\beta!} \sum_{\alpha \le p} A_\alpha \cdot H_{\alpha+\beta} \left(\frac{s_{\mathcal{B}} - t_{\mathcal{C}}}{\sqrt{\delta}} \right). \tag{5.10}$$

Let $\mathrm{Err}_T(p)$ denote the error in truncating the Taylor series after p^d terms, in the box $\mathcal{C}$ with center $t_{\mathcal{C}}$ and side length $r\sqrt{2\delta}$, i.e.,

$$\mathrm{Err}_T(p) = \sum_{\beta \ge p} C_\beta \left(\frac{t - t_{\mathcal{C}}}{\sqrt{\delta}} \right)^\beta.$$

Then, we have

$$|\mathrm{Err}_T(p)| \le K' \cdot Q_B \left(\frac{1}{p!} \right)^{d/2} \left(\frac{r^{p+1}}{1-r} \right)^d$$

where $K' \le 2K$ and $r \le 1/2$.

Proof. We can write C_β in the following way:

$$\begin{aligned}
C_\beta &= \frac{(-1)^{\|\beta\|_1}}{\beta!} \sum_{j \in \mathcal{B}} q_j \sum_{\alpha \le p} \frac{1}{\alpha!} \left(\frac{s_j - s_{\mathcal{B}}}{\sqrt{\delta}} \right)^\alpha \cdot H_{\alpha+\beta} \left(\frac{s_{\mathcal{B}} - t_{\mathcal{C}}}{\sqrt{\delta}} \right) \\
&= \frac{(-1)^{\|\beta\|_1}}{\beta!} \sum_{j \in \mathcal{B}} q_j \left(\sum_{\alpha \ge 0} - \sum_{\alpha > p} \right) \frac{1}{\alpha!} \left(\frac{s_j - s_{\mathcal{B}}}{\sqrt{\delta}} \right)^\alpha \cdot H_{\alpha+\beta} \left(\frac{s_{\mathcal{B}} - t_{\mathcal{C}}}{\sqrt{\delta}} \right) \\
&= B_\beta - \frac{(-1)^{\|\beta\|_1}}{\beta!} \sum_{j \in \mathcal{B}} q_j \sum_{\alpha > p} \frac{1}{\alpha!} \left(\frac{s_j - s_{\mathcal{B}}}{\sqrt{\delta}} \right)^\alpha \cdot H_{\alpha+\beta} \left(\frac{s_{\mathcal{B}} - t_{\mathcal{C}}}{\sqrt{\delta}} \right) \\
&= B_\beta + (C_\beta - B_\beta)
\end{aligned}$$

Next, we have

$$|\mathrm{Err}_T(p)| \leq \left|\sum_{\beta \geq p} B_\beta \left(\frac{t - t_{\mathcal{C}}}{\sqrt{\delta}}\right)^\beta\right| + \left|\sum_{\beta \geq p} (C_\beta - B_\beta) \cdot \left(\frac{t - t_{\mathcal{C}}}{\sqrt{\delta}}\right)^\beta\right| \tag{5.11}$$

Using Lemma 9.10, we can upper bound the first term in the Eq. (5.11) by,

$$K \cdot Q_B \left(\frac{1}{p!}\right)^{d/2} \cdot \left(\frac{r^{p+1}}{1-r}\right)^d$$

To bound the second term in Eq. (5.11), we can do the following

$$\begin{aligned}
&\left|\sum_{\beta \geq p} (C_\beta - B_\beta) \cdot \left(\frac{t - t_{\mathcal{C}}}{\sqrt{\delta}}\right)^\beta\right| \\
&\leq Q_B \cdot \sum_{\beta \geq p} \left|\left(\frac{t - t_{\mathcal{C}}}{\sqrt{\delta}}\right)^\beta\right| \cdot \frac{1}{\beta!} \sum_{\alpha > p} \frac{1}{\alpha!} \left|\left(\frac{s_j - s_{\mathcal{B}}}{\sqrt{\delta}}\right)^\alpha\right| \cdot \left|H_{\alpha+\beta}\left(\frac{s_{\mathcal{B}} - t_{\mathcal{C}}}{\sqrt{\delta}}\right)\right| \\
&\leq K Q_B \sum_{\alpha > p} \sum_{\beta > p} \frac{r^{\|\alpha\|_1}}{\sqrt{\alpha!}} \cdot \sqrt{\frac{(\alpha+\beta)!}{\alpha!\beta!}} \cdot \frac{r^{\|\beta\|_1}}{\sqrt{\beta!}}
\end{aligned}$$

Finally, the proof is complete since we know that

$$\frac{(\alpha+\beta)!}{\alpha!\beta!} \leq 2^{\|\alpha+\beta\|_1}.$$

□

The proof of the following Lemma is almost identical. We omit the details here.

Lemma 9.12. *Let $G_{s_j}(t)$ be defined as*

$$G_{s_j}(t) = q_j \cdot e^{-\|t - s_j\|_2^2/\delta}$$

has the following Taylor expansion at $t_{\mathcal{C}}$

$$G_{s_j}(t) = \sum_{\beta \geq 0} \mathcal{B}_\beta \left(\frac{t - t_{\mathcal{C}}}{\sqrt{\delta}}\right)^\beta,$$

where the coefficients B_β is defined as

$$B_\beta = q_j \cdot \frac{(-1)^{\|\beta\|_1}}{\beta!} \cdot H_\beta \left(\frac{s_j - t_{\mathcal{C}}}{\sqrt{\delta}}\right)$$

and the error in truncation after p^d terms is

$$|\mathrm{Err}_T(p)| = \left|\sum_{\beta \geq p} B_\beta \left(\frac{t - t_{\mathcal{C}}}{\sqrt{\delta}}\right)^\beta\right| \leq K \cdot q_j \cdot \left(\frac{1}{p!}\right)^{d/2} \cdot \left(\frac{r^{p+1}}{1-r}\right)^d$$

for $r < 1$.

Algorithm

The algorithm is based on subdividing B_0 into smaller boxes with sides of length $r\sqrt{2\delta}$ parallel to the axes, for a fixed $r \leq 1/2$. We can then assign each source s_j to the box $\mathcal{B}$ in which it lies and each target t, to the box $\mathcal{C}$ in which it lies.

For each target box $\mathcal{C}$, we need to evaluate the total field due to sources in all boxes. Since boxed $\mathcal{B}$ have side lengths $r\sqrt{2\delta}$, only a fixed number of source boxes $\mathcal{B}$ can contribute more than $Q\varepsilon$ to the field in a given target box $\mathcal{C}$, where $Q = \|q\|_1$ and ε is the precision parameter. If we cut off the sum over all $\mathcal{B}$ after including the $(2k+1)^d$ nearest boxes to $\mathcal{C}$, it incurs an error which can be upper bounded as follows

$$\begin{aligned}\sum_{j:\|t-s_j\|_\infty \geq kr\sqrt{2\delta}} |q_j| \cdot e^{-\|t-s_j\|_2^2/\delta} &\leq \sum_{j:\|t-s_j\|_\infty \geq kr\sqrt{2\delta}} |q_j| \cdot e^{-\|t-s_j\|_\infty^2/\delta} \\ &\leq \sum_{j:\|t-s_j\|_\infty \geq kr\sqrt{2\delta}} |q_j| \cdot e^{-(k\cdot r\sqrt{2\delta})^2/\delta} \\ &\leq Q \cdot e^{-2r^2k^2} \end{aligned} \tag{5.12}$$

where the first step follows from $\|\cdot\|_2 \geq \|\cdot\|_\infty$, the second step follows from $\|t-s_j\|_\infty \geq kr\sqrt{2\delta}$, and the last step follows from a straightforward calculation.

For a box $\mathcal{B}$ and a box $\mathcal{C}$, there are several possible ways to evaluate the interaction between $\mathcal{B}$ and $\mathcal{C}$. We mainly need the following three techniques:

1. $N_\mathcal{B}$ Gaussians, accumulated in Taylor series via definition B_β in Lemma 9.12
2. Hermite series, directly evaluated
3. Hermite series, accumulated in Taylor series in Lemma 9.11

Essentially, having any two of the above three techniques is sufficient to give an algorithm that runs in $(M+N)\log^{O(d)}(\|q\|_1/\varepsilon)$ time.

In the next a few paragraphs, we explain the details of the three techniques.

Technique 1. Consider a fixed source box $\mathcal{B}$. For each target box $\mathcal{C}$ within range, we must compute p^d Taylor series coefficients

$$C_\beta(\mathcal{B}) = \frac{(-1)^{|\beta|}}{\beta!} \sum_{j\in\mathcal{B}} H_\beta\left(\frac{s_j - t_C}{\sqrt{\delta}}\right).$$

Each coefficient requires $O(N_\mathcal{B})$ work to evaluate, resulting in a net cost $O(p^d N_\mathcal{B})$. Since there are at most $(2k+1)^d$ boxes within range, the total work for forming all the Taylor series is $O((2k+1)^d p^d N)$. Now, for each target t_i, one must evaluate the p^d-term Taylor series corresponding to the box in which t_i lies. The total running time of algorithm is thus

$$O((2k+1)^d p^2 N) + O(p^d M).$$

Technique 2. We form a Hermite series for each box $\mathcal{B}$ and evaluate it at all targets. Using Lemma 9.9, we can rewrite $G(t)$ as

$$\begin{aligned} G(t) &= \sum_{\mathcal{B}} \sum_{j \in \mathcal{B}} q_j \cdot e^{-\|t-s_j\|_2^2/\delta} \\ &= \sum_{\mathcal{B}} \sum_{\alpha \geq 0} A_\alpha(B) H_\alpha \left(\frac{t - s_{\mathcal{B}}}{\sqrt{\delta}} \right) + \mathrm{Err}_H(p) \end{aligned}$$

where $|\mathrm{Err}_H(p)| \leq \varepsilon$ and

$$A_\alpha(\mathcal{B}) = \frac{1}{\alpha!} \sum_{j \in \mathcal{B}} q_j \cdot \left(\frac{s_j - s_{\mathcal{B}}}{\sqrt{\delta}} \right)^\alpha. \tag{5.13}$$

To compute each $A_\alpha(\mathcal{B})$ costs $O(N_{\mathcal{B}})$ time, so forming all the Hermite expansions takes $O(p^d N)$ time. Evaluating at most $(2k+1)^d$ expansions at each target t_i costs $O((2k+1)^d p^d)$ time per target, so this approach takes

$$O(p^d N) + O((2k+1)^d p^d M)$$

time in total.

Technique 3. Let $N(B)$ denote the number of boxes. Note that $N(B) \leq \min((r\sqrt{2\delta})^{-d/2}, M)$.

Suppose we accumulate all sources into truncated Hermite expansions and transform all Hermite expansions into Taylor expansions via Lemma 9.11. Then we can approximate the function $G(t)$ by

$$\begin{aligned} G(t) &= \sum_{\mathcal{B}} \sum_{j \in \mathcal{B}} q_j \cdot e^{-\|t-s_j\|_2^2/\delta} \\ &= \sum_{\beta \leq p} C_\beta \left(\frac{t - t_{\mathcal{C}}}{\sqrt{\delta}} \right)^\beta + \mathrm{Err}_T(p) + \mathrm{Err}_H(p) \end{aligned}$$

where $|\mathrm{Err}_H(p)| + |\mathrm{Err}_T(p)| \leq Q \cdot \varepsilon$,

$$C_\beta = \frac{(-1)^{\|\beta\|_1}}{\beta!} \sum_{\mathcal{B}} \sum_{\alpha \leq p} A_\alpha(\mathcal{B}) H_{\alpha+\beta} \left(\frac{s_{\mathcal{B}} - t_{\mathcal{C}}}{\sqrt{\delta}} \right)$$

and the coefficients $A_\alpha(\mathcal{B})$ are defined as Eq. (5.13). Recall in Part 2, it takes $O(p^d N)$ time to compute all the Hermite expansions, i.e., to compute the coefficients $A_\alpha(\mathcal{B})$ for all $\alpha \leq p$ and all sources boxes $\mathcal{B}$.

Making use of the large product in the definition of $H_{\alpha+\beta}$, we see that the time to compute the p^d coefficients of C_β is only $O(dp^{d+1})$ for each box $\mathcal{B}$ in the range. Thus, we know for each target box $\mathcal{C}$, the running time is

$$O((2k+1)^d dp^{d+1}).$$

Finally we need to evaluate the appropriate Taylor series for each target t_i, which can be done in $O(p^d M)$ time. Putting it all together, this technique 3 takes time

$$O((2k+1)^d dp^{d+1} N(B)) + O(p^d N) + O(p^d M).$$

Result

Finally, in order to get ε additive error for each coordinate, we will choose $k = O(\log(\|q\|_1/\varepsilon))$ and $p = O(\log(\|q\|_1/\varepsilon))$.

Theorem 9.13 (fast Gaussian transform)**.** *Given N vectors $s_1, \cdots, s_N \in \mathbb{R}^d$, M vectors $t_1, t_2, \cdots, t_M \in \mathbb{R}^d$, a number $\delta > 0$, and a vector $q \in \mathbb{R}^n$, let function $G : \mathbb{R}^d \to \mathbb{R}$ be defined as $G(t) = \sum_{i=1}^N q_i \cdot e^{-\|t-s_i\|_2^2/\delta}$. There is an algorithm that runs in*

$$O\left((M+N)\log^{O(d)}(\|q\|_1/\varepsilon)\right)$$

time, and outputs M numbers $x_1, \cdots, x_M$ such that for each $j \in [M]$

$$G(t_j) - \varepsilon \le x_j \le G(t_j) + \varepsilon.$$

The proof of fast Gaussian transform also implies a result for the online version:

Theorem 9.14 (online version)**.** *Given N vectors $s_1, \cdots, s_N \in \mathbb{R}^d$, a number $\delta > 0$, and a vector $q \in \mathbb{R}^n$, let function $G : \mathbb{R}^d \to \mathbb{R}$ be defined as $G(t) = \sum_{i=1}^N q_i \cdot e^{-\|t-s_i\|_2^2/\delta}$. There is an algorithm that takes*

$$O\left(N\log^{O(d)}(\|q\|_1/\varepsilon)\right)$$

time to do preprocessing, and then for each $t \in \mathbb{R}^d$, takes $O(\log^{O(d)}(\|q\|_1/\varepsilon))$ time to output a number x such that

$$G(t) - \varepsilon \le x \le G(t) + \varepsilon.$$

9.3 Generalization

The fast multipole method described in the previous section works not only for the Gaussian kernel, but also for $\mathsf{K}(u,v) = f(\|u,v\|_2^2)$ for many other functions f. As long as f has the following properties, the result of Theorem 9.13 also holds for f:

- $f : \mathbb{R} \to \mathbb{R}_+$.
- f is non-increasing, i.e., if $x \ge y \ge 0$, then $f(x) \le f(y)$.
- f is decreasing fast, i.e., for any $\varepsilon \in (0,1)$, we have $f(\Theta(\log(1/\varepsilon))) \le \varepsilon$.
- f's Hermite expansion and Taylor expansions are truncateable: If we only keep $\log^d(1/\varepsilon)$ terms of the polynomial for K, then the error is at most ε.

Let us now sketch how each of these properties is used in discritizing the continuous domain into a finite number of boxes. First, note that Eq.(5.12) holds more generally for any function f with these properties. Indeed, we can bound the error as follows (note that $Q = \|q\|_1$):

$$\begin{aligned}
\sum_{j:\|t-s_j\|_\infty \ge kr\sqrt{2\delta}} |q_j| \cdot f(\|t-s_j\|_2/\sqrt{\delta}) &\le \sum_{j:\|t-s_j\|_\infty \ge kr\sqrt{2\delta}} |q_j| \cdot f(\|t-s_j\|_\infty/\sqrt{\delta}) \\
&\le \sum_{j:\|t-s_j\|_\infty \ge kr\sqrt{2\delta}} |q_j| \cdot f(\sqrt{2}kr) \\
&\le Q \cdot f(\sqrt{2}kr) \\
&\le \varepsilon \qquad (5.14)
\end{aligned}$$

where the first step follows from $\|\cdot\|_2 \geq \|\cdot\|_\infty$ and that f is non-increasing, the second step follows from $\|t - s_j\|_\infty \geq kr\sqrt{2\delta}$ and that f is non-increasing, and the last step follows from the fact that f is decreasing fast, and choosing $k = O(\log(Q/\varepsilon)/r)$.

We next give an example of how the truncatable expansions property is used. Here we only show to generalize Definition 9.8 to Definition 9.15 and generalize Lemma 9.9 to Definition 9.16; the other Lemmas in the proof can be extended in a similar way.

Definition 9.15. *Let $\mathcal{B}$ denote a box with center $s_{\mathcal{B}}$ and side length $r\sqrt{2\delta}$ with $r < 1$. If source s_j is in box $\mathcal{B}$, we say $j \in \mathcal{B}$. Then the Gaussian evaluation from the sources in box $\mathcal{B}$ is,*

$$G(t) = \sum_{j \in \mathcal{B}} q_j \cdot f(\|t - s_j\|_2/\sqrt{\delta}).$$

The Hermite expansion of $G(t)$ is

$$G(t) = \sum_{\alpha \geq 0} A_\alpha \cdot h_\alpha \left(\frac{t - s_{\mathcal{B}}}{\sqrt{\delta}}\right), \tag{5.15}$$

where the coefficients A_α are defined by

$$A_\alpha = \frac{1}{\alpha!} \sum_{j \in \mathcal{B}} q_j \cdot \left(\frac{s_j - s_{\mathcal{B}}}{\sqrt{\delta}}\right)^\alpha \tag{5.16}$$

Definition 9.16. *Let p denote an integer, let $\mathrm{Err}_H(p)$ denote the error after truncating the series $G(t)$ (as defined in Def. 9.15) after p^d terms, i.e.,*

$$\mathrm{Err}_H(p) = \sum_{\alpha \geq p} A_\alpha \cdot H_\alpha \left(\frac{t - s_{\mathcal{B}}}{\sqrt{\delta}}\right).$$

We say f is Hermite truncateable, if Then we have

$$|\mathrm{Err}_H(p)| \leq p^{-\Omega(pd)}$$

where $K = (1.09)^d$.

9.4 $\mathsf{K}(x, y) = 1/\|x - y\|_2^2$

Similar ideas yield the following algorithm:

Theorem 9.17 (FMM, [39, 218])**.** *Given n vectors $x_1, x_2, \cdots, x_n \in \mathbb{R}^d$, let matrix $A \in \mathbb{R}^{n \times n}$ be defined as $A_{i,j} = 1/\|x_i - x_j\|_2^2$. For any vector $h \in \mathbb{R}^n$, in time $O(n \log^{O(d)}(\|u\|_1/\varepsilon))$, we can output a vector u such that*

$$(Ah)_i - \varepsilon \leq u_i \leq (Ah)_i + \varepsilon.$$

10 Neural Tangent Kernel

In this section, we show that the popular Neural Tangent Kernel K from theoretical Deep Learning can be rearranged into the form $\mathsf{K}(x,y) = f(\|x-y\|_2^2)$ for an appropriate analytic function f, so our results in this chapter apply to it. We first define the kernel.

Definition 10.1 (Neural Tangent Kernel, [164])**.** *Given n points $x_1, x_2, \cdots, x_n \in \mathbb{R}^d$, and any activation function $\sigma : \mathbb{R} \to \mathbb{R}$, the neural tangent kernel matrix $\mathsf{K} \in \mathbb{R}^{n\times n}$ can be defined as follows, where $\mathcal{N}(0, I_d)$ denotes the Gaussian distribution:*

$$\mathsf{K}_{i,j} := \int_{\mathcal{N}(0,I_d)} \sigma'(w^\top x_i)\sigma'(w^\top x_j) x_i^\top x_j \mathrm{d}w.$$

In the literature of convergence results for deep neural networks [12, 13, 58, 111, 195, 203, 265], it is natural to assume that all the data points are on the unit sphere, i.e., for all $i \in [n]$ we have $\|x_i\|_2 = 1$ and datas are separable i.e., for all $i \neq j$, $\|x_i - x_j\|_2 \geq \delta$. One of the most standard and common used activation functions in neural network training is ReLU activation, which is $\sigma(x) = \max\{x, 0\}$. Using Lemma 10.2, we can figure out the corresponding kernel function. By Theorem 5.8, the multiplication task for neural tangent kernels is hard. In neural network training, the multiplication can potentially being used to speed the neural network training procedure(See [265]).

In the following lemma, we compute the kernel function for ReLU activation function.

Lemma 10.2. *If $\sigma(x) = \max\{x, 0\}$, then the Neural Tangent Kernel can be written as $\mathsf{K}(x,y) = f(\|x-y\|_2^2)$ for*

$$f(x) = \frac{1}{\pi}(\pi - \cos^{-1}(1 - 0.5x)) \cdot (1 - 0.5x)$$

Proof. First, since $\|x_i\|_2 = \|x_j\|_2$, we know that

$$\|x_i - x_j\|_2^2 = \|x_i\|_2^2 - 2\langle x_i, x_j\rangle + \|x_j\|_2^2 = 2 - 2\langle x_i, x_j\rangle.$$

By definition of σ, we know

$$\sigma(x) = \begin{cases} 1, & \text{if } x > 0; \\ 0, & \text{otherwise.} \end{cases}$$

Using properties of the Gaussian distribution $\mathcal{N}(0, I_d)$, we have

$$\begin{aligned} \int_{\mathcal{N}(0,I_d)} \sigma'(w^\top x_i)\sigma'(w^\top x_j)\mathrm{d}w &= \frac{1}{\pi}(\pi - \cos^{-1}(x_i x_j)) \\ &= \frac{1}{\pi}(\pi - \cos^{-1}(1 - 0.5\|x_i - x_j\|_2^2)) \end{aligned}$$

We can rewrite $\mathsf{K}_{i,j}$ as follows:

$$\begin{aligned}\mathsf{K}_{i,j} &= \int_{\mathcal{N}(0,I_d)} \sigma'(w^\top x_i)\sigma'(w^\top x_j)x_i^\top x_j \mathrm{d}w \\ &= (1 - 0.5\|x_i - x_j\|_2^2) \cdot \int_{\mathcal{N}(0,I_d)} \sigma'(w^\top x_i)\sigma'(w^\top x_j)\mathrm{d}w \\ &= (1 - 0.5\|x_i - x_j\|_2^2) \cdot \frac{1}{\pi}(\pi - \cos^{-1}(1 - 0.5x)) \\ &= f(\|x_i - x_j\|_2^2).\end{aligned}$$

□

Lemma 10.3. *Given n data points $x_1, \ldots, x_n \in \mathbb{R}^d$ on unit sphere. For any activation function $\sigma : \mathbb{R} \to \mathbb{R}$, the corresponding Neural Tangent Kernel $\mathsf{K}(x_i, x_j)$ is a function of $\|x_i - x_j\|_2^2$.*

Proof. Note that $w \sim \mathcal{N}(0, I_d)$, so we know $(x_i^\top w, x_j^\top w) \sim \mathcal{N}(0, \Sigma_{i,j})$, where the covariance matrix

$$\Sigma_{i,j} = \begin{bmatrix} x_i^\top x_i & x_i^\top x_j \\ x_j^\top x_i & x_j^\top x_j \end{bmatrix} = \begin{bmatrix} 1 & x_i^\top x_j \\ x_i^\top x_j & 1 \end{bmatrix} \in \mathbb{R}^{2\times 2},$$

since $\|x_i\|_2 = \|x_j\|_2 = 1$. Thus,

$$\mathsf{K}(x_i, x_j) = \mathbf{E}_{(a,b)\sim\mathcal{N}(0,\Sigma_{i,j})}[\sigma'(a)\sigma'(b)]x_i^\top x_j = g(x_i^\top x_j)$$

for some function g.

Note $x_i^\top x_j = -\frac{1}{2}\|x_i - x_j\|^2 + 1$, so $\mathsf{K}(x_i, x_j) = f(\|x_i - x_j\|_2^2)$ for some function f, which completes the proof. □

Bibliography

[1] Mridul Aanjaneya, Frédéric Chazal, Daniel Chen, Marc Glisse, Leonidas Guibas, and Dmitriy Morozov. Metric graph reconstruction from noisy data. *International Journal of Computational Geometry and Applications (IJCGA)*, 22(04):305–325, 2012. 1.2

[2] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 218–230. SIAM, 2014. 2.1.3

[3] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 39–51. Springer, 2014. 3.12

[4] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 218–230, 2015. 3.6

[5] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003. doi: 10.1016/S0022-0000(03)00025-4. URL `https://doi.org/10.1016/S0022-0000(03)00025-4`. 3.7, 3.13

[6] Michal Adamszek, Henry Adams, Florian Frick, Chris Peterson, and Corrine Previte-Johnson. Nerve complexes of circular arcs. *Discrete & Computational Geometry*, 56(2): 251–273, 2016. 4

[7] Pankaj K Agarwal and R Sharathkumar. Approximation algorithms for bipartite matching with metric and geometric costs. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing (STOC)*, pages 555–564, 2014. 1

[8] Pankaj K Agarwal, Herbert Edelsbrunner, Otfried Schwarzkopf, and Emo Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Discrete & Computational Geometry*, 6(3):407–422, 1991. 3.19

[9] Pankaj K. Agarwal, Hsien-Chih Chang, and Allen Xiao. Efficient algorithms for geometric partial matching. In *35th International Symposium on Computational Geometry (SoCG)*, pages 6:1–6:14, 2019. 1

[10] Morteza Alamgir and Ulrike von Luxburg. Shortest path distance in random k-nearest neighbor graphs. In *Proceedings of the 29th International Conference on Machine Learning*, 2012. 1, 1, 1.1, G

[11] Vedat Levi Alev, Nima Anari, Lap Chi Lau, and Shayan Oveis Gharan. Graph clustering using effective resistance. In *ITCS*, 2018. 3.5

[12] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. On the convergence rate of training recurrent neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1.5, 10

[13] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning (ICML)*, 2019. 1.5, 10

[14] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning (ICML)*, 2019. 2.1.1

[15] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. On the convergence rate of training recurrent neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2.1.1

[16] Josh Alman. An illuminating algorithm for the light bulb problem. In *SOSA*. arXiv preprint arXiv:1810.06740, 2019. 2.1.3

[17] Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–150, 2015. 1.2

[18] Josh Alman and Ryan Williams. Probabilistic rank and matrix rigidity. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 641–652, 2017. 2.1.3

[19] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2021. 1.5

[20] Josh Alman, Timothy M Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 467–476. IEEE, 2016. 2.1.3, 1, 2.1.3, 2.1.3, 9, 3.8

[21] Josh Alman, Timothy Chu, Aaron Schild, and Zhao Song. Algorithms and hardness for linear algebra on geometric graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 541–552. IEEE, 2020. 2.1.3

[22] N. Alon and V. D. Milman. Eigenvalues, expanders and superconcentrators. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science, 1984*, FOCS '84, pages 320–322, Washington, DC, USA, 1984. IEEE Computer Society. ISBN 0-8186-0591-X. doi: 10.1109/SFCS.1984.715931. URL `http://dx.doi.org/10.1109/SFCS.1984.715931`. A, A.3

[23] Reid Andersen and Yuval Peres. Finding sparse cuts locally using evolving sets. In *Proceedings of the forty-first annual ACM symposium on Theory of computing (STOC)*, 2009. 3.2

[24] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE, 2006. 3.2

[25] Alexandr Andoni. *Nearest neighbor search: the old, the new, and the impossible*. PhD thesis, Massachusetts Institute of Technology, 2009. 1

[26] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 459–468, 2006. 3.15

[27] Alexandr Andoni, Piotr Indyk, Thijs Laarhovn, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. In *29th Annual Conference on Neural Information Processing Systems (NIPS)*, 2015. URL https://arxiv.org/abs/1509.02897. 7

[28] Alexandr Andoni, Robert Krauthgamer, and Ilya Razenshteyn. Sketching and embedding are equivalent for norms. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing (STOC)*, pages 479–488, 2015. 2.1.2

[29] Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Navigating nets: Simple algorithms for proximity search. In *59th Annual Symposium on Foundations of Computer Science (FOCS)*, 2018. URL https://ilyaraz.org/static/papers/daher.pdf. 7

[30] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jorg Sander. Optics: Ordering points to identify cluster structure. In *ACM SIGMOD International Conference on Management of Data*, 1999. URL http://www.dbs.ifi.lmu.de/Publikationen/Papers/OPTICS.pdf. 5.2, G

[31] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950. 2.1.2

[32] Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC), Chicago, IL, USA, June 13-16, 2004*, pages 222–231. ACM, 2004. doi: 10.1145/1007352.1007355. URL https://doi.org/10.1145/1007352.1007355. A

[33] Sunil Arya and David M. Mount. A fast and simple algorithm for computing approximate euclidean minimum spanning trees. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1220–1233, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. ISBN 978-1-611974-33-1. URL http://dl.acm.org/citation.cfm?id=2884435.2884520. 6.2

[34] Sunil Arya, Gautam Das, David M. Mount, Jeffrey S. Salowe, and Michiel Smid. Euclidean spanners: Short, thin, and lanky. In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, STOC '95, pages 489–498, New York, NY, USA, 1995. ACM. ISBN 0-89791-718-9. doi: 10.1145/225058.225191. URL http://doi.acm.org/10.1145/225058.225191. 6.2

[35] Patrice Assouad. Plongements isométriques dans l1: aspect analytique. *Number*, 14: 1979–1980, 1980. 1.2, 2, 2.1.1, 2.1.2, 2.1.2, 2.2.3

[36] Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. On the fine-grained complexity of empirical risk minimization: Kernel methods and neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4308–4318, 2017. 1.2

[37] Arturs Backurs, Moses Charikar, Piotr Indyk, and Paris Siminelakis. Efficient density evaluation for smooth kernels. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 615–626, 2018. 1.2, 1.5, 5.7

[38] Paul Balister, Bela Bollobas, Amites Sarkar, and Mark Walters. Connectivity of random k-nearest-neighbour graphs. *Advances in Applied Probability*, 37(1):1–24, 2005. ISSN 00018678. URL http://www.jstor.org/stable/30037313. 1.1

[39] Rick Beatson and Leslie Greengard. A short course on fast multipole methods. *Wavelets, multilevel methods and elliptic PDEs*, 1:1–37, 1997. 1.5, 9, 9.17

[40] Mikhail Belkin and Partha Niyogi. Semi-supervised learning on riemannian manifolds. *Mach. Learn.*, 56(1-3):209–239, June 2004. ISSN 0885-6125. doi: 10.1023/B:MACH.0000033120.25363.1e. URL https://doi.org/10.1023/B:MACH.0000033120.25363.1e. A, A.3

[41] Mikhail Belkin and Partha Niyogi. Towards a theoretical foundation for laplacian-based manifold methods. In *International Conference on Computational Learning Theory*, pages 486–500. Springer, 2005. A, A.3

[42] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, December 2006. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1248547.1248632. 1

[43] Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. In *International Conference on Machine Learning*, pages 541–549. PMLR, 2018. 2.1.1

[44] Asa Ben-Hur, Cheng Soon Ong, Sören Sonnenburg, Bernhard Schölkopf, and Gunnar Rätsch. Support vector machines and kernels for computational biology. *PLoS computational biology*, 4(10):e1000173, 2008. 1.5

[45] Christian Berg, Jens Peter Reus Christensen, and Paul Ressel. *Harmonic analysis on semigroups: theory of positive definite and related functions*, volume 100. Springer, 1984. A.2, A.3

[46] Johann Bernoulli. Brachistochrone problem. *Acta Eruditorum*, June 1696. 1

[47] Serge Bernstein. Sur les fonctions absolument monotones. *Acta Mathematica*, 52(1): 1–66, 1929. 2.1.2, 2.1.8

[48] M Francesca Betta, Friedemann Brock, Anna Mercaldo, and M Rosaria Posteraro. Weighted isoperimetric inequalities on $\mathbb{R}^n$ and applications to rearrangements. *Mathematische Nachrichten*, 281(4):466–498, 2008. D

[49] Avleen Singh Bijral, Nathan D. Ratliff, and Nathan Srebro. Semi-supervised learning with

density based distances. In Fabio Gagliardi Cozman and Avi Pfeffer, editors, *UAI*, pages 43–50. AUAI Press, 2011. 1, 1, 1, 7

[50] Salomon Bochner. Monotone funktionen, stieltjessche integrale und harmonische analyse. *Mathematische Annalen*, 108:378–410, 1933. 1

[51] Karol Borsuk. On the imbedding of systems of compacta in simplicial complexes. *Fund. Math.*, 35:217–234, 1948. 4

[52] Sabri Boughorbel, Jean-Philippe Tarel, and Francois Fleuret. Non-mercer kernels for svm object recognition. In *BMVC*, pages 1–10, 2004. 1.5

[53] Sabri Boughorbel, J-P Tarel, and Nozha Boujemaa. Conditionally positive definite kernels for svm based image recognition. In *2005 IEEE International Conference on Multimedia and Expo*, pages 113–116, 2005. 1, 1.5

[54] Sabri Boughorbel, J-P Tarel, and Nozha Boujemaa. Generalized histogram intersection kernel for image recognition. In *IEEE International Conference on Image Processing 2005*, volume 3, pages III–161. IEEE, 2005. 1.5

[55] Sabri Boughorbel, Jean-Philippe Tarel, François Fleuret, and Nozha Boujemaa. The gcs kernel for svm-based image recognition. In *International Conference on Artificial Neural Networks*, pages 595–600. Springer, 2005. 1.5

[56] Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *SODA*, 2020. 1.5

[57] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *52nd Annual ACM Symposium on Theory of Computing (STOC)*, 2020. 1.5

[58] Jan van den Brand, Binghui Peng, Zhao Song, and Omri Weinstein. Training (overparametrized) neural networks in near-linear time. In *The 12th Innovations in Theoretical Computer Science Conference (ITCS)*, 2021. 1.5, 10

[59] Jan van den Brand, Binghui Peng, Zhao Song, and Omri Weinstein. Training (overparametrized) neural networks in near-linear time. In *12th Innovations in Theoretical Computer Science Conference (ITCS)*, 2021. 2.1.1

[60] Ulrich Brehm. Extensions of distance reducing mappings to piecewise congruent mappings on ℓ_m. *Journal of Geometry*, 16(1):187–193, 1981. 3

[61] Peter Buser. A note on the isoperimetric constant. *Annales scientifiques de l'École Normale Supérieure*, Ser. 4, 15(2):213–230, 1982. doi: 10.24033/asens.1426. URL `http://www.numdam.org/item/ASENS_1982_4_15_2_213_0`. A, A.3, A.3, H.2

[62] Paul B. Callahan and S. Rao Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '93, pages 291–300, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics. ISBN 0-89871-313-7. URL `http://dl.acm.org/citation.cfm?id=313559.313777`. 1.2, 6.2

[63] Paul B Callahan and S Rao Kosaraju. Faster algorithms for some geometric graph prob-

lems in higher dimensions. In *SODA*, pages 291–300, 1993. 1.2

[64] Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, 42(1):67–90, January 1995. ISSN 0004-5411. doi: 10.1145/200836.200853. URL http://doi.acm.org/10.1145/200836.200853. 6.2, 2, H.1

[65] Paul B Callahan and S Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995. 1.2, 6.5, 6, 6.6, 19

[66] Alireza Chakeri, Hamidreza Farhidzadeh, and Lawrence O Hall. Spectral sparsification in spectral clustering. In *23rd international conference on pattern recognition (ICPR)*, pages 2301–2306. IEEE, 2016. 2

[67] Timothy M Chan and Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 1246–1255. SIAM, 2016. 2.1.3, 3.6

[68] Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. Training and testing low-degree polynomial data mappings via linear svm. *Journal of Machine Learning Research*, 11(Apr):1471–1490, 2010. 1.5

[69] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009. 1

[70] Moses Charikar and Paris Siminelakis. Hashing-based-estimators for kernel density in high dimensions. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1032–1043, 2017. 1.2, 1.5

[71] Shuchi Chawla, Anupam Gupta, and Harald Räcke. Embeddings of negative-type metrics and an improved approximation to generalized sparsest cut. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 102–111. SIAM, 2005. URL http://dl.acm.org/citation.cfm?id=1070432.1070447. A

[72] Frédéric Chazal and André Lieutier. Smooth manifold reconstruction from noisy and non-uniform approximation with guarantees. *Computational Geometry: Theory and Applications*, 40:156–170, 2008. 1

[73] Frédéric Chazal and Steve Y. Oudot. Towards persistence-based reconstruction in Euclidean spaces. In *Proceedings of the 24th ACM Symposium on Computational Geometry*, pages 232–241, 2008. 1.2, 4

[74] Frédéric Chazal, David Cohen-Steiner, and André Lieutier. A sampling theory for compact sets in Euclidean space. *Discrete & Computational Geometry*, 41:461–479, 2009. 1

[75] Frédéric Chazal Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. Persistence-based clustering in riemannian manifolds. *J. ACM*, 60(6:41):97–106, 2013. doi: 10.1145/2535927. URL http://doi.acm.org/10.1145/2535927. 1.2

[76] Bernard Chazelle, Herbert Edelsbrunner, Leonidas J. Guibas, John E. Hershberger,

Raimund Seidel, and Micha Sharir. Selecting heavily covered points. *SIAM J. Comput.*, 23(6):1138–1151, 1994. 1.1

[77] Jeff Cheeger. A lower bound for the smallest eigenvalue of the laplacian. In *Problems in Analysis: A Symposium in Honor of Salomon Bochner*, pages 195–199. Princeton Univ. Press, Princeton, N. J., 1970. A, A.3

[78] Jeff Cheeger. A lower bound for the smallest eigenvalue of the laplacian. In *In Gunning, Robert C. Problems in analysis (Papers dedicated to Salomon Bochner, 1969). Princeton, N. J.: Princeton Univ. Press.*, pages 195–199, 1970. 4

[79] Ching-Shyang Chen, Chia-Ming Fan, and PH Wen. The method of approximate particular solutions for solving certain partial differential equations. *Numerical Methods for Partial Differential Equations*, 28(2):506–522, 2012. 2.1.2

[80] Jie Chen, Hawren Fang, and Yousef Saad. Fast approximate knn graph construction for high dimensional data via recursive lanczos bisection. *Journal of Machine Learning Research*, 10:1989–2012, 2009. 1.2, 7

[81] Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. In *Computational Complexity Conference (CCC)*, 2018. 1.2, 3.18, 3.8, 3.20

[82] Sitan Chen and Ankur Moitra. Algorithmic foundations for the diffraction limit. In *STOC*. `https://arxiv.org/pdf/2004.07659.pdf`, 2021. 2.2.1

[83] W. Chen, Y. Song, H. Bai, C. Lin, and E. Y. Chang. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(3):568–586, March 2011. doi: 10.1109/TPAMI.2010.88. 2

[84] Xue Chen, Daniel M Kane, Eric Price, and Zhao Song. Fourier-sparse interpolation without a frequency gap. In *57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 741–750. IEEE, `https://arxiv.org/pdf/1609.01361.pdf`, 2016. 2.2.1

[85] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952. 3.9

[86] P Chew. There is a planar graph almost as good as the complete graph. In *Proceedings of the Second Annual Symposium on Computational Geometry*, SCG '86, pages 169–177, New York, NY, USA, 1986. ACM. ISBN 0-89791-194-6. doi: 10.1145/10515.10534. URL `http://doi.acm.org/10.1145/10515.10534`. 1.2

[87] Anna Choromanska, Tony Jebara, Hyungtae Kim, Mahesh Mohan, and Claire Monteleoni. Fast spectral clustering via the nyström method. In *International Conference on Algorithmic Learning Theory*, pages 367–381. Springer, 2013. 2

[88] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations (ICLR)*, 2020. 8

[89] Paul Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-

Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 273–282. ACM, 2011. doi: 10.1145/1993636.1993674. URL `https://doi.org/10.1145/1993636.1993674`. A

[90] Timothy Chu, Gary L. Miller, and Donald Sheehy. Exact computation of a manifold metric, via lipschitz embeddings and shortest paths on a graph. In *SODA*, 2020. 2.1.2, 1

[91] Fan Chung. *Spectral graph theory*. 92. American Mathematical Soc., 1997. 6.1

[92] F.R.K. Chung. *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*. American Mathematical Society, 1997. A, A.3

[93] Michael B Cohen, Rasmus Kyng, Gary L Miller, Jakub W Pachocki, Richard Peng, Anup B Rao, and Shen Chen Xu. Solving sdd linear systems in nearly $m \log^{1/2} n$ time. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing (STOC)*, pages 343–352, 2014. 1, 1.5, 3.9

[94] Michael B. Cohen, Brittany Terese Fasy, Gary L. Miller, Amir Nayyeri, Donald R. Sheehy, and Ameya Velingker. Approximating nearest neighbor distances. In *Proceedings of the Algorithms and Data Structures Symposium*, 2015. 1.3, 1, 1, 1, 1, 1, 1.1, 1.1

[95] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51th Annual Symposium on the Theory of Computing (STOC)*, 2019. 1.5

[96] Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM Journal on Computing*, 11(3):467–471, 1982. 6

[97] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995. 2.1.1, 1.5

[98] Richard Courant and David Hilbert. *Methods of Mathematical Physics*. Interscience Publishers, Inc., 1953. 1

[99] Frank Critchley. On certain linear mappings between inner-product and squared-distance matrices. *Linear Algebra and its Applications*, 105:91–107, 1988. A.5

[100] Ernie Croot, Vsevolod F Lev, and Péter Pál Pach. Progression-free sets in are exponentially small. *Annals of Mathematics*, pages 331–337, 2017. 2.1.10, 2.1.3

[101] George B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, 13:339–347, 1947. 1.5

[102] Eric Darve. The fast multipole method: numerical implementation. *Journal of Computational Physics 160.1*, 2000. 9

[103] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. 3.14

[104] Philippe Delsarte. *An algebraic approach to the association schemes of coding theory*. PhD thesis, Philips Research Laboratories, 1973. F.1

[105] Michel Marie Deza and Monique Laurent. *Geometry of Cuts and Metrics*. Springer

Publishing Company, Incorporated, 1st edition, 2009. 2.2.2, 2.2.3, 2.2.3, A.4, A.4

[106] Persi Diaconis and Susan Holmes. Random walks on trees and matchings. *Electron. J. Probab.*, 7:17 pp., 2002. doi: 10.1214/EJP.v7-105. 2.2.1

[107] Ilias Diakonikolas, Daniel M. Kane, and Jelani Nelson. Bounded independence fools degree-2 threshold functions. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 11–20, 2010. doi: 10.1109/FOCS.2010.8. URL `https://doi.org/10.1109/FOCS.2010.8`. A.3

[108] Pedro Domingos. Every model learned by gradient descent is approximately a kernel machine. *arXiv preprint arXiv:2012.00152*, 2020. 1.2

[109] Wei Dong, Moses Charikar, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceeding of the International Conference on World Wide Web, pages 577-586*, 2011. 1.2, 7

[110] Jack Dongarra and Francis Sullivan. Guest editors' introduction: The top 10 algorithms. *Computing in Science & Engineering*, 2(1):22, 2000. 1, 9

[111] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *ICLR*, 2019. 2.1.1, 1.5, 10

[112] Zeev Dvir and Benjamin L Edelman. Matrix rigidity and the croot-lev-pach lemma. *Theory of Computing*, 15:1–7, 2019. 2.1.3

[113] Zeev Dvir and Allen Liu. Fourier and circulant matrices are not rigid. In *Computational Complexity Conference (CCC)*, volume 137, pages 17:1–17:23, 2019. 2.1.3

[114] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 4(28):511–533, 2002. 1, 1.1, 4, 4

[115] Jordan S Ellenberg and Dion Gijswijt. On large subsets of with no three-term arithmetic progression. *Annals of Mathematics*, pages 339–343, 2017. 2.1.3

[116] Nader Engheta, William D. Murphy, Vladimir Rokhlin, and Marius Vassiliou. The fast multipole method for electromagnetic scattering computation. *IEEE Transactions on Antennas and Propagation 40*, pages 634–641, 1992. 9

[117] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996. URL `http://dl.acm.org/citation.cfm?id=3001460.3001507`. 5.2, G

[118] Pavel I Etingof, Oleg Golberg, Sebastian Hensel, Tiankai Liu, Alex Schwendner, Dmitry Vaintrob, and Elena Yudovina. *Introduction to representation theory*, volume 59. American Mathematical Soc., 2011. 1.2, 2.2.1, A.6

[119] Lawrence C. Evans and Ronald F. Gariepy. *Measure Theory and Fine Properties of Functions*. CRC Press, 2015. D.3

[120] Felix Faber, Alexander Lindmaa, O Anatole von Lilienfeld, and Rickard Armiento. Crys-

tal structure representations for machine learning models of formation energies. *International Journal of Quantum Chemistry*, 115(16):1094–1101, 2015. 2.1.1

[121] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021. 8

[122] R. P. Feynman. Space-time approach to non-relativistic quantum mechanics. *Rev. Mod. Phys*, 20(367), 1948. 1

[123] Richard Feynman, Robert Leighton, and Matthew Sands. The Feynman Lectures, Vol III Chapter 8: The Hamiltonian Matrix, 1964. URL: `http://www.feynmanlectures.caltech.edu/III_08.html`. 2.2.1

[124] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23(98):298–305, 1973. G.3

[125] Yuval Filmus, Ryan O'Donnell, and Xinyu Wu. A log-sobolev inequality for the multislice, with applications. ITCS, 2019. 2.2.1

[126] François Fleuret and Hichem Sahbi. Scale-invariance of support vector machines based on the triangular kernel. In *3rd International Workshop on Statistical and Computational Theories of Vision*, pages 1–13, 2003. 1.5

[127] Carsten Franke and Robert Schaback. Solving partial differential equations by collocation using radial basis functions. *Applied Mathematics and Computation*, 93(1):73–82, 1998. 2.1.2

[128] Kurt Otto Friedrichs. The identity of weak and strong extensions of differential operators. *Transactions of the American Mathematical Society*, 55:132–151, 1944. A.3

[129] William Fulton and Joe Harris. *Representation Theory, A first course, Readings in Mathematics*, volume 129. Springer-Verlag, New York, NY, USA, 1991. 1.2

[130] Nicolás García Trillos and Dejan Slepčev. On the rate of convergence of empirical measures in ∞-transportation distance. *Canadian Journal of Mathematics*, 67(6): 1358âĂŞ1383, 2015. doi: 10.4153/CJM-2014-044-6. A.3, A.3, H.1

[131] Ellen Gasparovic, Maria Gommel, Emilie Purvine, Bei Wang, Yusu Wang, and Lori Ziegelmeier. A complete characterization of the 1-dimensional intrinsic cech persistence diagrams for metric graphs. https://arxiv.org/abs/1702.07379, 2017. 4

[132] Yoav Goldberg and Michael Elhadad. splitsvm: fast, space-efficient, non-heuristic, polynomial kernel computation for nlp applications. *Proceedings of ACL-08: HLT, Short Papers*, pages 237–240, 2008. 1.5

[133] Jose Maria Gonzalez-Barrios and Aldofo J. Quiroz. A clustering procedure based on the comparison between the k nearest neighbors graph and the minimal spanning tree. *Statistics and Probability Letters*, 2003. URL `https://www.sciencedirect.com/science/article/pii/S0167715202004212`. 1.1

[134] J. C. Gower and G. J. S. Ross. Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 18(1):54–64, 1969. ISSN 00359254, 14679876. URL `http://www.jstor.org/stable/2346439`.

5.2, G

[135] Leo Grady and Eric L Schwartz. Isoperimetric graph partitioning for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 28(3):469–475, 2006. A.3

[136] Fan Chung Graham and Shlomo Sternberg. Laplacian and vibrational spectra for homogeneous graphs. *Journal of Graph Theory*, 16:605–627, 1992. 2.2.1

[137] Leslie Greengard. *The rapid evaluation of potential fields in particle systems*. MIT press, 1988. 9

[138] Leslie Greengard. The numerical solution of the n-body problem. *Computers in physics*, 4(2):142–152, 1990. 9

[139] Leslie Greengard. Fast algorithms for classical physics. *Science*, 265(5174):909–914, 1994. 9

[140] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987. 1, 1.1, 2.1, 9

[141] Leslie Greengard and Vladimir Rokhlin. The rapid evaluation of potential fields in three dimensions. *Vortex Methods. Springer, Berlin, Heidelberg*, pages 121–141, 1988. 1, 2.1, 9

[142] Leslie Greengard and Vladimir Rokhlin. On the evaluation of electrostatic interactions in molecular modeling. *Chemica scripta*, 29:139–144, 1989. 1, 1.1, 2.1, 9

[143] Leslie Greengard and Vladimir Rokhlin. An improved fast multipole algorithm in thre dimensions. ., 1996. 9

[144] Leslie Greengard and John Strain. The fast gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94, 1991. 1.2, 2.1, 2.1, 9, 9.2

[145] Karsten Grove. Critical point theory for distance functions. *Proceedings of the Symposia in Pure Mathematics*, 54(3):357–385, 1993. 1

[146] Stephen Guattery and Gary Miller. On the quality of spectral separators. *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1998. 2.2.1

[147] Stephen Guattery and Gary L. Miller. On the performance of the spectral graph partitioning methods. In *SODA'95*, pages 233–242. ACM-SIAM, 1995. G.3

[148] Steve R Gunn. Support vector machines for classification and regression. *ISIS technical report*, 14(1):5–16, 1998. 1.5

[149] Bart Hamers. Kernel models for large scale applications. ., 2004. 1.5

[150] Sariel Har-Peled, Piotr Indyk, and Anastasios Sidiropoulos. Euclidean spanners in high dimensions. In *SODA*, 2013. 1.2, 7

[151] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Nearly optimal sparse Fourier transform. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 563–578. ACM, `https://arxiv.org/pdf/1201.2501.pdf`, 2012. 2.2.1

[152] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical al-

gorithm for sparse Fourier transform. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, pages 1183–1194. SIAM, https://groups.csail.mit.edu/netmit/sFFT/soda_paper.pdf, 2012. 2.2.1

[153] A Hedayat and Walter Dennis Wallis. Hadamard matrices and their applications. *The Annals of Statistics*, 6(6):1184–1238, 1978. A.2

[154] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 21–30, 2015. 1.2

[155] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1.5

[156] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. 3.10

[157] Heiko Hoffmann. Kernel pca for novelty detection. *Pattern recognition*, 40(3):863–874, 2007. 1.5

[158] Sung Jin Hwang, Steven B. Damelin, and Alfred O. Hero III. Shortest path through random points. *Ann. Appl. Probab.*, 26(5):2791–2823, 10 2016. doi: 10.1214/15-AAP1162. URL http://dx.doi.org/10.1214/15-AAP1162. 1, 1, 1.1, 5.2, G, I

[159] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. 3.6

[160] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006. 7.14

[161] Piotr Indyk and Michael Kapralov. Sample-optimal Fourier sampling in any constant dimension. In *IEEE 55th Annual Symposium onFoundations of Computer Science (FOCS)*, pages 514–523. IEEE, https://arxiv.org/pdf/1403.5804.pdf, 2014. 2.2.1

[162] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998. 1.2

[163] Piotr Indyk, Michael Kapralov, and Eric Price. (Nearly) Sample-optimal sparse Fourier transform. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 480–499. SIAM, 2014. 2.2.1

[164] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems (NeurIPS)*, pages 8571–8580, 2018. 2.1.1, 1.5, 10.1

[165] Kevin Jarrett, Koray Kavukcuoglu, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pages 2146–2153, 2009. 1.5

[166] Shunhua Jiang, Yunze Man, Zhao Song, and Danyang Zhuo. Graph neural network acceleration via matrix dimension reduction. In *Openreview*. https://openreview.net/forum?id=8IbZUle6ieH, 2020. 1.5

[167] Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster lps. *arXiv preprint arXiv:2004.07470*, 2020. 1.5

[168] Yaonan Jin, Daogao Liu, and Zhao Song. A robust multi-dimensional sparse fourier transform in the continuous setting. *arXiv preprint arXiv:2005.06156*, 2020. 2.2.1

[169] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984. 1.2, 3.13, 3.14

[170] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. In *Journal of the ACM (JACM)*, pages 497–515, 2004. 3.2

[171] Michael Kapralov. Sparse Fourier transform in any constant dimension with nearly-optimal sample complexity in sublinear time. In *Symposium on Theory of Computing Conference (STOC)*. https://arxiv.org/pdf/1604.00845.pdf, 2016. 2.2.1

[172] Michael Kapralov. Sample efficient estimation and recovery in sparse FFT via isolation on average. In *58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. https://arxiv.org/pdf/1708.04544, 2017. 2.2.1

[173] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing (STOC)*, pages 302–311, 1984. 1.5

[174] Matti Karppa, Petteri Kaski, and Jukka Kohonen. A faster subquadratic algorithm for finding outlier correlations. *ACM Transactions on Algorithms (TALG)*, 14(3):1–26, 2018. 2.1.3

[175] Jonathan A Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving sdd systems in nearly-linear time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing (STOC)*, pages 911–920, 2013. 1.5, 3.9

[176] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980. 1.5

[177] George Kimeldorf and Grace Wahba. Some results on tchebycheffian spline functions. *Journal of mathematical analysis and applications*, 33(1):82–95, 1971. 2.1.2

[178] M. Kirszbraun. Über die zusammenziehende und lipschitzsche transformationen. *Fundamenta Mathematicae*, 22(1):77–108, 1934. URL http://eudml.org/doc/212681. 3

[179] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations (ICLR)*, 2019. 8

[180] Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving SDD linear systems. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 235–244. IEEE Computer Society, 2010. doi: 10.1109/FOCS.2010.29. URL https://doi.org/10.1109/FOCS.2010.29. A

[181] Ioannis Koutis, Gary L Miller, and Richard Peng. Approaching optimality for solving sdd linear systems. *FOCS*, 2010. 1.2, 1.5, 3.9

[182] Ioannis Koutis, Gary L Miller, and Richard Peng. A nearly-m log n time solver for sdd linear systems. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 590–598, 2011. 1.5, 3.6, 3.9, 4.2

[183] Robert Krauthgamer and Tal Wagner. Cheeger-type approximation for sparsest *st*-cut. *ACM Trans. Algorithms*, 13(1):14:1–14:21, 2016. doi: 10.1145/2996799. URL `https://doi.org/10.1145/2996799`. A.3, H

[184] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, pages 1097–1105, 2012. 1.5

[185] Ria Kulshrestha. Transformers. https://towardsdatascience.com/transformers-89034557de14, 2020. 1.2, 2.1.3, 8

[186] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nyström method. *Journal of Machine Learning Research*, 13(Apr):981–1006, 2012. 2

[187] Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians-fast, sparse, and simple. In *57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 573–582, 2016. 1.5

[188] Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A. Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In *STOC*, 2016. 1.5

[189] Thijs Laarhoven. Graph-based time-space trade-offs for approximate near neighbors. In *Symposium on Computational Geometry*, 2018. 7

[190] Peter Lax. *Functional Analysis*. Wiley, 1st edition, 2002. ISBN 0471556041. A.1

[191] Hung Le and Shay Solomon. Truly optimal euclidean spanners. *FOCS*, 2019. URL `http://arxiv.org/abs/1904.12042`. 1.1, 6.2

[192] Michel Ledoux. Spectral gap, logarithmic Sobolev constant, and geometric bounds. *Surveys in differential geometry*, 9(1):219–240, 2004. A, A.3, H, H.2

[193] James R Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):37, 2014. A.3, H

[194] James R. Lee, Shayan Oveis Gharan, and Luca Trevisan. Multi-way spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM*, 61(6)(37), 2014. H

[195] Jason D Lee, Ruoqi Shen, Zhao Song, Mengdi Wang, and Zheng Yu. Generalized leverage score sampling for neural networks. In *NeurIPS*, 2020. 2.1.1, 1.5, 10

[196] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *FOCS*, 2013. 1.5

[197] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $O(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 424–433. IEEE, 2014. 1.5

[198] Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear

time. In *FOCS*, 2015. 1.5

[199] Yin Tat Lee and Santosh S. Vempala. Stochastic localization + stieltjes barrier = tight bound for log-Sobolev. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 1122–1129, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5559-9. doi: 10.1145/3188745.3188866. URL http://doi.acm.org/10.1145/3188745.3188866. A.3

[200] Yin Tat Lee and Santosh S. Vempala. The Kannan-LovÃąsz-Simonovits conjecture, 2018. A, A.3

[201] Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empricial risk minimization in the current matrix multiplication time. In *COLT*, 2019. 1.5

[202] Xiang-Yang Li, Peng-Jun Wan, and Yu Wang. Power efficient and sparse spanner for wireless ad hoc networks. In *Proceedings Tenth International Conference on Computer Communications and Networks*, 2001. 1.1

[203] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2.1.1, 1.5, 10

[204] O Anatole Von Lilienfeld. Quantum machine learning in chemical compound space. *Angewandte Chemie International Edition*, 57(16):4164–4169, 2018. 2.1.1

[205] O Anatole Von Lilienfeld, Raghunathan Ramakrishnan, Matthias Rupp, and Aaron Knoll. Fourier series of atomic radial distribution functions: A molecular fingerprint for machine learning models of quantum chemical properties. *International Journal of Quantum Chemistry*, 115(16):1084–1093, 2015. 2.1.1

[206] Chun Liu and Noel J Walkingron. An Eulerian description of fluids containing viscoelastic particles. *Archive for rational mechanics and analysis*, 159(3):229–252, 2001. A.3

[207] J. Liu, C. Wang, M. Danilevsky, and J. Han. Large-scale spectral clustering on graphs. In *In Twenty-Third International Joint Conference on Artificial Intelligence*, 2013. 1

[208] Ting Liu, Andrew W. Moore, Alexander Gray, and Ke Yang. An investigation of practical approximate nearest neighbor algorithms. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, pages 825–832, Cambridge, MA, USA, 2004. MIT Press. URL http://dl.acm.org/citation.cfm?id=2976040.2976144. 7

[209] Xuanqing Liu, Si Si, Xiaojin Zhu, Yang Li, and Cho-Jui Hsieh. A unified framework for data poisoning attack to graph-based semi-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1, 3

[210] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. 1

[211] Anand Louis, Prasad Raghavendra, Prasad Tetali, and Santosh Vempala. Algorithmic extensions of cheeger's inequality to higher eigenvalues and partitions. In *In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 315–326, 2011. 3.2

[212] Anand Louis, Prasad Raghavendra, Prasad Tetali, and Santosh Vempala. Many sparse cuts via higher eigenvalues. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, FOCS 2012, pages 1131–1140. ACM, 2012. A.3, H

[213] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, pages 261–277, 1988. 2.2.1

[214] Ulrike Von Luxburg. Tutorial on spectral clustering. *Statistics and Computing*, 17(4), 2007. 1, 1.2

[215] Shaogao Lv, Huazhen Lin, Heng Lian, and Jian Huang. Oracle inequalities for sparse additive quantile regression in reproducing kernel hilbert space. *The Annals of Statistics*, 46(2):781–813, 2018. 2.1.2

[216] Adam Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families I: bipartite ramanujan graphs of all degrees. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 529–537. IEEE Computer Society, 2013. doi: 10.1109/FOCS.2013.63. URL `https://doi.org/10.1109/FOCS.2013.63`. A

[217] G. Margulis. Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators. *Probl. of Inform. Transm.*, pages 51–60, 1988. 2.2.1

[218] Per-Gunnar Martinsson. Encyclopedia entry on fast multipole methods. In *University of Colorado at Boulder*, 2012. 1.5, 9, 9.17

[219] Peter Massopust. *Interpolation and approximation with splines and fractals*. Oxford University Press, Inc., 2010. 1.5

[220] Jiří Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8(3):315–334, 1992. 3.19

[221] Charles A Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. In *Approximation theory and spline functions*, pages 143–145. Springer, 1984. 1.5

[222] Sebastian Mika, Bernhard Schölkopf, Alex J Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel pca and de-noising in feature spaces. In *Advances in neural information processing systems (NIPS)*, pages 536–542, 1999. 1.5

[223] Gary L. Miller, Noel J. Walkington, and Alex L. Wang. Hardy-muckenhoupt bounds for laplacian eigenvalues. *CoRR*, abs/1812.02841, 2018. URL `http://arxiv.org/abs/1812.02841`. G.2

[224] Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. Subquadratic algorithms for succinct stable matching. In *International Computer Science Symposium in Russia*, pages 294–308. Springer, 2016. 2.1.3

[225] Ankur Moitra. The threshold for super-resolution via extremal functions. In *STOC*. `https://arxiv.org/pdf/1408.1681.pdf`, 2015. 2.2.1

[226] Peter Monk. *Finite Element Methods for Maxwell's Equations (Numerical Analysis and Scientific Computation Series)*. Oxford Press, 01 2003. ISBN 0198508883. A.3

[227] Vasileios Nakos, Zhao Song, and Zhengyu Wang. (Nearly) Sample-optimal sparse Fourier transform in any dimension; RIPless and Filterless. In *FOCS*. https://arxiv.org/pdf/1909.11123.pdf, 2019. 2.2.1

[228] J. Von Neumann and I. J. Schoenberg. Fourier integrals and metric geometry. *Transactions of the American Mathematical Society*, 50(2):226–251, Sep 1941. 3

[229] J. Von Neumann and I. J. Schoenberg. Fourier integrals and metric geometry. *Transactions of the American Mathematical Society*, 50(2):226–251, 1941. 2.1.2, 2.1.2, 1

[230] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pages 849–856, Cambridge, MA, USA, 2001. MIT Press. URL http://dl.acm.org/citation.cfm?id=2980539.2980649. 1.4, A, A.3, B

[231] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems (NeurIPS)*, pages 849–856, 2002. 1, 2, 1.5

[232] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, New York, NY, USA, 2014. ISBN 1107038324, 9781107038325. 2.2.1, F.1

[233] Ryan O'Donnell and John Wright. Efficient quantum tomography. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 899–912, 2016. 2.2.1

[234] Lorenzo Orecchia and Nisheeth K. Vishnoi. Towards an sdp-based approach to spectral methods: A nearly-linear-time algorithm for graph partitioning and decomposition. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 532–545, Philadelphia, PA, USA, 2011. Society for Industrial and Applied Mathematics. URL http://dl.acm.org/citation.cfm?id=2133036.2133078. A.3

[235] Lorenzo Orecchia and Zeyuan Allen Zhu. Flow-based algorithms for local graph clustering. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1267–1286, 2014. 3.2

[236] Lorenzo Orecchia, Leonard J Schulman, Umesh V Vazirani, and Nisheeth K Vishnoi. On partitioning graphs via single commodity flows. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, STOC '08, pages 461–470. ACM, 2008. A.3

[237] Mark JL Orr. Introduction to radial basis function networks, 1996. 2.1.2

[238] Enea Parini. An introduction to the cheeger problem. *Surv. Math. Appl.*, 6:9–21, 2011. D

[239] Pablo A. Parrilo. Algebraic techniques and semidefinite optimization, February 2012. 10

[240] Eric Price and Zhao Song. A robust sparse Fourier transform in the continuous setting. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 583–600. IEEE, https://arxiv.org/pdf/1609.00896.pdf, 2015. 2.2.1

[241] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems (NIPS)*, pages 1177–1184, 2008. 1.5

[242] Ilya Razenshteyn. *High-dimensional similarity search and sketching: algorithms and hardness*. PhD thesis, Massachusetts Institute of Technology, 2017. 1

[243] Lorenzo Rosasco, Mikhail Belkin, and Ernesto De Vito. On learning with integral operators. *Journal of Machine Learning Research*, 11(Feb):905–934, 2010. A.3

[244] Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1260–1268, 2018. 1.2, 1.2, 3.17

[245] Karthik C. S. and Pasin Manurangsi. On closest pair in euclidean metric: Monochromatic is as hard as bichromatic. In *ITCS*, 2019. 1.2, 3.16

[246] Sushant Sachdeva and Nisheeth K Vishnoi. Faster algorithms via approximation theory. *Foundations and Trends in Theoretical Computer Science*, 9(2):125–210, 2014. 5.2, 5.6

[247] Sajama and Alon Orlitsky. Estimating and computing density based distance metrics. In *ICML '05*, pages 760–767, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102447. URL `http://doi.acm.org/10.1145/1102351.1102447`. 1, 1, 1

[248] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017. 1.5

[249] Laurent Saloff-Coste. Random walks on finite groups. In *Probability on discrete structures*, pages 263–346. Springer, 2004. 2.2.1

[250] Thatchaphol Saranurak and Di Wang. Expander decomposition and pruning: Faster, stronger, and simpler. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2616–2635. SIAM, 2019. doi: 10.1137/1.9781611975482.162. URL `https://doi.org/10.1137/1.9781611975482.162`. A

[251] I Schoenberg. Positive definite functions on spheres. *Duke Math. J*, 1:172, 1942. 1.2, 2.1.1, 2.1.3, 2.1.2, 11

[252] I. J. Schoenberg. Remarks to maurice frechet's article" sur la definition axiomatique d'une classe d'espace distancies vector! ellement applicable sur l'espace de hilbertl. *Ann. of Math*, 36:724–732, 1935. 2.2.3, 2.2.4, 2.2.3, A.5, A.7

[253] I. J. Schoenberg. On certain metric spaces arising from euclidean spaces by a change of metric and their imbedding in hilbert space. *Annals of Mathematics*, 38(4):787–793, 1937. ISSN 0003486X. URL `http://www.jstor.org/stable/1968835`. 1

[254] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001. 1, 2.1.1, 2.1.2

[255] Bernhard Scholkopf, Kah-Kay Sung, Christopher JC Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and Vladimir Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE transactions on Signal Processing*, 45(11):2758–2765, 1997. 2.1.2

[256] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component

analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998. 1.5

[257] Karl Schwarzschild. Uber das gravitationsfeld eines massenpunktes nach der einsteinschen theorie. *Sitzungsberichte der Deutschen Akademie der Wissenschaften zu Berlin*, 1916. 1

[258] Jonah Sherman. Generalized preconditioning and network flow problems. In *SODA*, pages 772–780, 2017. 3.14

[259] J.B. Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 22:888–905, 01 1997. A

[260] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014. 1.5

[261] Alex J Smola. *Regression estimation with support vector learning machines*. PhD thesis, MasterâĂŹs thesis, Technische Universität München, 1996. 1.2, 1, 2.1.1

[262] Alex J Smola, Zoltan L Ovari, and Robert C Williamson. Regularization with dot-product kernels. In *Advances in neural information processing systems (NeurIPS)*, pages 308–314, 2001. 1, 2.1.1, 2.1.3, E.2, 11

[263] S Soboleff. Sur un théorème d'analyse fonctionnelle. *Matematicheskii Sbornik*, 46(3): 471–497, 1938. 2.1.1, 2.1.2, 2.1.2, 2.2.3, 2.2.3, A.3

[264] Zhao Song. *Matrix Theory : Optimization, Concentration and Algorithms*. PhD thesis, The University of Texas at Austin, 2019. 1.5

[265] Zhao Song and Xin Yang. Quadratic suffices for over-parametrization via matrix chernoff bound. *arXiv preprint arXiv:1906.03593*, 2019. 2.1.1, 1.5, 10

[266] Zhao Song and Zheng Yu. Oblivious sketching-based central path method for solving linear programming problems. In *Openreview*. https://openreview.net/forum?id=fGiKxvFeub, 2020. 1.5

[267] César R Souza. Kernel functions for machine learning applications. *Creative Commons Attribution-Noncommercial-Share Alike*, 3:29, 2010. 1, 1.5

[268] Daniel Spielman. Spectral graph theory and its applications, 2018. URL: `http://www.cs.yale.edu/homes/spielman/eigs.pdf`. Last visited on 2019/10/15. 2.2.1

[269] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011. 1, 1.2, 3.7, 3.7.1, 6, 7.2, 7.2, 4, 8

[270] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 81–90, New York, NY, USA, 2004. ACM. ISBN 1-58113-852-0. doi: 10.1145/1007352.1007372. URL `http://doi.acm.org/10.1145/1007352.1007372`. A.3

[271] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, 2004. 1.5, 3.2, 3.5, 3.9, 4.2, 4.4

[272] Daniel A. Spielman and Shang-Hua Teng. Spectral partitioning works: Planar graphs and finite element meshes. *Linear Algebra and its Applications*, 421(2):284 – 305, 2007. ISSN 0024-3795. doi: https://doi.org/10.1016/j.laa.2006.07.020. URL http://www.sciencedirect.com/science/article/pii/S0024379506003454. Special Issue in honor of Miroslav Fiedler. H.1

[273] Prashant Sridhar. An experimental study into spectral and geometric approaches to data clustering. Master's thesis, Carnegie Mellon University, October 2015. CMU CS Tech Report CMU-CS-15-149. 1.2, I

[274] Elias M Stein and Rami Shakarchi. *Real analysis: measure theory, integration, and Hilbert spaces*. Princeton University Press, 2009. A.3

[275] Werner Stuetzle. Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *Journal of Classification*, 20(1):025–047, May 2003. ISSN 1432-1343. doi: 10.1007/s00357-003-0004-6. URL https://doi.org/10.1007/s00357-003-0004-6. G

[276] Werner Stuetzle. A generalized single linkage method for estimating the cluster tree of a density. 2007. G

[277] H. J. Sussmann and J. C. Willems. 300 years of optimal control: from the brachystochrone to the maximum principle. *IEEE Control Systems Magazine*, 17(3):32–44, June 1997. ISSN 1066-033X. doi: 10.1109/37.588098. 1

[278] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. 1, 1.2

[279] Shang-Hua Teng. The laplacian paradigm: Emerging algorithms for massive graphs. In *International Conference on Theory and Applications of Models of Computation*, pages 2–14. Springer, 2010. 2.2.1

[280] NicolÃąs GarcÃ■a Trillos and Dejan SlepÄ■ev. A variational approach to the consistency of spectral clustering, 2015. A, A.3, B, H.1

[281] Michael Unser. Splines: A perfect fit for signal and image processing. *IEEE Signal processing magazine*, 16(6):22–38, 1999. 1.5

[282] Pravin M Vaidya. An algorithm for linear programming which requires $O(((m+n)n^2 + (m+n)^{1.5}n)L)$ arithmetic operations. In *FOCS*, 1987. 1.5

[283] Pravin M Vaidya. Speeding-up linear programming using fast matrix multiplication. In *30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 332–337. IEEE, 1989. 1.5

[284] Pravin M. Vaidya. A sparse graph almost as good as the complete graph on points in k dimensions. *Discrete & Computational Geometry*, 6(3):369–381, Sep 1991. ISSN 1432-0444. doi: 10.1007/BF02574695. URL https://doi.org/10.1007/BF02574695. 1.2

[285] F. A. Valentine. A Lipschitz condition preserving extension for a vector function. *American Journal of Mathematics*, 67(1):83–93, 1945. ISSN 00029327, 10806377. URL http://www.jstor.org/stable/2371917. 3

[286] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 11–20, 2012. 2.1.3

[287] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems (NeurIPS)*, pages 5998–6008, 2017. 2.1.3

[288] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *IEEE transactions on pattern analysis and machine intelligence*, 34(3):480–492, 2012. 1.5

[289] Pascal Vincent and Yoshua Bengio. Density sensitive metrics and kernels. In *Snowbird Workshop*, 2003. 1.3, 1, 1

[290] Ulrike von Luxburg. A tutorial on spectral clustering, 2007. 2

[291] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007. A

[292] Ulrike Von Luxburg, Mikhail Belkin, and Olivier Bousquet. Consistency of spectral clustering. *The Annals of Statistics*, pages 555–586, 2008. A.3

[293] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 8

[294] Jonathan Whiteley. Linear elliptic partial differential equations. In *Finite Element Methods*, pages 119–141. Springer, 2017. A.3

[295] R Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM Journal on Computing*, 47(5):1965–1985, 2018. 2.1.3, 6

[296] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005. 3.12, 3.6

[297] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians (ICM)*, 2018. 3.6

[298] D. Wishart. Mode analysis: A generalization of the nearest neighor which reduces chaining effects. 1969. 5.2, G

[299] Max A Woodbury. The stability of out-input matrices. *Chicago, IL*, 9, 1949. 3.8

[300] Max A Woodbury. Inverting modified matrices. *Memorandum report*, 42(106):336, 1950. 3.8

[301] Christian Wulff-Nilsen. Fully-dynamic minimum spanning forest with improved worst-case update time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, STOC 2017, page 1130âĂŞ1143, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450345286. doi: 10.1145/3055399.3055415. URL `https://doi.org/10.1145/3055399.3055415`. A

[302] Changjiang Yang, Ramani Duraiswami, Nail A. Gumerov, and Larry Davis. Improved fast gauss transform and efficient kernel density estimation. In *Proceedings Ninth IEEE*

International Conference on Computer Vision (ICCV), 2003. 9

[303] Changjiang Yang, Ramani Duraiswami, and Larry Davis. Efficient kernel machines using the improved fast gauss transform. In *NIPS*, 2004. 9

[304] Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982. 3.19

[305] Grigory Yaroslavtsev and Adithya Vadapalli. Massively parallel algorithms and hardness for single linkage clustering under l_p distances. Arxiv. ACM, 2017. URL `https://arxiv.org/pdf/1710.01431`. G

[306] Kai Zhong, Zhao Song, and Inderjit S Dhillon. Learning non-overlapping convolutional neural networks with multiple kernels. *arXiv preprint arXiv:1711.03440*, 2017. 1.5

[307] Kai Zhong, Zhao Song, Prateek Jain, Peter L Bartlett, and Inderjit S Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 4140–4149, 2017. 1.5

[308] Kai Zhong, Zhao Song, Prateek Jain, and Inderjit S Dhillon. Provable non-linear inductive matrix completion. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 11439–11449, 2019. 1.5

[309] Xiaojin Zhu. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, language technologies institute, school of Computer Science, 2005. 1, 3, 1

[310] Xiaojin Zhu and John Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Proceedings of the 22nd international conference on Machine learning (ICML)*, pages 1052–1059, 2005. 1

[311] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005. 1, 3

[312] Zeyuan Allen Zhu, Silvio Lattanzi, and Vahab Mirrokni. A local algorithm for finding well-connected clusters. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 396–404, 2013. 3.2