Algorithms and Hardness for Linear Algebra on Geometric Graphs

Josh Alman
jalman@seas.harvard.edu
Harvard U.

Timothy Chu
tzchu@andrew.cmu.edu
Carnegie Mellon U.

Aaron Schild
aschild@uw.edu
U. of Washington

Zhao Song
zhaos@utexas.edu
Columbia, Princeton, IAS

***Abstract*—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 paper, we initiate the study of 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.

I. 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 [GR87], [GR88], [GR89], voted one of the top ten algorithms of the twentieth century by the editors of *Computing in Science and Engineering* [DS00]. The second task is *spectral clustering* [NJW02], [LWDH13], 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 [Zhu05b], [CSZ09], [ZL05], which has seen increasing use in machine learning. One notable method for performing semi-supervised learning is the graph-based Laplacian regularizer method [LSZ+19b], [ZL05], [BNS06], [Zhu05a].

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 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 $f(z) = e^{-z}$ [vL07], [NJW02]. 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 [CFH16], [CSB+11], [KMT12]. Standard sparsification methods in the literature are heuristical, and include the widely used Nystrom method which uniformly samples rows and columns from the original

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00057

matrix [CJK+13].
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 [CFH16]. 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 [CFH16], [CJK+13], [KMT12].

3) **Semi-supervised learning**: An important subroutine in semi-supervised learning is completion based on ℓ_2-minimization [Zhu05a], [Zhu05b], [LSZ+19b]. 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 [SS11], [CKM+14] to multiply, sparsify, and solve systems. However, this requires a minimum of $\Omega(n^2)$ time, as G is a dense graph.

In this paper, 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 [Boc33], [Sch37], [NS41]. 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 [Sou10], [Zhu05a], [BTB05a]. See Section I-E 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 $\mathsf{K}(u,v) = \|u-v\|_2^q$ [Llo82], [AS14], [ACX19], [CMS20].

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 the full version 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 the full version. 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 I-E 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 I-D 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.

A. High-dimensional results

We begin in this subsection by stating our results about which functions have $\mathrm{poly}(d, \log(\alpha), \log(1/\varepsilon))n^{1+o(1)}$-time algorithms for multiplication and Laplacian solving and $\mathrm{poly}(d, \log(\alpha), 1/\varepsilon)n^{1+o(1)}$-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 the full version.

1) 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 I.1. *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.

Strong Exponential Time Hypothesis (SETH) is a common assumption in fine-grained complexity regarding the difficulty of solving the Boolean satisfiability problem; see the full version for more details. Theorem I.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 I.2. *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 [GR87], [GR89] solves one step of this n-body problem in time $(\log(n/\varepsilon))^{O(d)}n^{1+o(1)}$. Our Corollary I.2 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.

2) 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 I.3. *For any function f such that f is $(2, L)$-multiplicatively Lipschitz, building a $(1 \pm \varepsilon)$-spectral spar-*

sifier 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 $\rho^{-L} f(x) \le f(\rho x) \le \rho^L f(x)$ 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 I.3 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 I.4. *Consider an* $L > 1$*. There is some sufficiently large value* $C_L > 1$ *depending on* L *such that for any decreasing function* $f : \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$ *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 I.4 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 I.4 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 I.5. *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.*

3) 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 I.6. *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 I.7. *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 the full version, 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 I.7 then follows using this connection: it describes functions which we have shown have efficient sparsifiers but not efficient multipliers.

Corollary I.7, 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 I.8. *Consider an* $L > 1$*. There is some sufficiently large value* $C_L > 1$ *depending on* L *such that for any decreasing function* $f : \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$ *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 I.7, 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

[1] f is a nonnegative function if $f(x) \ge 0$ for all $x \ge 0$.

particular, doing semi-supervised learning, cannot be done in almost-linear time assuming SETH.

B. Our Techniques

1) 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 the full version for 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 the full version, following the seminal work of Greengard and Strain [GS91], which may be of independent interest.

As discussed in section I-A 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 I.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 the full version for more details.

Lower Bound Techniques: We now sketch the proof of Theorem I.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 [AW15], [Rub18], [Che18], [SM19] 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

$$\sum_{\ell=1}^{d} c_\ell \cdot f(a \cdot \ell + b). \tag{1}$$

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 (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 the full version for the details.

Comparison with Previous Lower Bound Techniques: Prior work (e.g. [CS17], [BIS17], [BCIS18]) has shown SETH-based fine-grained complexity results for matrix-related computations. For instance, [BIS17] showed hardness results for exact algorithms for many machine-learning related tasks, like kernel PCA and gradient computation in training neural networks, while [CS17] and [BCIS18] 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, [BIS17] shows hardness for kernel PCA only for Gaussian kernels. These limitations arise from the technique used. To show hardness, [BIS17] 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 [CS17] and [BCIS18] 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 the full version.

Our lower bound is also interesting when compared with the Online Matrix-Vector Multiplication (OMV) Conjecture of Henzinger et al. [HKNS15]. 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.

2) 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 [JL84], [IM98], the notion of well-separated pair decomposition from Callahan and Kosaraju [CK93], [CK95], and spectral sparsification via oversampling [SS11], [KMP10]. Combining these techniques carefully gives us the bounds in Theorem I.3.

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 I.3. Details can be found in the full version. 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 the full version 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 the full version)and that all dense graphs have an expander subgraph consisting of a large fraction of the vertices (see the full version). 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 $O(\text{poly}(d \log n)/n)$. This can be done via uniform sampling and Johnson-Lindenstrauss [SS11]. 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 the full version).

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 [Rub18], 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 the full version.

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.

C. Brief summary of our results in terms of p_f

Before proceeding to the body of the paper, we summarize our results. Recall that we consider three linear-algebraic problems in this paper, 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.

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 I.1 part 2), while $p_f < o(\log n)$ implies an almost-linear time algorithm (Theorem I.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 (see the full version), but functions exist that are hard in low dimensions (see the full version).
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 I.4). If $p_f < o(\log n)$, then an almost-linear time algorithm for sparsification exists (Theorem I.3).
   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 (see the full version). If $p_f < n^{o(1/d)}$, then there is a subquadratic time algorithm (see the full version).
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 I.7 combined with Theorem I.8) and an algorithm existing when $p_f < o(\log n)$ (Theorem I.6).
   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 *Sparsification* setting, then no subquadratic time algorithm exists (see the full version).

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.

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 I: Functions among f_1, f_2, f_3, f_4 that have almost-linear time algorithms

D. 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 I, 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.

E. 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 [Dan47], [Kha80], [Kar84], [Vai87], [Vai89], [LS14], [LS15], [CLS19], [LSZ19a], [Son19], [Bra20], [BLSS20], [SY20], [JSWZ20]. For the dense input matrix, the state-of-the-art algorithm [JSWZ20] takes $n^{\max\{\omega, 2+1/18\}} \log(1/\varepsilon)$ time, ω is the exponent of matrix multiplication. 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 [ST04], [KMP10], [KMP11], [KOSZ13], [LS13], [CKM+14], [KLP+16], [KS16]. 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. [CS17], [BCIS18] 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 [Sou10]. 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 paper: the Gaussian kernel [NJW02], [RR08], exponential kernel, Laplace kernel [RR08], rational quadratic kernel, multiquadric kernel [BG97], inverse multiquadric kernel [Mic84], [Mar12], circular kernel [BTFB05], spherical kernel, power kernel [FS03], log kernel [BG97], [Mar12], Cauchy kernel [RR08], and generalized T-Student kernel [BTF04].

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 the full version): the linear kernel [SSM98], [MSS+99], [Hof07], [Shl14] and the polynomial kernel [CV95], [GE08], [BHOS+08], [CHC+10].

Finally, the following relatively popular kernels are not of the form we directly study in this paper, and we leave extending our results to them as an important open problem: the Hyperbolic tangent (Sigmoid) kernel [HS97], [BTB05b], [JKL09], [KSH12], [ZSJ+17], [ZSD17], [SSB+17], spline kernel [Gun98], [Uns99], B-spline kernel [Ham04], [Mas10], Chi-Square kernel [VZ12], and the histogram intersection kernel and generalized histogram intersection [BTB05c]. More interestingly, our result also can be applied to Neural Tangent Kernel [JGH18], which plays a crucial role in the recent work about convergence of neural network training [LL18], [DZPS19], [AZLS19b], [AZLS19a], [SY19], [BPSW20], [LSS+20]. For more details, we refer the readers to the full version.

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.

Josh Alman is supported by a Michael O. Rabin Post-

doctoral Fellowship. Aaron Schild is supported by ONR #N00014-17-1-2429 and Packard fellowship. Zhao Song is partially supported by Ma Huateng Foundation, Schmidt Foundation, Simons Foundation, NSF, DARPA/SRC, Google and Amazon.

REFERENCES

[ACX19] Pankaj K. Agarwal, Hsien-Chih Chang, and Allen Xiao. Efficient algorithms for geometric partial matching. In *35th International Symposium on Computational Geometry, SoCG 2019, June 18-21, 2019, Portland, Oregon, USA.*, pages 6:1–6:14, 2019.

[AS14] 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*, pages 555–564. ACM, 2014.

[AW15] 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. IEEE, 2015.

[AZLS19a] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *ICML*, 2019.

[AZLS19b] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. On the convergence rate of training recurrent neural networks. In *NeurIPS*, 2019.

[BCIS18] 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. IEEE, 2018.

[BG97] Rick Beatson and Leslie Greengard. A short course on fast multipole methods. *Wavelets, multilevel methods and elliptic PDEs*, 1:1–37, 1997.

[BHOS+08] 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.

[BIS17] 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.

[BLSS20] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *STOC*, 2020.

[BNS06] 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.

[Boc33] Salomon Bochner. Monotone funktionen, stieltjessche integrale und harmonische analyse. *Mathematische Annalen*, 108:378–410, 1933.

[BPSW20] Jan van den Brand, Binghui Peng, Zhao Song, and Omri Weinstein. Training (overparametrized) neural networks in near-linear time. *arXiv preprint arXiv:2006.11648*, 2020.

[Bra20] Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *SODA*, 2020.

[BTB05a] 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. IEEE, 2005.

[BTB05b] 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. IEEE, 2005.

[BTB05c] 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.

[BTF04] Sabri Boughorbel, Jean-Philippe Tarel, and Francois Fleuret. Non-mercer kernels for svm object recognition. In *BMVC*, pages 1–10, 2004.

[BTFB05] 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.

[CFH16] Alireza Chakeri, Hamidreza Farhidzadeh, and Lawrence O Hall. Spectral sparsification in spectral clustering. In *2016 23rd international conference on pattern recognition (icpr)*, pages 2301–2306. IEEE, 2016.

[CHC+10] 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.

[Che18] Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. In *CCC*. arXiv preprint arXiv:1802.02325, 2018.

[CJK+13] 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.

[CK93] Paul B Callahan and S Rao Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *SODA*, volume 93, pages 291–300, 1993.

[CK95] 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.

[CKM+14] 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*, pages 343–352. ACM, 2014.

[CLS19] 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.

[CMS20] 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 '20*, 2020.

[CS17] 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. IEEE, 2017.

[CSB+11] 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*, 33(3):568–586, March 2011.

[CSZ09] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[Dan47] George B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, 13:339–347, 1947.

[DS00] Jack Dongarra and Francis Sullivan. Guest editors' introduction: The top 10 algorithms. *Computing in Science & Engineering*, 2(1):22, 2000.

[DZPS19] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *ICLR*. https://arxiv.org/pdf/1810.02054, 2019.

[FS03] 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.

[GE08] 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.

[GR87] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

[GR88] Leslie Greengard and Vladimir Rokhlin. The rapid evaluation of potential fields in three dimensions. *Vortex Methods. Springer, Berlin, Heidelberg*, pages 121–141, 1988.

[GR89] Leslie Greengard and Vladimir Rokhlin. On the evaluation of electrostatic interactions in molecular modeling. *Chemica scripta*, 29:139–144, 1989.

[GS91] Leslie Greengard and John Strain. The fast gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94, 1991.

[Gun98] Steve R Gunn. Support vector machines for classification and regression. *ISIS technical report*, 14(1):5–16, 1998.

[Ham04] Bart Hamers. Kernel models for large scale applications. ., 2004.

[HKNS15] 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.

[Hof07] Heiko Hoffmann. Kernel pca for novelty detection. *Pattern recognition*, 40(3):863–874, 2007.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[IM98] 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*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[JKL09] 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. IEEE, 2009.

[JL84] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[JSWZ20] Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster lps. *arXiv preprint arXiv:2004.07470*, 2020.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[Kha80] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[KLP+16] Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A. Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In *STOC*, 2016.

[KMP10] Ioannis Koutis, Gary L Miller, and Richard Peng. Approaching optimality for solving sdd linear systems. *FOCS*, 2010.

[KMP11] 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*, pages 590–598. IEEE, 2011.

[KMT12] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nyström method. *Journal of Machine Learning Research*, 13(Apr):981–1006, 2012.

[KOSZ13] 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*, pages 911–920. ACM, 2013.

[KS16] Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians-fast, sparse, and simple. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 573–582. IEEE, 2016.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[LL18] 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.

[Llo82] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[LS13] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *FOCS*, 2013.

[LS14] 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.

[LS15] Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear time. In *FOCS*, 2015.

[LSS+20] Jason D. Lee, Ruoqi Shen, Zhao Song, Mengdi Wang, and Zheng Yu. Generalized leverage score sampling for neural network. *manuscript*, 2020.

[LSZ19a] Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empricial risk minimization in the current matrix multiplication time. In *COLT*, 2019.

[LSZ+19b] 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.

[LWDH13] 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.

[Mar12] Per-Gunnar Martinsson. Encyclopedia entry on "fast multipole methods". In *University of Colorado at Boulder*. http://amath.colorado.edu/faculty/martinss/2014_CBMS/Refs/2012_fmm_encyclopedia.pdf, 2012.

[Mas10] Peter Massopust. *Interpolation and approximation with splines and fractals*. Oxford University Press, Inc., 2010.

[Mic84] 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.

[MSS+99] 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*, pages 536–542, 1999.

[NJW02] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[NS41] J. Von Neumann and I. J. Schoenberg. Fourier integrals and metric geometry. *Transactions of the American Mathematical Society*, 50(2):226–251, 1941.

[RR08] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184. https://people.eecs.berkeley.edu/~brecht/papers/07.rah.rec.nips.pdf, 2008.

[Rub18] Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1260–1268. ACM, 2018.

[Sch37] 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.

[Shl14] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[SM19] Karthik C. S. and Pasin Manurangsi. On closest pair in euclidean metric: Monochromatic is as hard as bichromatic. In *ITCS*. https://arxiv.org/pdf/1812.00901.pdf, 2019.

[Son19] Zhao Song. *Matrix Theory : Optimization, Concentration and Algorithms*. PhD thesis, The University of Texas at Austin, 2019.

[Sou10] César R Souza. Kernel functions for machine learning applications. *Creative Commons Attribution-Noncommercial-Share Alike*, 3:29, 2010.

[SS11] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[SSB+17] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

[SSM98] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[ST04] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the STOC*, volume 4, 2004.

[SY19] Zhao Song and Xin Yang. Quadratic suffices for over-parametrization via matrix chernoff bound. *arXiv preprint arXiv:1906.03593*, 2019.

[SY20] Zhao Song and Zheng Yu. Sketching as a tool for solving linear programs. In *Manuscript*, 2020.

[Uns99] Michael Unser. Splines: A perfect fit for signal and image processing. *IEEE Signal processing magazine*, 16(6):22–38, 1999.

[Vai87] 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.

[Vai89] Pravin M Vaidya. Speeding-up linear programming using fast matrix multiplication. In *30th Annual Symposium on Foundations of Computer Science*, pages 332–337. IEEE, 1989.

[vL07] Ulrike von Luxburg. A tutorial on spectral clustering, 2007.

[VZ12] 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.

[Zhu05a] Xiaojin Zhu. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, language technologies institute, school of Computer Science, 2005.

[Zhu05b] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.

[ZL05] 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. ACM, 2005.

[ZSD17] Kai Zhong, Zhao Song, and Inderjit S Dhillon. Learning non-overlapping convolutional neural networks with multiple kernels. *arXiv preprint arXiv:1711.03440*, 2017.

[ZSJ+17] 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-Volume 70*, pages 4140–4149. JMLR. org, 2017.