Electronic Colloquium on Computational Complexity, Report No. 178 (2022)

A Strongly Polynomial Algorithm for Approximate Forster Transforms and its Application to Halfspace Learning

Ilias Diakonikolas[†]
University of Wisconsin-Madison
`ilias@cs.wisc.edu`

Christos Tzamos[‡]
University of Wisconsin-Madison
`tzamos@cs.wisc.edu`

Daniel M. Kane[§]
University of California, San Diego
`dakane@cs.ucsd.edu`

December 6, 2022

Abstract

The Forster transform is a method of regularizing a dataset by placing it in *radial isotropic position* while maintaining some of its essential properties. Forster transforms have played a key role in a diverse range of settings spanning computer science and functional analysis. Prior work had given *weakly* polynomial time algorithms for computing Forster transforms, when they exist. Our main result is the first *strongly polynomial time* algorithm to compute an approximate Forster transform of a given dataset or certify that no such transformation exists. By leveraging our strongly polynomial Forster algorithm, we obtain the first strongly polynomial time algorithm for *distribution-free* PAC learning of halfspaces. This learning result is surprising because *proper* PAC learning of halfspaces is *equivalent* to linear programming. Our learning approach extends to give a strongly polynomial halfspace learner in the presence of random classification noise and, more generally, Massart noise.

[*]Author names are in randomized order.

[†]Supported by NSF Medium Award CCF-2107079, NSF Award CCF-1652862 (CAREER), a Sloan Research Fellowship, and a DARPA Learning with Less Labels (LwLL) grant.

[‡]Supported by NSF Award CCF-2008006 and NSF Award CCF-2144298 (CAREER).

[§]Supported by NSF Medium Award CCF-2107547, NSF Award CCF-1553288 (CAREER), and a grant from CasperLabs.

ISSN 1433-8092

1 Introduction

1.1 Forster Transforms and Their Applications

The Forster transform is a method of regularizing a dataset X (in particular, by placing it in *radial* isotropic position) while maintaining some of its essential properties. Forster transforms have been an essential tool in a diverse range of settings, including functional analysis [Bar98, GGdOW17], communication complexity [For02], coding theory [DSW17], mixed determinant/volume approximation [GS02], learning theory [HM13, HKLM20, DKT21, DPT21] and the Paulsen problem in frame theory [KLLR18, HM19]. The reader is referred to [AKS20] for a more detailed discussion.

Known algorithms for computing (approximate) Forster transforms [HM13, AKS20, DKT21] rely on black-box convex optimization (e.g., the ellipsoid algorithm) and consequently have *weakly* polynomial runtimes. Here we study the question of whether Forster transforms can be computed in *strongly* polynomial time. We then leverage Forster transforms for the problem of PAC learning halfspaces (both in the realizable setting and in the presence of semi-random label noise).

Intuitively speaking, a Forster transform is a mapping that turns a dataset into one with good *anti-concentration* properties. Specifically, given a dataset $X \subset \mathbb{R}_*^d$[1], a Forster transform of X is an invertible linear transformation $A \in \mathbb{R}^{d\times d}$ such that the set of points $Y = \{Ax/\|Ax\|_2, x \in X\}$ is in isotropic position (i.e., has identity second moment matrix). Formally, we have the following more general definition allowing for *approximate* isotropic position.

Definition 1.1 (Approximate Forster Transform)**.** Let X be a set of n nonzero points in $\mathbb{R}^d$ and $0 \le \epsilon \le 1$ be an error parameter. An *ϵ-approximate Forster transform* of X is an invertible linear transformation $A \in \mathbb{R}^{d\times d}$ such that, considering the mapping $f_A : \mathbb{R}_*^d \mapsto \mathbb{S}^d$ defined by $f_A(x) \overset{\text{def}}{=} Ax/\|Ax\|_2$, the matrix $M_A(X) \overset{\text{def}}{=} (1/n)\sum_{x\in X} f_A(x)f_A(x)^\top$ satisfies $\frac{1-\epsilon}{d}\, I \preceq M_A(X) \preceq \frac{1+\epsilon}{d} I$.

An *exact* Forster transform (corresponding to $\epsilon = 0$ in Definition 1.1) aims to linearly transform a given dataset so that the normalizations of these points are in isotropic position. This notion is known as “Forster's isotropic position” or “radial isotropic position” and can be viewed as an outlier-robust analogue of isotropic position. As already mentioned, radial isotropy has been extensively studied in functional analysis and computer science.

Remark 1.2. At a high-level, a Forster transform aims to transform a given dataset so that it becomes “well-conditioned” in a well-defined technical sense. We note that several other such transformations have been studied in the literature, including the “outlier-removal technique” of Dunagan and Vempala [DV04a] (improving on [BFKV96]) and the rescaling method of Dunagan and Vempala [DV04b] for linear programming. We provide a summary of these techniques and a comparison to radial isotropy in Section 1.5.

Existence Forster [For02] showed that if the set of points X is in general position, then a Forster transform exists. Interestingly, generalizations of Forster's theorem appear implicitly in [Bar98] and explicitly in [GS02]. We note that there are datasets for which a Forster transform does not exist. For example, if there is a $d/3$-dimensional subspace that contains half of the points in X, then after applying *any* such transformation to our dataset, this will still be the case; thus, there will be a $d/3$-dimensional subspace over which the trace of the second moment matrix is at least $1/2$. In a recent refinement of the aforementioned works, [HKLM20] showed that this is the only thing that can go wrong. That is, a Forster transform of a given dataset X exists unless there is a k-dimensional subspace, for some $0 < k < d$, containing at least a k/d-fraction of the points in X.

[1]We use $\mathbb{R}_*$ to denote the set $\mathbb{R} \setminus \{\mathbf{0}\}$.

Efficient Computability Forster's existence proof proceeds via a non-constructive iterative argument. By analyzing a convex program proposed by Barthe [Bar98], Hardt and Moitra [HM13] (see also [AKS20]) showed that the ellipsoid method yields a *weakly polynomial* time algorithm to compute an approximate Forster transform (when it exists). (More recently, [DKT21] pointed out that a simple explicit SDP can be used to obtain a similar guarantee.) We remind the reader that the term *weakly polynomial time* algorithm refers to the fact that the *number of arithmetic operations* performed by the algorithm scales polynomially with *the bit complexity of the numbers in the input.* Specifically, in our Forster setting, the number of arithmetic operations required by the ellipsoid method is $\mathrm{poly}(n, d, b, \log(1/\epsilon))$, where ϵ is the accuracy parameter of Definition 1.1, n is the size of the dataset X, and b is the bit complexity of X.

Starting from the convex programming formulation in [Bar98], Artstein-Avidan, Kaplan, and Sharir [AKS20] gave an SVD-based gradient-descent method for computing approximate Forster transforms. This method incurs a $\mathrm{poly}(1/\epsilon)$ runtime dependence and is still weakly polynomial, i.e., the number of arithmetic operations scales polynomially in the bit complexity b. Finally, it is interesting to remark that Forster's rescaling is a special case of operator scaling and tensor scaling (see [GdO18] for a survey). Efficient algorithms have been developed for these more general tasks, see, e.g., [AGL$^+$18, BFG$^+$18], albeit with weakly polynomial guarantees.

Weakly versus Strongly Polynomial Time As is standard for computational purposes, we assume that every integer or rational number appearing in the input is encoded using its binary representation. Let $N \in \mathbb{Z}_+$ denote the number of integer numbers given as input and $b \in \mathbb{Z}_+$ denote the bit complexity of the largest integer appearing in the input description. An algorithm for the underlying computational problem is called *weakly polynomial*, if its worst-case running time is bounded by a fixed-degree polynomial in the Turing machine model of computation.

The concept of *strongly polynomial* time was introduced by Megiddo [Meg83], under the name "genuinely polynomial". A strongly polynomial time algorithm satisfies the following properties (see, e.g., Section 1.3 of [GLS88]): (i) it uses only elementary arithmetic operations (specifically, integer addition, subtraction, multiplication, and division), (ii) the number of arithmetic operations is bounded above by a polynomial in N, and (iii) the algorithm is a polynomial space algorithm: that is, all numbers appearing in all intermediate computations are rational numbers with bit complexity bounded above by a polynomial in the input size (i.e., $\mathrm{poly}(N, b)$).

The key difference between strongly and weakly polynomial time lies in property (ii) above. In a weakly polynomial algorithm, the *number of arithmetic operations* is allowed to scale with the bit complexity of the numbers in the input. *In sharp contrast, in a strongly polynomial time algorithm no bit complexity dependence is allowed.*

Forster Transforms in *Strongly* Polynomial Time? Motivated by the fundamental nature and the varied applications of Forster transforms, here we ask the following question:

Is there a strongly *polynomial time algorithm to compute an approximate Forster transform of a given dataset (assuming one exists)?*

Our main algorithmic result (Theorem 1.5) answers this question in the affirmative by giving the first randomized strongly polynomial-time algorithm for computing approximate Forster transforms — corresponding to $\epsilon = \Omega(\mathrm{poly}(1/(n, d)))$ in Definition 1.1. Importantly, a constant value of ϵ suffices for our learning theory application to learning halfspaces. Obtaining a strongly polynomial time algorithm for inverse exponential values of ϵ is left as an interesting open problem (see Section 8 for a discussion).

1.2 Halfspaces and Efficient PAC Learnability

One of the main motivations behind this work was leveraging Forster transforms as a tool for the algorithmic problem of distribution-free PAC learning of halfspaces. We review the relevant background in the subsequent discussion.

Halfspaces We are concerned with the efficient learnability of halfspaces in Valiant's distribution-free PAC model [Val84]. A *halfspace* or Linear Threshold Function (LTF) is any Boolean-valued function $f : \mathbb{R}^d \mapsto \{\pm 1\}$ of the form $f(x) = \mathrm{sign}(w \cdot x - t)$, for some $w \in \mathbb{R}^d$ (known as the weight vector) and $t \in \mathbb{R}$ (known as the threshold). (The function $\mathrm{sign} : \mathbb{R} \mapsto \{\pm 1\}$ is defined as $\mathrm{sign}(u) = 1$ if $u \geq 0$, and $\mathrm{sign}(u) = -1$ otherwise.) Halfspaces are one of the most extensively studied classes of Boolean functions due to their central role in several areas, including complexity theory, learning theory, and optimization [Ros58, Nov62, MP68, Yao90, GHR92, FS97, Vap98, STC00, O'D14].

Background on PAC Learning The major goal of computational learning theory is to develop learning algorithms for expressive concept classes that are both statistically and computationally efficient. To facilitate the subsequent discussion, we formally define Valiant's PAC model.

Definition 1.3 (PAC Learning)**.** Let $\mathcal{C}$ be a class of Boolean-valued functions over $X = \mathbb{R}^d$ and $\mathcal{D}_X$ be a fixed but unknown distribution over X. Let f be an unknown target function in $\mathcal{C}$. A *PAC example oracle*, $\mathrm{EX}(f, \mathcal{D}_X)$, works as follows: Each time $\mathrm{EX}(f, \mathcal{D}_X)$ is invoked, it returns a labeled example (x, y), where $x \sim \mathcal{D}_X$ and $y = f(x)$. Let $\mathcal{D}$ denote the joint distribution on (x, y) generated by the above oracle. Given an accuracy parameter $\gamma > 0$ and access to i.i.d. samples from $\mathcal{D}$, the learner wants to output a hypothesis $h : \mathbb{R}^d \mapsto \{\pm 1\}$ such that with high probability the misclassification error of h is at most γ, i.e., we have that $\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[h(x) \neq y] \leq \gamma$.

The hypothesis h in Definition 1.3 does not necessarily belong to the class $\mathcal{C}$. Namely, we focus on the standard notion of *improper* learning, where the learner can output any efficiently computable hypothesis. The special case where h is required to lie in $\mathcal{C}$ is known as *proper* learning. While proper learning might be desirable for some applications (e.g., due to its interpretability), there exist natural concept classes for which proper learning is computationally hard and improper learning is easy (see, e.g., [KV94]). An improper hypothesis is as useful as a proper one for the purpose of predicting new function values.

Remark 1.4. The PAC model of Definition 1.3 is known as *realizable* because of the assumption that the labels are consistent with the target concept. While our main learning application is on the realizable learning of halfspaces in strongly polynomial time (Theorem 1.6), our positive result extends for learning halfspaces in the presence of random or semi-random label noise (Theorem 1.8).

PAC Learning Halfspaces and Linear Programming With this terminology, we return to our discussion on halfspaces. Suppose we are given a multiset of n labeled examples, $(x^{(i)}, y^{(i)})$, with $x^{(i)} \sim \mathcal{D}_X$ and $y^{(i)} = f^*(x^{(i)})$, where $f^*(x) = \mathrm{sign}(w^* \cdot x - t^*)$ is the target halfspace. Then we can find a consistent halfspace hypothesis $h(x) = \mathrm{sign}(\widehat{w} \cdot x - \widehat{t})$ (i.e., a halfspace that agrees with the training set) via a reduction to Linear Programming (LP); see, e.g., [MT94]. Indeed, each example $(x^{(i)}, y^{(i)})$ gives rise to the linear inequality $(w \cdot x^{(i)} - t)y^{(i)} \geq 0$ over variables $(w, t) \in \mathbb{R}^{d+1}$. This gives us an LP with $d + 1$ variables and n constraints, which is feasible (as (w^*, t^*) is a feasible solution by assumption). We can thus use any polynomial-time LP algorithm to compute a feasible solution $(\widehat{w}, \widehat{t})$. By standard VC-dimension generalization results (see, e.g., [KV94]), if the sample size n is sufficiently large, namely for some $n = \tilde{O}(d/\gamma)$, the halfspace hypothesis

$h(x) = \mathrm{sign}(\widehat{w} \cdot x - \widehat{t})$ with high probability satisfies $\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[h(x) \neq y] \leq \gamma$. This straightforward reduction gives a PAC learning algorithm for halfspaces on $\mathbb{R}^d$ with sample complexity $\tilde{O}(d/\gamma)$ and running time polynomial in the input size. Formally speaking, the running time of such an algorithm is *weakly polynomial*, i.e., its worst-case number of arithmetic operations scales with the bit complexity of the input examples.

Interestingly, the aforementioned reduction can be reversed. That is, one can use any PAC learner that outputs a halfspace hypothesis as a black-box to solve the linear feasibility problem $Aw \geq 0$, $w \neq 0$, where $A \in \mathbb{R}^{n\times d}$ and $w \in \mathbb{R}^d$, by considering each linear constraint as an example. Intuitively, the vector w can be viewed as the weight vector defining the target halfspace.

Learning Halfspaces in *Strongly* Polynomial Time? All known polynomial time algorithms for LP, including the ellipsoid algorithm and interior-point methods, are weakly polynomial. The existence of a strongly polynomial LP algorithm is a major open question in computer science, famously highlighted by Smale [Sma98]. The straightforward reduction of PAC learning halfspaces to LP leads to a *weakly* polynomial learner. Interestingly, the reduction in the opposite direction has lead various authors (see [Coh97] and recently [DGT19, CKMY20]) to suggest that learning halfspaces in strongly polynomial time is *equivalent* to strongly polynomial LP. The catch, of course, is that this equivalence only holds if we restrict ourselves to *proper* learners.

Several weakly polynomial time algorithms for PAC learning halfspaces have been developed over the past thirty years, starting with the pioneering works [BFKV96, Coh97, DV04b] and recently in [DGT19, CKMY20, DKT21]. (These works do not proceed by a black-box reduction to solving LPs.) These learners succeed not only in the realizable setting, but also in the presence of (semi)-random label noise. Importantly, all prior learners are weakly polynomial — even restricted to the realizable setting. This discussion serves as a motivation for the following question:

Is there a strongly *polynomial time algorithm for PAC learning halfspaces?*

The main learning-theoretic result of this paper (Theorem 1.6) answers the above question in the affirmative. This algorithmic result generalizes to yield strongly polynomial time algorithms for learning halfspaces in "benign" noise models, including Random Classification Noise (RCN) [AL88] and, more generally, Massart noise [MN06] (Theorem 1.8).

1.3 Our Results

The main algorithmic result of this work is the first randomized strongly polynomial time algorithm for computing an approximate Forster transform of a given dataset, assuming that one exists.

Theorem 1.5 (Approximate Forster Transforms in Strongly Polynomial Time)**.** *There exists a randomized algorithm that given a set $X \subset \mathbb{R}^d_*$ of size n and a parameter $\epsilon \in (0, 1)$, runs in time strongly polynomial in nd/ϵ, and has the following high probability guarantee: either the algorithm computes an ϵ-approximate Forster transform of X, or it correctly detects that no Forster transform of X exists by finding a proper subspace $W \subset \mathbb{R}^d$ such that $|X \cap W| > (n/d)\dim(W)$.*

In more detail, the algorithm of Theorem 1.5 performs $\mathrm{poly}(n, d, 1/\epsilon)$ arithmetic operations on $\mathrm{poly}(n, d, 1/\epsilon, b)$-bit numbers, where b is the bit complexity of the points in X. As discussed in the introduction, previous algorithms for this problem rely on the ellipsoid method and therefore are weakly polynomial even for constant values of ϵ. The running time of our algorithm has a polynomial dependence in $1/\epsilon$; hence, our algorithm does not run in polynomial time when ϵ is inverse super-polynomially small in n, d. Importantly, for our application in halfspace learning (and several other applications of Forster transforms) constant values of the parameter ϵ suffice.

By using the algorithm of Theorem 1.5 as a black-box (for $\epsilon = 1/2$), we establish our main learning result (see Theorem 7.5 for a more detailed statement).

Theorem 1.6 (PAC Learning Halfspaces in Strongly Polynomial Time)**.** *Let $\mathcal{D}$ be a distribution over labeled examples $(x, y) \in \mathbb{R}^d \times \{\pm 1\}$ such that the distribution over examples is arbitrary and the label y of example x satisfies $y = f(x)$, for an unknown halfspace $f : \mathbb{R}^d \mapsto \{\pm 1\}$. There is an algorithm that, given $\gamma > 0$, draws $n = \mathrm{poly}(d/\gamma)$ i.i.d. samples from $\mathcal{D}$, runs in strongly polynomial time, and returns a strongly polynomial time computable hypothesis $h : \mathbb{R}^d \mapsto \{\pm 1\}$ such that with high probability we have that $\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[h(x) \neq y] \leq \gamma$.*

Given the *equivalence* of *proper* halfspace learning and LP, we view this algorithmic result as fairly surprising. Theorem 1.6 gives the first strongly polynomial time PAC learning algorithm for halfspaces. In more detail, if b is the bit complexity of the examples (i.e., the maximum number of bits required to represent each coordinate of each example vector), our algorithm uses $\mathrm{poly}(n)$ arithmetic operations on $\mathrm{poly}(n, b)$-bit numbers. Finally, we note that the hypothesis h computed by our algorithm is a decision-list of $\mathrm{poly}(d/\gamma)$ many halfspaces. Importantly, for each point x, the value $h(x)$ is computable in strongly polynomial time (in n).

Remark 1.7. The list of concept classes for which efficient learners have been developed in Valiant's distribution-free PAC model is fairly short. The class of halfspaces is of central importance in this list. Specifically, a strongly polynomial algorithm for PAC learning halfspaces immediately implies (via the kernel trick) strongly polynomial learners for broader concept classes, including degree-k polynomial threshold functions for any $k = O(1)$ (see, e.g., [BEHW89]).

It is worth pointing out that the idea of using Forster transforms for halfspace learning was recently used in [DKT21] for the problem of PAC learning with Massart noise. In the Massart model [MN06], an adversary independently flips the label of each point x with unknown probability $\eta(x) \leq \eta < 1/2$. The learner of [DKT21] used a weakly polynomial Forster transform routine. By instead using our algorithm of Theorem 1.5, we obtain the following generalization of Theorem 1.6.

Theorem 1.8 (PAC Learning Massart Halfspaces in Strongly Polynomial Time)**.** *Let $\mathcal{D}$ be a distribution over labeled examples $(x, y) \in \mathbb{R}^d \times \{\pm 1\}$ such that the distribution over examples is arbitrary and the label y of example x satisfies (i) $y = f(x)$ with probability $1 - \eta(x)$, and (ii) $y = -f(x)$ with probability $\eta(x)$, for an unknown halfspace $f : \mathbb{R}^d \mapsto \{\pm 1\}$. Here $\eta(x)$ is an unknown function that satisfies $\eta(x) \leq \eta < 1/2$ for all x. There is an algorithm that, given $\gamma > 0$, draws $n = \mathrm{poly}(d/\gamma)$ i.i.d. samples from $\mathcal{D}$, runs in strongly polynomial time, and returns a strongly polynomial time computable hypothesis $h : \mathbb{R}^d \mapsto \{\pm 1\}$ such that with high probability we have that $\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[h(x) \neq y] \leq \eta + \gamma$.*

Theorem 1.8 generalizes Theorem 1.6 (which corresponds to the case of $\eta = 0$). For the special case of uniform noise (i.e., when $\eta(x) = \eta < 1/2$ for all x) — this is known as Random Classification Noise [AL88] — Theorem 1.8 achieves the information-theoretically optimal error and runs in *strongly* polynomial time. It thus qualitatively improves on the classical work of [BFKV96] who gave a weakly polynomial time algorithm with the same error guarantee.

Theorem 1.8 similarly improves prior work on learning halfspaces with Massart noise. Prior algorithms for learning Massart halfspaces have weakly polynomial runtimes and achieve the same error as Theorem 1.8, which is believed to be the computational limit for the problem. In more detail, the first (weakly) polynomial learner for Massart halfspaces was given in [DGT19] and achieves error $\eta+\gamma$, as our Theorem 1.8. While this error guarantee is not information-theoretically optimal in the Massart model (the optimal error is $\mathrm{OPT} = \mathbf{E}_x[\eta(x)]$), there exists strong evidence [DK20, NT22, DKMR22] that the bound of η cannot be improved by any polynomial time algorithm. Finally, we note that subsequent work to [DGT19] gave a proper learner for Massart halfspaces [CKMY20], which is inherently weakly polynomial.

1.4 Our Techniques

1.4.1 Strongly Polynomial Approximate Forster Transform

Overview of Algorithmic Approach Letting $f_A(x) \overset{\text{def}}{=} Ax/\|Ax\|_2$, given a dataset X of n points in $\mathbb{R}^d_*$, our goal is to efficiently compute an invertible linear transformation $A \in \mathbb{R}^{d\times d}$ such that the matrix $M_A(X) \overset{\text{def}}{=} (1/n)\sum_{x\in X} f_A(x)f_A(x)^\top$ is approximately equal to $(1/d)\,I$; in particular, we would like it to have eigenvalues in $[\frac{1-\epsilon}{d}, \frac{1+\epsilon}{d}]$. Since the trace of $M_A(X)$, $\mathrm{tr}(M_A(X))$, is always equal to 1, this goal is equivalent to finding a matrix A such that the squared Frobenius norm of $M_A(X)$, $\|M_A(X)\|_F^2$, is close to $1/d$ (Lemma 3.1). This observation gives rise to the natural idea of using an iterative algorithm to compute such an A. In particular, given a linear transformation A such that $\|M_A(X)\|_F^2$ is somewhat small, our goal is then to find another linear transformation $C \in \mathbb{R}^{d\times d}$ such that the corresponding second moment matrix $M_{CA}(X) = (1/n)\sum_{x\in X} f_{CA}(x)f_{CA}(x)^\top$ has squared Frobenius norm, $\|M_{CA}(X)\|_F^2$, somewhat smaller than $\|M_A(X)\|_F^2$. Equivalently, since for any point $x \in \mathbb{R}^d_*$ it holds that $f_{CA}(x) = f_C(f_A(x))$, we consider the set of transformed points $X_A = f_A(X) \overset{\text{def}}{=} \{f_A(x) : x \in X\}$ and aim to make the second moment matrix of $f_C(X_A)$ smaller than the second moment matrix of X_A. If for any invertible A we can find such a C, then by iteratively replacing A by CA we can achieve smaller and smaller values of $\|M_A(X)\|_F^2$, until in the limit it approaches $1/d$.

Since $\mathrm{tr}(M_A(X)) = 1$, if $\|M_A(X)\|_F^2$ is bounded away from $1/d$, some of the eigenvalues of $M_A(X)$ (which average to $1/d$) must differ substantially from $1/d$. This in turn implies that $M_A(X)$ must have a reasonably-sized *eigenvalue gap*. In particular, this means that there exist subspaces V and $V^\perp$, that are each spanned by eigenvectors of $M_A(X)$, such that the eigenvalues of $V^\perp$ exceed the eigenvalues on V by at least some reasonably large $\delta > 0$. Roughly speaking, if we can find a matrix C that decreases the squared Frobenius norm of $M_A(X)$ on $V^\perp \times V^\perp$ and increases the squared Frobenius norm on $V \times V$, this will improve the desired squared Frobenius norm.

A natural approach to achieve this goal is to let C be equal to $I_{V^\perp} + (1+\alpha)I_V$, the identity on $V^\perp$, and $(1+\alpha)$ times the identity on V, for some suitable $\alpha > 0$. It is not hard to see that this choice of C strictly decreases the second moment matrix on $V^\perp$, and strictly increases it on V. Unfortunately, it might also create cross-terms that will increase the Frobenius norm. To understand the effect of the cross-terms, it is important to consider how close vectors in X_A are to being in V or in $V^\perp$. In particular, let β be the maximum distance that any vector in X_A is from being in either V or $V^\perp$. If $\alpha = O(1)$, this moves approximately $\alpha\beta^2$ of the trace of $M_A(X)$ from $V^\perp$ to V, which improves (i.e., decreases) the squared Frobenius norm by roughly $\alpha\beta^2$ (times some inverse $\mathrm{poly}(dn/\epsilon)$ factors). On the other hand, this also creates cross-terms in the order of $\alpha\beta$, which increases the squared Frobenius norm by a quantity on the order of $\alpha^2\beta^2$. Thus, as long as α is less than β times a sufficiently small polynomial in dn/ϵ, we obtain an improvement in the squared Frobenius norm on the order of $\alpha\beta^2/\mathrm{poly}(dn/\epsilon)$.

This improvement suffices for our purposes, unless β happens to be very small. The latter occurs if all of the points in X_A are either very close to V or very close to $V^\perp$. In such a case, the simple choice of matrix C described in the previous paragraph may not be sufficient, as it will produce too many cross-terms. In order to make progress here, we require a different approach, which we describe next. To describe our approach for this case, we introduce additional terminology. We let X_A^{B} be the set of points in X_A that are close to $V^\perp$. Moreover, let U be the span of the $|V|$

smallest eigenvectors of the matrix $\sum_{x\in X_A^{\mathrm{B}}} xx^\top$, and let $U^\perp$ be the orthogonal subspace. We now define the new matrix C to be $I_{U^\perp} + (1+\alpha)I_U$, the identity on $U^\perp$ and some very large multiple $(1+\alpha)$ of the identity on U. We claim that this choice actually does not create much in the way of cross-terms. In particular, the matrix $\sum_{x\in X_A^{\mathrm{B}}}(Cx)(Cx)^\top = C^\top \sum_{x\in X_A^{\mathrm{B}}} xx^\top\, C$ will have no $U\times U^\perp$ term, since $\sum_{x\in X_A^{\mathrm{B}}} xx^\top$ does not — as U is an eigenspace of $\sum_{x\in X_A^{\mathrm{B}}} xx^\top$. The second moment matrix $\sum_{x\in X_A^{\mathrm{B}}} f_C(x)f_C(x)^\top$ will have some contribution to $U\times U^\perp$ cross-terms coming from the renormalization; but these will only be on the order of $(\alpha\,\beta)^4$. On the other hand, the matrix $\sum_{x\in X_A\setminus X_A^{\mathrm{B}}} f_C(x)f_C(x)^\top$ will have small $U\times U^\perp$ terms, because each $f_C(x)$ will nearly lie in $U^\perp$. If β is sufficiently small, this leads to roughly $(\alpha\,\beta)^2$ mass being moved from $U^\perp\times U^\perp$ to $U\times U$, while only creating off-diagonal terms on the order of $(\alpha\,\beta)^4$. Thus, this alternate choice of C can be used to decrease the squared Frobenius norm by $\mathrm{poly}(\alpha/(dn))$.

The preceding outline provides a procedure that produces a sequence of matrices $A_1, A_2, \ldots$ such that if $e_i = \|M_{A_i}(X)\|_F^2 - 1/d$, then $e_{i+1} < e_i - \mathrm{poly}(e_i/(dn))$. Therefore, after polynomially many iterations, we have that $\|M_{A_m}(X)\|_F^2 < 1/d + (\epsilon/d)^2$, which implies we have obtained an ϵ-approximate Forster transform. This gives us an efficient algorithm for computing an approximate Forster transform in the real RAM model, assuming the availability of an algorithm for exact eigendecomposition computation.

Additional Technical Obstacles The above iterative procedure forms the basis of our final strongly polynomial time algorithm. Unfortunately, as is, this procedure does not directly imply a strongly polynomial time algorithm for two reasons: First, we need to control the bit complexities of the matrices A_i (which might become exponentially large). Second, we need to show that our algorithm works with approximate eigendecompositions (which can further be implemented in strongly polynomial time). We elaborate on these issues in the following discussion.

Controlling the Bit Complexity via Rounding Recall that, in a strongly polynomial time algorithm, all intermediate numbers computed throughout the algorithm must fit in polynomial space. To handle the bit complexity in our setting, we establish the following statement. If the points in the initial dataset $X \subset \mathbb{R}^d_*$ of size n have bit complexity at most b, then the following holds: given a matrix $A \in \mathbb{R}^{d\times d}$ and any $\delta > 0$, we can approximate A by another matrix A' of bit complexity $\mathrm{poly}(b, d, n, \log(1/\delta))$ such that $\|M_{A'}(X)\|_F^2 < \|M_A(X)\|_F^2 + \delta$ (see Theorem 5.1). This structural result suffices for our purposes for the following reason: Replacing each intermediate matrix A_i (in our iterative procedure) by the corresponding A_i' obtained by rounding (for an appropriately small δ) at each step of our algorithm suffices to keep the bit-complexity under control.

To prove the desired structural result, we proceed as follows: First, if A has condition number at most $\exp(\mathrm{poly}(n, b, d))$, it suffices to merely approximate each entry of A to some $\mathrm{poly}(bdn/\log(1/\delta))$ bits of precision. The difficulty arises if the condition number of A is quite large — in fact, exponentially large in our other parameters. If the condition number of A is large, it is because there are large multiplicative gaps in the singular values of A. In such a case, there will be subspaces V and $V^\perp$ such that the $V^\perp$-component of any vector is multiplied by a huge amount relative to the V-component. In particular, any vector that was not exponentially close to V to begin with, after multiplying by A ends up essentially in $V^\perp$. Our basic strategy here is to decrease the size of this singular value gap of A to be at most (merely) exponential, without much affecting any of the normalized transformed vectors. Our goal is to scale down the subspace $V^\perp$ to decrease the multiplicative eigenvalue gap. However, we must ensure that the vectors of X that are sufficiently close to $V^\perp$ after applying A do not end up being essentially in V. To achieve this, we consider a subspace W spanned by such problematic vectors and build an improved matrix AT such that T does not affect vectors in W, but rescales significantly vectors lying in a subspace R that is very close to $V^{\perp}$. Via this step, we can reduce the condition number of A to be appropriately bounded without affecting the mapping f_A significantly; after that, we can make do with a suitably precise rounding to obtain the output matrix A'.

Approximate Eigendecomposition in Strongly Polynomial Time So far, we have assumed the availability of a routine for exact eigendecomposition. In fact, there are several places in the above intuitive overview of our algorithmic approach where we need to compute an eigenvalue decomposition of a matrix. This is required first when we need to find the initial eigenvalue gap in $M_A(X) \propto \sum_{x \in X_A} xx^{\top}$, and again later when we need to find the span of the large eigenvalues of $\sum_{x \in X_A^{\mathrm{B}}} xx^{\top}$. Unfortunately, computing exact eigenvalues is impossible in our model of computation (as doing so might require finding roots of high-degree polynomials). Fortunately, it is sufficient for us to find merely an *approximate* eigenvalue decomposition of these matrices. A subtle and important point is that our required notion of approximation is *significantly stronger than the typical guarantees explicitly available in the literature.* Interestingly, we show that the desired strongly polynomial guarantees can be achieved in our model using some variation of the power iteration method. This requires a novel proof of correctness, that we provide here.

We are now ready to describe our strongly polynomial approximate eigendecomposition routine in tandem with a sketch of its analysis (see Proposition 4.1). The standard power iteration method says that in order to approximate the principal eigenvector of a symmetric, PSD matrix M, it suffices to multiply a random vector v by a large power t of M. If we express v as a linear combination of eigenvectors of M, then multiplying by a large power of M scales each of these components by an amount depending on the eigenvalue. It is not hard to see that if there is a reasonable gap between the largest and second largest eigenvalues, then the vector $M^t v$ will likely end up close to a multiple of the largest eigenvector. Once an approximate principal eigenvector is computed, one can attempt to repeat the same procedure, i.e., projecting onto the orthogonal subspace to find the second largest eigenvalue; and so on. This iterative procedure is known to succeed in finding approximations to the eigenvectors and eigenvalues in question, so long as the eigenvalues are not too close to each other. On the other hand, if M has (nearly) degenerate eigenspaces, then this method may fail to separate eigenvectors with very similar eigenvalues. However, in this (near-)degenerate case, such an approximation is usually not needed, as the eigenvalues are close to begin with. One can hope that the matrix $\hat{M}$ corresponding to the computed eigendecomposition is close to M in an appropriate sense. In particular, standard results (see, e.g., [Par98]) show how to compute such an $\hat{M}$ satisfying $\|M - \hat{M}\|_2 \leq \epsilon \|M\|_2$.

Unfortunately, this notion of approximation is not sufficient for our purposes. For example, in the case where the parameter β is small in our Forster algorithm, it is important for us to compute the spaces V' and W' to *very good accuracy.* This is because the linear transformation that we apply will multiply elements of V' by a large factor of roughly $1/\beta$. This means that we need to compute V' to error on the order of β in order to ensure the accuracy of our result. More generally, we will need a qualitatively stronger guarantee for our approximate eigenvalue decomposition. In particular, we need that for some small $\epsilon > 0$, for any vector v, it holds that $|v^{\top}(M - \hat{M})v| \leq \epsilon(v^{\top} M v)$. This means that if v lies in a space spanned by eigenvectors of M with very small eigenvalues (as V' is above), then we need that $\hat{M}v$ to be correspondingly small. Fortunately, we can obtain this much stronger "multiplicative" guarantee via power iteration. The intuitive reason this works is essentially because if we have a space V' spanned by eigenvectors of M with eigenvalues at most β, then multiplying a random vector v by powers of M reduces the size of the projection of v onto V' by a power of β. This means that power iteration produces vectors that are very nearly orthogonal to V' with the error in this approximation scaling with β.

1.4.2 Learning Halfspaces in Strongly Polynomial Time

As already mentioned in the introduction, we leverage our algorithm for approximate Forster transforms to obtain the first strongly polynomial algorithm for PAC learning halfspaces. It turns out that this approach goes through both in the realizable case (Definition 1.3) and in the presence of (semi-random) Massart noise on the labels. In fact, it is not difficult to verify that by plugging in our new Forster algorithm into the learning algorithm of [DKT21], one directly obtains a strongly polynomial halfspace learner in the presence of Massart noise. For the sake of the completeness, here we focus on the realizable case and provide a simpler, self-contained algorithm and proof.

Note that it is without loss of generality to assume that the threshold of the target halfspace is zero (one can reduce the general case to the homogeneous case). The main challenge in PAC learning halfspaces is that the target halfspace may have very bad anti-concentration (aka "margin"). If the margin is not too small (i.e., at least inverse polynomial), simple iterative algorithms (e.g., perceptron) efficiently learn halfspaces (in strongly polynomial time). A natural idea is then to reduce the general case to the large margin case by appropriately transforming the data. A number of such reductions have been developed in the literature [BFKV96, DV04a, DV04b, DKT21]. The methods developed in [BFKV96, DV04a, DV04b] are inherently not strongly polynomial. Recently, [DKT21] pointed out that one can use Forster transforms for this purpose.

For our purposes, we require a stronger guarantee than what is provided by the vanilla perceptron algorithm. Specifically, we want a learning algorithm for halfspaces that correctly classifies at least some reasonable fraction of points, if the points are guaranteed to be well-conditioned (for example, in the sense of being unit vectors with $\mathbf{E}[xx^\top] \approx I$). By using an approximate Forster algorithm, we can transform the input points in order to make them well-conditioned, while preserving the notion of halfspaces. We can then apply our learner to this set in order to learn a classifier that works on some reasonable fraction of the points. Repeating this procedure iteratively on the unclassified points eventually gives a halfspace learning algorithm.

More precisely, the modified perceptron algorithm of [DV04b] is a strongly polynomial time algorithm with the following performance guarantee: given labeled examples consistent with an unknown linear classifier, the algorithm learns a classifier that correctly labels all points whose margin is not too small. It is not hard to see that, for points in approximate radial isotropic position, at least a $1/d$-fraction of points have not-too-small margin. Therefore, if we have a set of points in approximate radial isotropic position, the modified perceptron algorithm finds (in strongly polynomial time) an explicit halfspace that separates out a roughly $1/d$-fraction of the points all of the same sign. By standard generalization bounds, this gives us an algorithm that in strongly polynomial time learns a *partial classifier*, i.e., outputs a partial function that correctly classifies an $\Omega(1/d)$-fraction of the points while misclassifying an $O(\gamma/d)$-fraction. In other words, this procedure produces a partial classifier that labels at least a $1/d$-fraction of points and misclassifies at most a γ-fraction of these points.

To learn an arbitrary halfspace, we use our approximate Forster transform to put the points in approximate radial isotropic position without changing the notion of a halfspace on them. We then apply the above partial learner to these new points in order to obtain a non-trivial partial classifier that makes mistakes on only a γ-fraction of its classified set. We repeat this process on the unclassified points, using a new approximate Forster transform, to learn a non-trivial fraction of the unclassified points. Repeating this procedure iteratively as necessary, we eventually obtain a partial classifier that produces an answer on essentially all points of the domain and only makes mistakes on a γ-fraction of them.

1.5 Related Work

In this section, we summarize additional prior work that was not covered in the introduction.

Comparison to Strongly Polynomial Algorithm for Matrix Completion It is worthwhile to compare our techniques for the Forster transform to [LSW00], who developed the first strongly polynomial time algorithm for the matrix scaling problem. To put this problem in terms more analogous to ours, one is given a set of d vectors $x_1, x_2, \dots, x_d$ in $\mathbb{R}^d$. The goal is to find a *diagonal* matrix A such that if $y_i := Ax_i/\|Ax_i\|_1$ is the ℓ_1 normalization of Ax_i, then the absolute deviation of the j^{th} coordinates of the y's around 0 are (approximately) the same for all j. In particular, it should hold that $\sum_{i=1}^{d} |(y_i)_j| \approx 1$ for all $1 \leq j \leq d$. Note that for our problem, we have n (possibly greater than d) vectors, A can be *any* matrix, we take y_i to be the ℓ_2 normalization and we want the mean square deviation of the y's in *any* direction (not just along coordinate axes) to be approximately the same.

The algorithm in [LSW00] works roughly as follows. We construct A through an iterative sequence of improvements. Given a specific A, we compute the appropriate values of y_i and then compute the absolute deviations of each coordinate. If these are all close to each other, we are done. Otherwise, by sorting the deviations and finding the largest gap, we can split our coordinates into two sets, B and S, so that the deviation of any coordinate in B is substantially larger than the deviation of any coordinate in S. One then defines the diagonal matrix C to be $(1 + \delta)$ on the coordinates in S and 1 on the coordinates in B, and replaces A by $A' := CA$. It is not hard to see that by doing this, one increases the deviations along all coordinates in S while decreasing it along all coordinates in B (and keeping the total sum of deviations the same). By picking δ carefully, [LSW00] show that the variance of these coordinate-wise deviations can be decreased by some polynomial amount in each step. Thus, by iterating this method a polynomial number of times, one obtains a scaling where the coordinate-wise deviations are sufficiently close.

The starting point for our algorithm is somewhat similar. Given a matrix A, we try to find a matrix C such that the matrix $A' := CA$ is closer to satisfying our condition (in the sense that $\|M_{A'}(X)\|_F$ should be smaller than $\|M_A(X)\|_F$ by an additive inverse polynomial term). To do this, we compute subspaces V_S and V_B (by finding an eigenvalue gap in $M_A(X)$) such that the variance of the $y_i := Ax_i/\|Ax_i\|_2$ in any direction along V_B is substantially larger than along any direction in V_S. Ideally, we would like to take $C = I + \alpha I_{V_S}$ for some carefully selected α. While this does only increase the variance in directions along V_S and decrease it along V_B, in our setting this *also* creates off-diagonal terms that increase our potential. While it is always possible to ensure that this error does not overwhelm the progress we make by taking α small enough, in some cases (particularly where all of the y's are either very close to lying in V_B or very close to lying in V_S), this is not compatible with making polynomial progress in each step. In this other case, we need to use a subtly different method for finding C in order to minimize the contribution of these off-diagonal terms. Furthermore, unlike in [LSW00], the matrices C used might have large numerical complexity (perhaps on the order of the complexity of A). If we naively apply the iterative algorithm as is, it might lead to computations involving matrices with exponentially large bit complexity. In order to fix this, we also need to add a rounding step, whereby in each stage we reduce the numerical complexity of A down to some manageable level but without substantially affecting our potential.

Comparison to Other Data Transformations The Forster transform is one of several data transformations that have been studied in the literature to make a dataset "well-conditioned". Here we explain two similar in spirit such transformations, namely the "outlier removal" technique [BFKV96, DV04a] and the rescaling method of [DV04b]. Both of these techniques have been used to obtain weakly polynomial learners for halfspaces with random noise.

The "outlier-removal" technique was introduced in [BFKV96] and was significantly refined by Dunagan and Vempala [DV04a]. Given a dataset X and a parameter $\beta > 0$, a point in X is called a β-outlier if there exists a direction v such that the squared length of x along v is more than β times the average squared length of X along v. The goal of the method is to efficiently find a large subset of $X' \subseteq X$ such that X' has no β-outliers, for as small β as possible. This would give a reasonable sized sub-distribution on which the desired anti-concentration holds. As shown in [DV04a], the parameter β (which affects the quality of the resulting anti-concentration) needs to scale polynomially with the bit complexity b of the dataset X. Consequently, the resulting runtimes in applications of this method will be *inherently weakly polynomial*. Interestingly, this is the reason that the (random noise tolerant) halfspace learner of [BFKV96] is only weakly polynomial.

A different algorithm for learning halfspaces with random classification noise is implicit in the *rescaled perceptron* algorithm of Dunagan and Vempala [DV04b] for efficiently solving linear programs (see also [Bet04]). The key ingredient of their approach is a rescaling step that linearly transforms the data so that, roughly speaking, the margin increases in each iteration by a factor of $1 + 1/d$. Since the initial margin scales with the bit complexity, so does the total number of iterations. (Since this leads to a proper learning algorithm, a dependence on the bit complexity is expected; otherwise, one would obtain a strongly polynomial algorithm for LP!)

Strongly Polynomial Special Cases of LP A line of work, starting in the 80s, has developed strongly polynomial time algorithms for interesting special cases of LP, including minimum cost circulations [Tar85, GT89, Orl93], min cost flow and multi-commodity flow problems [Tar86, VY96], and generalized flow maximization [Vég14, OV17, OV20] (see also [DHNV20, DNV20]). Strongly polynomial time algorithms have also been developed for certain structured convex programs, see, e.g. [Vég16, GV19] in the context of equilibrium computation, and [LSW00] for matrix scaling.

1.6 Organization

The structure of this paper is as follows: In Section 2, we record basic notation and facts that will be used throughout this paper. Section 3 presents our Forster decomposition algorithm, assuming exact eigendecomposition and ignoring bit complexity issues. Section 4 establishes our strongly polynomial guarantees for approximate eigendecomposition. Section 5 shows that we can efficiently round the entries of the underlying matrix without losing much in the desired guarantees. Finally, Section 6 puts all the pieces together to obtain our strongly polynomial Forster algorithm. Section 7 presents our strongly polynomial halfspace learning algorithm. Finally, in Section 8 we summarize our results and provide directions for future work.

1.7 Acknowledgements

We would like to thank Ravi Kannan, Santosh Vempala, and Mihalis Yannakakis for encouragement and insightful conversations about this work. We are grateful to Daniel Dadush for sharing his expertise on optimization, and for detailed feedback that improved the presentation of this paper. We are indebted to Nikhil Srivastava for answering our questions about the complexity of eigenvalue decomposition, and for technical correspondence regarding our strongly polynomial eigendecomposition routine.

2 Preliminaries

Here we introduce some terminology and establish a basic technical fact (Fact 2.1) that will be used throughout this paper.

Basic Notation We use $\mathbb{Z}_+$ to denote the non-negative integers, $\mathbb{R}^d$ for the d-dimensional real coordinate space, $\mathbb{R}^d_*$ for $\mathbb{R}^d \setminus \{\mathbf{0}\}$, and $\mathbb{S}_d$ for the unit ℓ_2-sphere. For a set $S \subset \mathbb{R}$, we will denote $\max(S) \stackrel{\text{def}}{=} \max_{x \in S} x$ and $\min(S) \stackrel{\text{def}}{=} \min_{x \in S} x$.

For $x \in \mathbb{R}^d$, we use $\|x\|_2$ to denotes the ℓ_2-norm of x. We use $\operatorname{tr}(\cdot)$, $\|\cdot\|_F$, and $\|\cdot\|_2$ for the trace, Frobenius norm, and spectral norm of a square matrix. For matrices $A, B \in \mathbb{R}^{d\times d}$, we write $A \succeq B$ (or $B \preceq A$) to denote that $A - B$ is positive semidefinite (PSD). We use I for the $d \times d$ identity matrix, where the dimension will be clear from the context. If $M \in \mathbb{R}^{d\times d}$ is a PSD matrix, we denote by $\lambda_i(M)$ and $q_i(M)$ the i-th largest eigenvalue and corresponding eigenvector of M. That is, $\lambda_1(M) \geq \lambda_2(M) \geq \ldots \geq \lambda_d(M) \geq 0$ and $Mq_i(M) = \lambda_i(M)q_i(M)$ for all $i \in [d]$. We denote by $\Lambda(M)$ the set of eigenvalues of M and $Q(M)$ the set of eigenvectors. That is, $\Lambda(M) = \{\lambda_i(M), i \in [d]\}$ and $Q(M) = \{q_i(M), i \in [d]\}$. For $S \subseteq [d]$, we denote $\Lambda_S(M) = \{\lambda_i(M), i \in S\}$ and $Q_S(M) = \{q_i(M), i \in S\}$.

For a finite set of vectors $S \subset \mathbb{R}^d$, we use $\operatorname{span}(S)$ for their span. For a subspace $V \subset \mathbb{R}^d$, we use $\dim(V)$ for its dimension and $V^\perp$ for its orthogonal complement. For $x \in \mathbb{R}^d$ and a subspace V, we will denote by $\operatorname{proj}_V x$ the projection of x onto V. If $V = \operatorname{span}(S)$, we will sometimes use $\operatorname{proj}_S x$ to denote $\operatorname{proj}_V x$. For conciseness, we sometimes use $x^{(V)}$ for $\operatorname{proj}_V x$. We denote by I_V the $d \times d$ matrix with eigenvalues 1 in V and 0 in $V^\perp$ (the projection of I onto V).

Additional Notation and Basic Fact For a dataset $X \subset \mathbb{R}^d_*$ of size $|X| = n$ and a linear transformation $A \in \mathbb{R}^{d\times d}$, let $f_A : \mathbb{R}^d_* \to \mathbb{S}_d$ be defined by $f_A(x) \stackrel{\text{def}}{=} \frac{Ax}{\|Ax\|_2}$. We aim to find an invertible $A \in \mathbb{R}^{d\times d}$ such that f_A brings a given dataset X in (approximate) radial isotropic position. We denote $f_A(X) \stackrel{\text{def}}{=} \left\{\frac{Ax}{\|Ax\|_2} \mid x \in X\right\}$ We will use various "covariance-like" matrices for the initial dataset X and its subsets. For $X' \subseteq X$, we denote $M_A(X') \stackrel{\text{def}}{=} (1/n)\sum_{x\in X'} f_A(x)f_A(x)^\top$. For subspaces $V_1, V_2 \subset \mathbb{R}^d$ and $X' \subseteq X$, we denote by $M_A^{V_1,V_2}(X') \stackrel{\text{def}}{=} (1/n)\sum_{x\in X'} f_A^{(V_1)}(x) f_A^{(V_2)}(x)^\top$, where we used the shorthand notation $y^{(V)} = \operatorname{proj}_V y$. Note that the normalization factor is fixed in both cases. We start by recording some useful properties of the transformation f_A.

Fact 2.1. *Let $A, B \in \mathbb{R}^{d\times d}$ be full-rank matrices. For any $x \in \mathbb{R}^d_*$ and $a \in \mathbb{R}_*$, the following hold:*

(a) $f_{aA}(x) = f_A(ax) = f_A(x)$.

(b) $f_{BA}(x) = f_B(f_A(x))$.

(c) For $B \succeq I$, we have that $\|f_{BA}(x) - f_A(x)\|_2 \leq \|B - I\|_2$.

(d) Let $V \subseteq \mathbb{R}^d$ be a subspace and let $B = I + aI_V$ for some $a > 0$. Then, $f_{BA}^{(V)}(x) = \lambda(x)\, f_A^{(V)}(x)$, where $1 \leq \lambda(x) \leq 1 + a$, and $f_{BA}^{(V^\perp)}(x) = \mu(x)\, f_A^{(V^\perp)}(x)$, where $\frac{1}{1+a} \leq \mu(x) \leq 1$.

See Appendix A for the simple proof.

3 Approximate Forster Transform in Strongly Polynomial Time

In this section, we describe and analyze our algorithm that either computes an approximate Forster transform of a given dataset or certifies that no Forster transform exists. There are two technical caveats in the algorithm presented in this section: First, we assume the existence of exact routines for matrix eigendecomposition. Second, we do not bound the bit complexity of the associated numbers. Both of these technical issues are handled in subsequent sections.

3.1 Algorithm Pseudocode

The algorithm aims to find a matrix $A \in \mathbb{R}^{d\times d}$ such that the transformation $f_A : \mathbb{R}^d \to \mathbb{R}^d$ brings the set X in (approximate) radial isotropic position. Starting from the initial guess $A = I$, the algorithm iteratively improves the current matrix A until the desired approximation is obtained or a proper subspace W of $\mathbb{R}^d$ is found such that $|X \cap W|/n \geq \dim(W)/d$.

Algorithm 1 Main algorithm for computing Forster Transform

```
1: function ForsterTransform (set $X \subset \mathbb{R}^d_*$ of $n$ points, accuracy parameter $\epsilon$)
2:     Let $A \leftarrow I$        ▷ Initialization of transformation matrix $A$
3:     $M_A \leftarrow M_A(X) = (1/n)\sum_{x\in X} f_A(x)f_A(x)^\top$
4:     while $\|M_A\|_F^2 > \frac{1}{d} + \frac{\epsilon^2}{d^2}$ do
5:         Set $A \leftarrow$ ImproveTransform$(A, X)$
6:         Set $M_A \leftarrow M_A(X) = (1/n)\sum_{x\in X} f_A(x)f_A(x)^\top$
7:     return $A$
```

3.2 Analysis of Algorithm 1

Our Potential Function Our algorithm measures the improvements between consecutive iterations using the potential function

$$\Phi_X(A) \overset{\text{def}}{=} \|M_A\|_F^2 \tag{1}$$

corresponding to the squared Frobenius norm of the matrix

$$M_A \overset{\text{def}}{=} M_A(X) \overset{\text{def}}{=} (1/n)\sum_{x\in X} f_A(x)f_A(x)^\top .$$

Recall that approximate radial isotropy condition amounts to the condition $\frac{1-\epsilon}{d}I \preceq M_A \preceq \frac{1+\epsilon}{d}I$. Equivalently, we want that $\|M_A - \frac{1}{d}I\|_2 \leq \frac{\epsilon}{d}$ or that the eigenvalues of M_A lie in $[\frac{1-\epsilon}{d}, \frac{1+\epsilon}{d}]$. This is guaranteed to hold when the potential function becomes less than $1/d + \epsilon^2/d^2$, as shown in the following lemma.

Lemma 3.1. *Consider any dataset $X \subseteq \mathbb{R}^d_*$ and any full-rank matrix $A \in \mathbb{R}^{d\times d}$. The following properties hold for the potential $\Phi_X(A) = \|M_A\|_F^2$.*

1. $1/d \leq \Phi_X(A) \leq 1$.
2. *If $\Phi_X(A) \leq 1/d + \epsilon^2/d^2$ for some $\epsilon \in (0, 1)$, then for every eigenvalue λ of M_A it holds that $|\lambda - 1/d| \leq \epsilon/d$.*
3. *If $\Phi_X(A) > 1/d + \epsilon^2/d^2$ for some $\epsilon \in (0, 1)$, then (a) there exists an eigenvalue λ of M_A such that $|\lambda - 1/d| > \epsilon/d^2$ and (b) there exists a pair of consecutive eigenvalues λ_i and λ_{i+1} of M_A such that $\lambda_i - \lambda_{i+1} > \epsilon/d^3$.*

Algorithm 2 Find Improved Transform Matrix

```
1: function ImproveTransform (current matrix $A \in \mathbb{R}^{d\times d}$, $X \subset \mathbb{R}^d_*$, accuracy parameter $\epsilon$)
2:     Set $M_A \leftarrow M_A(X) = (1/n)\sum_{x\in X} f_A(x) f_A(x)^\top$
3:     Compute the set of eigenvalues, $\Lambda = \Lambda(M_A)$, and eigenvectors, $Q = Q(M_A)$, of $M_A$.
4:     Set $\gamma \leftarrow O(\frac{\epsilon^2}{d^4 n^2})$, where $n := |X|$
5:     Partition $(\Lambda, Q)$ into two sets of eigenvalues and corresponding eigenvectors, $(\Lambda_{\mathrm{B}}, Q_{\mathrm{B}})$ and
       $(\Lambda_{\mathrm{S}}, Q_{\mathrm{S}})$, maximizing $\min(\Lambda_{\mathrm{B}}) - \max(\Lambda_{\mathrm{S}})$.
           ▷ Consider the Following Two Cases
6:     if there exists $x \in X$ such that $\left\|\mathrm{proj}_{Q_{\mathrm{B}}} f_A(x)\right\|_2, \left\|\mathrm{proj}_{Q_{\mathrm{S}}} f_A(x)\right\|_2 \geq \gamma$ then
7:         Set $U \leftarrow \mathrm{span}(Q_{\mathrm{S}})$.
8:         Set $\alpha \leftarrow \frac{\epsilon}{8nd^3}$.
9:     else
10:        Set $X^{\mathrm{B}} \leftarrow \{x \in X : \|\mathrm{proj}_{Q_{\mathrm{B}}} f_A(x)\|_2 \geq \gamma\}$.
11:        Set $M_A^{\mathrm{B}} \leftarrow M_A(X^{\mathrm{B}}) \overset{\mathrm{def}}{=} (1/n)\sum_{x\in X^{\mathrm{B}}} f_A(x) f_A(x)^\top$.
12:        Let $Q_{\mathrm{b}}$ and $Q_{\mathrm{s}}$ be the sets of top $|Q_{\mathrm{B}}|$ and bottom $|Q_{\mathrm{S}}|$ eigenvectors of $M_A^{\mathrm{B}}$ respectively.
13:        Set $U \leftarrow \mathrm{span}(Q_{\mathrm{s}})$.
14:        Set $\beta \leftarrow \max_{x\in X^{\mathrm{B}}} \|f_A^{(U)}(x)\|_2$.
15:        if $\beta = 0$ then
           ▷ No Forster Transform Exists
16:            Output the subspace $\mathrm{span}(Q_{\mathrm{b}})$.
17:        else        ▷ Case where $\beta > 0$
18:            Set $\alpha \leftarrow \frac{1}{\beta}\epsilon/(3d^2 n) - 1$
19:    return $A' := (I + \alpha I_U)\, A$
```

Proof. Note that for any set X the matrix M_A is PSD (as an autocorrelation matrix). Let $\lambda_i = \lambda_i(M_A)$, $i \in [d]$, with $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d \geq 0$, be its eigenvalues. Then we have that

$$\sum_{i=1}^{d} \lambda_i = \mathrm{tr}(M_A) = (1/n) \sum_{x\in X} \mathrm{tr}\left(f_A(x) f_A(x)^\top\right) = 1 ,$$

where we used the linearity of the trace and the fact that all points $f_A(x)$ have unit ℓ_2-norm. Moreover, for the squared Frobenius norm of M_A it holds that $\|M_A\|_F^2 = \mathrm{tr}(M_A^2) = \sum_{i=1}^d \lambda_i^2$. Below we prove each of the stated properties.

1. Given that $\sum_{i=1}^d \lambda_i = 1$, the maximum possible value of $\|M_A\|_F^2 = \sum_{i=1}^d \lambda_i^2$ is equal to 1, and the minimum possible value is equal to $1/d$ (which is achieved when $\lambda_i = 1/d$, for all $i \in [d]$, i.e., when $M_A = (1/d)I$, as desired).

2. Since $\sum_{i=1}^d \lambda_i = 1$, it holds that $\sum_{i=1}^d \lambda_i^2 = 1/d + \sum_{i=1}^d (\lambda_i - 1/d)^2$. By the assumed upper bound on $\Phi_X(A)$, we get that $\sum_{i=1}^d (\lambda_i - 1/d)^2 \leq \epsilon^2/d^2$, and thus $\max_{i\in[d]} |\lambda_i - 1/d| \leq \epsilon/d$.

3. By the assumed lower bound on $\Phi_X(A)$, we get that $\sum_{i=1}^d (\lambda_i - 1/d)^2 > \epsilon^2/d^2$. By an averaging argument, there exists $j \in [d]$ such that with $(\lambda_j - 1/d)^2 > \epsilon^2/d^3$, and thus $|\lambda_j - 1/d| > \epsilon/d^2$. This proves (a). Since $\lambda_1 \geq \max\{\lambda_j, 1/d\}$ and $\lambda_d \leq \min\{\lambda_j, 1/d\}$, it follows that $\lambda_1 - \lambda_d > \epsilon/d^2$. This implies that there is a gap between consecutive eigenvalues of M_A, namely there exists $i \in [d-1]$ such that $\lambda_i - \lambda_{i+1} > \epsilon/d^3$. This proves (b).

□

Bounding the Decrease in Potential We now proceed with the analysis. We show that the algorithm ImproveTransform either correctly determines that no Forster transform exists or computes a transformation matrix with significantly reduced potential value. This statement implies correctness and simultaneously allows us to bound the running time of our algorithm.

The main result of this section is the following proposition.

Proposition 3.2. *Let $A \in \mathbb{R}^{d\times d}$ be a full-rank matrix and X be a set of n points in $\mathbb{R}^d_*$ such that $\Phi_X(A) > \frac{1}{d} + \frac{\epsilon^2}{d^2}$ for some $\epsilon \in (0,1)$. The algorithm* ImproveTransform *returns a matrix A' such that*

$$\Phi_X(A) - \Phi_X(A') \geq \Omega(\epsilon^5/(n^5 d^{11})) \tag{2}$$

or correctly determines that no Forster Transform of X exists, in which case it returns a subspace W such that $|X \cap W| > (n/d)\dim(W)$.

In the rest of this section, we provide a proof of Proposition 3.2.

Assuming that a Forster transform of X exists, the algorithm ImproveTransform returns the matrix $A' = (I + \alpha I_V)A$, where $V \subset \mathbb{R}^d$ is an appropriate proper subspace of $\mathbb{R}^d$ and $\alpha \in \mathbb{R}_{>0}$ is a carefully selected parameter (that depends on the structure of the dataset X). The algorithm distinguishes two cases: In the first case, α is a small positive quantity, equal to $\epsilon/(8nd^3)$, see Line 8 in Algorithm 2. In the second case, α is set to $\frac{1}{\beta}\epsilon/(3d^2n) - 1$, and can be significantly larger than 1 as it depends on a small parameter β which is a function of the dataset X. See Line 18 in Algorithm 2.

3.2.1 A Useful Structural Result

We will use the notation $M_A = M_A(X)$ and $M_{A'} = M_{A'}(X)$. To bound the desired quantity, $\Phi_X(A) - \Phi_X(A') = \|M_A\|_F^2 - \|M_{A'}\|_F^2$, we will make essential use of the following key lemma:

Lemma 3.3. *For any $X \subset \mathbb{R}^d_*$ and any full-rank matrix $A \in \mathbb{R}^{d\times d}$ the following holds. For any subspace $V \subset \mathbb{R}^d$ and any scalar $\alpha > 0$, for $A' \overset{def}{=} (I + \alpha I_V)A$, we have that*

$$\Phi_X(A) - \Phi_X(A') \geq 2\left(\lambda_k(M_A^{V^\perp,V^\perp}) - \lambda_1(M_A^{V,V}) - 2D_f\right)D_f - 2\|M_{A'}^{V,V^\perp}\|_F^2\,, \tag{3}$$

where $k = \dim(V^\perp)$ and $D_f \overset{def}{=} \frac{1}{n}\sum_{x\in X}\left(\|f_{A'}^{(V)}(x)\|_2^2 - \|f_A^{(V)}(x)\|_2^2\right)$.

Lemma 3.3 bounds the improvement in potential in terms of two opposing contributions. On the one hand, there is a decrease in the potential proportional to the amount of mass D_f transferred from the subspace $V^\perp$ to the subspace V times the eigenvalue gap between the subspaces V and $V^\perp$. On the other hand, there is an increase in potential due to the cross terms $V \times V^\perp$ that get created after the transformation by A'.

Proof of Lemma 3.3. By definition, we have that $\Phi_X(A) - \Phi_X(A') = \|M_A\|_F^2 - \|M_{A'}\|_F^2$. We decompose each matrix into block matrices specified by the subspaces V and $V^\perp$. We write M_A as $M_A^{V,V} + M_A^{V,V^\perp} + M_A^{V^\perp,V} + M_A^{V^\perp,V^\perp}$, where we recall that for subspaces S and T we defined $M_A^{S,T}$ as $I_S M_A I_T$. Since V and $V^\perp$ are orthogonal subspaces, we also have that

$$\|M_A\|_F^2 = \|M_A^{V,V}\|_F^2 + \|M_A^{V,V^\perp}\|_F^2 + \|M_A^{V^\perp,V}\|_F^2 + \|M_A^{V^\perp,V^\perp}\|_F^2.$$

We can thus express $\|M_A\|_F^2 - \|M_{A'}\|_F^2$ as the sum of the following three terms:

(i) $\|M_A^{V,V}\|_F^2 - \|M_{A'}^{V,V}\|_F^2$

(ii) $\|M_A^{V^\perp,V^\perp}\|_F^2 - \|M_{A'}^{V^\perp,V^\perp}\|_F^2$

(iii) $2\|M_A^{V,V^\perp}\|_F^2 - 2\|M_{A'}^{V,V^\perp}\|_F^2$

By Fact 2.1 part (d), for any $x \in \mathbb{R}^d_*$, we have that $f_{A'}^{(V)}(x)(f_{A'}^{(V)}(x))^\top \succeq f_A^{(V)}(x)(f_A^{(V)}(x))^\top$ and $f_{A'}^{(V^\perp)}(x)(f_{A'}^{(V^\perp)}(x))^\top \preceq f_A^{(V^\perp)}(x)(f_A^{(V^\perp)}(x))^\top$. Since $M_A^{V,V}$ is equal to $(1/n)$ times the sum of $f_A^{(V^\perp)}(x)(f_A^{(V^\perp)}(x))^\top$ over $x \in X$, we obtain $M_{A'}^{V,V} \succeq M_A^{V,V}$ and $M_{A'}^{V^\perp,V^\perp} \preceq M_A^{V^\perp,V^\perp}$.

We will use the following linear-algebraic fact to bound from below the first two terms.

Fact 3.4. *Let $A, B \in \mathbb{R}^{d\times d}$ be symmetric PSD matrices. If $A \succeq B$, then it holds that*

$$2\,\mathrm{tr}(A-B)\lambda_k(B) \le \|A\|_F^2 - \|B\|_F^2 \le 2\mathrm{tr}(A-B)\,\lambda_1(A)\ ,$$

where $k = \mathrm{rank}(A)$ and $\lambda_k(B)$ is the k-th largest eigenvalue of B.

Proof. We use $\bullet$ to denote the entrywise inner product between two matrices. Note that $A \bullet B = \mathrm{tr}(A^\top B)$ which is equal to $\mathrm{tr}(AB)$ if the matrices are symmetric. We have that

$$\|A\|_F^2 - \|B\|_F^2 = (A-B)\bullet(A+B) = \mathrm{tr}((A+B)(A-B)).$$

Since both $A+B$ and $A-B$ are PSD, multiplication by $A+B$ increases the eigenvalues of $A-B$ by at least a factor of $2\lambda_k(B)$ and at most $2\lambda_1(A)$. Thus, $\mathrm{tr}((A+B)(A-B))$ is bounded between $2\mathrm{tr}(A-B)\lambda_k(B)$ and $2\mathrm{tr}(A-B)\lambda_1(A)$. □

We start by bounding term (i) from below. Using Fact 3.4 applied to the matrices $M_{A'}^{V,V}$ and $M_A^{V,V}$, we get that

$$\begin{aligned}\|M_A^{V,V}\|_F^2 - \|M_{A'}^{V,V}\|_F^2 &\ge 2\lambda_1(M_{A'}^{V,V})\,\mathrm{tr}(M_A^{V,V} - M_{A'}^{V,V})\\ &= \frac{2}{n}\lambda_1(M_{A'}^{V,V})\sum_{x\in X}\left(\|f_A^{(V)}(x)\|_2^2 - \|f_{A'}^{(V)}(x)\|_2^2\right)\ . \end{aligned} \tag{4}$$

Similarly, we can bound below term (ii). Using Fact 3.4 applied to the matrices $M_A^{V^\perp,V^\perp}$ and $M_{A'}^{V^\perp,V^\perp}$, we get that

$$\begin{aligned}\|M_A^{V^\perp,V^\perp}\|_F^2 - \|M_{A'}^{V^\perp,V^\perp}\|_F^2 &\ge 2\lambda_k(M_{A'}^{V^\perp,V^\perp})\mathrm{tr}(M_A^{V^\perp,V^\perp} - M_{A'}^{V^\perp,V^\perp})\\ &= \frac{2}{n}\lambda_k(M_{A'}^{V^\perp,V^\perp})\sum_{x\in X}\left(\|f_A^{(V^\perp)}(x)\|_2^2 - \|f_{A'}^{(V^\perp)}(x)\|_2^2\right)\\ &= \frac{2}{n}\lambda_k(M_{A'}^{V^\perp,V^\perp})\sum_{x\in X}\left(\|f_{A'}^{(V)}(x)\|_2^2 - \|f_A^{(V)}(x)\|_2^2\right)\ , \end{aligned} \tag{5}$$

where $k = \mathrm{rank}(M_{A'}^{V^\perp,V^\perp})$ and the last equality follows since $\|f_A^{(V)}(x)\|_2^2 + \|f_A^{(V^\perp)}(x)\|_2^2 = 1$.
Finally, we straightforwardly bound from below the third term as follows:

$$2\|M_A^{V,V^\perp}\|_F^2 - 2\|M_{A'}^{V,V^\perp}\|_F^2 \ge -2\|M_{A'}^{V,V^\perp}\|_F^2\ . \tag{6}$$

Overall, recalling that $D_f = \frac{1}{n}\sum_{x\in X}(\|f_{A'}^{(V)}(x)\|_2^2 - \|f_A^{(V)}(x)\|_2^2)$, (4), (5), (6) give that

$$\Phi_X(A) - \Phi_X(A') \ge 2\left(\lambda_k(M_{A'}^{V^\perp,V^\perp}) - \lambda_1(M_{A'}^{V,V})\right)D_f - 2\|M_{A'}^{V,V^\perp}\|_F^2\ .$$

To complete the proof, we note that

$$\lambda_1(M_{A'}^{V,V}) \leq \lambda_1(M_A^{V,V}) + \|M_A^{V,V} - M_{A'}^{V,V}\|_2$$

and that

$$\lambda_k(M_{A'}^{V^\perp,V^\perp}) \leq \lambda_k(M_{A'}^{V^\perp,V^\perp}) + \|M_A^{V^\perp,V^\perp} - M_{A'}^{V^\perp,V^\perp}\|_2 = \lambda_k(M_{A'}^{V^\perp,V^\perp}) + \|M_A^{V,V} - M_{A'}^{V,V}\|_2$$

Finally, we have that

$$\begin{aligned}\|M_A^{V,V} - M_{A'}^{V,V}\|_2 &\leq \frac{1}{n}\sum_{x\in X} \left\| f_{A'}^{(V)}(x) f_{A'}^{(V)}(x)^\top - f_A^{(V)}(x) f_A^{(V)}(x)^\top \right\|_2 \\ &= \frac{1}{n}\sum_{x\in X}\left(\|f_{A'}^{(V)}(x)\|_2^2 - \|f_A^{(V)}(x)\|_2^2\right) ,\end{aligned}$$

where the last equality follows from Fact 2.1(d). Combining the above completes the proof of Lemma 3.3. □

In the following two subsections, we analyze the two cases of ImproveTransform separately. Note that ImproveTransform requires that we be able to do exact singular value decompositions in order to compute the subspace U. Our final algorithm will not be able to do this exactly and will need to make do with an approximate singular value decomposition (see Section 4). In order to make our extension easier, we will show that the potential decrease holds even when U is replaced by some V which satisfies some approximation of the properties that U does.

3.2.2 Case I: There exists $x \in X$ such that $\|\mathrm{proj}_{Q_\mathrm{B}} f_A(x)\|_2, \|\mathrm{proj}_{Q_\mathrm{S}} f_A(x)\|_2 \geq \gamma$

In order to analyze this case, we prove the following proposition:

Proposition 3.5. *Suppose that X is a set of n points in $\mathbb{R}_*^d$ and A an invertible $d \times d$ matrix. Suppose that $V \subset \mathbb{R}^d$ is a subspace so that for $\alpha, \rho > 0$ with $\alpha \leq \epsilon/(64nd^3)$:*

1. *The maximum over $x \in X$ of $\min(\|f_A^{(V)}(x)\|_2, \|f_A^{(V^\perp)}(x)\|_2)$ equals ρ.*
2. *$\lambda_{\min}(M_A^{V^\perp V^\perp}(X)) - \lambda_{\max}(M_A^{VV}(X)) \geq \frac{\epsilon}{2d^3}$.*
3. *$\|M_A^{VV^\perp}(X)\|_F \leq \alpha\rho$.*

Then for $C = (I + \alpha I_V)A$ we have that $\Phi_X(C) \leq \Phi_X(A) - \rho^2\epsilon/(8nd^3)$.

We note that if $\Phi_X(A) > \frac{1}{d} + \frac{\epsilon^2}{d^2}$, then by Lemma 3.1 part 3, the difference between the largest and smallest eigenvalues of $M_A(X)$ will be at least ϵ/d^2, and therefore the largest eigenvalue gap will be at least ϵ/d^3. Thus, for V taken to be the U given in Algorithm 2, Property 2 will hold. Furthermore, for as U is an eigenspace of $M_A(X)$, $M_A^{U,U^\perp}(X) = \mathbf{0}$ and Property 3 will hold.

The rest of this section will be devoted to proving Proposition 3.5.

To bound below the improvement in potential, we will make essential use of Lemma 3.3. We bound the relevant quantities in the following lemmas.

Lemma 3.6. *Letting $D_f = \frac{1}{n}\sum_{x\in X}\left(\|f_C^{(V)}(x)\|_2^2 - \|f_A^{(V)}(x)\|_2^2\right)$, we have that $\frac{\alpha\rho^2}{2n} \leq D_f \leq 2\alpha$.*

Proof. By Fact 2.1(c), it follows that $D_f \leq 2\alpha$, as $C = (I + \alpha I_V)A$ and $\|(I + \alpha I_V) - I\|_2 \leq \alpha$. This implies that $\|f_C^{(V)}(x) - f_A^{(V)}(x)\|_2 \leq \|f_C(x) - f_A(x)\|_2 \leq \alpha$ for all x. Thus (since $x \to x^2$ is 2-Lipschitz on $[0, 1]$), we have that $\|f_C^{(V)}(x)\|_2^2 - \|f_A^{(V)}(x)\|_2^2 \leq 2\alpha$ for all x, so $D_f \leq 2\alpha$.

To bound D_f from below, consider an $x^* \in X$ that maximizes the quantity

$$\min\{\|\mathrm{proj}_{Q_{\mathrm{B}}} f_A(x)\|_2, \|\mathrm{proj}_{Q_{\mathrm{S}}} f_A(x)\|_2\}$$

over $x \in X$. Recall that the maximum value of the above quantity equals ρ.

By assumption, we must have that $\rho \geq \gamma$. As Fact 2.1(d) implies that all terms in the sum defining D_f are nonnegative, we have that

$$D_f \geq \frac{1}{n}\left(\|f^{(V)}_{(I+\alpha I_V)A}(x^*)\|_2^2 - \|f_A^{(V)}(x^*)\|_2^2\right) .$$

Let $y = f_A(x^*)$, and recall that we use $y^{(V)} = f_A^{(V)}(x^*)$ and $y^{(V^\perp)} = f_A^{(V^\perp)}(x^*)$ for the projections of y onto V and $V^\perp$ respectively. With this notation, we can write

$$D_f \geq \frac{1}{n}\left(\|f^{(V)}_{I+\alpha I_V}(y)\|_2^2 - \|y^{(V)}\|_2^2\right) = \frac{1}{n}\left(\frac{(1+\alpha)^2\|y^{(V)}\|_2^2}{(1+\alpha)^2\|y^{(V)}\|_2^2 + \|y^{(V^\perp)}\|_2^2} - \|y^{(V)}\|_2^2\right) ,$$

where we used Fact 2.1(b) and the definition of the transformation f_A. The Pythagorean theorem and the definition of y give that $\|y^{(V)}\|_2^2 + \|y^{(V^\perp)}\|_2^2 = \|y\|_2^2 = 1$. We thus obtain

$$D_f \geq \frac{1}{n}\left(\frac{2\alpha\,\|y^{(V)}\|_2^2\,\|y^{(V^\perp)}\|_2^2}{(1+\alpha)^2}\right) \geq \frac{2\alpha\rho^2(1-\rho^2)}{(1+\alpha)^2 n} \geq \frac{\alpha\rho^2}{2n} \geq \frac{\alpha\gamma^2}{2n} ,$$

where the last inequality follows since α is sufficiently smaller than 1 and $\rho^2 \leq \frac{1}{2}$. □

Finally, we bound $\|M_C^{V,V^\perp}\|_F^2$ from above in the following lemma:

Lemma 3.7. *We have that* $\|M_C^{V,V^\perp}\|_F^2 \leq 4\alpha^2\rho^2$.

Proof. By the triangle inequality for the Frobenius norm, we have that

$$\|M_C^{V,V^\perp}\|_F^2 \leq \left(\|M_A^{V,V^\perp}\|_F + \|M_C^{V,V^\perp} - M_A^{V,V^\perp}\|_F\right)^2 .$$

We have that $\|M_A^{V,V^\perp}\|_F \leq \alpha\rho$, by assumption. We bound above the second term using the following sequence of inequalities:

$$\begin{aligned}
\|M_C^{V,V^\perp} - M_A^{V,V^\perp}\|_F^2 &= \left\|\frac{1}{n}\sum_{x\in X}\left(f_C^{(V)}(x) f_C^{(V)^\perp}(x)^\top - f_A^{(V)}(x) f_A^{(V^\perp)}(x)^\top\right)\right\|_F^2 \\
&\leq \max_{x\in X}\left\|f_C^{(V)}(x) f_C^{(V^\perp)}(x)^\top - f_A^{(V)}(x) f_A^{(V^\perp)}(x)^\top\right\|_F^2 \\
&= \max_{x\in X}\left\|\frac{1+\alpha}{\|(1+\alpha)f_A^{(V)}(x)\|^2 + \|f_A^{(V^\perp)}(x)\|^2} f_A^{(V)}(x) f_A^{(V^\perp)}(x)^\top - f_A^{(V)}(x) f_A^{(V^\perp)}(x)^\top\right\|_F^2 \\
&\leq \max_{x\in X}\left\|\alpha f_A^{(V)}(x) f_A^{(V^\perp)}(x)^\top\right\|_F^2 \\
&\leq \alpha^2\rho^2(1-\rho^2) \leq \alpha^2\rho^2 ,
\end{aligned}$$

where the third line uses Fact 2.1(b) and the definition of the transformation f_A. □

Combining the above lemmas, we obtain that

$$\Phi_X(A) - \Phi_X(C) \geq \left(\frac{\epsilon}{2d^3} - 4\alpha\right)\frac{\alpha\rho^2}{n} - 8\alpha^2\rho^2 .$$

We note that so long as $\alpha \leq \epsilon/(64nd^3)$ the above is at least

$$(\alpha\rho^2)\left(\left(\frac{\epsilon}{4d^3}\right)\frac{1}{n} - 8\alpha\right) \geq \alpha\rho^2\epsilon/(8nd^3).$$

This completes our proof of Proposition 3.5.

3.2.3 Case II: For all $x \in X$, either $\|\mathrm{proj}_{Q_{\mathrm{B}}} f_A(x)\|_2 \leq \gamma$ or $\|\mathrm{proj}_{Q_{\mathrm{S}}} f_A(x)\|_2 \leq \gamma$

In this case, all points $x \in X$ lie within a γ margin from the subspaces spanned by the vectors Q_{B} and Q_S. The algorithm updates the matrix A by considering only the set of "big" points X^{B}, i.e., the points in X whose images under f_A have sufficiently large projections on the subspace spanned by the large eigenvectors of M_A. In more detail, instead of using the eigenvectors Q_{B}, Q_S of the matrix $M_A = M_A(X)$, the algorithm uses the eigenvectors Q_b, Q_s of the matrix $M_A(X^{\mathrm{B}}) = (1/n)\sum_{x\in X^{\mathrm{B}}} f_A(x)f_A(x)^\top$, setting $U = \mathrm{span}(Q_s)$. This is done to ensure that the cross-terms $M_A^{U,U^\perp}(X^{\mathrm{B}})$ start out at 0 initially and remain small despite significant rescaling of the subspace U. Moreover, despite the change in the definition, we show (in Claim 3.9 and Claim 3.10) that the corresponding subspaces U and $U^\perp$ satisfy a similar margin condition to the subspaces spanned by Q_{B} and Q_S and that there is still a significant eigenvalue gap between U and $U^\perp$. The margin condition is shown in Claim 3.9 and Claim 3.10, and the eigenvalue bounds are proven in Lemmas 3.12 and 3.11.

These properties will allow us to bound the decrease in potential in this case. We will show the following result.

Proposition 3.8. *Suppose that X is a set of n points in $\mathbb{R}^d_*$ and A an invertible $d \times d$ matrix. Suppose that for some $k < n$ that $\lambda_k(M_A(X)) - \lambda_{k+1}(M_A(X)) \geq \epsilon/(2d^3)$. Let W be the span of the $d-k$ smallest eigenvalues of $M_A(X)$.*

Suppose furthermore that for some γ at most a sufficiently small multiple of $\epsilon^2/(d^4n^2)$ that every $x \in X$ satisfies $\min(\|f_A^{(W)}(x)\|_2, \|f_A^{(W^\perp)}(x)\|_2) \leq \gamma$. Let X^{B} denote the set of $x \in X$ so that $\|f_A^{(W)}(x)\|_2 \leq \gamma$ and $X^S = X \backslash X^{\mathrm{B}}$. Let $0 < \delta < \gamma$. Suppose that $V \subset \mathbb{R}^d$ is a $(d-k)$-dimensional subspace so that:

1. $\mathrm{tr}(M_A^{V,V}(X^{\mathrm{B}})) \leq \mathrm{tr}(M_A^{W,W}(X^{\mathrm{B}})) + \delta^2$,
2. $\mathrm{tr}(M_A^{V^\perp,V^\perp}(X^{\mathrm{B}})) \geq \mathrm{tr}(M_A^{W^\perp,W^\perp}(X^{\mathrm{B}})) - \delta^2$,
3. $\lambda_k(M_A^{V^\perp,V^\perp}(X^{\mathrm{B}})) \geq \lambda_k(M_A(X^{\mathrm{B}})) - \delta$,
4. $\|M_A^{V,V^\perp}(X^{\mathrm{B}})\|_F \leq \beta\delta$,

where $\beta = \max_{x\in X^{\mathrm{B}}} \|f_A^{(V)}(x)\|_2$. Then if $\beta = 0$, $V^\perp$ contains more than kn/d elements of X. Otherwise, setting $\alpha = \epsilon/(3\beta d^2 n) - 1$ and $C = (I + \alpha I_V)A$, we have that

$$\Phi_X(C) \leq \Phi_X(A) - \Omega(\epsilon^3/(d^7n^3)) .$$

We note that if V is taken to be the space of the bottom $d-k$ eigenvalues of $M_A(X^{\mathrm{B}})$ that the above properties trivially hold with $\delta = 0$. Properties 1 and 2 follow from the variational characterization of eigenspaces. Properties 3 and 4 hold trivially.

We know that elements of X^{B} are close to $W^{\perp}$ and elements of X^S are close to W. We will need to claim that elements of X^{B} are also close to $V^{\perp}$ and elements of X^S are close to V. We establish this in the next two claims.

Claim 3.9. *We have that* $\frac{1}{n}\sum_{x \in X^{\mathrm{B}}} \|f_A^{(V)}(x)\|_2^2 \le \gamma^2 + \delta^2$. *In particular, for any* $x \in X^{\mathrm{B}}$, *it holds that* $\|f_A^{(V)}(x)\|_2 \le \sqrt{2n}\gamma$.

Proof. This follows from Property 1 and the fact that

$$\mathrm{tr}(M_A^{W,W}(X^{\mathrm{B}})) = \frac{1}{n}\sum_{x \in X^{\mathrm{B}}} \|f_A^{(W)}(x)\|_2^2 \le \gamma^2.$$

□

Claim 3.10. *We have that* $\frac{1}{n}\sum_{x \in X \setminus X^{\mathrm{B}}} \|f_A^{(V^{\perp})}(x)\|_2^2 \le \gamma^2 + \delta^2$. *In particular, for any* $x \in X \setminus X^{\mathrm{B}}$, *it holds that* $\|f_A^{(V^{\perp})}(x)\|_2 \le \sqrt{2n}\gamma$.

Proof. Recalling that $W^{\perp}$ is the span of the principle eigenvectors of $M_A(X)$, by the variational characterization of eigenspaces, it maximizes the quantity $\mathrm{tr}(M_A^{Z,Z}(X)) = \frac{1}{n}\sum_{y \in f_A(X)} \|y^{(Z)}\|_2^2$ over all subspaces Z with $\dim(Z) = \dim(W^{\perp}) = k$. In particular, it holds that

$$\frac{1}{n}\sum_{y \in f_A(X)} \|y^{(W^{\perp})}\|_2^2 \ge \frac{1}{n}\sum_{y \in f_A(X)} \|y^{(V^{\perp})}\|_2^2 \,.$$

On the other hand, by Property 2, we have that

$$\frac{1}{n}\sum_{y \in f_A(X^{\mathrm{B}})} \|y^{(V^{\perp})}\|_2^2 = \mathrm{tr}(M_A^{V^{\perp},V^{\perp}}(X^{\mathrm{B}})) \ge \mathrm{tr}(M_A^{W^{\perp},W^{\perp}}(X^{\mathrm{B}})) - \delta^2 = \frac{1}{n}\sum_{y \in f_A(X^{\mathrm{B}})} \|y^{(W^{\perp})}\|_2^2 - \delta^2 \,.$$

Subtracting the above two inequalities, we get that

$$\frac{1}{n}\sum_{x \in X \setminus X^{\mathrm{B}}} \|f_A^{(V^{\perp})}(x)\|_2^2 \le \frac{1}{n}\sum_{x \in X \setminus X^{\mathrm{B}}} \|f_A^{(W^{\perp})}(x)\|_2^2 + \delta^2 \le \gamma^2 + \delta^2 \,.$$

This gives the claim. □

The case that $\beta > 0$**:** Here we assume that $\beta > 0$ and show that we can obtain an improvement in our potential function. To bound below the improvement in potential, we will make essential use of Lemma 3.3, and we bound the relevant quantities in a sequence of lemmas.

We begin by bounding below $\lambda_k(M_A^{V^{\perp},V^{\perp}}(X))$. In particular, we show that it is nearly as big as $\lambda_k(M_A(X))$. Morally, this holds because

$$\lambda_k(M_A(X)) = \lambda_k(M_A^{W^{\perp},W^{\perp}}(X)) \approx \lambda_k(M_A^{V^{\perp},V^{\perp}}(X)) \,.$$

Formally, we have the following.

Lemma 3.11. *We have that* $\lambda_k(M_A^{V^{\perp},V^{\perp}}(X)) \ge \lambda_k(M_A(X)) - 4\gamma$, *for* $k = \dim(V^{\perp})$.

Proof. We bound the k-th largest eigenvalue of $M_A^{V^\perp,V^\perp}$ via the following sequence of inequalities.

$$\begin{aligned}
\lambda_k(M_A^{V^\perp,V^\perp}(X)) &\geq \lambda_k(M_A^{V^\perp,V^\perp}(X^{\mathrm{B}})) && (7)\\
&\geq \lambda_k(M_A(X^{\mathrm{B}})) - \gamma && (8)\\
&\geq \lambda_k(M_A^{W^\perp,W^\perp}(X^{\mathrm{B}})) - 3\gamma && (9)\\
&\geq \lambda_k(M_A^{W^\perp,W^\perp}) - 3\gamma - \gamma^2 && (10)\\
&= \lambda_k(M_A) - 3\gamma - \gamma^2 \,. && (11)
\end{aligned}$$

Inequality (7) follows since

$$M_A^{V^\perp,V^\perp}(X) = M_A^{V^\perp,V^\perp}(X^{\mathrm{B}}) + M_A^{V^\perp,V^\perp}(X \setminus X^{\mathrm{B}}) \succeq M_A^{V^\perp,V^\perp}(X^{\mathrm{B}}) \,.$$

Inequality (8) follows from Property 3 and $\delta \leq \gamma$.
Inequality (9) follows since Inequality (10) follows since

$$\|M_A^{W^\perp,W^\perp}(X) - M_A^{W^\perp,W^\perp}(X^{\mathrm{B}})\|_2 = \|M_A^{W^\perp,W^\perp}(X \setminus X^{\mathrm{B}})\|_2.$$

This is at most γ^2 as it is bounded by

$$\sup_{\|v\|_2=1} v^\top \left(\frac{1}{n} \sum_{x \in X^S} f_A(x)^{(W^\perp)} (f_A(x)^{(W^\perp)})^\top \right) v \leq \sup_{\|v\|_2=1} \max_{x \in X^S} |v \cdot f_A(x)^{(W^\perp)}|^2 \leq \gamma^2 \,.$$

Equality (11) follows since $M_A^{W^\perp,W^\perp}$ and M_A agree in the top k eigenvalues by definition of the subspace W, as the span of the top k eigenvectors of M_A.

Lemma 3.11 now follows as $\gamma^2 \leq \gamma \leq 1$. □

Next we bound from above $\lambda_1(M_A^{V,V}(X))$. In particular, we show that it is not much larger than $\lambda_{k+1}(M_A(X))$. Morally, this holds because

$$\lambda_{k+1}(M_A(X)) = \lambda_1(M_A^{W,W}(X)) \approx \lambda_1(M_A^{V,V}(X)) \,.$$

Formally, we have the following.

Lemma 3.12. *We have that* $\lambda_1(M_A^{V,V}(X)) \leq \lambda_{k+1}(M_A(X)) + 8\gamma$.

Proof. We bound the largest eigenvalue of $M_A^{V,V}$ via the following sequence of inequalities.

$$\begin{aligned}
\lambda_1(M_A^{V,V}) &\leq \lambda_1(M_A^{V,V}(X \setminus X^{\mathrm{B}})) + 2\gamma^2 && (12)\\
&\leq \lambda_1(M_A(X \setminus X^{\mathrm{B}})) + 4\gamma + 2\gamma^2 && (13)\\
&\leq \lambda_1(M_A^{W,W}(X \setminus X^{\mathrm{B}})) + 6\gamma + 2\gamma^2 && (14)\\
&\leq \lambda_1(M_A^{W,W}(X)) + 6\gamma + 2\gamma^2 && (15)\\
&\leq \lambda_{k+1}(M_A(X)) + 8\gamma \,. && (16)
\end{aligned}$$

Inequality (12) follows since $\lambda_1(M_A^{V,V}) \leq \lambda_1(M_A^{V,V}(X \setminus X^{\mathrm{B}})) + \lambda_1(M_A^{V,V}(X^{\mathrm{B}}))$ and

$$\lambda_1(M_A^{V,V}(X^{\mathrm{B}})) = \|M_A^{V,V}(X^{\mathrm{B}})\|_2 \leq \frac{1}{n} \sum_{x \in X^{\mathrm{B}}} \|f_A^{(V)}(x)\|_2^2 \leq 2\gamma^2 \,,$$

where the last inequality follows from Claim 3.9 and since $\delta \le \gamma$.
Inequality (13) follows since $\|M_A(X \setminus X^{\mathrm{B}}) - M_A^{V,V}(X \setminus X^{\mathrm{B}})\|_2 \le 4\gamma$. Indeed, note that

$$\|M_A(X \setminus X^{\mathrm{B}}) - M_A^{V,V}(X \setminus X^{\mathrm{B}})\|_2 \le \frac{1}{n} \sum_{x \in X \setminus X^{\mathrm{B}}} \|f_A^{(V)}(x) f_A^{(V)}(x)^\top - f_A(x) f_A(x)^\top\|_2$$
$$\le \frac{2}{n} \sum_{x \in X \setminus X^{\mathrm{B}}} \|f_A^{(V)}(x) - f_A(x)\|_2 = \frac{2}{n} \sum_{x \in X \setminus X^{\mathrm{B}}} \|f_A^{(V^\perp)}(x)\|_2 \le 4\gamma \ ,$$

where the last inequality follows from Claim 3.10.
Inequality (14) follows since

$$\|M_A^{W,W}(X \setminus X^{\mathrm{B}}) - M_A(X \setminus X^{\mathrm{B}})\|_2 \le \frac{1}{n} \sum_{x \in X \setminus X^{\mathrm{B}}} \|f_A^{(W)}(x) f_A^{(W)}(x)^\top - f_A(x) f_A(x)^\top\|_2$$
$$\le \frac{2}{n} \sum_{x \in X \setminus X^{\mathrm{B}}} \|f_A^{(W)}(x) - f_A(x)\|_2 = \frac{2}{n} \sum_{x \in X \setminus X^{\mathrm{B}}} \|f_A^{(W^\perp)}(x)\|_2 \le 2\gamma \ ,$$

where the last inequality follows from the definition of $X \setminus X^{\mathrm{B}}$.
Inequality (15) follows since $M_A^{W,W}(X) \succeq M_A^{W,W}(X \setminus X^{\mathrm{B}})$.
Inequality (16) follows since $\lambda_1(M_A^{W,W}) = \lambda_{k+1}(M_A(X))$ and $\gamma^2 \le \gamma \le 1$.
This completes the proof of Lemma 3.12. □

Note that together Lemmas 3.11 and 3.12 show that $\lambda_k(M_A^{V^\perp,V^\perp}(X)) - \lambda_1(M_A^{V,V}(X))$ is nearly as large as $\lambda_k(M_A(X)) - \lambda_{k+1}(M_A(X)) \ge \epsilon/(2d^3)$.

Next we bound the off-diagonal terms of the transformed vectors.

Lemma 3.13. *We have that* $\|M_C^{V,V^\perp}\|_F^2 \le ((1+\alpha)^3\beta^3 + (1+\alpha)\beta\delta + 2\gamma)^2$.

Proof. By the triangle inequality, we have that

$$\|M_C^{V,V^\perp}\|_F \le \|M_C^{V,V^\perp}(X^{\mathrm{B}})\|_F + \|M_C^{V,V^\perp}(X \setminus X^{\mathrm{B}})\|_F \ .$$

We use this to bound the contributions from the vectors in X^{B} separately from the contributions from $X \setminus X^{\mathrm{B}}$.

For the contribution from X^{B}, we note that if $x \in X^{\mathrm{B}}$ and $y = f_A(x)$, then one obtains $f_C(x)$ by multiplying the V-part of y by $(1+\alpha)$ and rescaling slightly. Thus, the $V \setminus V^\perp$ component of $f_C(x) f_C(x)^\top$ is roughly $y^{(V)}(1+\alpha)y^{(V^\perp)T}$. Summing over all $y \in f_A(X^{\mathrm{B}})$ gives roughly $M_A^{V,V^\perp}(X^{\mathrm{B}})$, which is small by assumption.

In particular, we have that

$$\|M_C^{V,V^\perp}(X^{\mathrm{B}})\|_F \le \|M_C^{V,V^\perp}(X^{\mathrm{B}}) - (1+\alpha)M_A^{V,V^\perp}(X^{\mathrm{B}})\|_F + (1+\alpha)\beta\delta$$
$$= \left\| \frac{1}{n} \sum_{y \in f_A(X^{\mathrm{B}})} \frac{(1+\alpha) y^{(V)} y^{(V^\perp)T}}{\|y^{(V^\perp)} + (1+\alpha) y^{(V)}\|_2^2} - (1+\alpha)\frac{1}{n} \sum_{y \in f_A(X^{\mathrm{B}})} y^{(V)} y^{(V^\perp)T} \right\|_F + (1+\alpha)\beta\delta$$
$$= \left\| \frac{(1+\alpha)}{n} \sum_{y \in f_A(X^{\mathrm{B}})} \left(\frac{1}{\|y^{(V^\perp)} + (1+\alpha) y^{(V)}\|_2^2} - 1 \right) y^{(V)} y^{(V^\perp)T} \right\|_F + (1+\alpha)\beta\delta$$
$$\le \frac{(1+\alpha)}{n} \sum_{y \in f_A(X^{\mathrm{B}})} \left(1 - \frac{1}{\|y^{(V^\perp)} + (1+\alpha) y^{(V)}\|_2^2} \right) \|y^{(V)} y^{(V^\perp)T}\|_F + (1+\alpha)\beta\delta$$
$$\le ((1+\alpha)\beta)^3 + (1+\alpha)\beta\delta \ .$$

The first inequality follows from Property 4. The second equality follows from the fact that $f_C(x) = f_{I+\alpha I_V}(f_A(x))$, by Fact 2.1(b), and thus for $y = f_A(x)$ this is equal to $\frac{y^{(V)}+\alpha y^{(V^\perp)}}{\|y^{(V)}+(1+\alpha)y^{(V^\perp)}\|_2}$. The last inequality follows since $\|y^{(V)}(y^{(V^\perp)})^\top\|_F \le \beta$ and $\|y^{(V)} + (1+\alpha)y^{(V^\perp)}\|_2^2 \le 1 + (1+\alpha)^2\beta^2$.

To bound the contribution from $X \setminus X^{\mathrm{B}}$, we note that applying $f_{I+\alpha I_V}$ can only decrease the size of the $V^\perp$-component of a vector. Thus, for each $x \in X \setminus X^{\mathrm{B}}$, $f_C(x)$ will have a small $V^\perp$-component. In particular, we have

$$\begin{aligned}
\|M_C^{V,V^\perp}(X\setminus X^{\mathrm{B}})\|_F &= \left\| \frac{1}{n} \sum_{y \in f_C(X\setminus X^{\mathrm{B}})} y^{(V)}(y^{(V^\perp)})^\top \right\|_F \\
&\le \frac{1}{n} \sum_{y \in f_C(X\setminus X^{\mathrm{B}})} \|y^{(V)}(y^{(V^\perp)})^\top\|_F \\
&\le \frac{1}{n} \sum_{y \in f_C(X\setminus X^{\mathrm{B}})} \|(y^{(V^\perp)})^\top\|_2 \le 2\gamma \ ,
\end{aligned}$$

where the last inequality follows since for any $x \in X \setminus X^{\mathrm{B}}$, it holds that $\|f_C^{(V^\perp)}(x)\|_2 \le \|f_A^{(V^\perp)}(x)\|_2$, as follows from Fact 2.1(d).

Combining this with the above proves our lemma. $\square$

Finally, we need to bound D_f, showing that it is neither too big nor too small. This follows by noting that the greatest amount that any vector was modified is on the order of $\alpha\beta$. In particular, we have that:

Lemma 3.14. *We have that* $\frac{1}{n}\left(\frac{(1+\alpha)^2\beta^2}{1+(1+\alpha)^2\beta^2} - \gamma^2\right) \le D_f \le (1+\alpha)^2\beta^2 + 2\gamma^2$*, where*

$$D_f = \frac{1}{n} \sum_{y \in f_C(X)} \|y^{(V)}\|_2^2 - \frac{1}{n} \sum_{y \in f_A(X)} \|y^{(V)}\|_2^2 \ .$$

Proof. By the definition of β, there exists $x^* \in X^{\mathrm{B}}$ with $\|f_A^{(V)}(x^*)\|_2 = \beta$. Such a point satisfies

$$\|f_C^{(V)}(x^*)\|_2^2 = \|f_{I+\alpha I_V}^{(V)}(f_A(x^*))\|_2^2 = \frac{(1+\alpha)^2\|f_A^{(V)}(x^*)\|_2^2}{\|f_A^{(V^\perp)}(x^*)\|_2^2 + (1+\alpha)^2\|f_A^{(V)}(x^*)\|_2^2} = \frac{(1+\alpha)^2\beta^2}{1-\beta^2+(1+\alpha)^2\beta^2}.$$

This point will contribute at least

$$\frac{1}{n}\left(\|f_C^{(V)}(x^*)\|_2^2 - \|f_A^{(V)}(x^*)\|_2^2\right) \ge \frac{1}{n}\left(\frac{(1+\alpha)^2\beta^2}{1+(1+\alpha)^2\beta^2} - \gamma^2\right)$$

to D_f.

Moreover, for any other $x \in X^{\mathrm{B}}$, $\|f_C^{(V)}(x)\|_2^2 \le \|f_C^{(V)}(x^*)\|_2^2$.

Thus, for points $x \in X^{\mathrm{B}}$, we have that

$$0 \le \|f_C^{(V)}(x)\|_2^2 - \|f_A^{(V)}(x)\|_2^2 \le (1+\alpha)^2\beta^2 \ .$$

Finally, for all points $x \in X \setminus X^{\mathrm{B}}$, we have that $0 \le \|f_C^{(V)}(x)\|_2^2 - \|y_A^{(V)}(x)\|_2^2 \le 1 - (1-2\gamma^2) \le 2\gamma^2$. This implies the required bounds. $\square$

Combining the Lemmas 3.11, 3.12, 3.13, 3.14 with Lemma 3.3 and setting $\eta = (1+\alpha)\beta = \epsilon/(3d^2 n)$, we get that

$$\begin{aligned}
&\Phi_X(A) - \Phi_X(C) \geq \\
&\geq 2\left(\lambda_k(M_A(X)) - 4\gamma - \lambda_{k+1}(M_A(X)) - 8\gamma - 2\eta^2 - 4\gamma^2\right)\frac{1}{n}\left(\frac{\eta^2}{1+\eta^2} - \gamma^2\right) - 2(\eta^3 + \eta\delta + 2\gamma)^2 \\
&\geq \left(\frac{\epsilon}{2d^3} - 16\gamma - 2\eta^2\right)\left(\frac{\eta^2 - 2\gamma^2}{n}\right) - 2(\eta^3 + \eta\delta + 2\gamma)^2 \\
&\geq \left(\frac{\epsilon}{2d^3} - 16\gamma - 2\eta^2\right)\left(\frac{\eta^2 - 2\gamma^2}{n}\right) - 6\eta^6 - 6(\eta\delta)^2 - 24\gamma^2.
\end{aligned}$$

Given that both γ and η^2 are less than a sufficiently small multiple of $\frac{\epsilon}{d^3}$, the above is at least

$$\frac{\epsilon\eta^2}{3d^3 n} - 6\eta^6 - 6(\eta\delta)^2 - 24\gamma^2.$$

Given that δ is less than a sufficiently small multiple of $\frac{\epsilon}{dn}$, and γ a small multiple of $\epsilon^2/(d^4 n^2)$, this is

$$\Omega(\epsilon\eta^2/(d^3 n)) = \Omega(\epsilon^3/(d^7 n^3)).$$

The case of $\beta = 0$: We now argue that in the case that $\beta = 0$, no Forster transform exists, as the algorithm correctly identifies a subspace of dimension k containing more than a k/d fraction of the points of X.

Since we have that $\lambda_k(M_A(X)) - \lambda_{k+1}(M_A(X)) \geq \epsilon/(2d^3)$ by assumption, either $\lambda_k(M_A(X)) \geq \frac{1}{d} + \frac{\epsilon}{4d^3}$ or $\lambda_{k+1}(M_A(X)) \leq \frac{1}{d} - \frac{\epsilon}{4d^3}$. The algorithm returns the subspace $V^\perp$ of dimension k, which contains all points X^{B} as $\beta = 0$. We claim that $|X^{\mathrm{B}}|/n > k/d$, which would complete our analysis. This is essentially because the large eigenvalues of $M_A(X)$ on $V^\perp$ imply that X^{B} must have many points.

We consider two subcases below.

Case 1: $\lambda_k(M_A(X)) \geq \frac{1}{d} + \frac{\epsilon}{4d^3}$. In this case, we have that

$$\begin{aligned}
\frac{|X^{\mathrm{B}}|}{n} &= \mathrm{tr}(M_A(X^{\mathrm{B}})) \geq k\,\lambda_k(M_A(X^{\mathrm{B}})) \\
&\geq k\lambda_k(M_A^{V^\perp,V^\perp}(X^{\mathrm{B}})) - k\delta \\
&\geq k\lambda_k(M_A^{V^\perp,V^\perp}(X)) - k\delta - 2k\gamma^2 \\
&\geq k\lambda_k(M_A(X)) - 7k\gamma \\
&\geq k/d + k(\epsilon/(4d^3) - 7\gamma) > k/d\ ,
\end{aligned}$$

where the second line above follows from Property 3, the third line from Claim 3.10. The fourth line follows from Lemma 3.11, and the rest from $\epsilon/d^3 \gg \gamma > \delta$.

Case 2: $\lambda_{k+1}(M_A(X)) \leq \frac{1}{d} - \frac{\epsilon}{4d^3}$. In this case, we have that

$$\frac{|X^S|}{n} = \mathrm{tr}(M_A(X^S)) = \mathrm{tr}(M_A^{V,V}(X^S)) + \mathrm{tr}(M_A^{V^\perp,V^\perp}(X^S)).$$

By Claim 3.10 we have that $\mathrm{tr}(M_A^{V^\perp,V^\perp}(X^S)) \leq 2\gamma^2$. On the other hand since $\beta = 0$, all elements of X^{B} are orthogonal to V and thus

$$\mathrm{tr}(M_A^{V,V}(X^S)) = \mathrm{tr}(M_A^{V,V}(X)).$$

This is at most $(k-d)\lambda_1(M_A^{V,V}(A))$, which by Lemma 3.12 is at most $(k-d)\lambda_{k+1}(M_A(X))+8d\gamma$. Combining with the above, we get that

$$\frac{|X^S|}{n} \le (k-d)\lambda_{k+1}(M_A(X)) + 10d\gamma \le (k-d)/d - (k-d)\epsilon/(4d^3) + 10d\gamma < (k-d)/d \ .$$

Hence in this case as well $|X^{\mathrm{B}}|/n = 1 - |X^S|/n > k/d$.

This completes the proof of Proposition 3.8.

This completes the proof of Proposition 3.2. □

4 Approximate Eigendecomposition in Strongly Polynomial time

In this section, we give a simple algorithm that computes an approximate eigendecomposition with multiplicative error guarantees in strongly polynomial time.

Proposition 4.1. *Given a $d \times d$ PSD matrix M, an accuracy parameter $\epsilon > 0$ and a failure probability $\delta > 0$, there is an algorithm that computes orthogonal vectors $q_1, \ldots, q_d$ and scalars a_i such that the matrix $\hat{M} = \sum_{i=1}^d a_i q_i q_i^\top$ satisfies the following: for all $v \in \mathbb{R}^d$, it holds that*

$$|v^\top (M - \hat{M})v| \le \epsilon\,(v^\top M v) \ .$$

The algorithm performs $\mathrm{poly}(d/\epsilon, \log(1/\delta))$ *arithmetic operations on* $\mathrm{poly}(d/\epsilon, \log(1/\delta), b)$*-bit numbers, where b is the bit complexity of the entries of M.*

Proof. We assume throughout that d is sufficiently large and ϵ sufficiently small.

Our algorithm is based on the power method. In more detail, by taking a large power of M times a random vector, we can obtain an approximation of the principal eigenvalue. Taking a large power of M times another random vector and projecting onto the orthogonal complement of the first, gives us an approximation to the second eigenvector. Repeating this process, we obtain an approximation to the full eigendecomposition. Unfortunately, we cannot quite hope to learn the eigenvectors themselves in general. Specifically, in the case where some of the eigenvalues are close, we would require very large powers of M in order to distinguish them. However, in this case, it is not necessary to learn the eigenvalues exactly in order to obtain a good approximation.

It is well-known that the aforementioned standard approach can be used to get an approximate eigendecomposition such that $\|M - \hat{M}\|_2 \le \epsilon \|M\|_2$. Unfortunately, this guarantee is weaker than the result that we require. For our application, we require that if M has a large eigenvalue gap somewhere, then $\hat{M}$ very precisely finds this gap. Fortunately, if there is a large eigenvalue gap, this makes the power method that much stronger. Indeed, multiplying by a suitable power of M will cause the components of the small eigenvector to shrink by an amount proportional to a power of the gap size.

Our algorithm is presented in pseudocode below.

It is easy to see that this algorithm runs in the appropriate time and bit-complexity bounds. The difficulty is in showing that the resulting $\hat{M}$ satisfies the desired error bounds. We begin by giving this analysis under the assumption that M is non-singular.

We start by noting that if we take q_i' to be the normalization of q_i and take a_i' to be $(q_i')^\top M (q_i')$, then we have that $a_i'(q_i')(q_i')^\top = a_i q_i q_i^\top$, leading to the same matrix $\hat{M}$. (Note that we cannot use a_i' and q_i' in our algorithm only because normalizing q_i requires taking square roots; an operation that is not efficiently implementable in our model). We note that the q_i' are obtained from w_i by

Algorithm 3 Computing the approximate eigendecomposition of a matrix M

```
1: function EigenDecomposition(Matrix $M_{d\times d}$, accuracy parameter $\epsilon$, error probability $\delta$)
2:     Let $A$ be a random $d \times d$ matrix where the entries are i.i.d. uniform samples from $\{1, 2, \ldots, N\}$, for $N$ at least a sufficiently large constant multiple of $d/\delta$.
3:     Let $w_1, w_2, \ldots, w_d$ be the column vectors of $M^t A$, for $t$ a sufficiently large constant multiple of $d^6/\epsilon^2 \log(d/\delta)$.
4:     for $i = 1$ to $d$ do
5:         Let $q_i$ be the projection of $w_i$ onto the orthogonal complement of $w_1, w_2, \ldots, w_{i-1}$.
6:         Let $a_i = 0$ if $q_i = 0$ and $a_i = q_i^\top M q_i/(q_i \cdot q_i)$ otherwise.
7:     return $\{a_i, q_i\}$
```

applying Gram-Schmidt. From here on, we will consider the equivalent algorithm, where the q_i are obtained from the w_i by applying Gram-Schmidt.

Let the eigendecomposition of M be given by $M = \sum_{i=1}^{d} \lambda_i v_i v_i^\top$, where the v_i are an orthonormal basis and where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d \geq 0$. Let $\eta = \epsilon^2/d^3$.

We say that two consecutive eigenvectors v_i and v_{i+1} are in the same block if $\lambda_{i+1} \geq \lambda_i/(1+\eta)$. We say that λ_i and λ_j are in the same block for $i \leq j$, if λ_k and λ_{k+1} are in the same block for all $i \leq k < j$. Note that if λ_i and λ_j are in the same block, their ratio is at most $(1+\eta)^d$; and that if they are in different blocks, their ratio is at least $(1+\eta)$.

Let $\alpha_1, \ldots, \alpha_d$ be the columns of A. Let β_i be the unique vector in $\mathrm{span}(\alpha_1, \alpha_2, \ldots, \alpha_i)$ such that $v_j \cdot \beta_i = 0$ if $j < i$, and $v_i \cdot \beta_i = 1$. We note that if B is the matrix with columns β_i, applying Gram-Schmidt to the columns of $M^t B$ yields the same result as applying it to the columns of $M^t A$. We will need the following claim:

Claim 4.2. *With probability at least* $1 - \delta$ *over the choice of* A*, we have that for all* $1 \leq i \leq d$ *it holds that* $\|\beta_i\|_2 \leq O(d^2/\delta)^i$.

Proof. We note that $\beta_m = \sum_{i=1}^m t_i \alpha_i$, so that $v_j \cdot (\sum_{i=1}^m t_i \alpha_i) = \sum_{i=1}^m t_i (v_j \cdot \alpha_i)$ is equal to 0 for $j < m$, and equal to 1 for $j = m$. In particular, the t_i form the unique vector such that $D_m[t_1, t_2, \ldots, t_m]^\top = [0, 0, \ldots, 0, 1]^\top$, where D_m is the $m \times m$ matrix defined by $(D_m)_{j,i} = [v_j \cdot \alpha_i]_{1 \leq i,j \leq m}$. Using Cramer's rule, we find that each t_i is a sub-determinant of D_m divided by $\det(D_m)$. Since the sub-determinant has absolute value at most the product of the norms of its rows or $O(N\sqrt{m})^{m-1} = O(N\sqrt{d})^m$, it suffices to show that $|\det(D_m)| = \Omega(N\delta/d^{3/2})^m$.

In particular, we will prove that with probability at least $1 - \delta$ over the choice of A we have that $|\det(D_k)| = \Omega(N\delta/(d^{3/2}))^k$ for all k. In fact we show that conditioning on the values of $\alpha_1, \alpha_2, \ldots, \alpha_{m-1}$, the probability that $|\det(D_m)| > \Omega(N\delta/d^{3/2})|\det(D_{m-1})|$ is at least $1 - \delta/d$. If this holds for all m, our result will follow.

To show this, we note that conditioning on $\alpha_1, \alpha_2, \ldots, \alpha_{m-1}$ fixes all but the last column of D_m, which is linear in α_m. The determinant of D_m equals a sum over the last column of the relevant entry times an appropriate sub-determinant. In particular, $\det(D_m) = \sum_{i=1}^m C_i(v_i \cdot \alpha_m) = u \cdot \alpha_m$, where $v = \sum_{i=1}^m C_i v_i$ and C_i are the appropriate sub-determinants. Note that $v_m \cdot u = C_m = \det(D_{m-1})$. Therefore, $|u| \geq |\det(D_{m-1})|$. This implies that u must have an entry of size at least $|\det(D_{m-1})|/\sqrt{d}$. Fixing all other entries of α_m, we note that there is a probability of at least $1 - (\delta/d)$ that $|u \cdot \alpha_m| \gg N\delta/d^{3/2}$. If this holds, then $|\det(D_m)| \geq \Omega(N\delta/d^{3/2})|\det(D_{m-1})|$, as desired. This completes our proof. $\square$

We will show that so long as the conclusion of Claim 4.2 holds, our algorithm will produce an appropriate error guarantee.

We begin by defining

$$\gamma_i := M^t \beta_i / \lambda_i^t \ ,$$

where $\gamma_i = 0$ if $\lambda_i = 0$. Note that in this case λ_j will be 0 for all $j > i$, and thus $M^t\beta_i = 0$. Applying Gram-Schmidt to $M^t\beta_i$ or to γ_i gives the same result, and thus q_i can be thought of as the result of applying Gram-Schmidt to the γ_i. Furthermore, we note that

$$v_j \cdot \gamma_i = \begin{cases} 0 & \text{, if } j < i \\ 1 & \text{, if } j = i \\ O(d^2/\delta)^d (\lambda_j/\lambda_i)^t & \text{, otherwise.} \end{cases}$$

This implies that if λ_i and λ_j are in different blocks, then

$$|v_j \cdot \gamma_i| \leq s \, \min\left(\frac{\lambda_j}{\lambda_i}, \frac{\lambda_i}{\lambda_j}\right) \ ,$$

where

$$s = (d/\delta)^{-3d^3} \ .$$

We require the following claim.

Claim 4.3. *Letting q_i be obtained from γ_i by Gram-Schmidt, we have that if $1 \leq m \leq d$ and if λ_i and λ_m are in different blocks, then*

$$|v_i \cdot q_m| \leq s \, \min\left(\frac{\lambda_m}{\lambda_i}, \frac{\lambda_i}{\lambda_m}\right) M^m \ ,$$

where M is $(Cd^2/\delta)^{d+d^2}$, for a sufficiently large universal constant $C > 0$.

Proof. We proceed by induction on m. The case of $m = 1$ follows immediately from the above observation and the fact that $q_1 = \gamma_1/\|\gamma_1\|_2$ and that $\|\gamma_1\|_2 \geq v_1 \cdot \gamma_1 = 1$.

For the inductive step, we begin by assuming that our claim is true for all smaller values of m. We note that $q_m = r_m/\|r_m\|_2$, where

$$r_m := \gamma_m - \sum_{j=1}^{m-1} (\gamma_m \cdot q_j) q_j \ .$$

We note that if $m > j$ and λ_m and λ_j are in different blocks, then

$$\begin{aligned} |\gamma_m \cdot q_j| &\leq \sum_{k=1}^{d} |v_k \cdot \gamma_m||v_k \cdot q_j| \\ &\leq d\, s \max_k \min\left(\frac{\lambda_m}{\lambda_k}, \frac{\lambda_k}{\lambda_m}\right) \min\left(\frac{\lambda_j}{\lambda_k}, \frac{\lambda_k}{\lambda_j}\right) O(d^2/\delta)^d M^{m-1} \\ &\leq s\, O(d^2/\delta)^d \min\left(\frac{\lambda_m}{\lambda_j}, \frac{\lambda_j}{\lambda_m}\right) M^{m-1} \ , \end{aligned}$$

where the second line above follows from the inductive hypothesis, the fact that λ_k cannot be in the same block as both λ_m and λ_j, and the fact that $\|\gamma_m\|_2 \leq \|\beta_m\|_2 = O(d^2/\delta)^d$.

From this we conclude that if λ_m and λ_k are in different blocks, then

$$|r_m \cdot v_k| \leq |\gamma_m \cdot v_k| + \sum_{j=1}^{m-1} |q_j \cdot v_k||q_j \cdot \gamma_m| \ .$$

We know that

$$|\gamma_m \cdot v_k| \leq s \min\left(\frac{\lambda_m}{\lambda_k}, \frac{\lambda_k}{\lambda_m}\right) \ .$$

For λ_j in the same block as λ_m, we have that $|q_j \cdot v_k||q_j \cdot \gamma_m|$ is at most

$$|\gamma_m||q_j \cdot v_k| \leq s\, O(d^2/\delta)^d \min\left(\frac{\lambda_m}{\lambda_k}, \frac{\lambda_k}{\lambda_m}\right) M^{m-1} \ .$$

For λ_j and λ_m in different blocks, using the above bound on $|\gamma_m \cdot q_j|$, it is at most

$$s\, O(d^2/\delta)^d \min\left(\frac{\lambda_m}{\lambda_j}, \frac{\lambda_j}{\lambda_m}\right) \min\left(\frac{\lambda_k}{\lambda_j}, \frac{\lambda_j}{\lambda_k}\right) M^{m-1} \ ,$$

which means that

$$|q_j \cdot v_k||q_j \cdot \gamma_m| \leq s\, O(d^2/\delta)^d \min\left(\frac{\lambda_m}{\lambda_k}, \frac{\lambda_k}{\lambda_m}\right) M^{m-1} \ .$$

Summing over j, we find that for λ_m and λ_k in different blocks, we have that

$$|r_m \cdot v_k| \leq s\, O(d^2/\delta)^d \min\left(\frac{\lambda_m}{\lambda_k}, \frac{\lambda_k}{\lambda_m}\right) M^{m-1} \ . \tag{17}$$

We now just need to show that $\|r_m\|_2$ is not too small. To achieve this, we will show that q_m has a reasonably large projection onto the space orthogonal to $q_1, \ldots, q_{m-1}$. To show this, let ℓ be the smallest number such that λ_ℓ and λ_m are in the same block. For $\ell \leq i \leq m$, let r'_i denote the projection of γ_i onto the space orthogonal to $q_1, q_2, \ldots, q_{\ell-1}$. Namely, we define

$$r'_i := \gamma_i - \sum_{j=1}^{\ell-1} (\gamma_i \cdot q_j) q_j \ .$$

We note that since each $|\gamma_i \cdot q_j| \leq s\, O(d^2/\delta)^d M^d$, we have that $\|r'_i - \gamma_i\|_2 \leq s\, O(d^2/\delta)^d M^d$. We note that r_m is the component of r'_m orthogonal to $r'_\ell, \ldots, r'_{m-1}$. This equals the ratio of the volumes of the parallelepiped with sides $r'_\ell, \ldots, r'_m$ to the volume of the one with sides $r'_\ell, \ldots, r'_{m-1}$. The latter volume is at most $\|r'_\ell\|_2 \, \|r'_{\ell+1}\|_2 \cdots \|r'_{m-1}\|_2 \leq O(d^2/\delta)^{d^2}$. We can bound the former from below by the determinant of the matrix with entries $v_i \cdot r'_j$, for $\ell \leq i, j \leq m$. However, we note that

$$|v_i \cdot r'_j - v_i \cdot r_j| \leq \|r_j - r'_j\|_2 \leq s\, O(d^2/\delta)^d M^d \ .$$

On the other hand, the matrix with entries $v_i \cdot r_j$ is a lower diagonal matrix with 1's on the diagonal and entries of size at most $O(d^2/\delta)^d$. This matrix has determinant 1, and the difference between its determinant and that of the matrix with entries $v_i \cdot r'_j$ is at most $O(d^2/\delta)^{d^2}\, s\, M^d \leq 1/2$. Thus, the matrix with entries $v_i \cdot r'_j$ has determinant at least 1/2. This implies that $\|r_m\|_2 \geq O(d^2/\delta)^{-d^2}$. Combining this with Equation (17) yields

$$|q_m \cdot v_k| \leq s\, O(d^2/\delta)^{d+d^2} \min\left(\frac{\lambda_m}{\lambda_k}, \frac{\lambda_k}{\lambda_m}\right) M^{m-1} \leq s\, \min\left(\frac{\lambda_m}{\lambda_k}, \frac{\lambda_k}{\lambda_m}\right) M^m \ ,$$

whenever λ_m and λ_k are in different blocks. This completes our inductive step. □

Claim 4.3 implies that if λ_i and λ_j are in different blocks we have that

$$|v_i \cdot q_j| \le \min\left(\frac{\lambda_m}{\lambda_i}, \frac{\lambda_i}{\lambda_m}\right)(\epsilon/d)^3 .$$

We now will try to understand the size of the a_i's. We have the following sequence of (in)equalities:

$$\begin{aligned}
a_i &= q_i^\top M q_i \\
&= \sum_{j=1}^{d} \lambda_j |v_j \cdot q_i|^2 \\
&= \sum_{\lambda_j \text{ in same block as } \lambda_i} \lambda_i(1+O(d\eta))|v_j \cdot q_i|^2 + O\left(\sum_{\lambda_j \text{ in different block from } \lambda_i} (\epsilon/d)^6 \lambda_j (\lambda_i/\lambda_j)\right) \\
&= \lambda_i O(d\eta + \epsilon^6/d^6) + \lambda_i \sum_{\lambda_j \text{ in same block as } \lambda_i} |v_j \cdot q_i|^2 \\
&= \lambda_i O(d\eta + \epsilon^6/d^6) + \lambda_i - \lambda_i \sum_{\lambda_j \text{ not in same block as } \lambda_i} |v_j \cdot q_i|^2 \\
&= \lambda_i(1 + O(\epsilon^2/d^2)) ,
\end{aligned}$$

where we used the fact that the v_j's form an orthonormal basis, and therefore $1 = \|q_i\|_2^2 = \sum_j |v_j \cdot q_i|^2$.

Our result will now follow from the proceeding claim:

Claim 4.4. *For any $1 \le i, j \le d$, we have that $|v_i^\top (M - \hat{M}) v_j| < (\epsilon/d^2)\sqrt{\lambda_i \lambda_j}$.*

Proof. We begin with the case where λ_i and λ_j are not in the same block. We have that $v_i^\top M v_j = 0$ and that

$$|v_i^\top \hat{M} v_j| \le \sum_{k=1}^{d} a_k |v_i \cdot q_k||v_j \cdot q_k| = \sum_{k=1}^{d} O(\lambda_k)(\epsilon/d)^3 \min\left(\frac{\lambda_k}{\lambda_i}, \frac{\lambda_i}{\lambda_k}\right) \min\left(\frac{\lambda_k}{\lambda_j}, \frac{\lambda_j}{\lambda_k}\right) .$$

This in turn is at most

$$\sum_{k=1}^{d} O(\lambda_k)(\epsilon/d)^3 (\sqrt{\lambda_i/\lambda_k})(\sqrt{\lambda_j/\lambda_k}) < (\epsilon/d^2)\sqrt{\lambda_i \lambda_j} .$$

If λ_i and λ_j are in the same block, then we have that

$$v_i^\top \hat{M} v_j = \sum_{k=1}^{d} a_k (v_i \cdot q_k)(v_j \cdot q_k) .$$

The contribution from λ_k not in the same block is once again $O(\lambda_i \epsilon / d^2)$. The contribution from

λ_k in the same block can be bounded above as follows:

$$\begin{aligned}\sum_{\lambda_k \text{ in the same block as } \lambda_i} a_k(v_i \cdot q_k)(v_j \cdot q_k) &= \sum_{\lambda_k \text{ in the same block as } \lambda_i} \lambda_i(1+O(\epsilon^2/d^2))(v_i \cdot q_k)(v_j \cdot q_k) \\ &= O(\lambda_i \epsilon^2/d^2) + \lambda_i \sum_{\lambda_k \text{ in the same block as } \lambda_i} (v_i \cdot q_k)(v_j \cdot q_k) \\ &= O(\lambda_i \epsilon^2/d^2) + \lambda_i(v_i \cdot v_j) - \lambda_i \sum_{\lambda_k \text{ not in the same block as } \lambda_i} (v_i \cdot q_k)(v_j \cdot q_k) \\ &= O(\sqrt{\lambda_i \lambda_j}\epsilon^2/d^2) + \lambda_i \delta_{i,j} \\ &= O(\sqrt{\lambda_i \lambda_j}\epsilon^2/d^2) + v_i^\top M v_j \ .\end{aligned}$$

This completes the proof of the claim. □

To complete our analysis, let $v = \sum_{i=1}^d c_i v_i$. Then, we have that $v^\top M v = \sum_{i=1}^d c_i^2 \lambda_i \geq \max_i(|c_i|\sqrt{\lambda_i})^2$. On the other hand, we have that

$$|v^\top (M-\hat{M})v| \leq \sum_{i,j=1}^d |c_i|\,|c_j|\,|v_i^\top (M-\hat{M})v_j| \leq \sum_{i,j=1}^d (\epsilon/d^2)\,|c_i|\,|c_j|\,\sqrt{\lambda_i \lambda_j} \leq \epsilon \max_i(|c_i|\sqrt{\lambda_i})^2 \leq \epsilon(v^\top M v) \ .$$

This completes our analysis in the case where M is non-singular. When M is singular and rank k, then assuming that the conclusion of Claim 4.2 holds, we note that applying the same analysis to the vectors $M^t\alpha_1, \ldots, M^t\alpha_k$ as elements of the k-dimensional vector space Image(M), we get the desired result. □

5 Matrix Rounding

In this section, we establish our efficient rounding procedure, establishing the following:

Theorem 5.1 (Matrix Rounding)**.** *There is an algorithm that given (i) a set of n points $X \subseteq \{-2^b, \ldots, 2^b\}^d \backslash \{\mathbf{0}\}$ with $b \in \mathbb{Z}_+$, so that X spans $\mathbb{R}^d$, (ii) a full-rank $d \times d$ matrix $A \in \{-2^{rb}, \ldots, 2^{rb}\}^{d \times d}$, with $r \in \mathbb{Z}_+$, and (iii) an accuracy parameter $\epsilon \in (0,1)$, outputs a matrix A' with integer entries of magnitude at most $(\frac{d}{\epsilon})^{O(d^3 b)}$ such that for all points $x \in X$ it holds $\|f_A(x) - f_{A'}(x)\|_2 \leq \epsilon$. The algorithm performs* poly(d, n, r) *arithmetic operations on* poly$(d, n, r, b, \log(1/\epsilon))$*-bit numbers.*

This theorem will allow us to avoid having the matrices A in our main algorithm blow up in bit complexity since every round we can replace A by A' to reduce the bit complexity with at most a small loss of potential.

Notation For a matrix A and a subspace W, we let $A^{(W)} = AI_W$ be the matrix whose i-th row is the i-th row of A projected onto the subspace W. We let $\sigma_{\max}(A)$ to be the maximum singular value of a matrix A and $\sigma_{\min}(A)$ to be the minimum non-negative singular value.

We also use $\lceil x \rfloor$ to denote the integer closest to the real number x, and use $\lceil A \rfloor$ to denote the matrix obtained by applying $\lceil \cdot \rfloor$ to each entry of the matrix A.

The rounding process is presented in Algorithm 4. The main idea of the algorithm is to iteratively reduce the condition number of matrix A without significantly affecting the transformation on the pointset X. To achieve this, the algorithm identifies a subspace V such that V and $V^\perp$ have large multiplicative singular value gap $\sigma_{\min}(A^{(V^\perp)})/\sigma_{\max}(A^{(V)})$. It then aims to rescale the subspace $V^\perp$ so that the condition number decreases. As this could significantly affect the transformation for some points in X, it rescales instead a different subspace R that is very close to $V^\perp$, but at the same time leaves unaltered the transformation for the set of problematic points in X lying in a subspace W. Applying this technique iteratively, we reduce to the case where A has bounded condition number. At this point, we can simply replace A by the matrix obtained by an appropriate rounding of A's entries.

Algorithm 4 Rounding the matrix

```
1: function Round(Matrix $A \in \{-2^{rb}, \dots, 2^{rb}\}^{d\times d}$, Set $X$ of $n$ points in $\{0,\dots,2^b\}^d$, accuracy
   parameter $\epsilon$)
2:     Let $N \leftarrow (\frac{d}{\epsilon})^{O(d^3 b)}$
3:     while not terminated do
4:         Compute estimates $\bar{\sigma}_1$ and $\bar{\sigma}_d$ such that $\sigma_1(A) \le \bar{\sigma}_1 \le 2\sigma_1(A)$ and $\frac{1}{2}\sigma_d(A) \le \bar{\sigma}_d \le \sigma_d(A)$.
5:         if the condition number $\bar{\sigma}_1(A)/\bar{\sigma}_d(A) \ge N$ then return $\lceil \frac{d}{\epsilon} A/\bar{\sigma}_d \rfloor$,
6:         Round the entries of $A$ setting $A \leftarrow \lceil \frac{2^{O(rdb)}}{\epsilon} A/\bar{\sigma}_d \rfloor$
7:         Using approximate eigendecomposition find a subspace $V$ and a parameter $G$, such that
               $$\frac{1}{2}\max_{1\le i\le d-1}\frac{\sigma_i(A)}{\sigma_{i+1}(A)} \le G \le \frac{\sigma_{\min}(A^{(V^\perp)})}{\sigma_{\max}(A^{(V)})} \le \max_{1\le i\le d-1}\frac{\sigma_i(A)}{\sigma_{i+1}(A)}.$$
8:         Obtain $(w_i, p_i)_{i=1}^d$ by running EigendecompositionFromSet$(A, X)$
9:         Find the smallest index $m \ge 0$ such that $p_{m+1} \ge \frac{1}{d}G^{(m+1)/d}\sigma_{\max}(A^{(V)})$
10:        Let $W \leftarrow \text{span}\{w_1,\dots,w_m\}$, $g \leftarrow \frac{\min\{p_{m+1},\sigma_{\min}(A^{(V^\perp)})\}}{\max\{\sigma_{\max}(A^{(W)}),\sigma_{\max}(A^{(V)})\}}$
11:        Consider the subspace $R \leftarrow \text{span}(W \cup V^\perp) \cap W^\perp$,
12:          and note that by Claim 5.11 it holds that $I = I_R + I_W + I_{W^\perp \cap V}$
13:        Define a matrix $T$ that rescales the subspace $R$ by $\delta = \Theta(2^{-b})$ ,
14:          i.e. $T = \delta I_R + I_W + I_{W^\perp \cap V}$
15:        Set $A \leftarrow AT$
```

Algorithm 5 Eigendecomposition with respect to a Set

```
1: function EigendecompositionFromSet(Matrix $A$, Set $X$ of $n$ points)
2:     Let $w_1 = \arg\min_{x\in X}\frac{\|Ax\|_2}{\|x\|_2}$ and $p_1 = \min_{x\in X}\frac{\|Ax\|_2}{\|x\|_2}$
3:     for $i = 2$ to $d$ do
4:         Set $W_{i-1} = \text{span}\{w_1,\dots,w_{i-1}\}$
5:         Set $w_i = \arg\min_{x\in X\setminus W_{i-1}}\frac{\|Ax^{(W_{i-1}^\perp)}\|_2}{\|x^{(W_{i-1}^\perp)}\|_2}$
6:         Set $p_i = \min_{x\in X\setminus W_{i-1}}\frac{\|Ax^{(W_{i-1}^\perp)}\|_2}{\|x^{(W_{i-1}^\perp)}\|_2}$
7:     return $(w_i, p_i)_{i=1}^d$
```

This works by applying an iterative process to decrease the condition number of A followed by appropriate rounding. In this iteration, we take V to be a subspace spanned by the small singular vectors of A right before a large eigenvalue gap. We then define a sequence of vectors $w_1, w_2, \dots, w_n$, where w_i is (roughly) the element of X not in the span of $w_1, w_2, \dots, w_{i-1}$ with $\|Aw_i\|_2$ minimal; and take W to be the span of $w_1, w_2, \dots, w_m$, where $\|Aw_{m+1}\|_2$ is substantially larger than any of the previous values. This gives us a subspace W where any $x \in X$ is either in W or has $\|Ax\|_2$ substantially larger than the corresponding value for any small element of W. We then replace A by AT, where T acts as the identity on W and $W^\perp \cap V$, but shrinks things considerably in orthogonal directions. As elements in $x \in W$ are preserved by T and $x \in X \setminus W$ have most of their contributions to Ax coming from parts orthogonal to W and V, we show (see Proposition 5.4) that $f_A(x) \approx f_{AT}(x)$ for all $x \in X$, and thus this operation does not substantially change the potential. Furthermore, since T scales down the directions orthogonal to V (which are the large singular directions of A), we show (see Proposition 5.3) that the condition number of AT is substantially smaller than the condition number of A, which implies that repeating this process enough times will eventually terminate with a matrix with not-too-large condition number. Finally, in Proposition 5.2, we show that once we have reduced the condition number, rounding the appropriate matrix entries will not substantially change the Forster transform on any given vector.

The analysis of Algorithm 4 is based on three key propositions.

Proposition 5.2. *Let A be any matrix with smallest singular value $\sigma_d(A) > 0$, Given $\epsilon \in (0,1)$ and $\bar{\sigma}$ such that $\frac{1}{2}\sigma_d(A) \le \bar{\sigma} \le \sigma_d(A)$, the integer matrix $\hat{A} = \lceil \frac{d}{\bar{\sigma}\epsilon} A \rfloor$ satisfies*

- *$\kappa(\hat{A}) \le \kappa(A)(1 + 4\epsilon)$,*
- *for all $x \in \mathbb{R}^d$, $\|f_A(x) - f_{\hat{A}}(x)\|_2 \le 2\epsilon$, and*
- *the entries of $\hat{A}$ have magnitude $O(d\kappa(A)/\epsilon)$.*

Proposition 5.2 shows that so long as A has bounded condition number, letting A' be the rounding of an appropriate multiple of A yields a bounded precision matrix that nearly preserves all of the transformed points.

Proposition 5.3. *Let $A \in \mathbb{R}^{d \times d}$ be a full-rank matrix. Let V, W be subspaces of $\mathbb{R}^d$ so that*

$$\sigma_{\min}(A^{(V^\perp)}) \ge g \max\{\sigma_{\max}(A^{(W)}), \sigma_{\max}(A^{(V)})\}$$

for some $g > 10$. Define $T = \delta I_R + I_{R^\perp}$ for $\delta \ge 8g^{-1}$ and $R = \operatorname{span}(W \cup V^\perp) \cap W^\perp$. It holds that

$$\kappa(AT) \le 30\delta\kappa(A) \; .$$

Proposition 5.3 shows that after every iteration the condition number of the matrix A is significantly reduced by a factor of $O(\delta)$.

Proposition 5.4. *Let $A \in \mathbb{R}^{d \times d}$ be a full-rank matrix and X be a set of points. Let V, W be subspaces of $\mathbb{R}^d$ so that*

$$\frac{\min_{x \in X \setminus W} \|Ax^{(W^\perp)}\|_2 / \|x^{(W^\perp)}\|_2}{\max\{\sigma_{\max}(A^{(W)}), \sigma_{\max}(A^{(V)})\}} \ge g \text{ and } \min_{x \in X \setminus W} \|x^{(W^\perp)}\|_2 \ge \rho$$

for some $\rho \in (0,1)$ and $g > 1$. Define $T = \delta I_R + I_{R^\perp}$ for $\delta \in (0,1)$ and $R = \operatorname{span}(W \cup V^\perp) \cap W^\perp$. For all $x \in X$, it holds that

$$\|f_A(x) - f_{AT}(x)\|_2 \le \frac{16}{(g-1)\rho\delta} \; .$$

Proposition 5.4 shows that for every iteration of the algorithm, the update of A has a negligible effect on the transformation f_A.

We defer the proofs of the propositions to Sections 5.2, 5.3, 5.4, and proceed with the proof of Theorem 5.1.

5.1 Proof of Theorem 5.1

Before we proceed with the proof, we argue that the steps of the algorithm are well-defined. In particular, we must show that the choice of m is feasible. Indeed, the following claim shows that such an index $m \in \{0, \ldots, d-1\}$ as required in Line 9 always exists, as there is at least one p_i with $p_i \geq (G/d)\sigma_{\max}(A^{(V)})$.

Claim 5.5. *There is an index $i \in \{1, \ldots, d\}$ such that $p_i \geq (G/d)\sigma_{\max}(A^{(V)})$.*

Proof. Letting $u_i = w_i^{(W_{i-1}^{\perp})}/\|w_i^{(W_{i-1}^{\perp})}\|_2$, we note that the u_i's form an orthonormal basis of $\mathbb{R}^d$ and that $p_i = \|Au_i\|_2$. Therefore, we have that $\sqrt{\sum_{i=1}^d p_i^2} = \|A\|_F \geq \sigma_{\min}(A^{(V^{\perp})}) \geq G\sigma_{\max}(A^{(V)})$. Since $\sqrt{\sum_{i=1}^d p_i^2} \leq \sqrt{d}\max_{i=1}^d p_i$, this means there is at least one p_i with value at least $(G/d)\sigma_{\max}(A^{(V)})$. $\square$

We now proceed to bound the improvement on the condition number of matrix A at every iteration. Starting with a matrix A with condition number $\kappa(A)$, the algorithm finds a subspace V with a large multiplicative singular value gap G, i.e., $\frac{\sigma_{\min}(A^{(V^{\perp})})}{\sigma_{\max}(A^{(V)})} \geq G$. This gap G is at least $\kappa(A)^{1/d}$, as the following claim shows.

Lemma 5.6. *At any iteration of the algorithm, $G \geq \frac{1}{2}\kappa(A)^{1/d}$, and $g \geq \frac{1}{2d}\kappa(A)^{1/d^2}$.*

Proof. We prove each of the bounds separately.

We first bound G: Indeed, we have that

$$G = \frac{\sigma_{\min}(A^{(V^{\perp})})}{\sigma_{\max}(A^{(V)})} \geq \frac{1}{2}\max_{1\leq i\leq d-1}\frac{\sigma_i(A)}{\sigma_{i+1}(A)} \geq \frac{1}{2}\left(\frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}\right)^{1/d} = \frac{1}{2}\kappa(A)^{1/d} .$$

We now bound g: Recall that $g = \frac{\min\{p_{m+1}, \sigma_{\min}(A^{(V^{\perp})})\}}{\max\{\sigma_{\max}(A^{(W)}), \sigma_{\max}(A^{(V)})\}}$. To show the statement, we lower bound all four combinations of numerators and denominators separately.

- Term $\sigma_{\min}(A^{(V^{\perp})})/\sigma_{\max}(A^{(V)})$:

  By the definition of G, we have that

$$\sigma_{\min}(A^{(V^{\perp})}) \geq G\,\sigma_{\max}(A^{(V)}).$$

- Term $p_{m+1}/\sigma_{\max}(A^{(V)})$:

  Recall that the subspace W is defined by computing an eigendecomposition of A with respect to the set of points X to obtain $(w_i, p_i)_{i=1}^d$. It sets $W = \text{span}\{w_1, \ldots, w_m\}$ by choosing the smallest $m \geq 0$ so that $p_{m+1} \geq \frac{1}{d}G^{(m+1)/d}\sigma_{\max}(A^{(V)})$. This implies that

$$p_{m+1} \geq \frac{1}{d}G^{1/d}\sigma_{\max}(A^{(V)}) .$$

- Term $p_{m+1}/\sigma_{\max}(A^{(W)})$:

  We also have that $\max_{1\leq i\leq m} p_i \leq \frac{1}{d}G^{m/d} \leq G^{1/d}p_{m+1}$

  Moreover, we have that

$$d\max_{1\leq i\leq m} p_i \geq \sqrt{\sum_{i=1}^m p_i^2} = \|A^{(W)}\|_F \geq \|A^{(W)}\|_2 = \sigma_{\max}(A^{(W)}) .$$

  This implies that $\sigma_{\max}(A^{(W)}) \leq d\max_{i=1}^m p_i$ which in turn gives that $\sigma_{\max}(A^{(W)}) \leq dG^{1/d}p_{m+1}$.

- Term $\sigma_{\min}(A^{(V^\perp)})/\sigma_{\max}(A^{(W)})$:

  Moreover, the definition of m implies that for all $i \le m$, we have $p_i \le \frac{1}{d} G^{i/d} \sigma_{\max}(A^{(V)})$, and since $m < d$, we have

$$\max_{1 \le i \le m} p_i \le \frac{1}{d} G^{1-1/d} \sigma_{\max}(A^{(V)}) \le \frac{1}{d} G^{-1/d} \sigma_{\min}(A^{(V^\perp)}) \ .$$

  This implies that

$$\sigma_{\min}(A^{(V^\perp)}) \ge G^{1/d} \sigma_{\max}(A^{(W)}) \ .$$

Overall, we have that $g \ge \frac{1}{d} G^{1/d} \ge \frac{1}{2d} \kappa(A)^{1/d^2}$. □

We can now apply Proposition 5.3 to bound from above the condition number $\kappa(AT)$ of the matrix AT for $T = \delta I_R + I_{R^\perp}$. For $\delta \ge 8g^{-1}$, we get that

$$\kappa(AT) \le 30\delta\kappa(A) \le O(2^{-b})\kappa(A). \tag{18}$$

Initially, the condition number is bounded by $(2d)^{rdb}$, as the following claim shows:

Claim 5.7. *For any full-rank matrix $A \in \{-2^{rb}, \dots, 2^{rb}\}^{d \times d}$, we have that $\kappa(A) \le d^d 2^{rdb}$.*

Proof. Note that $\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$. We have that $\sqrt{\det(A^\top A)} = \prod_{i=1}^d \sigma_i(A)$, and thus we obtain $\sigma_{\max}(A)^{d-1}\sigma_{\min}(A) \ge \sqrt{\det(A^\top A)} \ge 1$, where the last inequality follows since A is full-rank and has integer entries. This implies that $\kappa(A) \le \sigma_{\max}(A)^d$. The statement follows since $\sigma_{\max}(A) \le \|A\|_F \le \sqrt{d^2 2^{2rb}} \le d2^{rb}$. □

Thus, since at any iteration we have that $g \ge \Omega(2^b)$, as $\kappa(A) \ge N \ge 2^{\Omega(d^2 b)}$, the condition number significantly improves at every iteration by a factor of $O(2^{-b})$; after $O(dr)$ iterations, it will become $\max\{2^{\Omega(d^2 b)}, N\}$.

We now proceed to bound the change in the transformation $\|f_A(x) - f_{AT}(x)\|_2$ for all $x \in X$, using Proposition 5.4. To do this, we need to lower bound ρ. In particular, we show that for all $x \in X \setminus W$, it holds that $\|x^{(W^\perp)}\|_2 \ge d^{-d} 2^{-db}$.

Claim 5.8. *For any iteration of* Round*, we have that* $\min_{x \in X \setminus W} \|x^{(W^\perp)}\|_2 \ge d^{-d} 2^{-db}$.

Proof. Note that at any iteration, the bit complexity of X stays the same as only the matrix A gets updated. Fix any $x \in X \setminus W$. Since W is the span of the linearly independent vectors $w_1, \dots, w_m$, for any $x \in X \setminus W$, $\|x^{(W^\perp)}\|_2 = \frac{\mathrm{Vol}(w_1,\dots,w_m,x)}{\mathrm{Vol}(w_1,\dots,w_m)}$, where $\mathrm{Vol}(z_1, \dots, z_k)$ is the k-dimensional volume of the parallelepiped defined by the vectors $z_1, \dots, z_k$.

Note that $\mathrm{Vol}(z_1, \dots, z_k)$ is given by $\sqrt{\det(Z^\top Z)}$, where Z is the matrix $[z_1|\dots|z_k]$. Since the vectors $w_1, \dots, w_m, x \in \{-2^b, \dots, 2^b\}^d$ are integer vectors and linearly independent, the corresponding volumes of the parallelepipeds they define have volume at least 1 and at most $d^d 2^{db}$. □

We thus get that we can apply Proposition 5.4 with $\rho = d^{-d} 2^{-db}$. In particular, we conclude that for all $x \in X$ we have

$$\|f_A(x) - f_{AT}(x)\|_2 \le \frac{16}{(g-1)\,\rho\delta} \le \frac{d^{O(db)}}{\kappa(A)^{1/d^2}} \ .$$

The total incurred error for the transformation across all iterations is at most

$$\|f_A(x) - f_{A'}(x)\|_2 \le \frac{d^{O(db)}}{N^{1/d^2}} \sum_{i=0}^{\infty} 2^{-bi/d^2} \le \frac{d^{O(db)}}{N^{1/d^2}} \ ,$$

where the first inequality is because $\kappa(A)$ ends at N and decreases by a factor of $\Omega(2^b)$ every time. This is at most $\epsilon/2$, if $N \geq \left(\frac{d}{\epsilon}\right)^{\Omega(d^3 b)}$.

Overall, we obtain a matrix A' with condition number at most $N \geq \left(\frac{d}{\epsilon}\right)^{\Omega(d^3 b)}$ such that $\|f_A(x) - f_{A'}(x)\|_2 \leq \epsilon/2$. Applying Proposition 5.2, we show that by rounding A' to have integer entries of magnitude at most $\left(\frac{d}{\epsilon}\right)^{O(d^3 b)}$, we get that $\|f_A(x) - f_{A'}(x)\|_2 \leq \epsilon$

To ensure that the numerical operations of every iteration are $\mathrm{poly}(d, n)$, we need to efficiently identify the subspace V. We note that we only need to compute the singular vectors approximately, so that the singular values are approximated within a small constant. We can achieve this in $\mathrm{poly}(d)$ operations using the algorithm of Proposition 4.1. Under this approximation, the multiplicative gap G satisfies $G \geq \frac{1}{2}\kappa(A)^{1/d}$, which results in the same order of improvement, when the condition number of A goes from $\kappa(A)$ to $\Theta(2^{-b})\kappa(A)$ at every iteration. Thus, overall, the number of arithmetic operations is $\mathrm{poly}(d, n, r)$, as the number of iterations is at most $O(dr)$.

To ensure that the bit complexity of the operations remains bounded, we need some additional care. It is easy to see that if the bit complexity of the points is $\mathrm{poly}(d, n, r, \log(1/\epsilon))$, it remains $\mathrm{poly}(d, n, r, \log(1/\epsilon))$ at the end of a single iteration. Yet, the increase may be significant over multiple iterations and the bit complexity may blow up exponentially.

To keep the bit complexity bounded, we can apply Proposition 5.2 with accuracy parameter $\epsilon/2^{O(rdb)}$, to round the resulting matrix A, so that it has entries with bit complexity $O(\log(\kappa(A)) + drb\log(1/\epsilon))$. This has negligible effect on the decrease of the condition number at every iteration (as it only increases it by at most a fixed constant), and ensures that the bit complexity of A at every iteration is bounded by a fixed polynomial in d, n, r, b and $\log(1/\epsilon)$. Moreover, the introduced error in the transformation f_A for every iteration is at most $\epsilon/2^{O(drb)}$, and thus over all $O(dr)$ iterations, it is at most $O(\epsilon)$, as desired.

We finally remark that all constants used in the algorithm can be computed as a function of the entries of X and A. In particular, $2^{O(b)} = \max_{x \in X} \|x\|_\infty^{O(1)}$ and $2^{O(rb)} = \max_{i,j} \|A_{ij}\|^{O(1)}$.

This completes the proof of Theorem 5.1. We now proceed with the proofs of Propositions 5.2, 5.3, and 5.4.

5.2 Proof of Proposition 5.2

Let $A_r \triangleq \frac{d}{\sigma\epsilon} A$. To prove the proposition, we first bound the condition number of $\hat{A}$ and then show that the transformation $f_{\hat{A}}$ is close to f_A.

Matrix A_r has singular values $\sigma_1(A_r) = \frac{d}{\sigma\epsilon}\sigma_1(A) \leq 2\frac{d}{\epsilon}\kappa(A)$ and $\sigma_d(A_r) = \frac{d}{\sigma\epsilon}\sigma_d(A) \geq \frac{d}{\epsilon}$. Moreover, after rounding the entries of matrix A_r to the nearest integer to obtain $\hat{A}$, for any unit vector v, we have that

$$\|(\hat{A} - A_r)v\|_2 \leq \frac{d}{2} ,$$

since $\hat{A} - A_r$ has entries of magnitude at most $1/2$.

Therefore, we get that $\hat{A}$ has singular values $\sigma_1(\hat{A}) \leq \sigma_1(A_r) + \frac{d}{2} \leq \sigma_1(A_r)(1+\epsilon/2)$ and $\sigma_d(\hat{A}) \geq \sigma_d(A_r) - \frac{d}{2} \geq \sigma_d(A_r)(1 - \epsilon/2)$. This implies that the condition number is at most $\kappa(A_r)(1 + 4\epsilon) = \kappa(A)(1 + 4\epsilon)$. Moreover, the magnitude of every entry is at most $\sigma_1(\hat{A}) \leq 2\frac{d}{\epsilon}\kappa(A)(1 + \epsilon/2)$.

To bound the effect of the rounding on the transformation, we note that by Fact 2.1, $f_{A_r} = f_A$

and thus for any x we have that

$$\begin{aligned}\|f_A(x) - f_{\hat{A}}(x)\|_2 = \|f_{A_r}(x) - f_{\hat{A}}(x)\|_2 &= \left\| \frac{A_r x}{\|A_r x\|_2} - \frac{\hat{A}x}{\|\hat{A}x\|_2} \right\|_2 \\ &\leq \left\| \frac{A_r x - \hat{A}x}{\|A_r x\|_2} \right\|_2 + \left\| \frac{\hat{A}x}{\|A_r x\|_2} - \frac{\hat{A}x}{\|\hat{A}x\|_2} \right\|_2 \\ &= \frac{\|(A_r - \hat{A})x\|_2}{\|A_r x\|_2} + \left| 1 - \frac{\|\hat{A}x\|_2}{\|A_r x\|_2} \right| \\ &\leq 2\frac{\|(A_r - \hat{A})x\|_2}{\|A_r x\|_2} \leq \frac{d}{\sigma_d(A_r)} \leq 2\epsilon \ .\end{aligned}$$

5.3 Proof of Proposition 5.3

For any matrix $\hat{T}$, we can bound the condition number of $\kappa(AT)$ by $\kappa(A\hat{T})\kappa(\hat{T}^{-1}T)$. We define $\hat{T} = \delta I_{V^\perp} + I_V$ as the matrix that rescales the subspace $V^\perp$ by δ. We argue that $\kappa(A\hat{T}) \leq 10\delta\kappa(A)$.

Claim 5.9. *For any $\delta \geq g^{-1}$, it holds that $\kappa(A\hat{T}) \leq 10\delta\kappa(A)$.*

Proof. We first show that $\sigma_{\max}(A\hat{T}) \leq 2\delta\sigma_{\max}(A)$. Indeed, for any unit vector $v \in \mathbb{R}^d$, we have that

$$\|A\hat{T}v\|_2 = \|\delta Av^{(V^\perp)} + Av^{(V)}\|_2 \leq \delta\|Av^{(V^\perp)}\|_2 + \|Av^{(V)}\|_2 \leq \delta\sigma_{\max}(A) + g^{-1}\sigma_{\max}(A) \ .$$

and since $\delta \geq g^{-1}$, this gives the required bound on $\sigma_{\max}(A\hat{T})$.

We now argue that $\sigma_{\min}(A\hat{T}) \geq \frac{1}{5}\sigma_{\min}(A)$. For any unit vector $v \in \mathbb{R}^d$, we have that

$$\begin{aligned}\|A\hat{T}v\|_2 = \|\delta Av^{(V^\perp)} + Av^{(V)}\|_2 &\geq \delta\|Av^{(V^\perp)}\|_2 - \|Av^{(V)}\|_2 \\ &\geq \delta\sigma_{\min}(A^{(V^\perp)})\|v^{(V^\perp)}\|_2 - \sigma_{\max}(A^{(V)})\|v^{(V)}\|_2 \\ &\geq \delta g\sigma_{\max}(A^{(V)})\|v^{(V^\perp)}\|_2 - \sigma_{\max}(A^{(V)})\|v^{(V)}\|_2 \\ &\geq \sigma_{\max}(A^{(V)})(\|v^{(V^\perp)}\|_2 - \|v^{(V)}\|_2) \ ,\end{aligned}$$

where the last inequality follows since $\delta \geq g^{-1}$. Since, $\sigma_{\min}(A) \leq \min_{v \in V: \|v\|_2 = 1} \|Av\|_2 \leq \sigma_{\max}(A^{(V)})$, we get that the above is at least $\sigma_{\min}(A)(\|v^{(V^\perp)}\|_2 - \|v^{(V)}\|_2)$. This is greater than $\frac{1}{5}\sigma_{\min}(A)$ if $\|v^{(V)}\|_2 < \frac{3}{5}$. Moreover, if $\|v^{(V)}\|_2 \geq \frac{3}{5}$, we have that

$$\|A\hat{T}v\|_2 \geq \sigma_{\min}(A)\|\hat{T}v\|_2 \geq \sigma_{\min}(A)\|v^{(V)}\|_2 \geq \frac{3}{5}\sigma_{\min}(A) \ ,$$

which again gives the required bound on $\sigma_{\min}(A\hat{T})$. □

To complete the proof of Proposition 5.3, we now bound $\kappa(\hat{T}^{-1}T)$.

Claim 5.10. *We have that $\kappa(\hat{T}^{-1}T) \leq 3$.*

Proof. Note that $\hat{T}^{-1} = I_V + 1/\delta I_{V^\perp}$ and that $T = \delta I_R + I_W + I_{W^\perp \cap V}$. Since,

$$\begin{aligned}\delta\hat{T}^{-1}I_R &= I_R - (1-\delta)I_V I_R, \\ \hat{T}^{-1}I_W &= I_W + (1 - 1/\delta)I_{V^\perp}I_W, \\ \hat{T}^{-1}I_{W^\perp \cap V} &= I_{W^\perp \cap V},\end{aligned}$$

and $I = I_R + I_W + I_{W^\perp \cap V}$, this implies that

$$\hat{T}^{-1}T = I + (1/\delta - 1)I_{V^\perp} I_W - (1-\delta) I_V I_R \ .$$

Our result will follow by bounding each of the terms $\|(1/\delta - 1)I_{V^\perp} I_W\|_2$, $\|(1-\delta)I_V I_R\|_2$ individually below $1/4$.

To bound the first term, we argue that for every unit vector $w \in W$, we have that

$$\|w^{(V^\perp)}\|_2 \le 2/g \ .$$

This is because $\|Aw\|_2 \le \sigma_{\max}(A^{(W)}) \le g^{-1}\sigma_{\min}(A^{(V^\perp)})$, but

$$\begin{aligned}\|Aw\|_2 &\ge \|Aw^{(V^\perp)}\|_2 - \|Aw^{(V)}\|_2 \ge \sigma_{\min}(A^{(V^\perp)})\|w^{(V^\perp)}\|_2 - \sigma_{\max}(A^{(V)})\|w^{(V^\perp)}\|_2 \\ &\ge \sigma_{\min}(A^{(V^\perp)})(\|w^{(V^\perp)}\|_2 - g^{-1}\|w^{(V)}\|_2) \ge \sigma_{\min}(A^{(V^\perp)})(\|w^{(V^\perp)}\|_2 - g^{-1}) \ .\end{aligned}$$

We thus get that $\|I_{V^\perp} I_W\|_2 \le 2g^{-1}$. Thus,

$$\|(1/\delta - 1)I_{V^\perp} I_W\|_2 \le 2/(\delta g) le 1/4 \ .$$

To bound $\|I_V I_R\|_2$, it suffices to show that for any unit vector $x \in R$, $\|I_V x\|_2$ is small. As $x \in \mathrm{span}(W \cup V^\perp)$, we can write $x = x_W + x_{V^\perp}$ with $x_W \in W$ and $x_{V^\perp} \in V^\perp$. Since $x \in W^\perp$, we have that

$$0 = x_W \cdot x = \|x_W\|_2^2 + x_W \cdot x_{V^\perp} \ .$$

Thus,

$$\|x_W\|_2^2 = |x_W \cdot x_{V^\perp}| \le \|x_{V^\perp}\|_2 \|I_{V^\perp} I_W x_W\|_2 \le (2/g)\|x_{V^\perp}\|_2 \|x_W\|_2 \ ,$$

which implies that $\|x_W\|_2 \le (2/g)\|x_{V^\perp}\|_2$. and that $\|x\|_2 \ge \|x_{V^\perp}\|_2 - \|x_W\|_2 \ge (g/2 - 1)\|x_W\|_2$. Therefore,

$$\|I_V x\|_2 = \|I_V x_W\|_2 \le \|x_W\|_2 \le (g/2-1)^{-1}\|x\|_2 \ .$$

Thus for $g > 10$, $\|(1-\delta)I_V I_R\|_2 \le 1/4$. □

5.4 Proof of Proposition 5.4

We first show the following claim that relates the subspace R to V and W.

Claim 5.11. *For any subspaces V, W of $\mathbb{R}^d$, and $R = \mathrm{span}(W \cup V^\perp) \cap W^\perp$, it holds that $I = I_R + I_W + I_{W^\perp \cap V}$.*

Proof. We first argue that vectors in W, R and $W^\perp \cap V$ are pairwise orthogonal. Indeed, any vector in the latter two subspaces belongs in $W^\perp$ and thus is orthogonal to W. Moreover, note that any vector $r \in R$ belongs in $\mathrm{span}(W \cup V^\perp)$ and can be written as $r = w + v$ for some $w \in W$ and $v \in V^\perp$. For any $u \in W^\perp \cap V$, it holds that $u \cdot r = u \cdot w + u \cdot v = 0$, because $u \cdot w = 0$ as $u \in W^\perp$ and $w \in W$ and $u \cdot v = 0$ as $u \in V$ and $v \in V^\perp$. We now argue that $\mathrm{span}(W, R, W^\perp \cap V) = \mathbb{R}^d$. Indeed, $\mathrm{span}(W, R) = \mathrm{span}(W, \mathrm{span}(W \cup V^\perp) \cap W^\perp) = \mathrm{span}(W, \mathrm{span}(W, \mathrm{proj}_{W^\perp} V^\perp) \cap W^\perp) = \mathrm{span}(W, \mathrm{proj}_{W^\perp} V^\perp) = \mathrm{span}(W, V^\perp)$. This is true, as for any subspaces A, B, $\mathrm{span}(A, B) = \mathrm{span}(A, \mathrm{proj}_{A^\perp} B)$. Thus, $\mathrm{span}(W, R, W^\perp \cap V) = \mathrm{span}(W, V^\perp, W^\perp \cap V) = \mathbb{R}^d$. □

We now proceed to show Proposition 5.4 assuming $x \in X \setminus W$ as in the case that $x \in W$, we have that $Tx = x$ and so $f_A(x) = f_{AT}(x)$. We argue that for any such point $x \in X \setminus W$, both $f_A(x)$ and $f_{AT}(x)$ are close to $\frac{Ax^{(R)}}{\|Ax^{(R)}\|_2}$. To do this, we use Claim 5.12, which shows that the contributions to Ax of the projections of x to W and $W^\perp \cap V$ are small.

Claim 5.12. *For any $x \in X \setminus W$,*

$$\|Ax^{(R)}\|_2 \geq (g-1)\rho \max\{\|Ax^{(W)}\|_2, \|Ax^{(W^\perp \cap V)}\|_2\} \ .$$

Proof. We analyze two cases separately.

We first bound $\|Ax^{(W^\perp \cap V)}\|_2$. If $x^{(W^\perp \cap V)} = 0$, we have that $\|Ax^{(W^\perp \cap V)}\|_2 = 0$. Otherwise, we have that

$$\|Ax^{(W^\perp)}\|_2 \geq g\|x^{(W^\perp)}\|_2 \sigma_{\max}(A^{(V)}) \geq g\|x^{(W^\perp)}\|_2 \frac{\|Ax^{(W^\perp \cap V)}\|_2}{\|x^{(W^\perp \cap V)}\|_2} \geq g\|Ax^{(W^\perp \cap V)}\|_2 \ .$$

By the triangle inequality and since $x^{(W^\perp)} = x^{(R)} + x^{(W^\perp \cap V)}$, we get that $\|Ax^{(R)}\|_2 \geq \|Ax^{(W^\perp)}\|_2 - \|Ax^{(W^\perp \cap V)}\|_2$. This implies that $\|Ax^{(R)}\|_2 \geq (g-1)\|Ax^{(W^\perp \cap V)}\|_2$ and $\|Ax^{(R)}\|_2 \geq (1-1/g)\|Ax^{(W^\perp)}\|_2$.

We now bound $\|Ax^{(W)}\|_2$. If $x^{(W)} = 0$, we have that $\|Ax^{(W)}\|_2 = 0$. Otherwise, we have that

$$\frac{\|Ax^{(W^\perp)}\|_2}{\|x^{(W^\perp)}\|_2} \geq g\sigma_{\max}(A^{(W)}) \geq g\frac{\|Ax^{(W)}\|_2}{\|x^{(W)}\|_2} \ .$$

Now since $\|Ax^{(R)}\|_2 \geq (1-1/g)\|Ax^{(W^\perp)}\|_2$, we get that

$$\|Ax^{(R)}\|_2 \geq (g-1)\frac{\|x^{(W^\perp)}\|_2}{\|x^{(W)}\|_2}\|Ax^{(W)}\|_2 \geq (g-1)\rho\|Ax^{(W)}\|_2 \ .$$

□

Using Claim 5.12, we now have that for $\mu = \frac{1}{(g-1)\rho}$:

$$\begin{aligned} \|f_A(x) - \frac{Ax^{(R)}}{\|Ax^{(R)}\|_2}\|_2 &\leq \|f_A(x) - \frac{Ax^{(R)}}{\|Ax\|_2}\|_2 + \|\frac{Ax^{(R)}}{\|Ax\|_2} - \frac{Ax^{(R)}}{\|Ax^{(R)}\|_2}\|_2 \\ &\leq \frac{\|Ax^{(W^\perp \cap V)}\|_2 + \|Ax^{(W)}\|_2}{\|Ax\|_2} + \left|\frac{\|Ax^{(R)}\|_2}{\|Ax\|_2} - 1\right| \\ &\leq 2\mu\frac{\|Ax^{(R)}\|_2}{\|Ax\|_2} + \left|\frac{\|Ax^{(R)}\|_2}{\|Ax\|_2} - 1\right| \end{aligned}$$

Since $\frac{\|Ax\|_2}{\|Ax^{(R)}\|_2} \in [1-2\mu, 1+2\mu]$, we get that:

$$\|f_A(x) - \frac{Ax^{(R)}}{\|Ax^{(R)}\|_2}\|_2 \leq \frac{4\mu}{1-2\mu} \ .$$

Similarly we get that $\|f_{AT}(x) - \frac{Ax^{(R)}}{\|Ax^{(R)}\|_2}\|_2 \leq \frac{4\mu/\delta}{1-2\mu/\delta}$, by noting that $ATx^{(R)} = \delta Ax^{(R)}$, $ATx^{(W^\perp \cap V)} = Ax^{(W^\perp \cap V)}$ and $ATx^{(W)} = Ax^{(W)}$.

Combining the above we get that,

$$\|f_A(x) - f_{AT}(x)\|_2 \leq \frac{4\mu}{1-2\mu} + \frac{4\mu/\delta}{1-2\mu/\delta} \leq 16\mu/\delta \ ,$$

for $\delta \in [0,1]$ and $\mu/\delta \in [0,1/4]$.

6 Full Algorithm: Proof of Theorem 1.5

In this section, we will put together the basic algorithm from Section 3 with the approximate eigendecomposition algorithm from Section 4 and the bit complexity reduction routine from Section 5 to prove Theorem 1.5. The final algorithm is given in pseudocode below.

Algorithm 6 Full Forster Transform Algorithm

```
1: function ForsterTransform (set $X \subset \mathbb{R}^d_*$ of $n$ points, accuracy parameter $\epsilon$)
2:     Let $A \leftarrow I$        ▷ Initialization of transformation matrix $A$
3:     while $\|M_A(X)\|_F^2 > \frac{1}{d} + \frac{\epsilon^2}{d^2}$ do
4:         Set $A \leftarrow$ ImproveTransform$(A, X, \epsilon, \delta)$, for $\delta$ a sufficiently small polynomial in $\epsilon/dn$.
5:         if ImproveTransform returned a subspace $V$ then
6:             return $V$.
7:         Set $A \leftarrow$ Round$(A, X, \zeta)$, for $\zeta$ a sufficiently small multiple of $\epsilon^5/(d^{10}n^5)$.
8:     return $A$
```

The full version of our ImproveTransform function is given in pseudocode below.

Algorithm 7 Find Improved Transform Matrix

```
1: function ImproveTransform (current matrix $A \in \mathbb{R}^{d\times d}$, $X \subset \mathbb{R}^d_*$, accuracy parameter $\epsilon$,
   error parameter $\delta$)
2:     Let $C$ be a sufficiently large constant.
3:     Set $n \leftarrow |X|$.
4:     Set $a_1, \dots, a_d, q_1, \dots, q_d \leftarrow$ EigenDecomposition$(M_A(X), \eta, \delta)$, for $\eta = \epsilon^4/(C^3 d^8 n^4)$.
5:     Sort $a_i\|q_i\|_2^2$ in descending order of size.
6:     Find $k$ maximizing $a_k\|q_k\|_2^2 - a_{k+1}\|q_{k+1}\|_2^2$.
7:     Let $W$ be the span of $q_{k+1}, \dots, q_d$.
8:     Let $\gamma$ be $\epsilon^2/(Cd^4n^2)$
         ▷ Consider the Following Two Cases
9:     if there exists $x \in X$ such that $\|\mathrm{proj}_W f_A(x)\|_2, \|\mathrm{proj}_{W^\perp} f_A(x)\|_2 \geq \gamma$ then
10:        Set $V \leftarrow W$.
11:        Set $\alpha \leftarrow \epsilon/(64nd^3)$.
12:    else
13:        Set $X^{\mathrm{B}} \leftarrow \{x \in X : \|\mathrm{proj}_{W^\perp} f_A(x)\|_2 \geq \gamma\}$.
14:        Let $c_1, c_2, \dots, c_d, r_1, r_2, \dots, r_d \leftarrow$ EigenDecomposition$(M_A(X^{\mathrm{B}}), \eta)$
15:        Let $V$ be the span of the $d-k$ vectors $r_i$ with the smallest values of $c_i\|r_i\|_2^2$.
16:        Set $\beta \leftarrow \max_{x\in X^{\mathrm{B}}} \|f_A^{(V)}(x)\|_2$.
17:        if $\beta = 0$ then
         ▷ No Forster Transform Exists
18:            return The subspace $V^\perp$.
19:        else        ▷ Case where $\beta > 0$
20:            Set $\alpha \leftarrow \epsilon/(3\beta d^2 n) - 1$
21:    return $(I + \alpha I_V)\, A$
```

The rest of this section will be devoted to proving the correctness of this algorithm.

To begin with, we note that if the algorithm returns a matrix A, it must be the case that $\|M_A(X)\|_F^2 \leq 1/d + \epsilon^2/d^2$, and so by Lemma 3.1, A will be an ϵ-Forster transform matrix. Also note that upon applying Round, we replace A with a matrix whose entries have bit complexity $\mathrm{poly}(bdn/\epsilon)$. From there it is not hard to see that all arithmetic computations performed by this algorithm are done to only polynomial precision. Finally, we note that in each iteration of the main while loop, our algorithm performs a polynomial number of arithmetic operations. Therefore, in order to prove correctness, we need to establish the following:

1. If our algorithm returns a subspace V, then $|X \cap V| > |X| \dim(V)/d$.
2. In each iteration of our while loop, the potential function $\Phi_X(A) := \|M_A(X)\|_F^2$ decreases by at least an inverse-polynomial amount.

We note that if this is the case, we will only need to call EigenDecomposition a polynomial number of times, and thus we may assume that all such calls succeed, which we will assume hereafter.

We begin with a basic consequence of our eigendecomposition lemma:

Lemma 6.1. *We have that $\|M_A(X) - \sum_{i=1}^d a_i q_i q_i^\top\|_F \leq \sqrt{d}\eta$.*

Proof. Letting $M = M_A(X)$ and $\hat{M} = \sum_{i=1}^d a_i q_i q_i^\top$, we have that for any unit vector v it holds that $|v^\top(M - \hat{M})v| \leq \eta\,(v^\top M v) \leq \eta$. This means that $\|M - \hat{M}\|_2 \leq \eta$. We note that since this is the maximum eigenvalue of $M - \hat{M}$, and since the Frobenius norm of $M - \hat{M}$ is the square root of the sum of squares of the eigenvalues, we have that $\|M - \hat{M}\|_F \leq \sqrt{d}\eta$, as desired. □

We next show that our approximate eigendecomposition exhibits an eigenvalue gap. In particular, we establish the following lemma.

Lemma 6.2. *We have that $a_k \|q_k\|^2 - a_{k+1}\|q_{k+1}\|^2 \geq (3/4)(\epsilon/d^3)$.*

Proof. Note that $\mathrm{tr}(M_A(X)) = 1$. Therefore, by Lemma 6.1 and letting $\hat{M} = \sum_{i=1}^d a_i q_i q_i^\top$, we have that that $|\mathrm{tr}(\hat{M}) - 1| \leq \sqrt{d}\,\|M_A(X) - \hat{M}\|_F \leq d\,\eta$. Moreover, we have that $\|\hat{M}\|_F \geq \|M_A(X)\|_F - \sqrt{d}\eta$. Thus, $\|\hat{M}\|_F^2 \geq 1/d + \epsilon^2/d^2 + O(\eta)$. On the other hand, we can write

$$\|\hat{M}\|_F^2 = \sum_{i=1}^d (a_i\,\|q_i\|_2^2)^2 = \sum_{i=1}^d (a_i\|q_i\|_2^2 - 1/d)^2 + 2\,\mathrm{tr}(\hat{M})/d - 1/d = \sum_{i=1}^d (a_i\|q_i\|_2^2 - 1/d)^2 + 1/d + O(d\eta)\ .$$

This implies that

$$\sum_{i=1}^d (a_i\,\|q_i\|_2^2 - 1/d)^2 \geq \epsilon^2/d^2 + O(d\eta)\ .$$

Thus, there must be some i with $\left|a_i\|q_i\|_2^2 - 1/d\right| \geq (99/100)\epsilon/d^2$. Since the average value of $a_i\|q_i\|_2^2 - 1/d$ is $(\mathrm{tr}(\hat{M})-1)/d = O(d\eta)$, the difference between the biggest and smallest values of $a_i\,\|q_i\|_2^2 - 1/d$ must differ by at least $(3/4)(\epsilon/d^2)$. Therefore, the biggest single gap between consecutive values of $\|q_i\|_2^2$ must be at least $(3/4)(\epsilon/d^3)$. This completes our proof. □

It is now easy to show that ImproveTransform decreases our potential in the case where there exists $x \in X$ such that $\|\mathrm{proj}_W f_A(x)\|_2\,, \|\mathrm{proj}_{W^\perp} f_A(x)\|_2 \geq \gamma$. To prove this, we would like to apply Proposition 3.5. In particular, in this case, we let $\rho = \max_{x\in X}(\min(\|\mathrm{proj}_W f_A(x)\|_2\,, \|\mathrm{proj}_{W^\perp} f_A(x)\|_2))$. By assumption, we have that $\rho > \gamma$ and $\rho = \max_{x\in X}(\min(\|\mathrm{proj}_W f_A(x)\|_2\,, \|\mathrm{proj}_{W^\perp} f_A(x)\|_2))$, as the first property in Proposition 3.5 requires.

Next we let $M = M_A(X)$, and $\hat{M} = \sum_{i=1}^d a_i q_i q_i^\top$. We note that

$$\lambda_{\min}(\hat{M}^{V^\perp,V^\perp}) - \lambda_{\max}(\hat{M}^{V,V}) = a_k \|q_k\|_2^2 - a_{k+1}\|q_{k+1}\|_2^2 \geq (3/4)(\epsilon/d^3) .$$

Therefore, by Lemma 6.1, we have that

$$\lambda_{\min}(M^{V^\perp,V^\perp}) - \lambda_{\max}(M^{V,V}) = a_k \|q_k\|_2^2 - a_{k+1}\|q_{k+1}\|_2^2 \geq \epsilon/(2d^3) ,$$

showing that Property 2 holds.

Finally, we note that

$$\hat{M}^{V,V^\perp} = \mathbf{0} .$$

Thus, by Lemma 6.1, we have that

$$\|M^{V,V^\perp}\|_F \leq \sqrt{d}\eta \leq \alpha \leq \alpha\rho ,$$

thus showing that the third property applies.

Therefore, applying Proposition 3.5, if there is an $x \in X$ such that $\|\mathrm{proj}_W f_A(x)\|_2 , \|\mathrm{proj}_{W^\perp} f_A(x)\|_2 \geq \gamma$, then setting $C = \textsc{ImproveTransform}(A, X, \epsilon)$, we have that

$$\Phi_X(C) \leq \Phi_X(A) - \rho^2\epsilon/(8nd^2) \leq \Phi_X(A) - \gamma^2\epsilon/(8nd^2) .$$

For the case where all $x \in X$ have $\min(\|\mathrm{proj}_W f_A(x)\|_2 , \|\mathrm{proj}_{W^\perp} f_A(x)\|_2) \leq \gamma$, we would like to apply Proposition 3.8. We begin by showing that all of the necessary properties apply.

For starters, letting $\hat{M} = \sum_{i=1}^d a_i q_i q_i^\top$ and $M = M_A(X)$, we have that

$$\lambda_k(\hat{M}) - \lambda_{k+1}(\hat{M}) = a_k\|q_k\|_2^2 - a_{k+1}\|q_{k+1}\|_2^2 \geq (3/4)(\epsilon/d^3) .$$

Since $\|M_A(X) - \hat{M}\|_F \leq \sqrt{d}\eta$, we have that

$$\lambda_k(M_A(X)) - \lambda_{k+1}(M_A(X)) \geq (3/4)(\epsilon/d^3) - 2\sqrt{d}\eta \geq \epsilon/(2d^3) .$$

The requirement that for each $x \in X$ that $\min(\|\mathrm{proj}_W f_A(x)\|_2 , \|\mathrm{proj}_{W^\perp} f_A(x)\|_2) \leq \gamma$ is a bit subtle, since the W used in Proposition 3.8 is the relevant eigenspace of $M_A(X)$, while our W is merely an approximation of it. Fortunately, it is not hard to show that these spaces are relatively close to each other.

Lemma 6.3. *If v is a unit eigenvector of M, then $\|\mathrm{proj}_W(v)\|_2 \leq 3d^{7/2}\eta/\epsilon \leq \gamma/\sqrt{d}$ if it is one of the top k eigenvectors, and $\|\mathrm{proj}_{W^\perp}(v)\|_2 \leq (3d^{7/2}\eta/\epsilon) \leq \gamma/\sqrt{d}$ otherwise.*

Proof. We have by definition that $Mv = \lambda v$ for some λ. We have that either $\lambda \leq \lambda_k(\hat{M}) - (3/8)(\epsilon/d^3)$ or $\lambda \geq \lambda_{k+1}(\hat{M}) + (3/8)(\epsilon/d^3)$. Without loss of generality, we assume the latter. We note that by Lemma 6.1 that $\|Mv - \hat{M}v\|_2 \leq \sqrt{d}\eta$, and thus $\|\hat{M}v - \lambda v\|_2 \leq \sqrt{d}\eta$.

Letting $v^{(W)}$ and $v^{(W^\perp)}$ denote the projections of v onto W and $W^\perp$, and noting that $\hat{M}v^{(W)} \in W$ and $\hat{M}v^{(W^\perp)} \in W^\perp$, we have that

$$\|\hat{M}v^{(W)} - \lambda v^{(W)}\|_2 \leq \sqrt{d}\eta .$$

On the other hand, we have that $\lambda I_W - \hat{M}^{W,W} \geq (3/8)(\epsilon/d^3)I_W$. Therefore, we have that

$$(3/8)(\epsilon/d^3)\|v^{(W)}\|_2 \leq \sqrt{d}\eta .$$

From this, we conclude that $\|v^{(W)}\|_2 \leq 3d^{7/2}\eta/\epsilon \leq \gamma/\sqrt{d}$. This completes our proof. □

From the preceding, we note that for any unit vector u that is a linear combination of either the top-k or bottom $d-k$ eigenvectors of M that u is γ-close to either $W^\perp$ or W respectively (since it is a sum of relevant eigenvectors). We have that for any $x \in X$ that either $\|f_A^{(W)}(x)\|_2 \leq \gamma$ or $\|f_A^{(W^\perp)}(x)\|_2 \leq \gamma$. In the former case, if u is a unit vector that is a linear combination of the bottom $d-k$ eigenvectors, then u is γ-close to W, so $u \cdot f_A(x) \leq 2\gamma$. This implies that the projection of $f_A(x)$ onto the eigenspace of the bottom $d-k$ eigenvectors has norm at most 2γ. Similarly, if $\|f_A^{(W^\perp)}(x)\|_2 \leq \gamma$, then the projection of $f_A(x)$ onto the eigenspace of the top-k eigenvectors is at most 2γ. This shows that the hypothesis of Proposition 3.8 involving the projections of these vectors onto what it calls W is satisfied with γ replaced by 2γ.

For Property 1 we let $\tilde{M} := \sum_{i=1}^d c_i r_i r_i^\top$, and note that $\|\tilde{M} - M_A(X^{\mathrm{B}})\|_F \leq \sqrt{d}\eta$. We note that V is the $(d-k)$-dimensional subspace minimizing $\mathrm{tr}(\tilde{M}^{V,V})$. In particular, this implies that for U the span of the bottom $d-k$ eigenvectors of $M_A(X)$, we have that

$$\mathrm{tr}(M_A^{V,V}(X^{\mathrm{B}})) \leq \mathrm{tr}(\tilde{M}^{V,V}) + d\eta \leq \mathrm{tr}(\tilde{M}^{U,U}) + d\,\eta \leq \mathrm{tr}(M_A^{U,U}(X^{\mathrm{B}})) + 2d\eta \leq \mathrm{tr}(M_A^{U,U}(X^{\mathrm{B}})) + \gamma^2/4 \ .$$

This shows that Property 1 holds for $\delta = \gamma/2$.

Property 2 holds similarly. Property 3 holds since

$$\lambda_k(M_A^{V^\perp,V^\perp}(X^{\mathrm{B}})) \geq \lambda_k(\tilde{M}^{V^\perp,V^\perp}) - \sqrt{d}\eta = \lambda_k(\tilde{M}) - \sqrt{d}\eta \geq \lambda_k(M_A(X^{\mathrm{B}})) - 2\sqrt{d}\eta \geq \lambda_k(M_A(X^{\mathrm{B}})) - \gamma/2 \ .$$

For Property 4, recall that $\beta = \max_{x \in X^{\mathrm{B}}} \|f_A^{(V)}(x)\|_2$. This implies that $\|M_A^{V,V}(X^{\mathrm{B}})\|_2 \leq \beta^2$. By the relative error property of Proposition 4.1, this implies that for $\tilde{M} := \sum_{i=1}^d r_i r_i^\top$ that $\|\tilde{M}^{V,V}\|_2 \leq 2\beta^2$. Also note that by definition $\tilde{M}^{V,V^\perp} = \mathbf{0}$. Next, let v be a unit vector in V and w a unit vector in $V^\perp$. We have that

$$(v \pm \beta w)^\top \tilde{M}(v \pm \beta w) = v^\top \tilde{M} v + \beta^2 w^\top \tilde{M} w \leq 3\beta^2 \ .$$

Thus, by the relative error property of Proposition 4.1, we have that

$$(v \pm \beta w)^\top M_A(X^{\mathrm{B}})(v \pm \beta w) = (v \pm \beta w)^\top \tilde{M}(v \pm \beta w) + O(\eta\beta^2) \ .$$

Taking the difference, we get that

$$\begin{aligned} 2\beta v^\top M_A(X^{\mathrm{B}}) w &= (v + \beta w)^\top M_A(X^{\mathrm{B}})(v + \beta w) - (v - \beta w)^\top M_A(X^{\mathrm{B}})(v - \beta w) \\ &= (v + \beta w)^\top \tilde{M}(X^{\mathrm{B}})(v + \beta w) - (v - \beta w)^\top \tilde{M}(v - \beta w) + O(\eta\beta^2) \\ &= (v^\top \tilde{M} v + \beta^2 w^\top \tilde{M} w) - (v^\top \tilde{M} v + \beta^2 w^\top \tilde{M} w) + O(\eta\beta^2) \\ &= O(\eta\beta^2) \ . \end{aligned}$$

Thus,

$$v^\top M_A(X^{\mathrm{B}}) w = O(\eta\beta) \ .$$

Summing over a basis of $v \in V$ and $w \in V^\perp$, we get that

$$\|M_A^{V,V^\perp}(X^{\mathrm{B}})\|_F = O(d\eta\beta) \leq (\gamma/2)\beta \ .$$

This shows that Property 4 holds.

Thus, we can apply Proposition 3.8 and find that if we return a subspace, it has the desired property; and otherwise that setting $C = \textsc{ImproveTransform}(A, X, \epsilon)$, we have that

$$\Phi_X(C) \leq \Phi_X(A) - \Omega(\epsilon^3/(d^7 n)) \ .$$

Thus, in either case, if $\textsc{ImproveTransform}(A, X, \epsilon)$ returns a matrix, the value of $\Phi_X(A)$ decreases by $\Omega(\epsilon^5/(d^{10}n^5))$. Since ζ is less than half of this, each iteration of $\textsc{ForsterTransform}$'s main while loop decreases $\Phi_X(A)$ by at least $\Omega(\epsilon^5/(d^{10}n^5))$. Therefore, our algorithm terminates in at most polynomially many iterations.

7 PAC Learning Halfspaces in Strongly Polynomial Time

In this section, we give our strongly polynomial improper PAC learner for halfspaces, thereby establishing Theorem 1.6.

7.1 Approximate Forster Decomposition

Theorem 1.5 is often difficult to use directly as it does not always guarantee a Forster ransform. This is necessary because if many points are concentrated on a subspace, it may be the case that no such transform exists. However, in this case we can at least find a dense subspace and hopefully can find a Forster transform on that subspace. In general, we have the following result:

Proposition 7.1 (Forster Decomposition)**.** *There is an algorithm that given a multiset X of n points in $\mathbb{R}^d_*$ and $\epsilon > 0$, runs in time strongly-polynomial in $d\,n/\epsilon$, and with high probability returns a subspace $V \subseteq \mathbb{R}^d$ with $V \neq \mathbf{0}$ and a linear transformation $A : V \to \mathbb{R}^{\dim(V)}$, such that*

1. $|X \cap V| \geq (n/d)\dim(V)$.
2. *The eigenvalues of* $\frac{1}{|X\cap V|}\sum_{x\in X\cap V} f_A(x)(f_A(x))^\top$ *are in* $[(1-\epsilon)/\dim(V), (1+\epsilon)/\dim(V)]$.

Proof. The algorithm here is quite simple, presented in pseudocode below.

Algorithm 8 Extended Forster Transform Algorithm

```
1: function ForsterSubspace (set $X \subset \mathbb{R}^d_*$ of $n$ points, accuracy parameter $\epsilon$)
2:     Let $V = \mathbb{R}^d$.
3:     Let $d' = \dim(V)$ and let $L$ be a linear isomorphism between $V$ and $\mathbb{R}^{d'}$ of bit complexity
       comparable to the bit complexity of $V$.
4:     Let $X' := \{L(x) : x \in X \cap (V)\}$.
5:     Run Algorithm 6 on $X' \subseteq \mathbb{R}^{d'}$.
6:     If it returns a subspace $W$, set $V \leftarrow L^{-1}W$ and return to Step 3.
7:     Otherwise, if it returns a matrix $A$, return $(V, AL)$.
```

The essential guarantee of this algorithm is that V is always a subspace of bounded bit complexity, such that $|X \cap V| \geq |X|\dim(V)/d$. This is clearly true initially. If it was true for V, and our algorithm finds a subspace W, it will also be true of $V' = L^{-1}W$. To see this, we note that

$$\begin{aligned}|V' \cap X| &= |\{x \in X \cap V : L(x) \in W\}| = |X' \cap W| \geq |X'|\dim(W)/d' \\ &\geq |X|\dim(V)\dim(W)/(\dim(V)d) = |X|\dim(W)/d\,.\end{aligned}$$

On the other hand, we note that W is generated by points in X', and thus V' is generated by points in X, which in turn implies the bounded complexity claim. In particular, this allows us to define an L with polynomial bit-complexity, which (along with the observation that $\dim(V)$ shrinks by at least one each iteration) makes the algorithm clearly strongly polynomial.

The correctness follows from the above proof that $|X \cap V| \geq |X|\dim(V)/d$, and the fact that A gives an ϵ-approximate Forster transform of X' on W. This completes the proof of Proposition 7.1. □

7.2 PAC Learning Halfspaces

Since we work in the distribution-independent setting, will assume without loss of generality that the target halfspace is homogeneous, i.e., has zero threshold. We can straightforwardly reduce the general case to the homogeneous case by increasing the dimension by 1. In particular, if we associate point $x \in \mathbb{R}^d$ with $x' = (x, -1) \in \mathbb{R}_*^{d+1}$, then we note that $w \cdot x - t = (w, t) \cdot (x, -1)$, and thus a general halfspace over the x vectors is equivalent to a homogeneous halfspace over the x'.

The basic idea of our PAC learning algorithm is that if we are given a set of points in approximate radial isotropic position, we can use a variant of the perceptron algorithm to efficiently compute a hypothesis that correctly classifies a reasonable fraction of these points. In particular, we will be using the following lemma, a version of which appears in [BFKV97, DV04b]:

Lemma 7.2. *Let S be a set of n labeled examples $(x, y) \in \mathbb{R}^d \times \{\pm 1\}$ such that there exists an unknown vector $w \in \mathbb{R}_*^d$ with $y = \mathrm{sign}(w \cdot x)$ for each $(x, y) \in S$, and let $\gamma > 0$ be a parameter. There exists an algorithm that given S and γ has running time strongly polynomial in $n\, d/\gamma$, and returns a vector $v \in \mathbb{R}_*^d$ that for all $(x, y) \in S$ with $|v \cdot x| \geq \gamma \|v\|_2 \|x\|_2$ satisfies $y = \mathrm{sign}(v \cdot x)$.*

Proof. We begin with the assumption that we know a vector v such that $v \cdot w \geq 3\|w\|_2/\gamma$ and $\|v\|_2^2 = O(d/\gamma^2)$. The algorithm is the following:

1. While there exists an $(x, y) \in S$ with $|v \cdot x| \geq \gamma \|v\|_2 \|x\|_2$ and $y \neq \mathrm{sign}(v \cdot x)$, do:
   (a) Let $\hat{x}$ be a positive multiple of x with ℓ_2-norm between 1 and 2.
   (b) $v \leftarrow v + y(\hat{x})$.
2. Return v.

It is clear that the returned value of v has the desired property and that each operation can be performed with limited precision. It remains to show that, under the given assumptions on v, this algorithm will terminate in a polynomial number of iterations.

For this, we note that in each iteration if we let v' be the new value of v, we have that

$$\|v'\|_2^2 = \|v\|_2^2 + 2yv \cdot (\hat{x}) + \|\hat{x}\|_2^2 \leq \|v\|_2^2 + 4 + 2y(v \cdot x)\|\hat{x}\|_2/\|x\|_2 \, .$$

Noting that y and $(v \cdot x)$ have opposite signs, the RHS above is at most

$$\|v\|_2^2 + 4 - 2|v \cdot x|/\|x\|_2 \leq \|v\|_2^2 + 4 - 2\|v\|_2 \gamma \, .$$

Therefore, so long as $\|v\|_2 \geq 3/\gamma$, we have that $\|v'\|_2^2 \leq \|v\|_2^2 - 2$.

On the other hand, we have that

$$v' \cdot w = v \cdot w + y(\hat{x} \cdot w).$$

Since y has the same sign as $x \cdot w$, which has the same sign as $\hat{x} \cdot w$, the above quantity is at least $v \cdot w$. This means that $v \cdot w$ only increases over the course of our algorithm, and therefore throughout the algorithm $\|v\|_2 \geq |v \cdot w|/\|w\|_2 \geq 3/\gamma$. Give the above, this implies that $\|v\|_2^2$ must decrease by at least 2 each iteration. This cannot happen more than $\|v\|_2^2$ times, and therefore the algorithm will terminate after at most $O(d/\gamma^2)$ iterations.

It remains to show how to efficiently find a v with $v \cdot w \geq 3\|w\|_2/\gamma$ and $\|v\|_2^2 = O(d/\gamma^2)$. We claim that it is always possible to take v to be an appropriately large constant multiple of $\sqrt{d}/\gamma$ times plus or minus a standard basis vector. This is because some coordinate of w must have absolute value at least $\|w\|_2/\sqrt{d}$. Thus, we can run the above algorithm in parallel for each such initial value of v and run until one of them returns an answer. □

Combining the modified perceptron algorithm of Lemma 7.2 with an approximate Forster transform, gives us a way to learn a reasonable fraction of the points for any linearly separable dataset.

Lemma 7.3. *Let S be a multiset of labeled examples $(x, y) \in \mathbb{R}^d_* \times \{\pm 1\}$ such that there exists an unknown vector $w \in \mathbb{R}^d_*$ with $y = \operatorname{sign}(w \cdot x)$ for each $(x, y) \in S$. There exists a strongly polynomial time algorithm that with high probability returns a subspace V of $\mathbb{R}^d$, a linear transformation $A : V \to \mathbb{R}^{\dim(V)}$, and a vector $v \in V$ such that for every $(x, y) \in S$ with $x \in V$ and $|v \cdot (Ax)| \geq \|v\|_2 \, \|Ax\|_2/(2\sqrt{d})$ we have that $y = \operatorname{sign}(v \cdot x)$. Furthermore, this holds for at least a $1/(4d)$-fraction of points $(x, y) \in S$.*

Proof. First, we apply the algorithm of Proposition 7.1 to the multiset $X = \{x \in \mathbb{R}^d : (x, y) \in S\}$ with $\epsilon = 1/2$, to obtain V and A. We then let $S' := \{(Ax, y) : (x, y) \in S, x \in V\}$. We note that for all $(z, y) \in S'$ we have that $y = \operatorname{sign}(w \cdot x) = \operatorname{sign}(((A^\top)^{-1} w) \cdot z)$. This means that we can apply the algorithm of Lemma 7.2 to S', which we do with $\gamma = 1/(2\sqrt{d})$ to obtain v.

By the statement of Lemma 7.2, we have that for each $(Ax, y) \in S'$ with $|v\cdot(Ax)| > \|v\|_2 \, \|Ax\|_2/(2\sqrt{d})$ that $y = \operatorname{sign}(v \cdot (Ax))$, as desired. It remains to show that this applies to a large fraction of points $(x, y) \in S$.

To establish this, we note that

$$\frac{1}{|S|} \sum_{(x,y)\in S, x\in V} \frac{(Ax)\,(Ax)^\top}{\|Ax\|_2^2} = \left(\frac{|S'|}{|S|}\right) \left(\frac{1}{|S'|} \sum_{(x,y)\in S, x\in V} \frac{(Ax)\,(Ax)^\top}{\|Ax\|_2^2} \right) \succeq \left(\frac{\dim(V)}{d}\right) \left(\frac{I}{2\dim(V)}\right) \succeq \frac{I}{2d} \,.$$

Therefore, we have that

$$\frac{1}{|S|} \sum_{(x,y)\in S} \mathbf{1}\{x \in V\} \left(\frac{v \cdot (Ax)}{\|Ax\|_2 \, \|v\|_2} \right)^2 \geq v^\top \left(\frac{I}{2d}\right) v/\|v\|_2^2 \geq \frac{1}{2d} \,.$$

This means that the average value over $(x, y) \in S$ of $g(x) := \mathbf{1}\{x \in V\} \left(\frac{v\cdot(Ax)}{\|Ax\|_2 \|v\|_2}\right)^2$ is at least $1/(2d)$. The contribution from terms with $g(x) < 1/(2\sqrt{d})$ is at most $1/(4d)$. Since $g(x) \leq 1$ for all x, this implies that at least a $1/(4d)$-fraction of points $(x, y) \in S$ have $g(x) \geq 1/(2\sqrt{d})$. This completes the proof of Lemma 7.3. $\square$

Ideally, we would like a version of Lemma 7.3 that works over a distribution rather than a finite set. This can be achieved by running the algorithm of Lemma 7.3 on a suitably large set of samples. To establish generalization guarantees, we leverage the fact that the collection of possible classifiers comes from a set of bounded VC-dimension.

Proposition 7.4. *Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{\pm 1\}$ such that for some unknown vector $w \in \mathbb{R}^d_*$ we have that for $(x, y) \sim \mathcal{D}$ that $y = \operatorname{sign}(w \cdot x)$ almost surely. Given $\epsilon, \delta > 0$ with $\epsilon < 1/(20d)$, there exists an algorithm that draws $n = O(d^2 \log(1/\delta)/\epsilon^2)$ i.i.d. samples from $\mathcal{D}$, runs in time strongly polynomial in n, d, and with probability at least $1 - \delta$ returns a vector subspace V in $\mathbb{R}^d$, a linear transformation $A : V \to \mathbb{R}^{\dim(V)}$ and a vector $v \in V$, such that:*

1. *The probability over $(x, y) \sim \mathcal{D}$ that $x \in V$, $|v \cdot (Ax)| \geq \|v\|_2 \, \|Ax\|_2/(2\sqrt{d})$, and $y \neq \operatorname{sign}(v\cdot x)$ is at most ϵ.*

2. *The probability over $(x, y) \sim \mathcal{D}$ that $x \in V$ and $|v \cdot (Ax)| \geq \|v\|_2 \, \|Ax\|_2/(2\sqrt{d})$ is at least $1/(5d)$.*

Proof. We take a set S of n i.i.d. samples from $\mathcal{D}$ and apply the algorithm of Lemma 7.3 to them for n a sufficiently large constant multiple of $(d^2 \log(1/\delta)/\epsilon^2)$. This is clearly a strongly polynomial time algorithm. It remains to prove correctness.

We note that the probability over (x, y) drawn uniformly from S that $x \in V$, $|v \cdot (Ax)| \geq \|v\|_2 \|Ax\|_2/(2\sqrt{d})$, and $y \neq \mathrm{sign}(v \cdot x)$ is 0. Furthermore, the probability over (x, y) drawn uniformly from S that $x \in V$ and $|v \cdot (Ax)| \geq \|v\|_2 \|Ax\|_2/(2\sqrt{d})$ is at least $1/(4d)$. It suffices to show that (with high probability over our samples) these probabilities over S are within ϵ of the corresponding probabilities if (x, y) were drawn from $\mathcal{D}$.

In fact, we will show that with probability $1 - \delta$ over our choice of samples the following holds: for any choice of V, A, and v, the probabilities over S and $\mathcal{D}$ of these events differ by at most ϵ. This will follow from the VC-Inequality [DL01], if we can show that these events come from classes of VC-dimension $O(d^2)$. We now proceed with the argument. We note that these events depend only on the following simpler events:

- Whether $y = 1$.
- Whether $x \in V$.
- Whether $(v \cdot x) > 0$.
- Whether $|v \cdot x|^2 \geq \|v\|_2^2 \|Ax\|_2^2/(4d)$.

The first of these is a specific event, so has VC-dimension 0. The second event checks membership in a subspace, which has VC-dimension d. The third event checks membership in a halfspace, hence also has VC-dimension d. The last of these events is a degree-2 threshold condition, which has VC dimension $O(d^2)$. Since the events we care about are logical combinations of finitely many events of VC-dimension $O(d^2)$, they come from classes with VC-dimension $O(d^2)$. This completes our proof. $\square$

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

Theorem 7.5. *Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d \times \{\pm 1\}$ such that for some unknown vector $w \in \mathbb{R}^d_*$ we have that for $(x, y) \sim \mathcal{D}$ that $y = \mathrm{sign}(w \cdot x)$ almost surely. Given $\epsilon, \delta > 0$ with $\epsilon < 1/(20d)$ there is an algorithm that draws $n = O(d^{9/2} \log(1/\epsilon) \log(d/\epsilon\delta)/\epsilon^2)$ i.i.d. samples from $\mathcal{D}$, runs in strongly polynomial time, and returns a strongly polynomial time computable function $f : \mathbb{R}^d \to \{\pm 1\}$ such that with probability $1 - \delta$ over the samples it holds that $\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f(x) \neq y] \leq \epsilon$.*

Proof. For simplicity, we allow our algorithm to output a function f valued in $\{0, 1, -1\}$. The algorithm is as follows:

It is easy to see that the sample complexity and runtime are as desired.

For correctness, we note that S is a set of i.i.d. samples from the distribution of $\mathcal{D}$ conditioned on $f_{i-1}(x) = 0$. By the conclusion of Proposition 7.4, this means that, with probability at least $1 - \delta/(2r)$ over the samples, the probability that $f_i(x) = 0$ is at most $(1 - 1/(2\sqrt{d}))$ times the probability that $f_{i-1}(x) = 0$. Furthermore, with probability at least $(1 - \delta/(2r))$, we have that $|S| < 2M\,\mathbf{Pr}[f_{i-1}(x) = 0]$. Combining the above, we see that with probability $1 - \delta$, when our algorithm returns an f, it is the case that $\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f(x) = 0] \leq \epsilon/2$.

Furthermore, if all the calls to the algorithm from Proposition 7.4 succeed, then the probability over $(x, y) \sim \mathcal{D}$ conditioned on $f_{i-1}(x) = 0$ that $f_i(x) = y$ is at least $1/(5d)$, while the probability that $f_i(x) = -y$ is at most $\epsilon/(10d)$.

Using this, we can show by induction on i that

$$\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f_i(x) = -y] < (\epsilon/2)\,\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f_i(x) = y]\ .$$

Algorithm 9 Halfspace Learning Algorithm

```
1: function LearnLTF (sample access to distribution $\mathcal{D}$ over $\mathbb{R}^d_* \times \{\pm 1\}$, accuracy parameter $\epsilon$)
2:     Let $f_0 \equiv 0$.
3:     Let $C > 0$ be a sufficiently large universal constant.
4:     Let $r = C\sqrt{d}\log(1/\epsilon) \in \mathbb{Z}_+$.
5:     for For $i = 1$ to $r$ do
6:         Take $M := Cd^4 \log(d/\epsilon\delta)/\epsilon^2$ samples from $\mathcal{D}$ and call the resulting multiset $T$.
7:         Let $S$ be the set of $(x, y) \in T$ such that $f_{i-1}(x) = 0$.
8:         if $|S| < \epsilon M/4$ then
9:             return $f_{i-1}$
10:        else
11:            Run the algorithm from Proposition 7.4 with parameters $\epsilon \leftarrow \epsilon/(10d)$ and $\delta \leftarrow \delta/(2r)$
           to obtain $V, A, v$, using $S$ as the set of samples.
12:            Let
```

$$f_i(x) := \begin{cases} f_{i-1}(x) & \text{, if } f_{i-1}(x) \neq 0 \\ \operatorname{sign}(v \cdot (Ax)) & \text{, if } f_{i-1} = 0, x \in V, \text{and } |v \cdot (Ax)|/(2\sqrt{d}) \\ 0 & \text{, otherwise.} \end{cases}$$

This combined with the result that $\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f_i(x) = 0] < \epsilon/2$ for the returned f_i, gives our final result. □

8 Conclusions and Open Problems

In this work, we designed the first strongly polynomial time algorithm for computing ϵ-approximate Forster transforms of a given dataset[2]. By using this algorithm is an essential ingredient, we gave the first strongly polynomial time algorithm for distribution-free PAC learning of halfspaces, both in the realizable setting and in the presence of semi-random label noise. This algorithmic result is surprising (even in the realizable case), as obtaining a strongly polynomial *proper* PAC learner is *equivalent* to strongly polynomial LP — a major unsolved problem in TCS.

A number of open problems suggest themselves:

- Our ϵ-approximate Forster transform algorithm has runtime scaling polynomially with $1/\epsilon$. That is, our algorithm runs in strongly polynomial time when ϵ is at least inverse polynomial in n, d. An obvious open question is to develop a strongly polynomial algorithm with a polylog$(1/\epsilon)$ runtime dependence. To achieve such a guarantee with our approach, one needs to circumvent two obstacles: First, one would need to reduce the number of iterations of our algorithm (that is controlled by the progress in our potential function). Second, one would require a strongly polynomial approximate eigendecomposition subroutine with a polylog$(1/\epsilon)$ runtime dependence.

- We believe that the following question is of independent interest: Is there a strongly polynomial time algorithm for approximate eigendecomposition with a polylog$(1/\epsilon)$ runtime dependence? Moreover, is there a deterministic algorithm?

[2]While our Forster algorithm is randomized, we remark that the only source of randomness is due to the method we use to compute an approximate eigendecomposition. It is plausible that deterministic algorithms exist for this purpose, in which case our Forster algorithm becomes deterministic as well.

- The running time of our algorithm is strongly polynomial in n, d, but the polynomial dependence is quite large (of the order of $(nd)^{10}$). While we did not make any effort to optimize the degree of the polynomials, it would be interesting to understand the quantitative limitations of our approach. Can our approach lead to algorithms with good practical performance?

- As mentioned in the introduction of this paper, Forster's rescaling can be viewed as a very special cases of operator scaling and tensor scaling [GdO18]. These tasks have attracted significant attention in recent years from various communities, and efficient (weakly polynomial) algorithms (in some cases with a poly$(1/\epsilon)$ dependence) have been developed, see, e.g., [AGL+18, BFG+18] and references therein. It would be interesting to explore whether our approach can be extended to yield strongly polynomial algorithms (when ϵ is not too small) for such generalizations.

References

[AGL+18] Z. Allen-Zhu, A. Garg, Y. Li, R. M. de Oliveira, and A. Wigderson. Operator scaling via geodesically convex optimization, invariant theory and polynomial identity testing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 172–181. ACM, 2018.

[AKS20] S. Artstein-Avidan, H. Kaplan, and M. Sharir. On radial isotropic position: Theory and algorithms. *CoRR*, abs/2005.04918, 2020.

[AL88] D. Angluin and P. Laird. Learning from noisy examples. *Mach. Learn.*, 2(4):343–370, 1988.

[Bar98] F. Barthe. On a reverse form of the brascamp-lieb inequality. *Inventiones mathematicae*, 134:335–361, 1998.

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(84):929–965, October 1989.

[Bet04] U. Betke. New combinatorial and polynomial algorithms for the linear feasibility problem. *Discrete & Computational Geometry*, 32:317–338, 2004.

[BFG+18] P. Bürgisser, C. Franks, A. Garg, R. M. de Oliveira, Michael Walter, and A. Wigderson. Efficient algorithms for tensor scaling, quantum marginals, and moment polytopes. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 883–897. IEEE Computer Society, 2018.

[BFKV96] A. Blum, A. M. Frieze, R. Kannan, and S. Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96*, pages 330–338, 1996.

[BFKV97] A. Blum, A. Frieze, R. Kannan, and S. Vempala. A polynomial time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22(1/2):35–52, 1997.

[CKMY20] S. Chen, F. Koehler, A. Moitra, and M. Yau. Classification under misspecification: Halfspaces, generalized linear models, and evolvability. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[Coh97] E. Cohen. Learning noisy perceptrons by a perceptron in polynomial time. In *Proceedings of the Thirty-Eighth Symposium on Foundations of Computer Science*, pages 514–521, 1997.

[DGT19] I. Diakonikolas, T. Gouleakis, and C. Tzamos. Distribution-independent PAC learning of halfspaces with massart noise. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 4751–4762, 2019.

[DHNV20] D. Dadush, S. Huiberts, B. Natura, and L. A. Végh. A scaling-invariant algorithm for linear programming whose running time depends only on the constraint matrix. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 761–774. ACM, 2020.

[DK20] I. Diakonikolas and D. M. Kane. Near-optimal statistical query hardness of learning halfspaces with massart noise. *CoRR*, abs/2012.09720, 2020. Conference version in COLT'22.

[DKMR22] I. Diakonikolas, D. M. Kane, P. Manurangsi, and L. Ren. Cryptographic hardness of learning halfspaces with massart noise. *CoRR*, abs/2207.14266, 2022. Conference version in NeurIPS'22.

[DKT21] I. Diakonikolas, D. M. Kane, and C. Tzamos. Forster decomposition and learning halfspaces with noise. *CoRR*, abs/2107.05582, 2021. Conference version appeared in NeurIPS'21.

[DL01] L. Devroye and G. Lugosi. *Combinatorial methods in density estimation.* Springer Series in Statistics, Springer, 2001.

[DNV20] D. Dadush, B. Natura, and L. A. Végh. Revisiting tardos's framework for linear programming: Faster exact solutions using approximate solvers. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 931–942. IEEE, 2020.

[DPT21] I. Diakonikolas, J. Park, and C. Tzamos. Relu regression with massart noise. *CoRR*, abs/2109.04623, 2021. Conference version appeared in NeurIPS'21.

[DSW17] Z. Dvir, S. Saraf, and A. Wigderson. Superquadratic lower bound for 3-query locally correctable codes over the reals. *Theory Comput.*, 13(1):1–36, 2017.

[DV04a] J. Dunagan and S. Vempala. Optimal outlier removal in high-dimensional spaces. *J. Computer & System Sciences*, 68(2):335–373, 2004.

[DV04b] J. Dunagan and S. Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 315–320, 2004.

[For02] J. Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *J. Comput. Syst. Sci.*, 65(4):612–625, 2002.

[FS97] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[GdO18] A. Garg and R. M. de Oliveira. Recent progress on scaling algorithms and applications. *Bull. EATCS*, 125, 2018.

[GGdOW17] A. Garg, L. Gurvits, R. M. de Oliveira, and A. Wigderson. Algorithmic and optimization aspects of brascamp-lieb inequalities, via operator scaling. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 397–409. ACM, 2017.

[GHR92] M. Goldmann, J. Håstad, and A. Razborov. Majority gates vs. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2. Springer, 1988.

[GS02] L. Gurvits and A. Samorodnitsky. A deterministic algorithm for approximating the mixed discriminant and mixed volume, and a combinatorial corollary. *Discrete & Computational Geometry*, 27:531–550, 2002.

[GT89] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *J. ACM*, 36(4):873–886, 1989.

[GV19] J. Garg and L. A. Végh. A strongly polynomial algorithm for linear exchange markets. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019*, pages 54–65. ACM, 2019.

[HKLM20] M. Hopkins, D. Kane, S. Lovett, and G. Mahajan. Point location and active learning: Learning halfspaces almost optimally. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 1034–1044. IEEE, 2020.

[HM13] M. Hardt and A. Moitra. Algorithms and hardness for robust subspace recovery. In *COLT 2013*, pages 354–375, 2013.

[HM19] L. Hamilton and A. Moitra. The paulsen problem made simple. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019*, volume 124 of *LIPIcs*, pages 41:1–41:6. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[KLLR18] T. C. Kwok, L. C. Lau, Y. T. Lee, and A. Ramachandran. The paulsen problem, continuous operator scaling, and smoothed analysis. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 182–189. ACM, 2018.

[KV94] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, Cambridge, MA, 1994.

[LSW00] N. Linial, A. Samorodnitsky, and A. Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. *Comb.*, 20(4):545–568, 2000. Conference version in STOC'98.

[Meg83] N. Megiddo. Towards a genuinely polynomial algorithm for linear programming. *SIAM Journal on Computing*, 12(2):347–353, 1983.

[MN06] P. Massart and E. Nedelec. Risk bounds for statistical learning. *Ann. Statist.*, 34(5):2326–2366, 10 2006.

[MP68] M. Minsky and S. Papert. *Perceptrons: an introduction to computational geometry.* MIT Press, Cambridge, MA, 1968.

[MT94] W. Maass and G. Turan. How fast can a threshold gate learn? In S. Hanson, G. Drastal, and R. Rivest, editors, *Computational Learning Theory and Natural Learning Systems*, pages 381–414. MIT Press, 1994.

[Nov62] A. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on Mathematical Theory of Automata*, volume XII, pages 615–622, 1962.

[NT22] R. Nasser and S. Tiegel. Optimal SQ lower bounds for learning halfspaces with Massart noise. *CoRR*, abs/2201.09818, 2022. Conference version in COLT'22.

[O'D14] R. O'Donnell. *Analysis of Boolean Functions.* Cambridge University Press, 2014.

[Orl93] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2):338–350, 1993.

[OV17] N. Olver and L. A. Végh. A simpler and faster strongly polynomial algorithm for generalized flow maximization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 100–111. ACM, 2017.

[OV20] N. Olver and L. A. Végh. A simpler and faster strongly polynomial algorithm for generalized flow maximization. *J. ACM*, 67(2):10:1–10:26, 2020.

[Par98] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Society for Industrial and Applied Mathematics, Philadelphia, 1998.

[Ros58] F. Rosenblatt. The Perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

[Sma98] S. Smale. Mathematical problems for the next century. *The Mathematical Intelligencer*, 20:7–15, 1998.

[STC00] J. Shawe-Taylor and N. Cristianini. *An introduction to support vector machines.* Cambridge University Press, 2000.

[Tar85] E. Tardos. A strongly polynomial minimum cost circulation algorithm. *Comb.*, 5(3):247–256, 1985.

[Tar86] E. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.

[Val84] L. G. Valiant. A theory of the learnable. In *Proc. 16th Annual ACM Symposium on Theory of Computing (STOC)*, pages 436–445. ACM Press, 1984.

[Vap98] V. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, New York, 1998.

[Vég14] L. A. Végh. A strongly polynomial algorithm for generalized flow maximization. In *Symposium on Theory of Computing, STOC 2014*, pages 644–653. ACM, 2014.

[Vég16] L. A. Végh. A strongly polynomial algorithm for a class of minimum-cost flow problems with separable convex objectives. *SIAM J. Comput.*, 45(5):1729–1761, 2016.

[VY96] S. A. Vavasis and Y. Ye. A primal-dual interior point method whose running time depends only on the constraint matrix. *Math. Program.*, 74:79–120, 1996.

[Yao90] A. Yao. On ACC and threshold circuits. In *Proceedings of the Thirty-First Annual Symposium on Foundations of Computer Science*, pages 619–627, 1990.

APPENDIX

A Proof of Fact 2.1

We show each property separately.

(a) This follows directly from the fact that the transformation $f_A(x) = \frac{Ax}{\|Ax\|_2}$ is scale-invariant.

(b) We have that $f_B(f_A(x)) = f_B\left(\frac{Ax}{\|Ax\|_2}\right) = f_B(Ax) = \frac{BAx}{\|BAx\|_2} = f_{BA}(x)$, where the second equality follows from part (a).

(c) Let $\alpha = \|B - I\|_2$ and $y = f_A(x)$. Note that $\|y\|_2 = 1$. By property (b), we equivalently want to show that $\|f_B(y) - y\|_2 \leq \alpha$. We have that

$$\|f_B(y) - y\|_2 = \left\|\frac{B}{\|By\|_2}y - y\right\|_2 \leq \max_{v \in \mathbb{R}^d : \|v\|_2 = 1} \left\|\frac{B}{\|By\|_2}v - v\right\|_2 = \left\|\frac{B}{\|By\|_2} - I\right\|_2 .$$

The desired statement follows from the fact that the matrix $\frac{B}{\|By\|_2}$ has eigenvalues between $\frac{1}{1+\alpha}$ and $1+\alpha$, as B has eigenvalues between 1 and $1+\alpha$ and $\|By\|_2 \in [1, 1+\alpha]$.

(d) We have that

$$f_{BA}^{(V)}(x) = \text{proj}_V f_B(f_A(x)) = \text{proj}_V \frac{(I + aI_V)f_A(x)}{\|Bf_A(x)\|_2} = \frac{(1+a)f_A^{(V)}(x)}{\|Bf_A(x)\|_2} .$$

Since $1 \leq \|Bf_A(x)\|_2 \leq 1 + a$, it follows that $1 \leq \lambda(x) \overset{\text{def}}{=} \frac{(1+a)}{\|Bf_A(x)\|_2} \leq 1 + a$, as desired. Similarly, we can write

$$f_{BA}^{(V^\perp)}(x) = \frac{f_A^{(V^\perp)}(x)}{\|Bf_A(x)\|_2} .$$

Since $1 \leq \|Bf_A(x)\|_2 \leq 1 + a$, it follows that $\frac{1}{1+a} \leq \mu(x) \overset{\text{def}}{=} \frac{1}{\|Bf_A(x)\|_2} \leq 1$.

This completes the proof of Fact 2.1.