Learning Arithmetic Formulas in the Presence of Noise: A General Framework and Applications to Unsupervised Learning

Pritam Chandra ✉
Microsoft Research, Bangalore, India

Ankit Garg ✉
Microsoft Research, Bangalore, India

Neeraj Kayal ✉
Microsoft Research, Bangalore, India

Kunal Mittal ✉
Princeton University, NJ, USA

Tanmay Sinha ✉
Microsoft Research, Bangalore, India

Abstract

We present a general framework for designing efficient algorithms for unsupervised learning problems, such as mixtures of Gaussians and subspace clustering. Our framework is based on a meta algorithm that learns arithmetic formulas in the presence of noise, using lower bounds. This builds upon the recent work of Garg, Kayal and Saha (FOCS '20), who designed such a framework for learning arithmetic formulas without any noise. A key ingredient of our meta algorithm is an efficient algorithm for a novel problem called *Robust Vector Space Decomposition*. We show that our meta algorithm works well when certain matrices have sufficiently large smallest non-zero singular values. We conjecture that this condition holds for smoothed instances of our problems, and thus our framework would yield efficient algorithms for these problems in the smoothed setting.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Arithmetic Circuits, Robust Vector Space Decomposition, Subspace Clustering, Mixtures of Gaussians

Digital Object Identifier 10.4230/LIPIcs.ITCS.2024.25

Related Version *Full Version*: `https://arxiv.org/abs/2311.07284` [6]

1 Introduction

Unsupervised learning involves discovering hidden patterns and structure in data without using any labels or direct human supervision. Here we consider data that has a nice mathematical structure or is generated from a mathematically well-defined distribution. An example of the former is when the data points can be grouped into meaningful clusters based on some similarity patterns and the goal is to find the underlying clusters. An example of the latter is mixture modeling, which assumes that the data is generated from a mixture of succinctly described probability distributions, such as Gaussian distributions, and the goal is to learn the parameters of these distributions from samples. A general framework for solving many unsupervised learning problems is the method of moments, which leverages the statistical moments[1] of the data to infer the underlying structure or the underlying

[1] Recall that moments are measures of the shape and variability of a data set. They are used to describe the the location and dispersion of the data. When the dataset consists of a collection of points

$$A = \{\mathbf{a}_i = (a_{i1}, a_{i2}, \ldots, a_{in}) \in \mathbb{R}^n \quad | \quad i \in [N]\},$$

some examples of (low-order) moments are $\underset{\mathbf{a}_i \in A}{\mathbb{E}}[a_{i1}]$, $\underset{\mathbf{a}_i \in A}{\mathbb{E}}[a_{i1} \cdot a_{i2}]$, etc.

parameters of the model. For many unsupervised learning problem scenarios wherein the underlying data has some nice mathematical structure, the moments of the data are well-defined functions of the parameters. Heuristic arguments then suggest that the converse should typically hold, i.e. the parameters of the structure/distribution are *typically* uniquely determined by a few low order moments of the data. In this broad direction, the main challenge then is to design algorithms to (approximately) recover the underlying parameters from the (empirical) moments[2]. We further want the algorithm to be *efficient*, noise-tolerant (i.e. work well even when the moments are known only approximately rather than exactly) and are even *outlier-tolerant* (i.e. work well even when a few data points do not conform to the underlying structure/distribution). But even the simplest problems in this area tend to be NP-hard and remain so even when there is no noise and no outliers. So one cannot realistically hope for an algorithm with provable worst-case guarantees. But what one can hope are algorithms that are guaranteed to typically work well, i.e. either for *random* problem instances or even more desirably for instances chosen in a smoothed fashion. Accordingly, many different algorithms have been designed for each such problem in unsupervised learning with varying levels of efficiency, noise-tolerance, outlier-tolerance and provable guarantees. In this work we give a *single meta-algorithm* that applies to many such unsupervised learning problems. The starting point of our work is the observation that many such problems reduce to the task of learning an appropriate subclass of arithmetic formulas.

Connecting unsupervised learning to arithmetic complexity. We now give a few more details of how such a reduction works for the setting in which the data points are drawn from a distribution having a nice mathematical structure. Let $\mathcal{D}$ be a distribution over points in $\mathbb{R}^n$. We introduce n formal variables $(x_1, x_2, \ldots, x_n)$ and denote it as $\mathbf{x}$. For a suitably chosen integer $d \geq 2$, form a degree-d polynomial $f(\mathbf{x})$ which encodes the d-th order moments[3] of the distribution in some suitable way. For example, in some applications the coefficient of a monomial of $f(\mathbf{x})$ is simply (a canonically scaled version of) the corresponding moment. At this point, such a formal polynomial is a mere bookkeeping device for the d-th order moments of $\mathcal{D}$. For many nice, well-structured distributions such as mixtures of Gaussians, however this polynomial (or variants thereof) turns out to have a remarkable property - it can be computed/represented by a small arithmetic formula! Special cases of this remarkable phenomenon were noted earlier when it was observed that many problems in (unsupervised) learning reduce to the problem of learning set-multilinear depth-three formulas, better known as tensor decomposition. Such connection(s) inspired a whole body of work on tensor decomposition with applications including independent component analysis, learning Hidden Markov Models, learning special cases of mixtures of Gaussians, latent Dirichlet allocation, dictionary learning, etc. (cf. the surveys [19, 14, 1]).

Noise-tolerance. Notice however that we are given a finite set of points sampled from the distribution $\mathcal{D}$, so we do not have the (d-th order) moments of $\mathcal{D}$ exactly but only approximately. Thus for such applications we need the algorithm for learning arithmetic formulas to also be *noise-tolerant*, i.e. given a polynomial $\tilde{f}(\mathbf{x})$ that is *close to*[4] a polynomial $f(\mathbf{x})$ that has a small arithmetic formula ϕ, we want to learn/reconstruct a arithmetic

[2] In scenarios where the data is a finite sample drawn from a distribution $\mathcal{D}$ over $\mathbb{R}^n$, the empirical moments (which can be very easily and efficiently computed) are estimates of, but *not* equal to, the true underlying moments.

[3] We are making the mild assumption here that the d-th order moments of $\mathcal{D}$ are bounded.

[4] Under a natural notion of distance between a pair of polynomials akin to Euclidean distance between the coefficient vectors of the pair of polynomials - see section 2 in [6].

formula $\tilde{\phi}$ from the same subclass as that of ϕ whose output polynomial is close to $\tilde{f}(\mathbf{x})$ (and therefore to $f(\mathbf{x})$ as well). Recently, [10] gave a meta-algorithm for learning many different subclasses of formulas including the ones relevant for unsupervised learning (assuming that certain nondegeneracy conditions hold). But it has one important shortcoming that was also pointed out in [4]: the techniques of [10] were algebraic and it was unclear if they could handle noise arising out of the fact that the moments are known only approximately and not exactly. Qualitatively, our main result builds upon and suitably adapts the algorithm [10] to make it noise-tolerant. Quantitatively, in the noisy setting, we provide bounds on the quality of the output of our algorithm that depend on singular values of certain matrices that underlie the algorithm. We expect that for most applications, the relevant singular values would be well-behaved for random instances and maybe even for smoothed/perturbed worst-case instances. If so, our algorithm would work and yield good quality outputs on such instances. Accordingly, we then go on on to analyze the singular values of the relevant matrices pertaining to subspace clustering[5]. We also expect (suitable adaptations of) our algorithm to be tolerant to the presence of a few outliers but we do not pursue this direction here and leave it for future work.

Illustrative example – mixtures of Gaussians. Let us make the above discussion concrete via the example of learning mixtures of Gaussians which in itself is a very well-studied problem with history going back to more than a hundred years. Suppose we are given a dataset consisting of a finite set of points $A \subset \mathbb{R}^n$

$$A = \{\mathbf{a}_i = (a_{i1}, a_{i2}, \ldots, a_{in}) \in \mathbb{R}^n \quad | \quad i \in [N]\}. \tag{1}$$

The points are drawn independently at random from an unknown mixture of s Gaussians $\mathcal{D} := \sum_{i=1}^{s} w_i \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$, which means that the i-th component of the mixture has weight[6] $w_i \in [0,1]$, mean $\boldsymbol{\mu}_i \in \mathbb{R}^n$ and covariance matrix $\Sigma_i \in \mathbb{R}^{n \times n}$. Our goal is to estimate the parameters w_i and $\boldsymbol{\mu}_i$ and Σ_i ($i \in [s]$) from the given samples/data A. Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be a tuple of formal variables and consider the polynomial $f(\mathbf{x}) := \mathbb{E}_{\mathbf{a} \sim \mathcal{D}}\left[\langle \mathbf{x}, \mathbf{a} \rangle^d\right]$. It is (a scalar multiple of) a *slice of* the formal moment generating function defined as $\mathbb{E}_{\mathbf{a} \sim \mathcal{D}}\left[\exp(\langle \mathbf{x}, \mathbf{a} \rangle)\right]$. Notice that the coefficients of a given monomial (over $\mathbf{x}$) in $f(\mathbf{x})$ equals the corresponding moment of the distribution (upto some canonical scaling). Then in this case, $f(\mathbf{x})$ has the following small formula[7]:

$$f(\mathbf{x}) = \sum_{i \in [s]} w_i G_d(\ell_i(\mathbf{x}), Q_i(\mathbf{x})),$$

where $\ell_i(\mathbf{x}) := \langle \boldsymbol{\mu}_i, \mathbf{x} \rangle$, $Q_i(\mathbf{x}) := \frac{1}{2}\mathbf{x}^T \Sigma_i \mathbf{x}$ and G_d is a fixed bivariate polynomial depending on d. In the zero-mean case (i.e. when $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \ldots = \boldsymbol{\mu}_s = \mathbf{0}$), the formula for $f(\mathbf{x})$ is

$$f(\mathbf{x}) = \sum_{i \in [s]} \frac{d!}{(d/2)!} w_i Q_i(\mathbf{x})^{d/2}$$

when d is even (and 0 if d is odd). In this way, if the sample size was infinite (or equivalently that if we knew the true moments of the distribution), learning mixtures of Gaussians would

[5] A recent work [3] analyzed the singular values of matrices arising in a related (but also different) algorithm that was tailor-made for the mixtures of (zero-mean) Gaussians and verified that for random instances the singular values are indeed well-behaved.

[6] The weights satisfy $\sum_{i \in [s]} w_i = 1$.

[7] This formula for $f(\mathbf{x})$ can be inferred from the fact that for a single Gaussian distribution $N(\boldsymbol{\mu}, \Sigma)$, its moment generating function is in fact equal to $\exp(\mathbf{x}^T \cdot \boldsymbol{\mu} + \frac{1}{2}\mathbf{x}^T \cdot \Sigma \cdot \mathbf{x})$.

reduce to the problem of learning/reconstructing the subclass of arithmetic formulas indicated by the rhs of the above expression for $f(\mathbf{x})$. But we don't have access to the exact moments. Using the empirical moments, we can get hold of an approximate version of f,

$$\widetilde{f}(\mathbf{x}) := \mathbb{E}_{\mathbf{a}\sim A}\left[\langle \mathbf{x}, \mathbf{a}\rangle^d\right] = \frac{1}{N}\sum_{i\in[N]}\left[\langle \mathbf{x}, \mathbf{a}_i\rangle^d\right].$$

We will have $\widetilde{f}(\mathbf{x}) = f(\mathbf{x}) + \eta(\mathbf{x})$ for a noise polynomial $\eta(\mathbf{x})$ whose magnitude will be inversely proportional to square root of the number of samples N. In this way, learning mixtures of Gaussians reduces to the problem of reconstructing the indicated subclass of arithmetic formulas in the presence of noise.

Learning arithmetic formulas in the presence of noise – problem formulation. The above discussion motivates us to consider the problem of learning (arbitrary subclasses of) arithmetic formulas in the presence of noise. In many practical settings the output gate of the underlying formula is a (generalized[8]) addition gate so that the problem can be formulated as follows. We are given a polynomial $\widetilde{f}(\mathbf{x})$ of the form $\widetilde{f}(\mathbf{x}) := T_1(\mathbf{x}) + \cdots + T_s(\mathbf{x}) + \eta(\mathbf{x})$, for *structured* polynomials $T_i(\mathbf{x})$'s and a noise polynomial $\eta(\mathbf{x})$. Our goal is to approximately recover each summand $T_i(\mathbf{x})$. For example, for the case of mixture of spherical Gaussians we would have $T_i(\mathbf{x}) = w_i \cdot \langle \boldsymbol{\mu}_i, \mathbf{x}\rangle^3$ (see Remark 1.1). For the case of mixture of zero-mean Gaussians we would have $T_i(\mathbf{x}) = w_i \cdot Q_i(\mathbf{x})^{d/2}$ and so on. In the noiseless setting, i.e. when $\eta(\mathbf{x}) = 0$, the paper [10] designed a meta-algorithm applicable to learning many interesting subclasses using a general framework exploiting lower bound techniques in arithmetic complexity theory. The algorithm worked under certain relatively mild non-degeneracy assumptions. However, their algorithm had some algebraic components and it was not clear how to design an algorithm in the noisy case when the noise polynomial $\eta(\mathbf{x})$ is non-zero.[9] Our main contribution is that we show how to modify the general framework in [10] to the noisy setting. We also show how to use this framework to design efficient algorithms for two well studied problems in unsupervised learning: mixtures of (zero mean) Gaussians and subspace clustering.

▶ Remark 1.1.

(a). Simpler reductions. The ability to handle arbitrary subclasses of arithmetic formulas not only yields a common (meta) algorithm that applies to a wide variety of problems in unsupervised learning but it also often makes the reductions simpler. For example, in the discussion above the reduction of learning mixtures of arbitrary Gaussians to learning the appropriate subclass of arithmetic formulas is perhaps simpler than the reduction of learning mixtures of spherical Gaussians[10] to tensor decomposition. We sketch this reduction now. Consider $f(\mathbf{x}) := \mathbb{E}_{\mathbf{a}\sim\mathcal{D}}\left[\langle \mathbf{x}, \mathbf{a}\rangle^3 - 3\left(\sum_{i\in[n]} x_i^2\right)\cdot\langle \mathbf{x}, \mathbf{a}\rangle\right]$. When $\mathcal{D}$ is a mixture of spherical Gaussians, expanding and simplifying this expression, we can get that $f(\mathbf{x}) = \sum_{i=1}^{s} w_i\langle \boldsymbol{\mu}_i, \mathbf{x}\rangle^3$.

(b). Mixtures of general Gaussians. We expect that our algorithm can be extended to general mixtures of Gaussians (different means and/or covariance matrices) but its analysis will likely get much more cumbersome, so we avoid this more general case for the sake of simplicity.

[8] A generalized addition gate can compute any fixed linear combination of its inputs.

[9] In most settings, one would like the running time of the algorithm to be inverse polynomial in the magnitude of the noise, to have a polynomial dependence on the number of samples in the final learning problem.

[10] A spherical Gaussian is one where the covariance matrix Σ_i is the identity matrix.

(c). **Handling outliers.** The ability to handle arbitrary subclasses of arithmetic formulas can also allow the algorithm to be tolerant to the presence of outliers. To see this, consider the case of zero-mean Gaussians and suppose that the given set of data points A contains a subset $\hat{A} \subset A$ of outliers of size $\hat{N} \ll N$. In that case the empirical moment polynomial $\widetilde{f}(\mathbf{x})$ would have the following structure:

$$\widetilde{f}(\mathbf{x}) = \frac{N-\hat{N}}{N} \cdot \frac{d!}{(d/2)!} \cdot \left(\sum_{i \in [s]} w_i Q_i(\mathbf{x})^{d/2} \right) + \frac{1}{N} \cdot \left(\sum_{\mathbf{a}_j \in \hat{A}} (\mathbf{x} \cdot \mathbf{a}_j)^d \right) + \eta(\mathbf{x}).$$

We expect that our algorithm can be adapted to learn the class of formulas corresponding to the right side of the above expression however the analysis of such an algorithm can get cumbersome. For the sake of keeping the length of this paper to within reasonable bounds, we do not do the analysis of the outlier tolerance of our algorithm.

(d). **Other mixtures models.** The connection between learning mixtures of Gaussians and learning an appropriate subclass of arithmetic formulas arose out of the fact that (any slice of) the moment generating function of a multivariate Gaussian has a simple algebraic expression. For some other distributions also the (slices of) moment generating function or some other related function like the cumulant generating function or the characteristic function have a nice algebraic expression and we can expect our approach to be applicable for such mixtures also.

(e). **Mixtures of structured point sets and those sampled from probability distributions.** Consider a set of points $A \subset \mathbb{R}^n$ that can be partitioned into two subsets $A = A_1 \uplus A_2$ such that A_1 is some structured set of points (such as being contained in the union of a small number of low-dimensional subspaces for example) and A_2 is chosen from some mixture model (such as being chosen from a mixture of Gaussians for example). When say the moment polynomials of both the structured set A_1 and the sampled set A_2 admit small formulas from a tractable subclass of formulas, we can expect our methods to apply. In particular, we expect (a suitable adaptation of) our algorithm to be to handle the case where points in A_1 are chosen from a union of low-dimensional subspaces in an non-degenerate way without conforming to any nice distribution and points in A_2 conform to a (mixture of) Gaussians. We leave the task of handling such mixed datasets and analyzing the relevant algorithms as a possible direction for future work.

(f). **Potential application – Topic Modeling.** It turns out that there are some other problems in unsupervised learning which reduce to robustly learning an appropriate subclass of arithmetic formulas. We expect that a suitable instantiation/adaptation of our algorithm should apply for these applications but we do not pursue these applications here and leave it as a direction for future work. One such problem is called topic modelling. It is known that learning some simple topic models reduce to tensor decomposition. It turns out that learning some general topic models as proposed in [20] reduce to the problem of learning set-multilinear formulas of larger depth.

(g). **Potential application – Learning (Mixtures of) Polynomial Transformations.** Another such application is the problem of learning polynomial transformations as studied in [7] which also reduces to learning a certain subclass of arithmetic formulas[11]. The generality of our approach makes us expect that it should apply to this task also as well as to its generalizations like learning **mixtures of** polynomial transformations. We do not pursue this potential application here but leave it as a direction for future work.

[11] The work of [7] does not state it this way but this can be inferred from the observations underlying their work.

2 Overview – Arithmetic Formula Learning algorithm

Background. Arithmetic formulas are a natural model of computing polynomials using the basic operations of addition ($+$) and multiplication ($\times$). A natural problem about arithmetic formulas is that of learning: given a polynomial $f(\mathbf{x})$[12], find the smallest (or somewhat small) arithmetic formula computing $f(\mathbf{x})$. We consider formulas in their alternating normal form: i.e. the formula consists of alternating layers of addition and multiplication gates. The learning problem boils down to recovering the polynomials computed at each child of a node v given the polynomial computed at v. When v is a multiplication node then generically, the polynomials computed at its children are irreducible[13] in which case the efficient multivariate polynomial factorization algorithm of Kaltofen and Trager [13] recovers the children's outputs. Even when there is noise, the robust factorization algorithm of [12] can recover the factors approximately[14]. Thus the main challenge is to recover the children of addition gates. This connects us to the problem discussed in the previous section with the *structured* polynomials being the polynomials computed at the children gates. In the noiseless setting, a meta algorithm for this problem was given in [10]. We provide a meta algorithm in the noisy case and show worst-case bounds on the quality of the output in terms of singular values of certain matrices [15]. The abstract problem is as follows. Given a polynomial $\widetilde{f}(\mathbf{x})$ that can be expressed as

$$\widetilde{f}(\mathbf{x}) = T_1(\mathbf{x}) + T_2(\mathbf{x}) + \ldots + T_s(\mathbf{x}) + \eta(\mathbf{x}), \tag{2}$$

where T_i's are *structured* polynomials and the noise/perturbation polynomial $\eta(\mathbf{x})$ has *small norm*, can we approximately recover the T_i's via an efficient algorithm?

Learning from lower bounds. [10] showed how the linear maps used in the known arithmetic formula lower bound proofs could be used to recover the T_i's in the noiseless ($\eta = 0$) setting, *assuming that appropriate non-degeneracy conditions hold.* [10] observed that the assumption that the T_i's are structured can effectively be operationalized via the existence of a known set of linear maps $\mathcal{L}$ from the vector space of polynomials to some appropriate vector space W_1 such that $\dim(\langle \mathcal{L} \cdot T_i \rangle)$ is[16] *small* for every simple polynomial T_i. When we apply such a set of linear operator $\mathcal{L}$ to (2) with $\eta = 0$, we get:

$$\left\langle \mathcal{L} \cdot \widetilde{f}(\mathbf{x}) \right\rangle \subseteq \langle \mathcal{L} \cdot T_1(\mathbf{x}) \rangle + \langle \mathcal{L} \cdot T_2(\mathbf{x}) \rangle + \ldots + \langle \mathcal{L} \cdot T_s(\mathbf{x}) \rangle . \tag{3}$$

[10] observe that generically two things tend to happen.

[12] There are various input models all of which lead to interesting questions. Some of the common ones are as a black box or described explicitly as a list of coefficients.

[13] Random multivariate polynomials are almost surely irreducible and with that as intuition, one expects the output of a formula with output being an addition gate to almost surely be an irreducible polynomial when the underlying field constants are chosen randomly. However proving this can be technically involved for any given subclass of formulas.

[14] The work of [13] aims to devise a factorization algorithm that is empirically as robust as possible and does not contain theoretical bounds on how much the output factors get perturbed as a function of the noise added to a true factorization. Nevertheless such a bound can be inferred from their work. The bound would depend on the appropriate singular values of an instance-dependent matrix called the Ruppert matrix that comes up in their algorithm.

[15] The matrices whose singular values are used to bound the quality of the output depend on the input instance as well as on the choice of linear operators used to instantiate our framework

[16] Here, $\langle S \rangle$ denotes the $\mathbb{R}$-linear span of a set S that consists of vectors or linear maps. Also $\mathcal{L} \cdot T_i$ denotes the set of vectors obtained by applying each linear map in $\mathcal{L}$ to T_i.

▶ **Assumption 2.1** (First blessing of dimensionality[17]). *If* $\sum_{i\in[s]} \dim(\langle \mathcal{L} \cdot T_i(\mathbf{x})\rangle) \ll \dim(W_1)$ *then almost surely (over the independent random choice of the T_i's), it holds that the subspaces $\langle \mathcal{L} \cdot T_i(\mathbf{x})\rangle$ form a direct sum, i.e.*

$$\dim(\langle \mathcal{L} \cdot T_1(\mathbf{x})\rangle + \langle \mathcal{L} \cdot T_2(\mathbf{x})\rangle + \ldots + \langle \mathcal{L} \cdot T_s(\mathbf{x})\rangle) = \sum_{i\in[s]} \dim(\langle \mathcal{L} \cdot T_i(\mathbf{x})\rangle).$$

▶ **Assumption 2.2** (Second blessing of dimensionality[18]). *If* $\sum_{i\in[s]} \dim(\langle \mathcal{L} \cdot T_i(\mathbf{x})\rangle) \ll \dim(\langle \mathcal{L}\rangle)$ *then almost surely (over the independent random choice of the T_i's), it holds that for all $i \in [s]$:*

$$\langle \mathcal{L} \cdot (T_1(\mathbf{x}) + \ldots + T_s(\mathbf{x}))\rangle \supseteq \langle \mathcal{L} \cdot T_i(\mathbf{x})\rangle .$$

Under these nondegeneracy assumptions we then have (for $\eta = 0$):

$$U \overset{\text{def}}{=} \left\langle \mathcal{L} \cdot \widetilde{f}(\mathbf{x}) \right\rangle = \langle \mathcal{L} \cdot T_1(\mathbf{x})\rangle \oplus \langle \mathcal{L} \cdot T_2(\mathbf{x})\rangle \oplus \ldots \oplus \langle \mathcal{L} \cdot T_s(\mathbf{x})\rangle . \tag{4}$$

We observe that in the noisy case, on input $\widetilde{f}$, finding the best $(\dim(U))$-rank subspace through the set of points $(\mathcal{L} \circ \widetilde{f})$ yields a subspace $\widetilde{U}$ that is pretty close to U (lemma D.1 in [6] gives quantitative bounds). Coming back to the noiseless case, [10] then observe that linear maps constructed for the purpose of proving lower bounds also yield a set of linear maps $\mathcal{B}$ such that

$$V \overset{\text{def}}{=} \langle \mathcal{B} \cdot U\rangle = \langle \mathcal{B} \cdot U_1\rangle \oplus \cdots \oplus \langle \mathcal{B} \cdot U_s\rangle, \tag{5}$$

where $U_i \overset{\text{def}}{=} \langle \mathcal{L} \cdot T_i(\mathbf{x})\rangle$. This motivated the following problem which they call *Vector Space Decomposition*. Given a set of linear maps $\mathcal{B}$ between two vector spaces U and V, find a (maximal) decomposition $U = U_1 \oplus \cdots \oplus U_s$, $V = V_1 \oplus \cdots \oplus V_s$ s.t. $\mathcal{B} \cdot U_i \subseteq V_i$ for all $i \in [s]$. In most applications, such a decomposition turns out to be unique (up to some obvious symmetries like permuting the subspaces) and hence an algorithm for vector space decomposition finds the intended decomposition.

Reduction to Vector Space Decomposition in the noisy setting. We then formulate and give an algorithm for a robust/noise-tolerant version of vector space decomposition. But there is an important difficulty that crops up in trying to use the problem of robust vector space decomposition as formulated below to the setting of learning arithmetic formulas in the presence of noise. $\mathcal{B}$ is a collection of maps from W_1 to W_2 where $U \subseteq W_1, V \subseteq W_2$ and $\mathcal{B} \cdot U$ equals V. However, $\mathcal{B} \cdot \widetilde{U}$ will typically *not* be contained in $\widetilde{V}$. In fact the dimension of the image of $\widetilde{U}$ under the action of $\mathcal{B}$ (denoted $\dim\left(\left\langle \mathcal{B} \cdot \widetilde{U}\right\rangle\right)$) will typically be much larger than the dimension of $\widetilde{V}$ (denoted $\dim(\widetilde{V})$). To overcome this difficulty, our idea is to compose maps in $\mathcal{B}$ with the projection[19] map to $\widetilde{V}$ to obtain a tuple of maps $\widetilde{\mathcal{B}}$ from $\widetilde{U}$ to

[17] The intuition is that (pseudo)-randomly chosen small-dimensional subspaces of a large-dimensional ambient space should form a direct sum.

[18] The intuition is that in most applications when the underlying dimension $n = |\mathbf{x}|$ is large enough then the dimension of the set of operators $\mathcal{L}$ is large relative to $\dim(\langle \mathcal{L} \cdot T_i(\mathbf{x})\rangle)$ for any i. In such a situation if the T_i's are chosen generically then $\mathcal{L}$ tends to contain many operators that kill all the other T_j's (for $j \neq i$) so that $\langle \mathcal{L} \cdot f(\mathbf{x})\rangle$ tends to contain each of the subspaces $\langle \mathcal{L} \cdot T_i(\mathbf{x})\rangle$.

[19] Projection to $\widetilde{V}$ here implicitly uses a decomposition of the ambient space W_2 into $\widetilde{V}$ and its orthogonal complement (defined via some canonical inner product on W_2 that is clear from context). It is the unique map in $\mathrm{Lin}(W_2, W_2)$ which is identity on $\widetilde{V}$ and whose kernel is the the orthogonal complement of $\widetilde{V}$.

$\widetilde{V}$. In general such a composition can completely spoil the structure of the set of maps $\mathcal{B}$ but our conceptual insight here is that in this situation, one can set up a natural correspondence between $\mathrm{Lin}(U, V)$ and $\mathrm{Lin}(\widetilde{U}, \widetilde{V})$ that can be used to infer that the projection-composed maps $\widetilde{\mathcal{B}}$ *are slight perturbations* of the corresponding maps in $\mathcal{B}$ (lemma 4.1 in [6] gives quantitative bounds). This insight gives us the reduction. Then, the robust vector space decomposition algorithm yields a decomposition

$$\widetilde{U} = \widetilde{U}_1 \oplus \widetilde{U}_2 \oplus \ldots \oplus \widetilde{U}_s \tag{6}$$

where $\widetilde{U}_1, \widetilde{U}_2, \ldots, \widetilde{U}_s$ are slightly perturbed versions of $\langle \mathcal{L} \cdot T_1(\mathbf{x}) \rangle, \langle \mathcal{L} \cdot T_2(\mathbf{x}) \rangle, \ldots, \langle \mathcal{L} \cdot T_s(\mathbf{x}) \rangle$ respectively (corollary 4.1 in [6] gives quantitative bounds). In particular this implies that for each $L \in \mathcal{L}$ we can obtain a vector close to $L \cdot T_1(\mathbf{x})$ by *projecting*[20] $L \cdot \widetilde{f}(\mathbf{x})$ to $\widetilde{U}_1$. This implies that we can approximately recover $T_1(\mathbf{x})$ itself via an appropriate pseudo-inverse computation. Similarly, we can recover all the $T_i(\mathbf{x})$'s up to some error (Theorem 14 in [6] gives quantitative bounds). Before stating the quantitative bound on this error (Theorem 3.3) let us discuss the subroutine of robust vector space decomposition which is perhaps of interest in itself and might have wider applicability.

3 Overview – Vector Space Decomposition algorithm

We refer to the noise-tolerant version of vector space decomposition as *Robust Vector Space Decomposition (RVSD)*. The setting is the following: let W_1 and W_2 be vector spaces, and let $U = U_1 \oplus \cdots \oplus U_s \subseteq W_1$ and $V = V_1 \oplus \cdots \oplus V_s \subseteq W_2$ be subspaces. Let $\mathcal{B} = (B_1, B_2, \ldots, B_m)$ be an m-tuple of linear operators, with each $B_j : U \to V$ being a linear map from U to V. Suppose that, under the action of $\mathcal{B}$, each U_i is mapped inside V_i; that is, for each $i \in [s]$, it holds that $\langle \mathcal{B} \cdot U_i \rangle \subseteq V_i$. We consider the problem of recovering the U_i's approximately given noisy access to U, V and $\mathcal{B}$. Specifically[21]

Robust Vector Space Decomposition (RVSD). We are given as input the integer s, two vector spaces $\widetilde{U} \subseteq W_1$ and $\widetilde{V} \subseteq W_2$, and a m-tuple of operators $\widetilde{\mathcal{B}} = (\widetilde{B}_1, \widetilde{B}_2, \ldots, \widetilde{B}_m)$ from $\widetilde{U}$ to $\widetilde{V}$, such that $\mathrm{dist}(\widetilde{U}, U)$, $\mathrm{dist}(\widetilde{V}, V)$ and $\mathrm{dist}(\widetilde{\mathcal{B}}, \mathcal{B})$[22] are "small". Our goal is to *efficiently* find an s-tuple $\widetilde{\mathbf{U}} = (\widetilde{U}_1, \widetilde{U}_2, \ldots, \widetilde{U}_s)$ of subspaces in $\widetilde{U} \subseteq W_1$, such that (upto a reordering of the components) for each $i \in [s]$, $\mathrm{dist}(\widetilde{U}_i, U_i)$ is "small"[23].

Now we give some rough ideas that go behind our Robust Vector Space Decomposition algorithm. For more details, the reader is referred to Section 4 in [6] . Let us first consider the noiseless setting, in which we are given an integer s, the vector spaces $U \subseteq W_1, V \subseteq W_2$, and a m-tuple of operators $\mathcal{B} = (B_1, \ldots, B_m)$ from $U \to V$; the goal is to find a decomposition $U = U_1 \oplus \cdots \oplus U_s$ and $V = V_1 \oplus \cdots \oplus V_s$, such that each U_i is mapped into V_i under the action of $\mathcal{B}$, i.e.

$$U = U_1 \oplus U_2 \oplus \ldots \oplus U_s \quad \text{and } V = V_1 \oplus V_2 \oplus \ldots \oplus V_s, \quad \langle \mathcal{B} \cdot U_i \rangle \subseteq V_i \quad \forall i \in [s]. \tag{7}$$

[20] Projection to $\widetilde{U}_1$ here refers to using the decomposition given by (6). It is applying the unique map in $\mathrm{Lin}(\widetilde{U}, \widetilde{U})$ which is identity on $\widetilde{U}_1$ and whose kernel is $(\widetilde{U}_2 \oplus \widetilde{U}_3 \oplus \ldots \oplus \widetilde{U}_s)$.

[21] As discussed above, it is often the case that a set of operators $(B_1, \ldots, B_m)$, with each $B_i : W_1 \to W_2$, satisfying the above property are exactly known. In this case, we can instantiate the Robust Vector Space Decomposition problem with suitable projections of these operators on the set of linear maps from $U \to V$, and $\widetilde{U} \to \widetilde{V}$ respectively. For more details, the reader is referred to Section 4.3 in [6].

[22] In the formulation here, the distance $\mathrm{dist}(\widetilde{\mathcal{B}}, \mathcal{B})$ is defined by extending all operators to map W_1 into W_2.

[23] As we note in Remark 6 in [6] , our algorithms can be used to find $(V_1, \ldots, V_s)$ approximately as well, but we omit that here since our applications do not need it.

The adjoint algebra and its properties. Based on [17, 8], [10] defined a notion called the *adjoint algebra*[24] whose structure can be used to understand (the potentially many) decompositions. Let us recall this notion.

▶ **Definition 3.1** (Adjoint algebra). *The adjoint algebra, corresponding to the vector spaces U, V, and the tuple of operators $\mathcal{B}$, denoted $\mathrm{Adj}_{U,V}(\mathcal{B})$ is defined to be the set of all tuples of linear maps (D, E), with $D : U \to U$, $E : V \to V$, such that $B_j \cdot D = E \cdot B_j$ for all $j \in [m]$.*

Observe that the adjoint algebra always contains the space of scaling maps[25]: that is, the set of maps $D : U \to U, E : V \to V$ such that D (resp. E) simply scales each U_i (resp. V_i) by some scalar λ_i, for each $i \in [s]$. We observe that in most applications these maps are all that the adjoint algebra contains, and in this case, there is a simple algorithm to solve the vector space decomposition, and the obtained decomposition is unique:

▶ **Proposition 3.2** (Proposition A.3[26] in [10]). *Suppose that U, V admit a decomposition into direct sum of s spaces under the action of $\mathcal{B}$ as in (7). If $\dim(\mathrm{Adj}_{U,V}(\mathcal{B})) = s$, then it holds that:*

1. *$\mathrm{Adj}_{U,V}(\mathcal{B})$ equals the set of scaling maps (as defined above) and,*
2. *The decomposition given by (7) is the unique irreducible decomposition, i.e. if*

$$U = \hat{U}_1 \oplus \hat{U}_2 \oplus \ldots \oplus \hat{U}_{\hat{s}} \text{ and } V = \hat{V}_1 \oplus \hat{V}_2 \oplus \ldots \oplus \hat{V}_{\hat{s}}, \quad \hat{s} \geq s,$$

and

$$\left\langle \mathcal{B} \cdot \hat{U}_i \right\rangle \subseteq \hat{V}_i, \quad \forall i \in [\hat{s}],$$

then $\hat{s} = s$ and upto reordering if necessary, $\hat{U}_i = U_i$ and $\hat{V}_i = V_i$ for all $i \in [s]$.

Noiseless algorithm. Note that given $\mathcal{B}$ (and U, V) computing $\mathrm{Adj}_{U,V}(\mathcal{B})$ is easy and simply involves solving for D and E that satisfy the linear constraints specified in definition 3.1. Further under the assumption that $\mathrm{Adj}_{U,V}(\mathcal{B})$ equals the set of scaling maps (this we refer to as strong uniqueness), the required subspaces $U_1, U_2, \ldots U_s$ can be obtained as the eigenspaces corresponding to distinct eigenvalues of the linear map $D : U \mapsto U$ which is the component of a random element (D, E) of $\mathrm{Adj}_{U,V}(\mathcal{B})$.

Making the algorithm robust. There is a relatively straightforward way to make this algorithm robust: we use the maps in $\widetilde{\mathcal{B}}$ to compute a vector space[27] that is in some sense an approximation to the original adjoint algebra. Finally, we recover the U_i's approximately as (the sum of a few) eigenspaces of suitably chosen elements of this approximate adjoint algebra. In the noiseless setting it suffices to chose random elements of the adjoint algebra but in the noisy setting this does not work very well. This is because the error incurred in the recovery of an eigenvector/eigenspace of an operator is inversely related to the corresponding eigengap(s) (see lemma A.8 in [6]). Simply picking a random element of the adjoint algebra

[24] The adjoint algebra is a generalization of the notion of the centralizer algebra in matrix/group theory to the case when the image space of the set of linear maps is different from the domain space.
[25] This observation is due to [8] and forms the starting point of the [10] algorithm for vector space decomposition.
[26] This proposition is a special case of the more general proposition A.3 in [10] wherein the blocks of $\mathrm{Adj}_{U,V}(\mathcal{B})$ consist of scalar matrices only.
[27] This space is typically not closed under multiplication and so does not form an algebra.

$\mathrm{Adj}_{U,V}(\mathcal{B})$ leads to a rather small eigengap and we therefore incur a rather large error both theoretically and practically (i.e. in both the worst case noise scenario and the random noise scenarios). Our insight here is that the multiplicative structure of the adjoint algebra can be exploited to find operators in it with (some) large eigengaps and this yields an algorithm that is more robust. Indeed, our initial experiments suggest that the resulting algorithm when applied to tensor decomposition empirically performs better (in terms of error in the output) than any of the known algorithms for tensor decomposition. The details and quantitative bounds are provided in section 4 in [6].

Our Results. The noise-tolerance and performance of our meta-algorithm is captured by the following theorem which bounds the error incurred in terms of various parameters involved.

▶ **Theorem 3.3** (**Learning Noisy Arithmetic Circuits**, Informal version of Theorem 14 in [6]). *Let $f(\mathbf{x}) = T_1(\mathbf{x}) + \cdots + T_s(\mathbf{x})$ be a polynomial such that each $T_i \in \mathbb{R}[\mathbf{x}]^{=d}$ belongs to a circuit class $\mathcal{C}$ that admits operators $\mathcal{L}$ and $\mathcal{B}$ satisfying the following properties:*

- *$\mathcal{L}$ consists of linear maps $L : \mathbb{R}[\mathbf{x}]^{=d} \to W_1$ such that $U \overset{def}{=} \langle \mathcal{L} \cdot f \rangle = U_1 \oplus \cdots \oplus U_s$, $\dim(U) = d_U$, where $U_i \overset{def}{=} \langle \mathcal{L} \cdot T_i \rangle$.*
- *$\mathcal{B}$ consists of linear maps $B : W_1 \to W_2$ satisfying $V \overset{def}{=} \langle \mathcal{B} \cdot \mathcal{L} \cdot f \rangle = V_1 \oplus \cdots \oplus V_s$, $\dim(V) = d_V$, where $V_i \overset{def}{=} \langle \mathcal{B} \cdot \mathcal{L} \cdot T_i \rangle$.*
- *The decomposition of (U, V) under $\mathcal{B}$ is strongly unique, i.e. $\dim(\mathrm{Adj}_{U,V}(\mathcal{B})) = s$.*

We also need the robust versions of the above assumptions and that $\mathcal{L}$ and $\mathcal{B}$ are appropriately normalized. Let M, N be matrices with columns $L \cdot f, L \in \mathcal{L}$ and $B \cdot L \cdot f, B \in \mathcal{B}, L \in \mathcal{L}$ respectively. Suppose that the d_U^{th} and the d_V^{th} largest singular values of M and N, respectively, are bounded from below by some $\sigma > 0$. Similarly, for an appropriate operator corresponding to the adjoint algebra, we need an appropriate singular value lower bounded by σ.

Let $\widetilde{f}(\mathbf{x}) = f(\mathbf{x}) + \eta(\mathbf{x})$ be a polynomial such that $\|\eta\| \leq \epsilon$[28]. Then, there is an efficient algorithm, which on input $\widetilde{f}$, recovers $\widetilde{T_1}, \widetilde{T_2}, \ldots, \widetilde{T_s}$, such that for any $\delta > 0$, with probability at least $1 - \delta$, (upto reordering) for each $i \in [s]$ it holds that

$$\left\| T_i - \widetilde{T_i} \right\| \leq \mathrm{poly}\left(s, d, d_U, d_V, 1/\delta, 1/\sigma\right) \cdot \epsilon.$$

▶ Remark 3.4.

1. **Error for random noise.** The above bound on the output error is for the case when the noise $\eta(\mathbf{x})$ is chosen in an adversarial (i.e. worst-case) fashion, subject of course to the indicated upper bound on its norm. In practice $\eta(\mathbf{x})$ often behaves like a random vector so that the output error is in practice significantly less[29] than the worst-case bound in the above theorem. Our intuition is that when $\eta(\mathbf{x})$ is random the output error should be less by a factor of $\mathrm{poly}(dim(\langle \mathcal{L} \rangle))$ compared to when $\eta(\mathbf{x})$ is adversarially chosen. We leave it as a potential direction for future investigation.
2. **Noise-tolerance.** As noted earlier, our initial experiments indicate that for the well-studied special case of tensor decomposition our algorithm seems to be more noise-tolerant than existing algorithms. We remark here that for subspace clustering, one can have a somewhat different reduction to vector space decomposition which also incorporates the

[28] Under an appropriate norm called the Bombieri norm as defined in Section 2 in [6]. The Bombieri norm is a suitably scaled version of the ℓ_2 norm that has many desirable properties including being invariant under a unitary transformation of the underlying variables.

[29] This situation is reminiscent of the well-studied spiked tensor problem in machine learning which can be thought of as a very special case of our problem.

affinity-based information to obtain a more noise-tolerant clustering algorithm. It might be interesting to do an empirical comparison of noise-tolerance of (such adaptations of) our algorithm to existing algorithms for various applications of interest.

3. **Running Time.** The algorithm boils down to computing singular value decompositions and/or pseudoinverses of certain matrices and thus its running time[30] is upper bounded by the cube of the dimension of the largest vector space involved.
4. We suggest a potential way to speed up the above algorithm in section 6.

4 Application 1: Subspace Clustering

Subspace clustering is the following problem - we are given a set of N points $A = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N\} \subseteq \mathbb{R}^n$ that admit a partition

$$A = A_1 \uplus A_2 \uplus \ldots \uplus A_s,$$

such that the points in each A_j ($j \in [s]$) span a low-dimensional (relative to the number of points in A_j) space $\langle A_j \rangle$. The goal is to find such a partition.

Even for $n = 3$, this problem is NP-hard in the worst case [15]. Despite this, it has been intensely studied and we refer the reader to the surveys [16], [18] and the references therein. Most state of the art techniques rely on constructing an affinity matrix, which measures how likely two points are to be in the same subspace, followed by spectral clustering using the affinity matrix. Most such algorithms have little theoretical analysis about the robustness and recovery guarantees.

A non-degeneracy condition and a reduction. Suppose now that the span of the A_j's satisfy the following non-degeneracy condition: they form a direct sum, i.e.

$$\langle A \rangle = \langle A_1 \rangle \oplus \langle A_2 \rangle \oplus \ldots \oplus \langle A_s \rangle . \tag{8}$$

We will see that in this case subspace clustering reduces to vector space decomposition in the following way. For a point $\mathbf{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{R}^n$, let $\mathbf{a} \cdot \mathbf{x} \in \mathbb{R}[\mathbf{x}]$ denote the linear form $a_1x_1 + a_2x_2 + \cdots + a_nx_n$, in the formal variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. For $d \geq 1$ we denote by $A^{\otimes d}$ the set $\left\{(\mathbf{a} \cdot \mathbf{x})^d \ : \ \mathbf{a} \in A\right\} \subseteq \mathbb{R}[\mathbf{x}]^{=d}$. Consider the space of first-order partial differential operators $\mathcal{B} = \boldsymbol{\partial}^{=1}$ acting on the subspace of polynomials

$$\left\langle A^{\otimes 2} \right\rangle = \left\langle (\mathbf{a}_1 \cdot \mathbf{x})^2, (\mathbf{a}_2 \cdot \mathbf{x})^2, \ldots, (\mathbf{a}_N \cdot \mathbf{x})^2 \right\rangle \subseteq \mathbb{R}[\mathbf{x}]^{=2}.$$

The image space is then

$$\left\langle A^{\otimes 1} \right\rangle = \langle (\mathbf{a}_1 \cdot \mathbf{x}), (\mathbf{a}_2 \cdot \mathbf{x}), \ldots, (\mathbf{a}_N \cdot \mathbf{x}) \rangle \subseteq \mathbb{R}[\mathbf{x}]^{=1}.$$

Note that our non-degeneracy condition can be restated as saying that

$$\left\langle A^{\otimes 1} \right\rangle = \left\langle {A_1}^{\otimes 1} \right\rangle \oplus \left\langle {A_2}^{\otimes 1} \right\rangle \oplus \ldots \oplus \left\langle {A_s}^{\otimes 1} \right\rangle .$$

This implies that the subspaces $\left\langle {A_j}^{\otimes 2} \right\rangle$ also form a direct sum. Its also easily seen that the image of each $\left\langle {A_j}^{\otimes 2} \right\rangle$ under $\mathcal{B} = \boldsymbol{\partial}^{=1}$ is precisely $\left\langle {A_j}^{\otimes 1} \right\rangle$. Thus the vector space $\left\langle A^{\otimes 2} \right\rangle$ admits a decomposition under the action of $\mathcal{B}$. Furthermore, under the additional mild assumption that each $\left\langle {A_j}^{\otimes 2} \right\rangle$ is *indecomposable* under the action of $\mathcal{B}$ it turns out (using Corollary B.1 in [6]) that the decomposition is unique and thus the subspace clustering problem reduces to the problem of vector space decomposition.

[30] This is in the model where operations over real numbers are of unit cost. A more precise bound on the running time in terms of the dimensions of the various relevant vector spaces can be

A weaker non-degeneracy condition. Note that the non-degeneracy condition given by (8) is rather restrictive - it implies in particular that the number of subspaces s cannot exceed n, the dimension of the ambient space. We can get a weaker non-degeneracy condition by considering the action of first-order partial differential operators $\mathcal{B} = \boldsymbol{\partial}^{=1}$ on the space $\langle A^{\otimes d}\rangle$ instead (for some suitable choice of $d \geq 2$). The image space is then $\langle A^{\otimes(d-1)}\rangle$. As before, under the (now weaker) non-degeneracy condition that

$$\left\langle A^{\otimes(d-1)}\right\rangle = \left\langle {A_1}^{\otimes(d-1)}\right\rangle \oplus \left\langle {A_2}^{\otimes(d-1)}\right\rangle \oplus \ldots \oplus \left\langle {A_s}^{\otimes(d-1)}\right\rangle,$$

the vector space $\langle A^{\otimes d}\rangle$ admits a decomposition under the action of $\mathcal{B}$. Furthermore, as before, under the additional mild assumption that each $\left\langle {A_j}^{\otimes d}\right\rangle$ is indecomposable under the action of $\mathcal{B}$ it turns out (Corollary B.1 in [6]) that the decomposition is unique and thus the subspace clustering problem reduces to the problem of vector space decomposition (see Theorem 9 in [6]).

Robust subspace clustering. The robust or noisy version of the subspace clustering problem is the following. Given a set of points $\widetilde{A} = \{\tilde{\mathbf{a}}_1, \tilde{\mathbf{a}}_2, \ldots, \tilde{\mathbf{a}}_N\} \subseteq \mathbb{R}^n$ suppose that each point $\tilde{\mathbf{a}}_i$ is *close to* an (unknown point) $\mathbf{a}_i \in \mathbb{R}^n$ such that the resulting set of points $A = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N\} \subseteq \mathbb{R}^n$ can be clustered using s subspaces, i.e.

$$A = A_1 \uplus A_2 \uplus \ldots \uplus A_s,$$

where each A_j spans a *low-dimensional* subspace $\langle A_j\rangle$. The computational task is to *approximately* recover each subspace $\langle A_j\rangle$, that is, output $\widetilde{\mathbf{W}} = (\widetilde{W}_1, \widetilde{W}_2, \ldots, \widetilde{W}_s)$ such that (upto reordering) each $\widetilde{W}_j$ is close to $\langle A_j\rangle$ for each $j \in [s]$. We can reduce this problem to the robust vector space decomposition as follows. Let $m_d \stackrel{\text{def}}{=} \dim(\langle A^{\otimes d}\rangle)$ and $m_{d-1} \stackrel{\text{def}}{=} \dim(\langle A^{\otimes(d-1)}\rangle)$. Given $\widetilde{A}$ we algorithmically compute the best fitting subspace $\widetilde{U}$ (resp. $\widetilde{V}$) of dimension m_d (resp. m_{d-1}) to $\widetilde{A}^{\otimes d}$ (resp. to $\widetilde{A}^{\otimes(d-1)}$). It turns out then that $\widetilde{U}$ (resp. $\widetilde{V}$) is *close to* $\langle A^{\otimes d}\rangle$ (resp. to $\langle A^{\otimes(d-1)}\rangle$) (Lemmas B.7 and B.8 in [6] give the quantitative bounds). Applying the robust version of vector space decomposition on $(\widetilde{U}, \widetilde{V}, \mathcal{B})$, the subspaces that we obtain are close to $\left\langle {A_j}^{\otimes d}\right\rangle$ ($j \in [s]$) and from these we can, in turn, also approximately recover $\langle A_j\rangle$ (Proposition B.1 in [6]), as required. This yields the following theorem.

▶ **Theorem 4.1** (**Robust Subspace Clustering**, Informal version of Theorem 10 in [6]). *Let $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_N\} \subseteq \mathbb{R}^n$ be a finite set of N points of unit norm, which can partitioned as $A = A_1 \uplus \cdots \uplus A_s$, where each $\langle A_i\rangle$ is subspace of dimension at most t.*

Let $d \geq 2$ be an integer, let $\mathbf{U} = (U_1, \ldots, U_s)$ (resp. $\mathbf{V} = (V_1, \ldots, V_s)$) be an s-tuple of subspaces with $U_j = \left\langle {A_j}^{\otimes d}\right\rangle$ (resp. $V_j = \left\langle {A_j}^{\otimes d-1}\right\rangle$) for each $j \in [s]$. Let $U = \langle \mathbf{U}\rangle$ (resp. $V = \langle \mathbf{V}\rangle$) have dimension m_d (resp. m_{d-1}).

Suppose that:

- *$U = U_1 \oplus \cdots \oplus U_s$, $V = V_1 \oplus \cdots \oplus V_s$, and for each $j \in [s]$, it holds that* $\dim(U_j) = \binom{\dim(\langle A_j\rangle)+d-1}{d}$, $\dim(V_j) = \binom{\dim(\langle A_j\rangle)+d-2}{d-1}$.
- *σ_A is the minimum of $\sigma_{m_d}(M_{A,\,d})$ and $\sigma_{m_{d-1}}(M_{A,\,d-1})$, where $M_{A,d}$ (resp. $M_{A,\,d-1}$) is the matrix whose columns are the polynomials $(\mathbf{a}_i \cdot \mathbf{x})^d$ (resp. $(\mathbf{a}_i \cdot \mathbf{x})^{d-1}$) (see Definition B.4 in [6]).*
- *$\kappa(\mathbf{U})$ denotes the condition number of the tuple of subspaces $\mathbf{U}$ (see Section 2 in [6]).*

- $\sigma_{-(s+1)}(\mathfrak{A})$ *is the* $(s+1)^{th}$ *smallest singular value of the adjoint algebra map (see Definition 4.2 in [6]), corresponding to the action of* $\mathcal{B} = (B_1, \ldots, B_n)$ *on* $\mathbf{U}, \mathbf{V}$*, where* B_i *corresponds to the operator* ∂_{x_i}*.*

Let $\widetilde{A} = \{\tilde{\mathbf{a}}_1, \tilde{\mathbf{a}}_2, \ldots, \tilde{\mathbf{a}}_N\} \subseteq \mathbb{R}^n$ *be a set of unit norm vectors such that* $\|\mathbf{a}_i - \tilde{\mathbf{a}}_i\|_2 \leq \epsilon$ *for each* $i \in [N]$*. Then, there is an algorithm, which on input* $\widetilde{A}$*, runs in time* $\mathrm{poly}(N, n^d)$*, and recovers subspaces* $(\widetilde{W}_1, \widetilde{W}_2, \ldots, \widetilde{W}_s)$*, such that with probability at least* $1 - \delta$*, (upto reordering) for each* $j \in [s]$ *it holds that*

$$\mathrm{dist}(\widetilde{W}_j, \langle A_j \rangle) \leq \mathrm{poly}\left(t, N, d, s, 1/\delta,\ \kappa(\mathbf{U}),\ 1/\sigma_A,\ 1/\sigma_{-(s+1)}(\mathfrak{A})\right) \cdot \epsilon.$$

We show how to lower bound $\sigma_{-(s+1)}(\mathfrak{A})$ (Theorem 11 in [6]). The main technical component is an inductive argument to analyze singular values of basic adjoint operators, which is inspired by the inductive argument in recent works on analyzing eigenvalues for random walks on simplicial complexes (e.g. [2]). Next we see what our algorithms would yield in the smoothed case and state some explicit conjectures about singular values of relevant smoothed matrices.

Smoothed analysis of subspace clustering. We first describe the input model. For simplicity, we assume that each of the subspaces have the same dimension (equal to t).

1. **Perturbation model for subspaces.** We have a tuple of s hidden subspaces of $\mathbb{R}^n$, $\mathbf{W} = (W_1, W_2, \ldots, W_s)$, each of dimension t. Let $P_1, P_2, \ldots, P_s \in \mathbb{R}^{n \times t}$ be matrices with orthonormal columns, such that the column span of P_i is W_i. Each subspace W_i is perturbed by perturbing P_i by a random (and independent) Gaussian matrix $G_i \sim \mathcal{N}(0, \rho^2/n)^{n \times t}$. Let $\hat{P}_i = P_i + G_i$, and $\hat{W}_1, \hat{W}_2, \ldots, \hat{W}_s$ be the column spans of $\hat{P}_1, \hat{P}_2, \ldots, \hat{P}_s$ respectively.
2. **Perturbation models for points from each subspace.** Sample (possibly adversarially) sets of points $A_1, A_2, \ldots, A_s$ from $\hat{W}_1, \hat{W}_2, \ldots, \hat{W}_s$ respectively, of unit norm. For each $i \in [s]$, perturb each point in A_i with respect to $\hat{W}_i$ to get the set of points $\hat{A}_i$. Formally, this means perturbing points in A_i by $\hat{B}_i \cdot v$, where $\hat{B}_i$ is an $n \times t$ matrix describing an orthonormal basis for $\hat{W}_i$ and $v \sim \mathcal{N}(0, \rho^2/t)^t$ (independently generated for each point), and normalizing. Let $\hat{A} = \hat{A}_1 \cup \hat{A}_2 \cup \cdots \cup \hat{A}_s$.
3. **Adding noise.** For each $\mathbf{a} \in \hat{A}$, add noise (possibly adversarially) and normalize to get a unit norm point $\mathbf{a}'$ such that $\|\mathbf{a} - \mathbf{a}'\|_2 \leq \epsilon$. We are given as input $\hat{A}'$, the set of noise-added points.

Given the set of points $\hat{A}'$, the goal is to recover subspaces $\tilde{\mathbf{W}} = (\widetilde{W}_1, \widetilde{W}_2, \ldots, \widetilde{W}_s)$ such that $\mathrm{dist}(\hat{\mathbf{W}}, \tilde{\mathbf{W}})$ is small. Next we state a couple of conjectures about minimum singular values of smoothed random matrices that we encounter:

▶ **Conjecture 4.2.** *Let* $\mathbf{v}_{i1}, \ldots, \mathbf{v}_{it}$ *be an orthonormal basis for* $\hat{W}_i$ *generated as above. Define the linear forms* $\ell_{ij}(\mathbf{x}) = \langle \mathbf{v}_{ij}, \mathbf{x} \rangle$*. Consider the* $\binom{n+d-1}{d} \times s\binom{t+d-1}{d}$ *matrix* M *where the columns are divided into* s *chunks and in the* i^{th} *chunk, the columns are all the monomials of degree* d *in the polynomials* $\ell_{i1}, \ldots, \ell_{it}$*. Also suppose* $s\binom{t+d-1}{d} \leq (1-\delta)\binom{n+d-1}{d}$ *for a constant* $\delta > 0$*. Then for constant* d*, with high probability,* $\sigma_{s\binom{t+d-1}{d}}(M) \geq \mathrm{poly}(\rho, 1/n)$*.*

▶ **Conjecture 4.3.** *Consider arbitrary vectors* $v_1, \ldots, v_s \in \mathbb{R}^t$ *of unit norm and their smoothed versions* $\hat{v}_1, \ldots, \hat{v}_s$*, where* $\hat{v}_i = v_i + g_i$*,* $g_i \sim \mathcal{N}(0, \rho^2/t)^t$ *(and then further normalized to unit norm). Consider the* $s \times \binom{t+d-1}{d}$ *matrix* M *where the* i^{th} *row contains the polynomial* $\langle \hat{v}_i, \mathbf{x} \rangle^d$*. Suppose* $s \geq (1+\delta)\binom{t+d-1}{d}$ *for a constant* $\delta > 0$*. Then for constant* d*, with high probability,* $\sigma_{\binom{t+d-1}{d}}(M) \geq \mathrm{poly}(\rho, 1/t)$*.*

▶ **Theorem 4.4** (**Smoothed analysis of subspace clustering**, Theorem 12 in [6] restated)**.** *Suppose Conjectures 4.2 and 4.3 are true. Then for constant d, Algorithm 5 in [6] on input $(\hat{A}', d, s, m_d, m_{d-1})$ outputs $\tilde{\mathbf{W}} = (\widetilde{W}_1, \ldots, \widetilde{W}_s)$ such that with high probability,*

$$\mathrm{dist}(\widetilde{W}_j, \hat{W}_j) \leq \mathrm{poly}\,(n, t, 1/\rho) \cdot \epsilon$$

Regarding the two conjectures, Conjecture 4.3 is closely linked to the paper [5]. There they considered the setting where $s \leq (1-\delta)\binom{t+d-1}{d}$ and proved a similar lower bound for $\sigma_s(M)$. In both the settings there is slack, so it is plausible that the techniques of [5] can be adapted to prove Conjecture 4.3. But we don't know how to do that. In Conjecture 4.2, the matrix M is such that both the rows and columns share random variables. Most of the smoothed analysis till now focuses on matrices where either rows or columns have different sets of variables involved, and this makes it amenable to the leave-one-out distance method. Still, in Conjecture 4.2, the sharing of variables is not completely arbitrary. One can divide rows into chunks so that different chunks have different sets of variables. However, even this setting seems to require new techniques to analyze.

5 Application 2: Learning Mixtures of Gaussians

In this section we will see how the problem of computing the parameters of a mixture of Gaussians reduces to (several instances of) vector space decomposition.

Reduction to a special case of formula learning. It is implicit in [11] that learning a mixture of s zero-mean Gaussians reduces to robustly expressing a given homogeneous polynomial $p(\mathbf{x})$ as a sum of s powers of quadratics, i.e.

$$p(\mathbf{x}) = p_1(\mathbf{x})^d + p_2(\mathbf{x})^d + \ldots + p_s(\mathbf{x})^d, \tag{9}$$

where the p_i's are homogeneous quadratic polynomials. Following the ideas in [10], we give a direct reduction[31] to vector space decomposition as follows.

Obtaining a vector space that is the direct sum of unknown spaces. Following [10], we apply partial derivatives followed by a *random projection* to obtain a vector space that is a direct sum of s unknown subspaces, one corresponding to each $p_i(\mathbf{x})$. Specifically, let $\mathcal{L}$ be the set of operators corresponding to taking k-th order partial derivatives followed by *a random restriction*[32]. Applying $\mathcal{L}$ to both sides of equation (9), we get

$$\langle \mathcal{L} \cdot p(\mathbf{x}) \rangle \subseteq \left\langle \mathcal{L} \cdot p_1(\mathbf{x})^d \right\rangle + \left\langle \mathcal{L} \cdot p_2(\mathbf{x})^d \right\rangle + \ldots + \left\langle \mathcal{L} \cdot p_s(\mathbf{x})^d \right\rangle,$$

It turns out that (Lemma C.1 in [6]) under relatively mild nondegeneracy conditions on the choice of the p_i's, the vector space sum on the right hand side of the above equation is actually a direct sum and the containment is actually an equality, i.e.

$$\langle \mathcal{L} \cdot p(\mathbf{x}) \rangle = \left\langle \mathcal{L} \cdot p_1(\mathbf{x})^d \right\rangle \oplus \left\langle \mathcal{L} \cdot p_2(\mathbf{x})^d \right\rangle \oplus \ldots \oplus \left\langle \mathcal{L} \cdot p_s(\mathbf{x})^d \right\rangle.$$

We now carefully choose another set of operators $\mathcal{B}$ such that the subspace $U \overset{\text{def}}{=} \langle \mathcal{L} \cdot p(\mathbf{x}) \rangle$ admits a *unique* decomposition under the action of $\mathcal{B}$.

[31] In [10], there is an additional "multi-gcd" step which we avoid here.

[32] W can think of a random projection as keeping a subset $\mathbf{y} \subseteq \mathbf{x}$ of the variables alive and setting the rest to zero.

Choice of $\mathcal{B}$. The set of operators $\mathcal{L}$ maps polynomials in $\mathbf{x}$ to polynomials in a subset of variables $\mathbf{y} \subseteq \mathbf{x}$. Under the above mentioned nondegeneracy conditions, it also turns out that for each $i \in [s]$, $\langle \mathcal{L} \cdot p_i(\mathbf{x})^d \rangle$ is of the form $U_i \overset{\text{def}}{=} \langle \mathbf{y}^{=k} \cdot q_i(\mathbf{y})^{d-k} \rangle \subseteq \mathbb{R}[\mathbf{y}]^{=(2d-k)}$. With this in mind, we choose $\mathcal{B}$ as the following set of operators: first order partial derivatives followed by multiplication[33] by polynomials of degree 1. In detail: $\mathcal{B}$ consists of $|\mathbf{y}|^2$ operators with the (i,j)-th operator $(i,j \in [|\mathbf{y}|])$ being

$$B_{ij} : \mathbb{R}[\mathbf{y}]^{=(2d-k)} \mapsto \mathbb{R}[\mathbf{y}]^{=(2d-k)}, \quad B_{ij} \cdot q(\mathbf{y}) = y_j \cdot (\partial_{y_i} q(\mathbf{y})) \text{ for any } q(\mathbf{y}) \in \mathbb{R}[\mathbf{y}]^{=(2d-k)}.$$

It turns out that for any $i \in [s]$, under the action of $\mathcal{B}$, the image of

$$U_i \overset{\text{def}}{=} \langle \mathbf{y}^{=k} \cdot q_i(\mathbf{y})^{d-k} \rangle \text{ is the subspace } V_i \overset{\text{def}}{=} \left\langle \mathbf{y}^{=(k+2)} \cdot q_i(\mathbf{y}))^{d-k-1} \right\rangle$$

and that the U_i's and V_j's form direct sums (Lemma C.2 in [6]). Furthermore, under mild non-degeneracy conditions such a decomposition is unique (Corollary C.1 in [6]) implying that our vector space U has a unique decomposition into s subspaces under the action of $\mathcal{B}$. Lastly, from each U_i we can recover the corresponding $q_i(\mathbf{y})$ which is a restriction of $p_i(\mathbf{x})$ to a chosen subspace. Any polynomial can be recovered from its restriction to a small number of chosen subspaces and we use this to recover each $p_i(\mathbf{x})$ $(i \in [s])$, as required. In this way, the problem of learning mixtures of Gaussians reduces to robust vector space decomposition.

Robust version. Our general algorithm for learning arithmetic circuits with noise (Theorem 3.3) can be used to make the above algorithm robust. We will also need to use the algorithm of [3] in this case to combine the various projections of p_i's. Our algorithm will depend on condition numbers of certain matrices which can be deduced from the operators used in the above algorithm. Lemmas C.1, C.2 and C.3 in [6] show that at least the ranks of these matrices are as expected. Lemmas C.1 and C.2 in [6] are from [10]. Lemma C.3 in [6] is new and is the main technical contribution for this section, and shows that the relevant adjoint algebra is of the correct dimension. Also [3] analyze similar matrices corresponding to Lemmas C.1 and C.2 in [6] and prove the required condition number bounds in the fully random case. For the singular values of the adjoint operator (robustification of Lemma C.3 in [6], we believe similar techniques as Theorem 11 in [6] should work to give us a bound but the setting is more challenging and we don't know how to prove a bound here yet.

Comparison to [10] and [3]. The algorithms of [10, 3] for learning mixtures of Gaussians roughly proceed as follows (for simplicity, we only consider the the noiseless case here).

Given a polynomial $p(x) = \sum_{i=1}^{s} p_i(\mathbf{x})^d$, where each p_i is a quadratic polynomial:

1. Apply a set of operators $\mathcal{L}$ to $p(\mathbf{x})$, where $\mathcal{L}$ corresponds to taking some k-th order partial derivatives followed by a random restriction: as described before, each $\langle \mathcal{L} \cdot p_i(\mathbf{x}) \rangle$ is of the form $\langle \mathbf{y}^{=k} \cdot q_i(\mathbf{y})^{d-k} \rangle \subseteq \mathbb{R}[\mathbf{y}]^{=(2d-k)}$. This step is essentially the same in both [10, 3]. We note however that [3] actually do not work under the non-degeneracy condition of the spaces $\langle \mathcal{L} \cdot p_i(\mathbf{x}) \rangle$'s forming a direct sum, and instead explicitly characterize the structure of the intersections $\langle \mathcal{L} \cdot p_i(\mathbf{x}) \rangle \cap \langle \mathcal{L} \cdot p_j(\mathbf{x}) \rangle$. This allows them to deal with a broader range of parameters compared to [10].

[33] The relevant literature on arithmetic formula lower bounds would refer to the set of operators $\mathcal{B}$ as *shifted partials* and denote it by $\mathbf{y}^{=1} \cdot \boldsymbol{\partial}_{\mathbf{y}}^{=1}$.

2. The next step is a "multi-gcd" step, which is used to find the vector space $\langle q_i(\mathbf{y})^{d-k} \rangle + \cdots + \langle q_s(\mathbf{y})^{d-k} \rangle$. This step is already present in the algorithm of [10], however [3] give a significantly simpler algorithm for this step, along with an analysis for the robust version of this step.
3. The next step, which is in some sense the "main part" of the algorithm, is where the two algorithms [10] and [3] differ:
   a. The algorithm of [10] considers another application of k-th order partial derivatives + random restriction on this vector space, and uses vector space decomposition with respect to this set of operators. This allows them to recover the component polynomials.
   b. The algorithm of [3] follows the approach in [11], and does a "desymmetrization + tensor-decomposition" step. This roughly enables them to convert the sum of polynomials to a sum of tensors, and then apply standard tensor decomposition methods to obtain the required components.
4. The final step is to repeat the above procedure multiple times, using a different random restriction each time, and then aggregating the obtained $q_i(\mathbf{y})$'s into $p_i(\mathbf{x})$, as described before.

Our algorithm essentially follows the same first and final step as both these algorithms. It significantly deviates from the two algorithms in Steps 2 and 3:

1. While we follow the same vector space decomposition paradigm as [10], our algorithm completely eliminates the use of the multi-gcd step. Instead, we use a very simple set of operators, namely order one partial derivatives + order one shifts, directly on the vector space $\langle \mathcal{L} \cdot p(\mathbf{x}) \rangle$. Hence, our approach provides a much more direct reduction to vector space decomposition.
2. In comparison to [3], we first eliminate the use of the multi-gcd step, and further we do not go through the desymmetrization step at all. Instead, our framework of vector space decomposition allows us to deal with symmetric polynomials throughout the algorithm; this inherently seems much more natural since the inputs and outputs all deal only with polynomials (symmetric tensors).

Finally, we note the the above described simplification allows us to obtain a much better range of parameters compared to [10], whereas we still expect them to be slightly worse than [3].

6 Conclusion and Future Directions

In this work we showed how to adapt the algorithm of [10] for learning subclasses of arithmetic formulas to make it noise-tolerant. This turns out to have a number of applications arising out of the remarkable fact that in these applications, a suitably defined polynomial formed out of the statistics of the data has a small arithmetic formula. We feel that our approach has the potential to give algorithms which are fast, noise-tolerant, outlier-tolerant and come with provable guarantees[34] for many such applications and is therefore worthy of further investigation. We now pose some problems that might encourage or guide such further study.

[34] For most such applications the worst-case instances are intractable so the best we can hope for are algorithms whose performance can be bounded using singular values of certain instance-dependent matrices.

Making the vector space decomposition algorithm faster. Consider a set of operators $\mathcal{B}$ mapping a real vector space U to another real vector space V. Our algorithm for decomposition of U (and V) under the action of $\mathcal{B}$ involved computations with the adjoint algebra which entailed working in the vector spaces of linear maps $\mathrm{Lin}(U, U)$ and $\mathrm{Lin}(V, V)$. These spaces of linear maps have larger dimension than that of U and V themselves and consequently, our approach for decomposing U has running time pertaining to the cost of doing linear algebra over spaces of dimension $(\dim(U)^2 + \dim(V)^2)$. Let us first make an observation. Suppose that the decomposition induced by $\mathcal{B}$, namely:

$$U = U_1 \oplus U_2 \oplus \ldots \oplus U_s, \quad V = V_1 \oplus V_2 \oplus \ldots \oplus V_s$$

had the property that the U_i's (respectively also the V_i's) were orthogonal complements of each other (under some canonical inner product on the spaces U and V). Consider the collection of linear maps $\mathcal{L} \subseteq \mathrm{Lin}(U, U)$ defined as $\mathcal{L} := \left\{ B_j^T \cdot B_i \ : \ B_i, B_j \in \mathcal{B} \right\}$. Then each U_i is an invariant subspace (i.e. an eigenspace) of every operator in $\mathcal{L}$. In such a situation we typically expect the following simple algorithm to work: simply pick three random maps $B_1, B_2, B_3 \in \langle \mathcal{B} \rangle$ and compute[35] $L := B_2^T \cdot B_1$ and $M := B_3^T \cdot B_1$. Then for each eigenvector u of L, compute the span of the orbit of u under the action of M. The distinct subspaces so obtained should typically give us the required subspaces $U_1, U_2, \ldots, U_s$. Clearly such an algorithm, *when it works*, would be much faster. We expect that for most applications, the above algorithm should work but we don't know.

▶ **Problem 1.** *For problems such as subspace clustering and learning mixtures of Gaussians, if the relevant U_i's (respectively also the V_i's) are orthogonal to each other, does the above algorithm correctly recover the U_i's?*

▶ **Problem 2.** *Whats the best way to make this algorithm noise-tolerant?*

Finally, in situations where the U_i's (resp the V_i's) are not orthogonal to each other we can clearly make them so by using appropriate inner products on U and V. But how do we find such an inner product? We expect the operator scaling algorithm of [9] to yield such an inner product(!)

▶ **Problem 3.** *For (noiseless) subspace clustering, does the operator scaling algorithm of [9] applied on the relevant $\mathcal{B}$ yield inner products under which the relevant subspaces are orthogonal?*

Mixture of Gaussians. As mentioned in Remark 1.1(b) earlier, we expect that our algorithm can be extended to handle general mixtures of Gaussians with differing means and covariance matrices. Let us formally state this as an open problem.

▶ **Problem 4** (Random instances of general mixtures of Gaussians.). *Let $n, s \geq n$ be integers. For $i \in [s]$ suppose that we pick $\boldsymbol{\mu}_i \in \mathbb{R}^n$ and covariance matrices $\Sigma_i \in \mathbb{R}^{n \times n}$ independently at random[36]. Let $\mathcal{D} := \sum_{i=1}^{s} \frac{1}{s} \cdot \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$ be the equi-weighted mixture of Gaussians with the above randomly chosen parameters. Design an efficient algorithm that given samples from D recovers the $\boldsymbol{\mu}_i$'s and Σ_i's approximately.*

[35] The linear maps B_2^T, B_3^T from V to U are defined using the canonical inner products on these two spaces.

[36] Any reasonable distribution would do but for concreteness say we pick $\boldsymbol{\mu}_i \sim \mathcal{N}(0, I_n)$ and we pick $\Sigma_i = B^T \cdot B$, where $B \sim \mathcal{N}(0,1)^{n \times n}$.

Our work as well as that of [3] leave open the problem of doing a smoothed analysis of the corresponding algorithm for mixtures of zero-mean Gaussians. To encourage this direction of research, let us state this explicitly in the form of a conjecture.

▶ **Conjecture 6.1** (Smoothed analysis of our algorithm for mixture of zero-mean Gaussians.). *Our algorithm efficiently recovers the unknown parameters for smoothed instances of mixtures of zero-mean Gaussians.*

Handling outliers and other applications. In Remark 1.1, we conjectured that our approach/framework should enable the design of efficient algorithms that can handle outliers and also be useful for many more applications in unsupervised learning. It would be nice to have concrete results in such directions.

References

1 Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M. Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(1):2773–2832, January 2014.

2 Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials II: High-dimensional walks and an FPRAS for counting bases of a matroid. In *STOC'19—Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1–12, 2019.

3 Mitali Bafna, Jun-Ting Hsieh, Pravesh K Kothari, and Jeff Xu. Polynomial-time power-sum decomposition of polynomials. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 956–967. IEEE, 2022.

4 Ainesh Bakshi, Ilias Diakonikolas, He Jia, Daniel M. Kane, Pravesh K. Kothari, and Santosh S. Vempala. Robustly learning mixtures of k arbitrary Gaussians. In *STOC '22—Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1234–1247. ACM, New York, 2022.

5 Aditya Bhaskara, Aidao Chen, Aidan Perreault, and Aravindan Vijayaraghavan. Smoothed analysis in unsupervised learning via decoupling. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 582–610. IEEE, 2019.

6 Pritam Chandra, Ankit Garg, Neeraj Kayal, Kunal Mittal, and Tanmay Sinha. Learning arithmetic formulas in the presence of noise: A general framework and applications to unsupervised learning, 2023. arXiv:2311.07284.

7 Sitan Chen, Jerry Li, Yuanzhi Li, and Anru R. Zhang. Learning polynomial transformations via generalized tensor decompositions. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1671–1684. ACM, 2023. doi:10.1145/3564246.3585209.

8 Alexander L. Chistov, Gábor Ivanyos, and Marek Karpinski. Polynomial time algorithms for modules over finite dimensional algebras. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, ISSAC '97, Maui, Hawaii, USA, July 21-23, 1997*, pages 68–74, 1997.

9 Ankit Garg, Leonid Gurvits, Rafael Mendes de Oliveira, and Avi Wigderson. Operator scaling: Theory and applications. *Found. Comput. Math.*, 20(2):223–290, 2020. doi:10.1007/s10208-019-09417-z.

10 Ankit Garg, Neeraj Kayal, and Chandan Saha. Learning sums of powers of low-degree polynomials in the non-degenerate case. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 889–899. IEEE, 2020. Open source version at arXiv:2004.06898.

11 Rong Ge, Qingqing Huang, and Sham M. Kakade. Learning mixtures of gaussians in high dimensions. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, pages 761–770, 2015. Open source version at `arXiv:1503.00424`.

12 Erich Kaltofen, John P. May, Zhengfeng Yang, and Lihong Zhi. Approximate factorization of multivariate polynomials using singular value decomposition. *Journal of Symboilic Computation*, 43(5):359–376, 2008. `doi:10.1016/J.JSC.2007.11.005`.

13 Erich Kaltofen and Barry M. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *Journal of Symboilic Computation*, 9(3):301–320, 1990.

14 Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

15 Nimrod Megiddo and Arie Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1(5):194–197, 1982. `doi:10.1016/0167-6377(82)90039-6`.

16 Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: A review. *SIGKDD Explor. Newsl.*, 6(1):90–105, June 2004.

17 Youming Qiao. Block diagonalization for adjoint action. Private communication, 2018.

18 Wentao Qu, Xianchao Xiu, Huangyue Chen, and Lingchen Kong. A survey on high-dimensional subspace clustering. *Mathematics*, 11(2), 2023. `doi:10.3390/math11020436`.

19 Aravindan Vijayaraghavan. Efficient tensor decompositions. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*, pages 424–444. Cambridge University Press, 2020. `arXiv:2007.15589`.

20 Hanna Wallach. Topic modeling: Beyond bag-of-words. In *ICML 2006 - Proceedings of the 23rd International Conference on Machine Learning*, volume 2006, pages 977–984, January 2006. `doi:10.1145/1143844.1143967`.