Unsupervised Learning via Algebraic Circuit Reconstruction

CS224 Project (Algorithms for Data Science, Fall 2023)

Prashanth Amireddy

December 26, 2023

Abstract

We give an exposition of a recent series of works solving unsupervised learning problems such as learning Gaussian mixtures and subspace clustering by framing them as algebraic circuit reconstruction problems.

1 Introduction

Tensor decomposition is a fundamental problem with several applications in machine learning. In the symmetric version of this problem, we are given a polynomial $f(\mathbf{x}) = \sum_{i=1}^{s} \ell_i(\mathbf{x})^3$ where $\ell_i(\mathbf{x})$ are linear real polynomials, the goal is to find these linear polynomials. This can be solved in a "non-degenerate" setting, i.e., when ℓ_i's are linearly independent, in polynomial time by using Jennrich's algorithm [H+70, LRA93]. A natural generalization of this problem is that of learning sum of powers of quadratic forms. Here, for a given polynomial $f(\mathbf{x}) = \sum_{i=1}^{s} Q_i(\mathbf{x})^{d/2}$ for some homogeneous quadratic polynomials $Q_i(\mathbf{x})$, we need to compute the Q_i's. Apart from being a natural generalization of symmetric tensor decomposition into a more complex algebraic circuit model, this is also closely related to learning Gaussian mixtures from their moments. Learning mixtures of any Gaussian mixtures in the smoothed setting is a f Tensor decomposition is a fundamental problem with several applications in machine learning. In the symmetric version of this problem, we are given a polynomial $f(\mathbf{x}) = \sum_{i=1}^{s} \ell_i(\mathbf{x})^3$ where $\ell_i(\mathbf{x})$ are linear real polynomials, the goal is to find these linear polynomials. This can be solved in a "non-degenerate" setting, i.e., when ℓ_i's are linearly independent, in polynomial time by using Jennrich's algorithm [H+70, LRA93]. A natural generalization of this problem is that of learning sum of powers of quadratic forms. Here, for a given polynomial $f(\mathbf{x}) = \sum_{i=1}^{s} Q_i(\mathbf{x})^{d/2}$ for some homogeneous quadratic polynomials $Q_i(\mathbf{x})$, we need to compute the Q_i's. Apart from being a natural generalization of symmetric tensor decomposition into a more complex algebraic circuit model, this is also closely related to learning Gaussian mixtures from their moments. Learning *any* "smoothed" mixture of polynomially bounded number of Gaussians in polynomial time is an important problem in learning theory that is still wide-open! We will focus on one particular approach that makes progress towards this goal under some additional assumptions; for example, that the means of the Gaussians are zero.

A recent series of works exploits the above connection to algebraic circuit complexity to derive new learning algorithms for Gaussian mixtures. In this report, we will primarily focus on the work of Garg, Kayal and Saha [GKS20] that gives a new polynomial time algorithm for learning zero-mean Gaussian mixtures given exact access to its moments.

Theorem 1.1 ([GKS20] Lemma 3.1, modified)**.** *Suppose n is the dimension and $s = \mathrm{poly}(n)$. Given black-box access to the exact $O(1)$-order moments of a non-degenerate mixture of zero-mean Gaussians $\mathcal{D} = \sum_{i=1}^{s} w_i \cdot \mathcal{N}(\mathbf{0}, \Sigma_i)$, its parameters $(w_i, \Sigma_i)_{i\in[s]}$ can be recovered by a polynomial (in the bit complexity of the parameters) time randomized algorithm.*

The main conceptual contribution of [GKS20], and other related works [KS19, CGK+23], is to give a meta-algorithm that converts a lower bound proof for an algebraic model into a learning algorithm for the same class. In Section 3, we will give an overview of this meta-algorithm for learning sum of powers of quadratics, by making some simplifying assumptions such as the weights of all the components being identical, that $s = \mathrm{poly}(n)$ and that the degrees of the "inner polyonmials" is two (this suffices for learning Gaussian mixtures), deferring the proofs of some of the technical parts to the original paper.

Related works. Ge, Huang and Kakade [GHK15] give a polynomial time algorithm in the smoothed setting when the number of components $s \leqslant O(\sqrt{n})$. The authors proceed by handling the zero-mean Gaussians first and then extend it to the general mean case. The zero-mean case involves estimating the ($\leqslant 6$)-th order moments of the distribution and exploiting their algebraic structure to learn the parameters of the mixture.

The work of Chandra et al. [CGK+23] closely builds upon [GKS20] and covers even more unsupervised learning tasks including subspace clustering and potentially learning polynomial transformations and topic modelling. For all these applications, the idea is the same: Suppose we want to fit a model/distribution to a given set of points, $A \subseteq \mathbb{R}^n$. This could be fitting a Gaussian mixtures model or a subspace clustering problem[1]. Then, we first encode the data A as a (low-degree) multivariate polynomial $P(\mathbf{x}) \in \mathbb{R}_d[\mathbf{x}]$; this encoding is natural in most cases. As it turns out, for most of these unsupervised learning tasks, $P(\mathbf{x})$ admits a "simple/small" algebraic circuit if and only if A has the desired structure, be it that its moments agree with that of Gaussian mixtures or that it is a union of low-dimensional subspaces for the subspace clustering problem. Then, one can use the algebraic circuit reconstruction principles to obtain the individual components of the circuit, which in most cases immediately yield the parameters of the structure (e.g., the covariance matrices or the low-dimensional components) we are trying to learn about A.

Another crucial contribution of [CGK+23] is noise-robustness. While the previous works only work when given exact access to the moments (for the Gaussian mixtures problem), practically we can only empirically estimate the moments, which adds a noise term to the function we are trying to learn. The main step that needs to be made robust to noise is Vector Space Decomposition (discussed in Section 3). Under some assumptions about the condition numbers of certain random matrices being small, the authors give an algorithm for robust subspace clustering in the smoothed setting, i.e., the projection matrix for each component subspace is perturbed by a Gaussian noise while maintaining the dimension. Similarly, for learning Gaussian mixtures with estimated/approximate moments, the authors prove the correctness of their algorithm under certain conjectured singular value bounds, again in the smoothed setting (i.e., each covariance matrix is obtained by perturbing a random instance with a Gaussian noise).

These results are closely related to the work of Bafna et al. [BHKX22]. Their algorithm proceeds in a similar way of using partial derivatives to obtain more structured subspaces. However,

[1] In subspace clustering, the goal is to partition the (noisy) points of A into the constituent low dimensional subspaces.

it has the advantage of learning Gaussian mixtures from much lower-order moments than [GKS20, CGK+23], at least in the random setting i.e., when the coefficients of the polynomials are picked independently from a normal distribution. The resulting algorithm is robust to noise in the moments, which is proved by bounding the singular values of certain random matrices by using the graph matrix decomposition method.

2 Preliminaries

Basic notation. We define $[n] \triangleq \{1, 2, \ldots, n\}$. For a finite set A and integer d, we use $\binom{A}{d}$ to refer to the set of subsets of A of size d. An algebraic circuit is an algebraic expression with addition and multiplication operations; in this article, we only need to work with algebraic circuits of the form $R_1(\mathbf{x})^e + R_2(\mathbf{x})^e + \cdots + R_s(\mathbf{x})^e$ where $R_i(\mathbf{x})$'s are homogeneous polynomials of the same degree.

Probability. For $\mu \in \mathbb{R}^n$ and a psd matrix $\Sigma \in \mathbb{R}^{n\times n}$, $\mathcal{N}(\mu, \Sigma)$ stands for the Gaussian distribution with mean μ and covariance Σ. The d-th order moments of a distribution $\mathcal{D}$ over $\mathbb{R}^n$ are $\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x}^{\boldsymbol{\alpha}}]$ for all $\boldsymbol{\alpha} \in \mathbb{Z}^n_{\geqslant 0}$ such that $|\boldsymbol{\alpha}|_1 = d$, where $\mathbf{x}^{\boldsymbol{\alpha}}$ is a monomial in $\mathbf{x}$ with exponents represented by a vector $\boldsymbol{\alpha} \in \mathbb{Z}^n_{\geqslant 0}$.

Algebra. For $\mathbf{x} \triangleq (x_1, x_2, \ldots, x_n)$, $\mathbb{R}_d[\mathbf{x}]$ denotes the set of all real homogeneous polynomials over $\mathbf{x}$ of degree d – this is a vector space over $\mathbb{R}$ of dimension $\binom{n+d-1}{d}$. For $P = P(\mathbf{x}) \in \mathbb{R}_d[\mathbf{x}]$ and $0 \leqslant k \leqslant d$, $\partial^k(P) \subseteq \mathbb{R}_{d-k}[\mathbf{x}]$ denotes the set of all k-th order partial derivatives of P. For any set of polynomials $\mathcal{P} \subseteq \mathbb{R}_d[\mathbf{x}]$, $\langle\mathcal{P}\rangle$ denotes its span and $\dim(\cdot)$ stands for dimension. The Schwartz-Zippel lemma states that for any non-zero polynomial $P(\mathbf{x})$ of degree d and any non-empty $S \subseteq \mathbb{R}$, it holds that

$$\Pr_{\mathbf{a}\sim S^n}[p(\mathbf{a}) = 0] \leqslant d/|S|.$$

Linear algebra. We use the notation $A \succeq 0$ to denote that a real symmetric matrix A is positive semidefinite (psd). The operator norm of a real matrix A is denoted by $\|A\|$. For two subspaces U and V of a vector space W, we write $U + V = U \oplus V$ if U and V are linearly independent; equivalently $\dim(U + V) = \dim(U) + \dim(V)$.

3 Learning Gaussian mixtures

Here, we are given exact access to low-order moments of a mixture of zero-mean Gaussians and the goal is to recover the parameters of the distribution. More formally, suppose

$$\mathcal{D} = \sum_{i=1}^{s} w_i \cdot \mathcal{N}(\mathbf{0}, \Sigma_i)$$

is the (unknown) distribution, where $w_i \geqslant 0$, $\sum_i w_i = 1$ and $\Sigma_i \succeq 0$.

We will prove the following theorem.

Theorem 3.1 (Learning Gaussian mixtures)**.** *Given* exact *access to all the $O(\log s/\log n)$-order moments of a* non-degenerate *Gaussian mixture $\mathcal{D} = \sum_{i=1}^{s} w_i \cdot \mathcal{N}(\mathbf{0}, \Sigma_i)$, its parameters w_i and Σ_i can be estimated in* $\mathrm{poly}(n, s)$ *time.*

In particular, when $s = \mathrm{poly}(n)$, we get an algorithm with $\mathrm{poly}(n)$ time complexity assuming exact access to $O(1)$-order moments. (This is the setting we will mostly restrict to in this paper.)

A few remarks about the above theorem statement:

Remark 3.2.

1. The non-degeneracy conditions will be discussed in more detail later, but for now, we only remark that these conditions will be satisfied for any "generic" choices of the parameters. For example, it could be the condition that some matrix formed by the parameters is non-singular.
2. The above algorithm is randomized, and succeeds with probability $1 - o(1)$.
3. We assume that s is already known (if not, we can try each value of s).

We now describe how the above task can be reduced to that of learning an *algebraic circuit* representation given black-box access to a polynomial. Let $d = O(\log s / \log n) = O(1)$ be a sufficiently large even integer to be fixed later. The moment generating function of $\mathcal{D}$ is[2]

$$\mathbb{E}[e^{\langle \mathbf{x}, \mathcal{D} \rangle}] = \sum_{i=1}^{s} w_i e^{\mathbf{x}^\top \Sigma_i \mathbf{x}/2}.$$

Letting $Q_i(\mathbf{x}) \triangleq \mathbf{x}^\top \Sigma_i \mathbf{x}/2$ and comparing the degree-d homogeneous parts in the Taylor expansion of both sides, we have

$$\frac{\mathbb{E}[\langle \mathbf{x}, \mathcal{D} \rangle^d]}{d!} = \sum_{i=1}^{s} \frac{w_i Q_i(\mathbf{x})^{d/2}}{(d/2)!}.$$

Since, we are assuming black-box access to the moments of $\mathcal{D}$, we have black-box access to the polynomial on the RHS of the above equation, i.e.,

$$f(\mathbf{x}) \triangleq \sum_{i=1}^{s} w_i Q_i(\mathbf{x})^{d/2}.$$

Thus, we have reduced the problem to that of learning Q_i's (from which we can read off the covariance matrices Σ_i's) in the above representation, given black-box access to the function $f(\mathbf{x})$. However, note that the above form may not be unique as we can scale up w_i's and scale down Q_i's accordingly. Nevertheless, for simplicity we can assume that $w_i = 1/s$ for all $i \in [s]$. To handle the general case of unknown weights, we can take two different values for d and "pair-up" the learned Q_i's to figure out the weights. Hence, it suffices to show the following:

Theorem 3.3 (Learning sum of powers of polynomials)**.** *Given black-box access to $f(\mathbf{x}) = \sum_{i=1}^{s} Q_i(\mathbf{x})^{d/2}$ where Q_i's are homogeneous degree-2 polynomials, there is a $\mathrm{poly}(n)$ algorithm that outputs Q_i's up to a permutation, assuming that Q_i's satisfy some non-degeneracy condition (to be specified later within the proof).*

We describe the algorithm in Algorithm 1 and breakdown its analysis into the following steps. For simplicity, we will assume that we have explicit access to the polynomial f (as a list of coefficients). All the operations we are going to perform with f can be made black-box.

[2] This is where we are using the assumption that the Gaussians have zero mean; otherwise, the resulting algebraic circuit reconstruction problem will involve non-homogeneous polynomials making it harder to analyze.

Algorithm 1 Learning sum of powers of quadratics

Input: Access to a polynomial $f(\mathbf{x}) = \sum_{i=1}^{s} Q_i(\mathbf{x})^{d/2}$

Output: The polynomials $Q_i(\mathbf{x})$ for $i \in [s]$, up to a permutation

1. Compute a basis for $U = \langle \mathcal{L}_1 \circ f \rangle = U_1 \oplus U_2 \oplus \cdots \oplus U_s$ (see Step 1 below)
2. Compute a basis for $V = V_1 \oplus V_2 \oplus \cdots \oplus V_s$, where $V_i = \langle G_i^e \rangle$ (see Step 2 below)
3. Compute a basis for $W = W_1 \oplus W_2 \oplus \cdots \oplus W_s$, where $W_i = \mathcal{L}_2 \circ V_i$ and $W = \mathcal{L}_2 \circ V$
4. Decompose V and W into independent subspaces V_i's and W_i's respectively, under the action of $\mathcal{L}_2$, with maximal possible terms s.
5. Recover Q_i^e (up to a constant factor) from $V_i = \langle \pi_L(Q_i)^e \rangle$, and then Q_i's for all $i \in [s]$.

3.1 Step 1: The first set of linear maps $\mathcal{L}_1$

In order to prove the above result, the idea is to use the structure of each term $T_i(\mathbf{x}) \triangleq Q_i(\mathbf{x})^{t/2}$ on the RHS. Suppose there exists $k \in [d/2]$ and a set of linear maps $\mathcal{L} = \{L_1, L_2, \ldots, L_t\}$ from $\mathbb{R}_d[\mathbf{x}]$ to $\mathbb{R}_{d-k}[\mathbf{x}]$ [3] such that

$$\langle \mathcal{L}_1 \circ f \rangle = \langle \mathcal{L}_1 \circ T_1 \rangle \oplus \langle \mathcal{L}_1 \circ T_2 \rangle \oplus \cdots \oplus \langle \mathcal{L}_1 \circ T_s \rangle . \tag{1}$$

Informally, this means that the linear forms are such that each term T_i "simplifies" into a low dimensional subspace whereas f doesn't. In a sense, $\dim \langle \mathcal{L}_1 \circ f \rangle$ is a measure of the "complexity" of f. Using such linear maps $\mathcal{L}_1$, we can prove a lower bound

$$s \geqslant \frac{\dim \langle \mathcal{L}_1 \circ f \rangle}{\max_i \dim \langle \mathcal{L}_1 \circ T_i \rangle}.$$

Indeed, this is how most known algebraic circuit lower bounds are proven. Thus a starting point for what linear maps to choose is to simply use the same linear maps that the lower bound proofs use. For the model we are considering here, i.e., sum of powers of quadratics, it turns out that the following measure works:

$$\mathcal{L}_1 \triangleq \{\pi_L(\partial^k)\},$$

where $\pi_L : \mathbb{R}[\mathbf{x}] \to \mathbb{R}[\mathbf{z}]$ corresponds to the substitution $x_i = \ell_i(\mathbf{z})$ for a linear map $L = (\ell_1(\mathbf{z}), \ell_2(\mathbf{z}), \ldots, \ell_n(\mathbf{z}))$, with $\mathbf{z} = (z_1, z_2, \ldots, z_{n_0})$ being new formal variables. Here n_0 is strictly less than n that we will fix later (see Subsection 3.6 on non-degeneracy conditions). This measure (i.e., the one corresponding to $\mathcal{L}_1$) is called the *affine projections of partials* (APP) of f. That is,

$$\mathsf{APP}(f) \triangleq \dim \left\langle \{\pi_L(\partial^k(f))\} \right\rangle.$$

The choice of k, n_0 and L are hidden from the above definition.

We now elaborate on how the choice of these linear maps i.e., the partial differentiation operators, are motivated by Jennrich's algorithm. Below, we briefly describe a variant of Jennrich's algorithm for symmetric tensor decomposition. We shall discuss it in terms of polynomials rather than tensors to make the connection clear.

[3] The final maps we shall choose will have a co-domain of $\mathbb{R}_{d-k}[\mathbf{z}]$ but this is just a matter of renaming the variables

Here, we have a polynomial $f(\mathbf{x}) = \sum_{i=1}^{s} \ell_i(\mathbf{x})^d$ and the goal is to learn the linear polynomials $\ell_i(\mathbf{x})$ for $i \in [s]$, assuming that ℓ_i's are linearly independent[4]. Furthermore, we also assume that for any $j \leqslant \lfloor d/2 \rfloor$, the polynomials $\ell_i(\mathbf{x})^j$ are linearly independent (these may be treated as additional non-degeneracy conditions); therefore we have $s \leqslant n$. Then we can show that $\langle \partial^1(f) \rangle = \langle \ell_1(\mathbf{x})^{d-1} \rangle \oplus \langle \ell_2(\mathbf{x})^{d-1} \rangle \oplus \cdots \oplus \langle \ell_s(\mathbf{x})^{d-1} \rangle$ and $\langle \partial^2(f) \rangle = \langle \ell_1(\mathbf{x})^{d-2} \rangle \oplus \langle \ell_2(\mathbf{x})^{d-2} \rangle \oplus \cdots \oplus \langle \ell_s(\mathbf{x})^{d-2} \rangle$. Now, to find the ℓ_i's, we find the common eigenvectors of the linear maps (or equivalently, find a common diagonalization of the matrices represented by these linear maps) from $\langle \partial^1(f) \rangle$ and $\langle \partial^2(f) \rangle$ given by ∂^1. This gives us the polynomials $\ell_i(\mathbf{x})^{d-1}$ from which we can recover $\ell_i(\mathbf{x})$.

The following note explains the need for projections after taking the derivatives when defining the APP measure.

Remark 3.4. It is necessary to reduce the number of variables for the above approach to work since if some term $T_i(\mathbf{x}) = (x_1^2 + x_2^2 + \cdots + x_n^2)^{d/2}$, the measure $\dim \langle \partial^k(T_i) \rangle$ can be proven to be "maximal" i.e., $\binom{n+k-1}{k}$, which isn't conducive for (1) to hold.

We have the following properties for APP.

Proposition 3.5. For degree-d homogeneous polynomials f and g,

- $\mathsf{APP}(f+g) \leqslant \mathsf{APP}(f) + \mathsf{APP}(g)$
- $\mathsf{APP}(f) \leqslant \min\{\binom{n+k-1}{k}, \binom{n_0+d-k-1}{d-k}\}$, and this is tight (indeed a "randomly" chosen f has APP close to the RHS)

Proof sketch. Note that

$$\{\pi_L(\partial^k(f+g))\} \subseteq \left\langle \{\pi_L(\partial^k(f))\} \right\rangle + \left\langle \{\pi_L(\partial^k(g))\} \right\rangle,$$

since ∂^k is a linear operator over the space $\mathbb{R}_d[\mathbf{x}]$. Taking $\dim(\cdot)$ on both sides and using $\dim(U+V) \leqslant \dim(U) + \dim(V)$ proves the first item.

The second item follows since the number of linear operators i.e., the size of $\{\pi_L(\partial^k(f))\}$ is $\binom{n+k-1}{k}$ and the degree of the polynomials after taking k derivatives is $d-k$. Thus $\langle \{\pi_L(\partial^k(f))\} \rangle$ is a subspace of $\mathbb{R}_{d-k}[\mathbf{z}]$ whose dimension is $\binom{n_0+d-k-1}{d-k}$. □

Let us denote $U_i \triangleq \mathcal{L}_1 \circ T_i$ for $i \in [s]$ and $U \triangleq \langle \mathcal{L}_1 \circ f \rangle$. Then, we can compute (a basis for U) using (black-box) access to f. However, the main challenge is to also recover the subspaces U_i's (and then the terms T_i subsequently).

By using the product and chain rule of derivatives, it is not hard to show that $U_i \subseteq \langle \mathbf{z}^k \cdot \pi_L(Q_i)^{d/2-k} \rangle$. Further, for the direct sum (1) to hold, we have the following stronger condition.

Proposition 3.6. With probability[5] $1 - o(1)$ over the choice of the linear map L (i.e., the coefficients of the linear functions are picked uniformly at random from an arbitrary, but sufficiently large set of reals), we have that

- $U = U_1 \oplus U_2 \oplus \cdots \oplus U_s$, and

[4] We may assume that $d = 3$, but some of the non-degeneracy conditions that follow might only work for large enough d; this needs to be verified.

[5] Even though we state this in terms of a random choice of L, it suffices if the coefficients used in L satisfy an appropriate non-degeneracy condition.

- $U_i = \left\langle \mathbf{z}^k \cdot \pi_L(Q_i)^{d/2-k} \right\rangle$ (With probability 1, LHS is a subset of RHS)

Hence, $\dim(U) = s \cdot \binom{n_0+k-1}{k}$.

Proof sketch. Note that $\partial_{x_1}(Q_i^{d/2}) \in \left\langle Q_i^{d/2-1} \cdot \mathbf{x} \right\rangle$ by applying chain rule. Applying more derivatives and the product rule, we can see that $\partial^m(Q_i^{d/2}) \in \left\langle \mathbf{x}^k \cdot Q_i^{d/2-k} \right\rangle$. This can be formally proved by an induction on k. Then, applying the substitution of $\mathbf{z}$ variables gives that $U_i \subseteq \left\langle \mathbf{z}^k \cdot \pi_L(Q_i)^{d/2-k} \right\rangle$. To show that they are actually equal under a non-degenerate case, we can argue that each level of the induction (on k) does not unexpectedly decrease the dimension. It only remains to prove that U_i's are independent. This is a bit technical, however the intuition is that as long as U has a sufficiently large dimension that allows for U_i's to be independent, this is possible (under some non-degeneracy condition). We will prove this in Subsection 3.6. □

3.2 Step 2: Multi-gcd

Let $e = d/2 - k$, $G_i \triangleq \pi_L(Q_i)$, $V_i \triangleq \langle G_i^e \rangle$ and $V \triangleq V_1 + V_2 + \cdots + V_s$. We can show

Proposition 3.7. With probability $1 - o(1)$ over the choice of L, we have that $V = V_1 \oplus V_2 \oplus \cdots \oplus V_s$.

Here, we skip the details justifying how to compute (a basis of) of V, but only mention that this is the same as computing a "multi-gcd" of elements in (the known vector space) U. Although this multi-gcd step may be avoided by working with U_i's and U instead of V_i's and V respectively, taking multi-gcd makes the subsequent steps easier.

3.3 Step 3: The second set of linear maps $\mathcal{L}_2$

Recall that

$$V = V_1 \oplus V_2 \oplus \cdots \oplus V_s.$$

Let $g(\mathbf{z}) \in V$ be a randomly[6] chosen element of V. Now, suppose there exists a set of linear maps $\mathcal{L}_2$ from V (which is a subspace of $\mathbb{R}[\mathbf{z}]^{2e}$) to some subspace W of $\mathbb{R}[\mathbf{w}]^{2e-k}$ where $\mathbf{w} = (w_1, w_2, \ldots, w_{m_0})$ and $m_0 \leqslant n_0$, such that the following property holds:

$$W = W_1 \oplus W_2 \oplus \cdots \oplus W_s,$$

where $W_i \triangleq \mathcal{L}_2 \circ V_i$ and $W \triangleq W_1 + W_2 + \cdots + W_s$. Then, the idea is to use the above structure, i.e., the fact that the linear maps are all some block diagonal matrices under appropriate bases for V and W, and recover V_i's and W_i's by doing a simultaneous block diagonalization. However, note that one can't hope to perform this with any $\mathcal{L}_2$; for example taking just the identity map can never reveal anything about V_i's. We need the decomposition to be unique (in its most reduced form, i.e., it has to be further indecomposable). Taking the following linear maps works where $P = (p_1(\mathbf{w}), p_2(\mathbf{w}), \ldots, p_{n_0}(\mathbf{w}))$ is a random linear substitution:

$$\mathcal{L}_2 \triangleq \{\pi_P(\partial^k)\}.$$

This follows by another application of Proposition 3.6.

[6] Again, we only need some non-degeneracy condition to be satisfied by g, but it turns out that taking random instances (i.e., the coefficient of each basis element is picked from a large enough set uniformly at random) are non-degenerate w.h.p.; this follows by the Schwartz-Zippel lemma.

3.4 Step 4: Vector space decomposition

We now have access to a basis of the vector spaces V and W and our goal is to decompose them into further-indecomposable (i.e., maximal s possible) subspaces such that we have, $V = V_1 \oplus V_2 \oplus \cdots \oplus V_s$ and $W = W_1 \oplus W_2 \oplus \cdots \oplus W_s$, where recall that $W_i \triangleq \mathcal{L}_2 \circ V_i$. Additionally, we have that $\dim(V_i) = 1$. In order to show uniqueness and efficient computation of the above decomposition, we analyze the *adjoint algebra* associated with $\mathcal{L}_2$.

Suppose we have computed a basis $\{g_1, g_2, \ldots, g_s\}$ of V. We have $q \triangleq \dim(W_i) = \binom{m_0+k-1}{k}$, as $W_i = \left\langle \mathbf{w}^k \cdot \pi_P(G_i)^{e-k} \right\rangle$. Hence $\dim(W) = sq$. We shall represent each linear map $L \in \mathcal{L}_2$ as a $sq \times s$ matrix using the basis $\{g_1, g_2, \ldots, g_s\}$ for V and an arbitrary basis for W. We then define the adjoint algebra corresponding to $\mathcal{L}_2$ as

$$\mathrm{adj}(\mathcal{L}_2) \triangleq \{(D, E) \in \mathbb{R}^{s \times s} \times \mathbb{R}^{sq \times sq} : LD = EL, \text{ for all } L \in \mathcal{L}_2\}.$$

Using the specific maps $\mathcal{L}_2$ we work with, i.e., the affine projections of partials, we can show that the adjoint algebra is trivial in the following sense.

Proposition 3.8. Let $A \in \mathbb{R}^{s \times s}$ be the change of basis (of V) matrix from $\{G_1^e, G_2^e, \ldots, G_s^e\}$ to $\{g_1, g_2, \ldots, g_s\}$. Then we have

$$A^{-1}\mathrm{adj}(\mathcal{L}_2)_1 A = \mathcal{D},$$

where $\mathrm{adj}(\mathcal{L}_2)_1 \triangleq \{D : (D, E) \in \mathrm{adj}(\mathcal{L}_2)\}$ and $\mathcal{D} \triangleq \{\mathrm{diag}(a_1, a_2, \ldots, a_s) : a_1, a_2, \ldots, a_s \in R\}$.

Proof sketch. At least one direction of the above proposition is easy to prove: consider an arbitrary $B \triangleq \mathrm{diag}(a_1, a_2, \ldots, a_n) \in \mathrm{adj}(\mathcal{L}_2) \in \mathcal{D}$. To show that $ABA^{-1} \in \mathrm{adj}(\mathcal{L}_2)_1$, we note that in the basis $\{G_1^e, G_2^e, \ldots, G_s^e\}$, applying the linear maps $\mathcal{L}_2$ results in a block diagonal $sq \times s$ matrix where each "block" is a $q \times 1$ matrix. Hence, any diagonal matrix B results in an element (B, B') in $\mathrm{adj}(\mathcal{L}_2)$, where B' is an appropriate block diagonal matrix obtained by "repeating" B q many times.

To show the other direction, we will use the following fact (without proof): for any $(j, i) \in [sq] \times [s]$ there always exists a linear map $L \in \langle \mathcal{L}_2 \rangle$ such that, when it is represented in the $\{G_1^e, G_2^e, \ldots, G_s^e\}$ basis, there is a 1 at the (j, i)-th entry and 0 everywhere else.

Then using $LD = EL$ for the above L, we conclude that $A^{-1}DA$ has to be a diagonal matrix. To see this, in the basis $\{G_1^e, G_2^e, \ldots, G_s^e\}$, note that LD is just the i-th column of D placed as the j-th row and 0 everywhere else, whereas EL is the j-th row of E placed as the i-th column and 0 everywhere else. Hence, $LD = EL$ implies that D has to be diagonal and E has to be block diagonal ($q \times q$ blocks). □

By Proposition 3.8, we can compute the matrix A by finding the eigenvectors of a random element from $\mathrm{adj}(\mathcal{L}_2)_1$. This follows because $\mathrm{adj}(\mathcal{L}_2)_1 A = A\mathcal{D}$ implies that the columns of A are eigenvectors of the elements of $\mathrm{adj}(\mathcal{L}_2)_1$. Now, we first compute a basis for $\mathrm{adj}(\mathcal{L}_2)_1$ (which is a subspace of $s \times s$ real matrices). Then, for a random element, the eigenvalues are all distinct, hence we can recover the eigenvectors, i.e., the basis change matrix A. Once we have A, we have computed the polynomials G_i^e for all $i \in [s]$.

3.5 Step 5: Recovering the quadratics

Now, we give a brief idea on how to find Q_i^e (up to a constant factor) from $G_i^e = \pi_L(Q_i)^e$. At a high level, this is done by repeating the above steps with $\mathrm{poly}(d)$ many independent choices of the substitution L and "gluing together" the corresponding G_i's; we skip further details here. Once

we have Q_i^e, the quadratic form Q_i can be found by using a polynomial factorization algorithm. Finally this gives us $Q_i(\mathbf{x})$ up to a constant factor, for all $i \in [s]$. Then these factors can be found by solving a linear system.

3.6 Non-degeneracy conditions

Most of the non-degeneracy conditions, which are sometimes hidden in a "high probability" statements boil down to a bunch of vectors in a vector space being linearly independent. Let us look at the first non-degeneracy condition we needed, i.e., in Proposition 3.6: Here, the dimension of $U = \langle \mathcal{L}_1 \circ f \rangle$ is at most $\min\{\binom{n+k-1}{k}, \binom{n_0+d-k-1}{d-k}\}$ (by Proposition 3.5) whereas for the direct sum to hold, we must have that $\dim(U) = s \cdot \binom{n_0+k-1}{k}$. Therefore, it suffices if we take k to be very small and $n_0 = n^\epsilon$ for some constant ϵ. Then,

$$\begin{aligned}
\dim(U) &= s \cdot \binom{n_0 + k - 1}{k} \\
&\approx \mathrm{poly}(n) \cdot n_0^k \\
&\leqslant n^{\epsilon k + O(1)} \\
&\leqslant \binom{n + k - 1}{k} \leqslant \binom{n_0 + d - k - 1}{d - k},
\end{aligned}$$

as desired. This determines a way of choosing the values for d, k and n_0 depending on the value of s. The value of $m_0 < n_0$ can also be set in a similar way, in order to satisfy the direct sum property for W_i's, i.e., the image of the linear maps from $\mathcal{L}_2$. There are a couple more non-degeneracy conditions (implicitly or explicitly imposed by the algorithm) that are needed for Step 2 (multi-gcd) to work, but we ignore those details here.

3.7 Time complexity

The total running time of the above algorithm is polynomial in n (recall that $s = \mathrm{poly}(n)$ and $d = O(1)$). To see this, we note that Step 1 (and Step 3) only involves computing a set of linear maps by taking partial derivatives, and some substitutions to reduce the number of variables. Since d and thus the order of derivatives k are constants, this can be done in polynomial time. These operations can be efficiently performed in the black-box setting as well. Step 4 is the other crucial component of the algorithm. Again, it only involves computing a basis of a fixed subspace and eigenvectors of a given matrix, both of which can be done in polynomial time.

References

[BHKX22] 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. 2

[CGK+23] 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. *arXiv preprint arXiv:2311.07284*, 2023. 2, 3

[GHK15] Rong Ge, Qingqing Huang, and Sham M Kakade. Learning mixtures of gaussians in high dimensions. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 761–770, 2015. 2

[GKS20] Ankit Garg, Neeraj Kayal, and Chandan Saha. Learning sums of powers of low-degree polynomials in the non-degenerate case. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 889–899. IEEE, 2020. 1, 2, 3

[H+70] Richard A Harshman et al. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. 1970. 1

[KS19] Neeraj Kayal and Chandan Saha. Reconstruction of non-degenerate homogeneous depth three circuits. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 413–424, 2019. 2

[LRA93] Sue E Leurgans, Robert T Ross, and Rebecca B Abel. A decomposition for three-way arrays. *SIAM Journal on Matrix Analysis and Applications*, 14(4):1064–1083, 1993. 1