Electronic Colloquium on Computational Complexity, Report No. 45 (2020)

Learning sums of powers of low-degree polynomials in the non-degenerate case

Ankit Garg
Microsoft Research India
garga@microsoft.com

Neeraj Kayal
Microsoft Research India
neeraka@microsoft.com

Chandan Saha
Indian Institute of Science
chandan@iisc.ac.in

April 15, 2020

Abstract

We develop algorithms for writing a polynomial as sums of powers of low degree polynomials. Consider an n-variate degree-d polynomial f which can be written as

$$f = c_1 Q_1^m + \ldots + c_s Q_s^m,$$

where each $c_i \in \mathbb{F}^\times$, Q_i is a homogeneous polynomial of degree t, and $tm = d$. In this paper, we give a $\text{poly}((ns)^t)$-time learning algorithm for finding the Q_i's given (black-box access to) f, if the $Q_i's$ satisfy certain *non-degeneracy* conditions and n is larger than d^2. The set of degenerate Q_i's (i.e., inputs for which the algorithm does not work) form a non-trivial variety and hence if the Q_i's are chosen according to any reasonable (full-dimensional) distribution, then they are non-degenerate with high probability (if s is not too large). This problem generalizes symmetric tensor decomposition, which corresponds to the $t = 1$ case and is widely studied, having many applications in machine learning. Our algorithm (for $t = 2$) allows us to solve the moment problem for mixtures of zero-mean Gaussians in the non-degenerate case.

Our algorithm is based on a scheme for obtaining a learning algorithm for an arithmetic circuit model from lower bound for the same model, provided certain non-degeneracy conditions hold. The scheme reduces the learning problem to the problem of decomposing two vector spaces under the action of a set of linear operators, where the spaces and the operators are derived from the input circuit and the complexity measure used in a typical lower bound proof. The non-degeneracy conditions are certain restrictions on how the spaces decompose. Such a scheme is present in a rudimentary form in an earlier work [KS19]. Here, we make it more general and detailed, and potentially applicable to learning other circuit models.

An exponential lower bound for the representation above (also known as homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}$ circuits) is known using the shifted partials measure. However, the number of linear operators in shifted partials is exponential and also the non-degeneracy condition emerging out of this measure is unlikely to be satisfied by a random $\Sigma \wedge \Sigma\Pi^{[t]}$ circuit when the number of variables is large with respect to the degree. We bypass this hurdle by proving a lower bound (which is nearly as strong as the previous bound) using a novel variant of the partial derivatives measure, namely *affine projections of partials* (APP). The non-degeneracy conditions appearing from this new measure are satisfied by a random $\Sigma \wedge \Sigma\Pi^{[t]}$ circuit. The APP measure could be of independent interest for proving other lower bounds.

ISSN 1433-8092

Contents

1 Introduction

Arithmetic circuits form a natural model for computing polynomials. They compute polynomials using basic arithmetic operations such as addition and multiplication.[1] Formally, an arithmetic circuit is a directed acyclic graph such that the sources are labelled with variables or constants from the underlying field, the internal nodes (gates) are labelled with the arithmetic operations and the sink(s) outputs the polynomial(s) computed by the circuit.[2] *Size* of a circuit is the number of edges in the underlying graph, and *depth* is the length of a longest path from a source to a sink node. The three main questions of interest regarding arithmetic circuits are the following:

- **Lower bounds.** Is there an "explicit" polynomial that requires super-polynomial sized arithmetic circuits to compute? This[3] is the famed VP vs VNP question (an arithmetic analogue of the P vs NP question[4]).

- **Polynomial Identity Testing (PIT).** Here the question is, given[5] an arithmetic circuit, determine if its output is identically zero. There is an easy randomized algorithm for this problem (plug in random values and check if the output is zero). Finding a deterministic algorithm is a major open question in this field.

- **Reconstruction.** Here the question is, given[6] a polynomial, find the smallest (or approximately smallest) arithmetic circuit computing it.

For all the above questions, there is very little progress on them for general arithmetic circuits. So, a lot of effort has gone into studying them for restricted classes of arithmetic circuits (like constant depth, multilinear, set-multilinear, non-commutative circuits etc.). We refer the interested reader to the excellent surveys [SY10, CKW11, Sap15] on this topic.

A lot of interconnections are known between the three above-mentioned problems, some of which we touch upon below.

- **Lower bounds and PIT.** There are several connections that go both ways between lower bounds and PIT. It is known that lower bounds for general arithmetic circuits would imply PIT algorithms via the hardness vs randomness tradeoff (in the algebraic setting) [KI04, DSY10]. Furthermore, non-trivial (deterministic) PIT algorithms also imply lower bounds [HS80, KI04, Agr05]. While these concrete connections are not always present for restricted circuit models, several PIT algorithms have been inspired by corresponding lower bounds, e.g., [RS05, FS13, OSlV16, For15a].

- **Lower bounds and reconstruction.** It is known that worst case reconstruction of a circuit model implies lower bound for the same model [FK09, Vol16] (also see the discussion in

[1]One can also allow division, but it is a classical result that one can eliminate division operations from an arithmetic circuit without too much blow up in the circuit size [Str73].

[2]Sometimes, one can let the edges going into addition gates to be labelled with field constants to allow scalar linear combinations. But, all these models can be interconverted to each other without too much blowup in the circuit size.

[3]One could also consider various other notions of explicitness while framing the lower bounds question.

[4]Or rather #P vs NC.

[5]Either as a black-box or explicitly.

[6]Again, either as a black box or explicitly.

Section 1.4). In the other direction, several reconstruction algorithms are inspired by lower bounds for the corresponding models [KS06, FS13, GKL11, GKQ14, KNST17, KNS19, KS19] (see also the discussion in Section 1.5).

- **PIT and reconstruction.** In one direction, a deterministic (worst case) reconstruction algorithm clearly implies a deterministic PIT algorithm (both in the black-box model) since the reconstruction algorithm would have to output an extremely small circuit when the circuit computes the zero polynomial. Randomized (or average-case) reconstruction algorithms may not have anything to do with deterministic PIT algorithms, of course. In the other direction, as discussed in [SY10], black-box PIT algorithms seemingly can help in designing reconstruction algorithms. This is because a black-box PIT algorithm outputs a list of evaluation points such that any circuit from the class being considered evaluates to a non-zero value on at least one of points, and hence any two circuits in the class computing different polynomials evaluate to a different value on at least one of the points.[7] So, the list of evaluation points determines the circuit and now it remains to be seen if one can efficiently reconstruct the circuit from these evaluations.[8] The reconstruction algorithms for sparse polynomials and constant top fan-in depth three circuits and read-once algebraic branching programs are some examples of reconstruction using PIT ideas [KS01, Shp09, KS09a, FS13]. Of course, deterministic PIT algorithms can also be sometimes used to get deterministic reconstruction algorithms when randomized ones are known [KS06, FS13].

To summarize, the three main problems in arithmetic complexity are richly interrelated and progress on one question spurs progress on the others. Hence, it is imperative to find more connections between these problems. This paper continues the line of work in [KS19] on building a new connection between lower bounds and reconstruction. We build on the work of [KS19] to further develop a meta framework[9] that yields reconstruction algorithms in the *non-degenerate* setting from lower bounds for the corresponding circuit models. In addition to developing this framework further, we implement this framework to learn sums of powers of low degree polynomials in the non-degenerate case (described in Section 1.1). We remark that assuming some kind of non-degeneracy conditions might be essential for designing efficient learning algorithms; otherwise for most circuit models, one will have to assume constant top fan-in to get polynomial time algorithms. This is because of various hardness results about reconstruction in the worst case (see Section 1.4). The usefulness of assuming non-degeneracy conditions is best illustrated by the following example.

Consider the model of homogeneous depth three powering circuits. This corresponds to the representation

$$f(\mathbf{x}) = \sum_{i=1}^{s} \ell_i(\mathbf{x})^d,$$

where ℓ_i's are linear polynomials. Finding such a decomposition with the minimum possible s is NP-hard even for degree $d = 3$ (this corresponds to symmetric tensor decomposition) [Hås90, Shi16]. Regardless of the NP-hardness, one can design algorithms for this model under reasonable

[7] Assuming the class is closed under subtractions.

[8] Of course, a random set of points forms a hitting set and it seems hard to reconstruct the circuit given its evaluations on random points. However, the hitting sets constructed for deterministic PIT algorithms typically have a lot of special structures which could be exploited for reconstruction.

[9] In [KS19], this framework is present in a rudimentary form.

assumptions and such algorithms are widely used in machine learning (see Section 1.3). One such algorithm, attributed to Jennrich [Har70, LRA93], says that given $f(\mathbf{x}) = \sum_{i=1}^{s} \ell_i(\mathbf{x})^3$ with $s \leq n$ and ℓ_i's linearly independent, we can find the ℓ_i's in polynomial time.[10] A couple of things to notice about the assumptions are:

- $s \leq n$: The number of summands that the algorithm can handle is (up to a small constant) the best known lower bound we can prove for this model (sums of cubes of linear forms or order-3 symmetric tensor decomposition).
- The set of inputs for which the algorithm does not work, i.e., when ℓ_i's are linearly dependent, form a non-trivial variety (if $s \leq n$). So, the algorithm would work for "random" ℓ_i's with high probability.

We hope to generalize the above kind of non-degenerate case learning algorithms to other circuit models. The circuit size which one might be able to handle if one implements the meta framework will depend on the lower bound one can prove for the circuit model. Since tensor decomposition algorithms (which corresponds to reconstruction for a very simple arithmetic circuit model) are widely used in machine learning, our meta framework raises the exciting possibility of importing techniques from arithmetic complexity to machine learning via reconstruction of various circuit models in the non-degenerate case. We mention one such possibility in Section 1.3.

Let us briefly describe the roadmap for the rest of this section now. In Section 1.1, we describe our main results about learning sums of powers of low degree polynomials in the non-degenerate case. In Section 1.2, we describe our techniques: the meta framework for turning lower bounds into reconstruction algorithms, the implementation for sums of powers of low degree polynomials and the non-degeneracy conditions needed for our algorithm to work. In Section 1.3, we describe the connection to mixtures of Gaussians. Finally, in Sections 1.4 and 1.5, we review hardness results about reconstruction and some previous work.

1.1 The model and our results

We study the learning problem for an interesting subclass of depth four arithmetic circuits which is a generalization of depth three powering circuits or symmetric tensors. A circuit in this class, computing an n-variate degree-d polynomial $f \in \mathbb{F}[\mathbf{x}]$, is an expression

$$f = c_1 Q_1^m + \ldots + c_s Q_s^m, \qquad (1)$$

where each $c_i \in \mathbb{F}^\times$, Q_i is a homogeneous polynomial[11] of degree t, and $tm = d$. Such a circuit is called a homogeneous $\Sigma \wedge \Sigma \Pi^{[t]}(s)$ circuit.[12] The parameter t is typically much smaller than d.

[10]There are natural extensions of this result for larger values of d that can handle a larger number of components (roughly matching the best lower bounds we can prove for this model), e.g., see [DLCC07, ABG+14, BCMV14, KS19]. Other algorithms include [Kay11, GKP18].

[11]The result in this paper holds even if $f = c_1 Q_1^{m_1} + \ldots + c_s Q_s^{m_s}$, where each Q_i (not necessarily homogeneous) has degree $t_i \leq t$ and $t_i m_i = d$ for all $i \in [s]$. We present the analysis assuming homogeneity and uniform exponent m for simplicity of exposition.

[12]Technically, the expressions of this kind are known as $\Sigma \wedge \Sigma \Pi^{[t]}(s)$ *formulas*. But, there is only a minor distinction between formulas and circuits in the constant depth case. Furthermore, in the random circuit setting, even this minor distinction is not there.

We show that a homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}$ circuit can be reconstructed efficiently if it satisfies certain *non-degeneracy* conditions. We defer stating these conditions precisely to the end of this section, but it is worth mentioning that a *random* $\Sigma \wedge \Sigma\Pi^{[t]}$ circuit is non-degenerate with high probability. In other words, if the coefficients of the monomials in $Q_1, \ldots, Q_s$, in Equation (1), are chosen uniformly at random from a sufficiently large subset of $\mathbb{F}$ then the resulting circuit is non-degenerate with high probability. In this sense, *almost all* homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}$ circuits can be reconstructed efficiently. The following theorem is proved in Section 2. We will assume that factoring univariate polynomials over $\mathbb{F}$ can be done in randomized polynomial time[13].

Theorem 1 (Learning *non-degenerate* sums of powers of low degree polynomials)**.** *Let $n, d, s, t \in \mathbb{N}$ such that $n \geq d^2$, $2 \leq t \leq \sqrt{\frac{\log d}{10 \cdot \log \log d}}$, $|\mathbb{F}| \geq (ns)^{150 \cdot t}$, $\mathrm{char}(\mathbb{F}) = 0$ or $> 2d$ and $s \leq \min(n^{\frac{d}{1100 \cdot t^2}}, \exp(n^{\frac{1}{30 \cdot t^2}}))$. Then, there is a randomized algorithm which when given black-box access to an n-variate degree-d non-degenerate polynomial $f = c_1 Q_1^m + \ldots + c_s Q_s^m$, where each $c_i \in \mathbb{F}^{\times}$, Q_i is a homogeneous polynomial of degree t, and $tm = d$ and the total number of monomials in Q_i's is σ, outputs (with high probability) $Q'_1, \ldots, Q'_s$ such that there exist a permutation $\pi : [s] \to [s]$ and non-zero constants $c'_1, \ldots, c'_s$ so that $Q'_i = c'_i Q_{\pi(i)}$ for all $i \in [s]$. The running time of the algorithm is $\mathrm{poly}(n, \sigma, s^t)$.*[14]

Remarks.

1. *Non-degeneracy.* The non-degeneracy conditions are explicitly mentioned in Section 1.2.4.

2. *Bounds on t and s.* The upper bounds on the parameters t and s in Theorem 1 originate from our analysis (especially, the part in Section 2.3 showing that a random $\Sigma \wedge \Sigma\Pi^{[t]}$ circuit is non-degenerate with high probability). We have not optimized this analysis in an attempt to keep it relatively simple. Given that the lower bounds (stated in Theorem 2) hold for a large range of t and s, it may be possible to tighten our analysis significantly.

3. *$t > 1$ substantially different from $t = 1$.* While we state the theorem for slightly super-constant values of t, one should think of t being a constant as the main setting (in which case, the running time of our algorithm is polynomial). Even the $t = 2$ case, which was open before, is substantially different from the $t = 1$ case (as discussed in Section 1.2.3) and is relevant to the problem of mixtures of Gaussians (see Section 1.3).

4. *Uniqueness of Q_i's.* A corollary of the analysis of our algorithm is that for a non-degenerate $f = c_1 Q_1^m + \ldots + c_s Q_s^m$ (which holds if the Q_i's are chosen randomly), this representation is of the smallest size and also unique. That is, if $f = \widetilde{c}_1 \widetilde{Q}_1^m + \ldots + \widetilde{c}_s \widetilde{Q}_{\widetilde{s}}^m$ with $\widetilde{s} \leq s$, then $\widetilde{s} = s$ and there exist a permutation $\pi : [s] \to [s]$ and non-zero constants $d_1, \ldots, d_s$ such that $\widetilde{Q}_i = d_i Q_{\pi(i)}$ for all $i \in [s]$.

Non-degeneracy conditions are satisfied if we choose the coefficients of the Q_i's randomly. This gives us the following corollary.

Corollary 1.1 (Learning *random* sums of powers of low degree polynomials)**.** *Let $n, d, s, t \in \mathbb{F}$ be as in Theorem 1. There is a randomized algorithm which when given black-box access to an n-variate degree-d*

[13] Univariate polynomials over finite fields can be factored in randomized polynomial time [Ber70], and over Q, they can be factored in deterministic polynomial time [LLL82].

[14] Here, $\exp(x) = 2^x$. Once we know $Q'_1, \ldots, Q'_s$, we can determine the non-zero constants $d_1, \ldots, d_s$ such that $f = d_1 Q_1'^m + \ldots + d_s Q_s'^m$ in randomized polynomial time.

polynomial $f = c_1 Q_1^m + \ldots + c_s Q_s^m$, where each $c_i \in \mathbb{F}^\times$, Q_i is a homogeneous polynomial of degree t, and $tm = d$ and the coefficients of Q_i's are chosen uniformly and independently at random from a set $S \subset \mathbb{F}$ of size $|S| \geq (ns)^{150 \cdot t}$, outputs (with high probability) $Q_1', \ldots, Q_s'$ such that there exist a permutation $\pi : [s] \to [s]$ and non-zero constants $c_1', \ldots, c_s'$ so that $Q_i' = c_i' Q_{\pi(i)}$ for all $i \in [s]$. The running time of the algorithm is $\mathrm{poly}((ns)^t)$.

1.2 Techniques: Learning from lower bounds

The novelty of our approach lies in the use of lower bound techniques in the design of learning algorithms. Such connections are known for certain classes of Boolean circuits, in particular AC^0 and $\mathrm{AC}^0[p]$ circuits [LMN93, CIKK16]. The influence of lower bound techniques on learning is also apparent in the case of ROABP reconstruction [BBB+00, KS06]. However, our approach differs substantially from these previous works and also uses lower bounds to design algorithms in the *non-degenerate* case. At a high level, our technique can be summarized as a fancy *reduction to linear algebra*. In Section 1.2.1, we will see how lower bounds are typically proven in arithmetic complexity. Section 1.2.2 describes our meta framework of turning lower bounds into learning algorithms. In Section 1.2.3, we discuss how implement the framework for learning sums of powers of low degree polynomials. Finally in Section 1.2.4, we state the non-degeneracy conditions we require explicitly.

1.2.1 A typical lower bound proof

Many of the circuit classes for which good lower bounds are known are of the form $T_1 + \ldots + T_s$, where each polynomial T_i is "simple" in some sense[15]. The lower bound problem for such a class $\mathcal{C}$ is to find an explicit polynomial f such that any representation of the form $f = T_1 + T_2 + \ldots + T_s$, where each T_i is a simple polynomial, requires s to be large. A typical lower bound strategy finds such an f by constructing a set of linear maps $\mathcal{L}$ from the vector space of polynomials to some appropriate vector space such that the following properties hold:

- $\dim(\langle \mathcal{L} \circ T \rangle)$ is *small* (say $\leq r$) for every simple polynomial T, [16]
- $\dim(\langle \mathcal{L} \circ f \rangle)$ is *large* (say $\geq R$).

Then, we have

$$\begin{aligned} & f = T_1 + T_2 + \ldots + T_s \\ & \Rightarrow \langle \mathcal{L} \circ f \rangle \subseteq \langle \mathcal{L} \circ T_1 \rangle + \cdots + \langle \mathcal{L} \circ T_s \rangle \\ & \Rightarrow \dim(\langle \mathcal{L} \circ f \rangle) \leq \dim(\langle \mathcal{L} \circ T_1 \rangle) + \cdots + \dim(\langle \mathcal{L} \circ T_s \rangle). \end{aligned} \tag{2}$$

This implies that $s \geq R/r$. Now, let us see how the set of linear maps $\mathcal{L}$ could potentially play a key role in learning class $\mathcal{C}$.

[15]For example, in case of $\Sigma \wedge \Sigma\Pi^{[t]}(s)$ circuits, T_i is a power of a degree-t polynomial. One can also get such representations from general circuits by various depth reduction theorems [AV08, Koi12, Tav13].

[16]Here, $\langle S \rangle$ denotes the $\mathbb{F}$-linear span of a set of polynomials S.

1.2.2 Reduction to vector space decomposition - a recipe for learning

The corresponding learning problem for $\mathcal{C}$ is the following: Given a polynomial f that can be expressed as $f = T_1 + \ldots + T_s$, where each T_i is a simple polynomial, can we efficiently recover the T_i's? It turns out that the set of linear maps $\mathcal{L}$ (used to prove lower bounds) can now be used to devise an efficient learning algorithm via the following meta-algorithm, which works if the expression $T_1 + \ldots + T_s$ is "non-degenerate". Let us explain what we mean by non-degeneracy. One might expect that if the T_i's are chosen randomly, then for some choice of linear maps $\mathcal{L}$, the subspace condition in Equation (2) becomes an equality and the sums become direct sums, i.e.,

$$\langle \mathcal{L} \circ f \rangle = \langle \mathcal{L} \circ T_1 \rangle \oplus \cdots \oplus \langle \mathcal{L} \circ T_s \rangle. \tag{3}$$

Existence of linear maps $\mathcal{L}$ satisfying Equation (3) for random T_i's is the starting point for our learning framework. A couple of things are important to state here:

- If Equation (3) is satisfied for even a single choice of T_i's, then it is satisfied for random T_i's because of the Schwarz-Zippel lemma [Sch80, Zip79].
- Equation (3) implies a *tight separation* within class $\mathcal{C}$. So, a prerequisite for a lower bound method to be useful for our learning framework is that it should be able to prove a tight separation for that model. In fact, Equation (3) is usually proven by exhibiting an explicit polynomial for which the linear maps in question yield a tight separation (see Lemma 1.1).

We will also need that $\mathcal{L} = \mathcal{L}_2 \circ \mathcal{L}_1$, i.e., $\mathcal{L}$ is a combination of two sets of linear maps[17] and Equation (3) holds for both $\mathcal{L}_1$ and $\mathcal{L}$. We say that the expression $T_1 + \cdots + T_s$ is non-degenerate if Equation (3) holds for both $\mathcal{L}_1$ and $\mathcal{L}$. Now given[18] $f = T_1 + \cdots + T_s$, we have access to

$$\begin{aligned} U &:= \langle \mathcal{L}_1 \circ f \rangle = \langle \mathcal{L}_1 \circ T_1 \rangle \oplus \ldots \oplus \langle \mathcal{L}_1 \circ T_s \rangle \quad \text{and} \\ V &:= \langle \mathcal{L}_2 \circ \mathcal{L}_1 \circ f \rangle = \langle \mathcal{L}_2 \circ (\mathcal{L}_1 \circ T_1) \rangle \oplus \ldots \oplus \langle \mathcal{L}_2 \circ (\mathcal{L}_1 \circ T_s) \rangle. \end{aligned} \tag{4}$$

If we can recover the $\langle \mathcal{L}_1 \circ T_i \rangle$ for all i, then usually one can recover the T_i's[19]. Towards this, a crucial property of the linear maps $\mathcal{L}_2$ (from U to V) is that $\mathcal{L}_2$ maps each component space $\langle \mathcal{L}_1 \circ T_i \rangle$ to the corresponding component space of V. This motivates the following problem.

Problem 1 (Vector space decomposition)**.** Given two vector spaces U and V and a set of linear maps $\mathcal{L}$ from U to V, find a decomposition

$$\begin{aligned} U &= U_1 \oplus \ldots \oplus U_s \\ V &= V_1 \oplus \ldots \oplus V_s, \end{aligned}$$

such that $\langle \mathcal{L} \circ U_i \rangle \subseteq V_i$ for all $i \in [s]$ (if such a decomposition exists). Moreover, we can ask that each of the pairs (U_i, V_i) be further indecomposable with respect to $\mathcal{L}$.

[17] For example, k^{th} order partial derivatives are a composition of $(k-1)^{\text{th}}$ order partial derivatives and first order partial derivatives, i.e., $\partial_{\mathbf{x}}^k = \partial_{\mathbf{x}}^{k-1} \circ \partial_{\mathbf{x}}$.

[18] This framework should be applicable given only black-box access to f using standard tricks (as we show for our problem) and we won't go into these details in this overview.

[19] For example, one can recover a homogeneous polynomial if given all its degree-k partial derivatives.

Amazingly, polynomial time algorithms are known for a symmetric version of this problem, where $U = V$ and we require $U_i = V_i$.[20] The algorithm was discovered in [CIK97] based on the algorithms developed for decomposition of algebras (e.g., see [FR85, Rón90, Ebe91]). The algorithm works over finite fields, $\mathbb{C}$ and $\mathbb{R}$ (if the input is over $\mathbb{Q}$, then the algorithm outputs a decomposition over an extension field). We give a simple reduction in Section B that reduces the vector space decomposition problem to the symmetric version. However, since we are in a specialized setting, we can design a simpler algorithm using the ideas in [CIK97] that also works over $\mathbb{Q}$ (this is important for some potential applications like mixtures of Gaussians).

Thus, we are capable of doing a decomposition like the one in Equation (4). But, why should we end up with the same decomposition? Certainly there are cases where the decompositions are not unique. For example, if $U = V$ and $\mathcal{L}$ just consists of the identity map, then any decomposition into one-dimensional spaces is a valid one. However, there is a characterization of all decompositions in the symmetric setting (Krull-Schmidt theorem, Theorem 3 in Section A) and it extends to vector space decomposition via our reduction (Corollary B.2). In many settings (including the one in this paper), this characterization helps in proving the uniqueness of decomposition.

Finally, the meta algorithm is stated in Algorithm 1, which works under the assumptions:

1. The following direct sum structure holds,

$$\begin{aligned} U &:= \langle \mathcal{L}_1 \circ f \rangle = \langle \mathcal{L}_1 \circ T_1 \rangle \oplus \ldots \oplus \langle \mathcal{L}_1 \circ T_s \rangle \quad \text{and} \\ V &:= \langle \mathcal{L}_2 \circ \mathcal{L}_1 \circ f \rangle = \langle \mathcal{L}_2 \circ (\mathcal{L}_1 \circ T_1) \rangle \oplus \ldots \oplus \langle \mathcal{L}_2 \circ (\mathcal{L}_1 \circ T_s) \rangle. \end{aligned} \tag{5}$$

2. (5) is the *unique* indecomposable decomposition for the vector spaces U and V w.r.t. $\mathcal{L}_2$.

3. One can recover T_i from $\langle \mathcal{L}_1 \circ T_i \rangle$ efficiently.

Next, we will discuss how we prove these assumptions for our setting, namely sums of powers of low degree polynomials.

Algorithm 1 Meta algorithm: Learning from lower bounds

```
Input: $f = T_1 + \cdots + T_s$.
Output: $T'_1, \ldots, T'_s$ such that there exists a permutation $\sigma : [s] \to [s]$ so that $T'_i = T_{\sigma(i)}$.

1: Take an appropriate set of linear maps $\mathcal{L} = \mathcal{L}_2 \circ \mathcal{L}_1$. Compute $U := \langle \mathcal{L}_1 \circ f \rangle$ and $V := \langle \mathcal{L}_2 \circ \mathcal{L}_1 \circ f \rangle$.
2: Obtain a (further indecomposable) vector space decomposition of $U$ and $V$ with respect to $\mathcal{L}_2$, namely $U = U_1 \oplus \cdots \oplus U_s$ and $V = V_1 \oplus \cdots \oplus V_s$.
3: Compute $T'_i$ from $U_i$ (assuming $U_i = \langle \mathcal{L}_1 \circ T'_i \rangle$).
```

1.2.3 Implementation for sums of powers of low degree polynomials

In this section, we discuss how we implement the meta algorithm described above for sums of powers of low degree polynomials. As discussed, the main ingredient in the learning algorithm

[20] The symmetric version is known as *module decomposition* in the literature.

is the lower bound. So, at first we need to understand how lower bounds are proven for this model [Kay12b, GKKS14, KSS14]. Let us first consider the setting of sums of powers of linear forms[21], i.e., $t = 1$ [Nis91, NW97]. Our goal is to find a degree-d homogeneous polynomial f such that any expression of the form:

$$f = \ell_1^d + \cdots + \ell_s^d,$$

with ℓ_i's linear, requires a large value of s. The set of linear maps here will be $\mathcal{L} = \partial_{\mathbf{x}}^{\lfloor d/2 \rfloor}$, i.e., all partial derivatives of order $\lfloor d/2 \rfloor$. Then, it is easy to see that $\dim(\langle \mathcal{L} \circ \ell^d \rangle) \leq 1$ for all linear polynomials ℓ. Thus, any f has a lower bound of $s \geq \dim(\langle \mathcal{L} \circ f \rangle)$. One can easily design polynomials with large dimension for the partial derivatives, e.g., the elementary symmetric polynomial of degree d in n variables satisfies $\dim(\langle \mathcal{L} \circ f \rangle) = \binom{n}{\lfloor d/2 \rfloor}$.[22]

For a long time, it was not known how to generalize these super-polynomial lower bounds even to the $t = 2$ case. What goes wrong is that $\dim(\langle \mathcal{L} \circ Q^m \rangle)$ is no longer small, for $\mathcal{L} = \partial_{\mathbf{x}}^k$, when Q is a degree-t homogeneous polynomial with $t \geq 2$. For example, $\dim(\langle \mathcal{L} \circ (x_1^2 + \cdots + x_n^2)^m \rangle) \geq \binom{n}{k}$ for $k \leq m \leq n$. However, one can still say something about the partial derivatives. If we take any $\alpha \in \mathbb{Z}_{\geq}^n$ with $|\alpha| := \sum_{i=1}^n \alpha_i = k$, then Q^{m-k} divides $\partial_{\mathbf{x}}^{\alpha} Q^m$ for $k \leq m$. Hence, any $\partial_{\mathbf{x}}^{\alpha} Q^m$ is of the form $Q^{m-k} R$, where R is homogeneous of degree $k(t-1)$. Now, the main observation in [Kay12b] was that we can make use of this special property of powers of low degree polynomials by using the *shifted* partial derivatives measure which is defined as follows,

$$\mathsf{SP}_{k,\ell}(f) := \dim \left\langle \mathbf{x}^{\ell} \cdot \partial_{\mathbf{x}}^{k} f \right\rangle.$$

That is, we take all k^{th} order partial derivatives of f and then multiply by all degree-ℓ monomials and then take the dimension of their span. Now, for any α, β such that $|\alpha| = k$ and $\beta = \ell$, $\mathbf{x}^{\beta} \partial_{\mathbf{x}}^{\alpha} Q^m$ is of the form $Q^{m-k} R$, where R is a homogeneous polynomial of degree $\ell + k(t-1)$. Hence

$$\mathsf{SP}_{k,\ell}(Q^m) \leq \binom{n + \ell + k(t-1) - 1}{\ell + k(t-1)},$$

where if k, ℓ are not too large, then one can expect $\mathsf{SP}_{k,\ell}(f)$, for an appropriately chosen f, to be close to the number of operators $\binom{n+\ell-1}{\ell}\binom{n+k-1}{k}$. Indeed, with appropriate choices of k and ℓ, this can be used to prove exponential lower bounds for the model sums of powers of low degree polynomials [Kay12b] and other more general models as well [GKKS14, KSS14, KS14, FLMS15, KLSS17, KS17b]. In all of these lower bounds, the value of ℓ is chosen to be comparable or larger than n. This makes the number of linear maps exponential and hence not suitable for designing an efficient algorithm. In fact, even ignoring the large number of linear maps, with such a large value of ℓ, the shifted partials measure is unlikely to satisfy the direct sum property in Equation (5) (see Section C) when the number of variables is large with respect to the degree. A natural way to decrease the number of linear maps is to project to a smaller number of variables. To our surprise, if we project down to a smaller number of variables, one does not need shifts at all to prove the lower bound! We call this new measure *affine projections* of partials, which we define next.[23]

[21] Also known in the literature as symmetric tensor decomposition or the Waring rank problem.

[22] If we use this lower bound with the recipe in Section 1.2.2, then one gets a close variant of Jennrich's algorithm for symmetric tensor decomposition.

[23] The word affine is added to avoid confusion with another kind of projection (namely multilinear projection) which is usually done in the literature to prove lower bounds for depth four arithmetic circuits.

Affine projections of partials – a novel adaptation of the partial derivatives measure. Let f be a polynomial in variables $\mathbf{x} = (x_1, \ldots, x_n)$ and $L = (\ell_1(\mathbf{z}), \ldots, \ell_n(\mathbf{z}))$ a tuple of linear forms[24] in variables $\mathbf{z} = (z_1, \ldots, z_{n_0})$. The parameter n_0 would be much smaller than n in this paper. Let π_L be the following affine projection map from $\mathbb{F}[\mathbf{x}]$ to $\mathbb{F}[\mathbf{z}]$,

$$\pi_L(f) := f(\ell_1(\mathbf{z}), \ldots, \ell_n(\mathbf{z})).$$

For a set $S \subseteq \mathbb{F}[\mathbf{x}]$, the projection π_L is naturally defined as,

$$\pi_L(S) := \{\pi_L(f) : f \in S\}.$$

Recall that $\partial_{\mathbf{x}}^k f$ is the set of all k-th order partial derivatives of f and $\langle S \rangle$ is the $\mathbb{F}$-linear span of a set of polynomials S. The *affine projections of partials* (APP) measure is defined as,

$$\text{(The measure)} \qquad \mathsf{APP}_{k,n_0}(f) := \max_L \dim \left\langle \pi_L(\partial_{\mathbf{x}}^k f) \right\rangle, \tag{6}$$

where the maximum is taken over all n-tuple $L = (\ell_1(\mathbf{z}), \ldots, \ell_n(\mathbf{z}))$ of linear forms in $\mathbb{F}[\mathbf{z}]$. It is easy to verify that for any $f, g \in \mathbb{F}[\mathbf{x}]$ the following linearity property is satisfied,

$$\text{(Linearity of the measure)} \qquad \mathsf{APP}_{k,n_0}(f + g) \leq \mathsf{APP}_{k,n_0}(f) + \mathsf{APP}_{k,n_0}(g). \tag{7}$$

The APP measure can be alternatively defined using a random affine projection π_L. The observation below is an easy consequence of the Schwartz-Zippel lemma [Sch80, Zip79].

Observation 1.1. *If $|\mathbb{F}| \geq 10 \cdot (d-k) \cdot \binom{n+k-1}{k}$ and every coefficient of the linear forms in $L = (\ell_1(\mathbf{z}), \ldots, \ell_n(\mathbf{z}))$ is chosen from a set $S \subseteq \mathbb{F}$ of size $10 \cdot (d-k) \cdot \binom{n+k-1}{k}$ then with probability at least 0.9,*

$$\mathsf{APP}_{k,n_0}(f) = \dim \left\langle \pi_L(\partial_{\mathbf{x}}^k f) \right\rangle.$$

Remark 1. Similarity with the skewed partials measure. *The* APP *measure is akin to the skewed partials (*SkP*) measure introduced in [KNS16] – SkP is a special case of* APP*. We show that the lower bound proof works with the* SkP *measure, but to ensure that the hard polynomial $f_{n,d}$ is multilinear we need affine projections (particularly, p-projections). However, the main reason for us to work with general/random affine projections (instead of* SkP *or p-projections) is to make the learning algorithm require as weak a non-degeneracy condition as possible.*

Let us give some intuition as to why the APP measure can yield lower bounds for sums of powers of low degree polynomials. As we say above, any $\partial_{\mathbf{x}}^{\alpha} Q^m$, with $|\alpha| = k$, is of the form $Q^{m-k}R$, where R is homogeneous of degree $k(t-1)$ (recall Q is homogeneous of degree t). This implies that $\mathsf{APP}(Q^m) \leq \binom{n_0+k(t-1)-1}{k(t-1)}$. However, for an appropriately chosen homogeneous degree-d polynomial f, one can expect that $\mathsf{APP}(f)$ could be as large as $\min\{\binom{n_0+d-k-1}{d-k}, \binom{n+k-1}{k}\}$. The first upper bound is from the fact that after derivatives and projection, we are in the space of degree-$(d-k)$ polynomials in n_0 variables and the second is from the fact that we have $\binom{n+k-1}{k}$ linear maps in APP. With appropriate choices of n_0, k and the polynomial f, we can prove the following lower bound result which comes close to the best known lower bounds via shifted partials.

[24]More generally, $\ell_1(\mathbf{z}), \ldots, \ell_n(\mathbf{z})$ are affine forms, but for this work it suffices to take them as linear forms.

Theorem 2 (Lower bound for homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}$ circuits using APP)**.** *The* APP *measure, defined above, can be used to prove the following lower bound for homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}$ circuits.*

- *High t case: Let $n, d, t \in \mathbb{N}$ such that $n \geq d^2$ and $\ln \frac{n}{d} \leq t \leq \frac{d}{4 \cdot e^{10} \cdot \ln d}$. There is a family of n-variate degree-d multilinear polynomials $\{f_{n,d}\}$ in* VNP *such that any homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}(s)$ circuit computing $f_{n,d}$ must have*
$$s = \left(\frac{n}{d}\right)^{\Omega\left(\frac{d}{t \ln t}\right)}.$$

- *Low t case: Let $n, d, t \in \mathbb{N}$ such that $n \geq d^{20}$ and $1 \leq t \leq \min\left\{\frac{\ln n}{6e \cdot \ln d}, d\right\}$. There is a family of n-variate degree-d multilinear polynomials $\{f_{n,d}\}$ in* VNP *such that any homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}(s)$ circuit computing $f_{n,d}$ must have*
$$s = n^{\Omega\left(\frac{d}{t}\right)}.$$

Remark 2. More general circuit model. *The above lower bound (proved in Section 4) is for the class of homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}$ circuits, which contains the class of homogeneous $\Sigma\wedge\Sigma\Pi^{[t]}$ circuits. In fact, the* APP *measure can be used to give a super-polynomial lower bound for general homogeneous depth four circuits – we skip the proof of this fact here.*

Of course, an $n^{\Omega\left(\frac{d}{t}\right)}$ lower bound is already known for homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}$ circuit using the shifted partials measure [KSS14, FLMS15]. We get nearly the same lower bound by replacing "shifts" by an affine projection. As discussed above, this change is essential to satisfy the direct sum property in Equation (5). In terms of lower bounds, this means that we show an explicit polynomial which can be computed by a homogeneous $\Sigma\wedge\Sigma\Pi^{[t]}(s)$ circuit but not by *any* homogeneous $\Sigma\wedge\Sigma\Pi^{[t]}(s-1)$ circuit.

Lemma 1.1 (Tight separation for homogeneous $\Sigma\wedge\Sigma\Pi^{[t]}$ circuits)**.** *Suppose $n, d, s, t, \mathbb{F}$ satisfy the conditions in Theorem 1. Then, there is a family of explicit*[25] *n-variate degree-d polynomials computable by homogeneous $\Sigma\wedge\Sigma\Pi^{[t]}(s)$ circuits but not by* any *homogeneous $\Sigma\wedge\Sigma\Pi^{[t]}(s-1)$ circuit.*

The above lemma (whose proof is implicit in Section 2.3) allows us to argue that a random $\Sigma\wedge\Sigma\Pi^{[t]}$ circuit is non-degenerate with high probability (Item 1 in the assumptions for Algorithm 1). Let us now briefly explain how we prove the uniqueness of decomposition and describe our algorithm for recovering a term from the corresponding vector space of polynomials (Items 2 and 3).

Uniqueness of vector space decomposition. The *adjoint algebra* helps us prove uniqueness of decomposition in our setting. Let us discuss what that is. Recall that we have vector spaces U, V and a set of linear maps $\mathcal{L}_2$ from U to V. The adjoint algebra of $\mathcal{L}_2$ is defined as follows:

$$\text{adj}(\mathcal{L}_2) := \{(D, E) \ : \ D : U \to U,\ E : V \to V \text{ and } K \circ D = E \circ K \text{ for all } K \in \mathcal{L}_2\}. \quad (8)$$

Suppose $U = U_1 \oplus \cdots \oplus U_s$, $V = V_1 \oplus \cdots \oplus V_s$ is an indecomposable decomposition with respect to $\mathcal{L}_2$. If it so happens that for every $(D, E) \in \text{adj}(\mathcal{L}_2)$, there exist constants $\alpha_1, \ldots, \alpha_s$ such that $Du_i = \alpha_i u_i$ for all $u_i \in U_i$, then the decomposition is unique (this follows from Corollary A.1 and B.2). We show this is the case in our setting. We deviate from the framework in Section

[25]That is, in time poly(n, s), we can output the $\Sigma\wedge\Sigma\Pi^{[t]}(s)$ circuit computing the polynomial.

1.2.2 a little bit to simplify the analysis. Due to this, the vector spaces in our case turn out to be $V = \langle \pi_L(Q_1)^{m-k}, \ldots, \pi_L(Q_s)^{m-k} \rangle$ and $W = W_1 \oplus \ldots \oplus W_s$, $W_i := \langle \pi_P(\partial^k_{\mathbf{z}}\ \pi_L(Q_i)^{m-k}) \rangle$, and $\mathcal{L}_2 = \pi_P(\partial^k_{\mathbf{z}}\ )$ are linear maps from V to W (changing notation to match with Section 2). Here, L is a random projection onto n_0 variables $\mathbf{z}$, and P is a random projection onto m_0 variables $\mathbf{w}$.

Recovery of the terms from the corresponding vector spaces. Due to the above-mentioned deviation from the framework, the final problem we have to solve is the following: Given access to the random projections $\pi_L(Q_1), \ldots, \pi_L(Q_s)$, recover the Q_i's. First of all, if one is given multiple projections $\pi_L(Q)$, then it is not hard to recover Q. However, we have multiple polynomials and for each random projection, we could be given the polynomials in an arbitrary order. This makes the recovery slightly non-trivial. For details of how we solve this, see the analysis of Steps 7-8 of Algorithm 2 in Section 2.2.

Next, we explicitly state the non-degeneracy conditions we require. The details of our algorithm and analysis can be found in Section 2. As mentioned, we deviate from the general recipe to simplify the analysis, but the recipe provides the intuition and forms the backbone of the algorithm.

1.2.4 Non-degeneracy conditions

In this section, we state the non-degeneracy conditions that our algorithm requires. Let C be a homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}(s)$ circuit computing an n-variate degree-d polynomial

$$f = c_1 Q_1^m + \ldots + c_s Q_s^m. \tag{9}$$

Notations. Let $\mathbf{z} = (z_1, \ldots, z_{n_0})$ be a set of $n_0 = \lfloor n^{\frac{1}{3 \cdot t}} \rfloor$ variables and $\mathbf{w} = (w_1, \ldots, w_{m_0})$ a set of $m_0 = \lfloor n^{\frac{1}{15 \cdot t^2}} \rfloor$ variables. Let $L = (\ell_1(\mathbf{z}), \ldots, \ell_n(\mathbf{z}))$ be a tuple of n linear forms in $\mathbb{F}[\mathbf{z}]$ and $P = (p_1(\mathbf{w}), \ldots, p_{n_0}(\mathbf{w}))$ a tuple of n_0 linear forms in $\mathbb{F}[\mathbf{w}]$. For every such tuples of linear forms L and P, we can define the following spaces and polynomials by setting $k = \left\lceil \frac{130 \cdot t \cdot \log s}{\log n} \right\rceil$:

- $U := \langle \pi_L(\partial^k_{\mathbf{x}}\ f) \rangle$ and $U_i := \langle \pi_L(\partial^k_{\mathbf{x}}\ Q_i^m) \rangle$,
- $G_i := \pi_L(Q_i)$ and $g_0(\mathbf{z}) := G_1^e + \ldots + G_s^e$, where $e = m - k$,
- $\widetilde{U_i} := \left\langle \mathbf{z}^{2k(t-1)} \cdot G_i^e \right\rangle$, where $\mathbf{z}^{2k(t-1)}$ is the set of all $\mathbf{z}$-monomials of degree $2k(t-1)$,
- $W := \langle \pi_P(\partial^k_{\mathbf{z}}\ g_0) \rangle$ and $W_i := \langle \pi_P(\partial^k_{\mathbf{z}}\ G_i^e) \rangle$.

Observe that

$$U \subseteq U_1 + \ldots + U_s, \quad \text{and} \quad U_i \subseteq \left\langle \mathbf{z}^{k(t-1)} \cdot \pi_L(Q_i)^{m-k} \right\rangle \quad \text{implying} \quad \dim U_i \le \binom{n_0 + k(t-1) - 1}{k(t-1)},$$

$$W \subseteq W_1 + \ldots + W_s, \quad \text{and} \quad W_i \subseteq \left\langle \mathbf{w}^{k(t-1)} \cdot \pi_P(G_i)^{e-k} \right\rangle \quad \text{implying} \quad \dim W_i \le \binom{m_0 + k(t-1) - 1}{k(t-1)}.$$

Definition 1.1 (Non-degeneracy)**.** The circuit C, given by Equation (9), is *non-degenerate* if there exist L and P (as above) such that the following conditions are satisfied:

1. $U = U_1 \oplus \ldots \oplus U_s$ and $\dim U_i = \binom{n_0+k(t-1)-1}{k(t-1)}$ for all $i \in [s]$,
2. $W = W_1 \oplus \ldots \oplus W_s$ and $\dim W_i = \binom{m_0+k(t-1)-1}{k(t-1)}$ for all $i \in [s]$,
3. $\widetilde{U_1} + \ldots + \widetilde{U_s} = \widetilde{U_1} \oplus \ldots \oplus \widetilde{U_s}$ for all $i \in [s]$,
4. $[G_1^e]_{z_1=0}, \ldots, [G_s^e]_{z_1=0}$ are $\mathbb{F}$-linearly independent.

Condition 1 and 2 constitute the main part of non-degeneracy. Condition 3 and 4 have been added to aid our analysis and keep it relatively simple; it may be possible to dispense with these conditions completely perhaps by altering Conditions 1 and 2 slightly.

We prove the following lemma in Section 2.3, which says that if we choose the Q_i's randomly in Equation (9), then the circuit is non-degenerate with high probability.

Lemma 1.2 (Random $\Sigma \wedge \Sigma\Pi^{[t]}$ circuits are non-degenerate). *Suppose the coefficients of Q_i's are chosen uniformly and independently at random from a set $S \subset \mathbb{F}$ of size $|S| \geq (ns)^{150 \cdot t}$. Then, with probability $1 - o(1)$, the non-degeneracy conditions in Definition 1.1 are satisfied.*

1.3 Moment problem for mixtures of Gaussians

In this section, we discuss an application to learning parameters of mixtures of Gaussians from the moments. Symmetric tensor decomposition (and also general tensor decomposition) has a lot of applications in both supervised and unsupervised machine learning, e.g., in independent component analysis, learning latent variable models, hidden Markov models, topic models and mixture of Gaussians [MR05, CJ10, HKZ12, AHK12, HK13, AGH+14, ABG+14, BCMV14]. In our language, (symmetric) tensor decomposition corresponds to the $t = 1$ case of "sums of powers of degree-t polynomials". It is conceivable that learning algorithms which handle more general circuit classes as compared to tensor decomposition will handle a richer class of learning models than mentioned above. One example is given by the mixture of Gaussians model which has a rich history, e.g., see [Pea94, RR95, Das99, PFJ03, VW04, BS10, MV10, ABG+14, BCMV14, GHK15, RV17]. While special cases of this problem can be solved by reduction to (symmetric) tensor decomposition [HK13, ABG+14, BCMV14], general cases correspond to "sums of powers of quadratics" (and slightly more general models), i.e., the $t = 2$ case. This observation is implicit in [GHK15].

Let us briefly describe the mixture of Gaussians problem. We are given samples from a mixture of Gaussians in n dimensions, $\mathcal{D} = \sum_{i=1}^{s} w_i \mathcal{N}(\mu_i, \Sigma_i)$, where w_i, μ_i, Σ_i denote the weight, mean and covariance matrix of the i^{th} Gaussian respectively. The goal is to recover the parameters $(w_i, \mu_i, \Sigma_i)_{i=1}^{s}$ upto some error. There have been many algorithms developed for the mixture of Gaussians problem and they make varying assumptions about the input parameters. These assumptions can be grouped into three broad categories:

1. **Worst case**, e.g., [Pea94, BS10, MV10]. Here, no assumptions are made on the input parameters. Due to this, the running time is exponential in the number of components s. In the worst case, even information theoretically, one needs exponential (in s) number of samples to learn the parameters [MV10, ABG+14].

2. **Separation assumptions**, e.g., [Das99, SK01, AM05, KSV05, DS07, BV08, KK10, RV17]. Here, one assumes that the parameters (either the means or the covariance matrices) are well separated. Running times are typically polynomial in the number of components s.

3. **Smoothed setting**, e.g., [HK13, ABG$^+$14, BCMV14, GHK15]. Here, one does smoothed analysis for the problem, i.e., one starts with worst case parameters and perturbs them with some noise, and the goal is to solve the resulting instance. Somewhat surprisingly, the problem becomes easier as the dimension becomes larger.[26] Running times are typically polynomial in the dimension n, as long as the number of components $s \leq \text{poly}(n)$.[27]

Of course, it is best to have algorithms in the worst case but this usually means running time exponential time in the number of components.[28] Making reasonable assumptions on the parameters helps in designing more efficient algorithms (when the number of components are growing). The two assumptions here, separation and the smoothed setting, are incomparable. For example, the algorithms developed in the smoothed setting can also handle instances where the parameters are not well separated. Our contribution towards the mixture of Gaussians problem should be understood in the context of smoothed analysis of the problem. We also remark that there has been a lot of work in designing algorithms for mixtures of Gaussians in the robust setting, i.e., when some of the samples are corrupted adversarially [DKK$^+$19, KS17a, KSS18, HL18]. These algorithms usually work by estimating the moments from the corrupted data, so are in some sense complementary to algorithms that recover the parameters from the moments.

In the smoothed setting, the papers [ABG$^+$14, BCMV14] give a polynomial time algorithm for the special case of mixtures of Gaussians with diagonal covariance matrices (i.e., the different dimensions of every component of the mixture are independent) when the number of components $s \leq \text{poly}(n)$. For Gaussians mixtures with general covariance matrices, [GHK15] gave a polynomial time algorithm when $s \leq \sqrt{n}$. We make a step towards getting a polynomial time algorithm for Gaussians mixtures with general covariance matrices when $s \leq \text{poly}(n)$. Our algorithm for learning sums of powers of low degree polynomials (specifically sums of powers of quadratics) allows us to recover the parameters of a *non-degenerate* mixture of *zero-mean* Gaussians given access to its $O(1)$-order *exact* moments when $s \leq \text{poly}(n)$.

Lemma 1.3 (Learning mixtures of zero-mean Gaussians). *There is a randomized polynomial time algorithm which when given access to exact $O(1)$-order moments of a mixture of non-degenerate*[29] *zero-mean Gaussians, $\mathcal{D} = \sum_{i=1}^{s} w_i \mathcal{N}(\mu_i, \Sigma_i)$, recovers the parameters $(w_i, \Sigma_i)_{i=1}^{s}$ with probability $1 - o(1)$.*

We believe that a few modifications to our algorithm will allow one to get a polynomial time algorithm for general mixtures of Gaussians in the smoothed case. Let us remark on the main differences between our current algorithmic guarantee and the above goal.

[26]This is because we can get more information from lower order moments in the higher dimensional case, which are easier to estimate.

[27]The exponent of the polynomial running time will depend on the exponent of the polynomial controlling the relation between s and n.

[28]This should be compared with some of the worst case reconstruction algorithms which run in time exponential in the top fan-in of the circuit.

[29]The non-degeneracy condition is satisfied if the entries of Σ_i's are choosen uniformly and independently, conditioned on Σ_i's being symmetric, from a set $S \subset \mathbb{Q}$ with $|S| \geq \text{poly}(n, s)$, and of course all w_i's are non-zero.

- **Extension to general means.** Our current statement only holds for zero-mean Gaussians because applying the method of moments to zero-mean mixtures naturally leads to sums of powers of quadratics. This is just to keep the analysis simple and clean. We believe our algorithm should extend to sums of *products* of low degree polynomials and the learning problems coming from moments of mixtures of general mean Gaussians lie somewhere between sums of powers of quadratics and sums of products of quadratics.

- **Exact vs inexact moments.** Our algorithm assumes that the $O(1)$-order moments of the mixture are given exactly. Using samples, we can only approximate the moments ($O(1)$-order moments can be approximated to $1/\text{poly}(n)$ accuracy using $\text{poly}(n)$ samples). We leave it open for future work to modify our algorithm to handle $1/\text{poly}(n)$ error.

- **Smoothed vs non-degenerate setting.** Note that we state our result in the non-degenerate setting. This seems to be the right kind of assumption when given access to exact moments. If one is given access to inexact moments, one will need to control appropriate condition numbers whence smoothed setting is the right assumption to make.

We hope that our techniques will lead to progress on the smoothed analysis of mixtures of general Gaussians and also influence algorithms which work under other kinds of assumptions.

1.4 Hardness of learning circuits in the worst case

In this section, we review some of the hardness results on learning circuits in order to gauge the difficulty of the problem for our circuit model and to place our result in context. Hardness of learning has been intensely studied for Boolean circuits as compared to arithmetic circuits. We state a few of these results from the Boolean world with the intent of drawing analogy.

MCSP and approximate MCSP. Circuit reconstruction is the arithmetic analogue of exact learning [Ang87] Boolean circuits from membership queries. Exact learning is closely related to the minimum circuit size problem (MCSP). MCSP for Boolean circuits is the following: Given the $N = 2^n$ size truth-table of an n-variate Boolean function f and a number s, check if f is computable by a Boolean circuit of size at most s. Analogously, in case of MCSP for arithmetic circuits, we are given the $N = \binom{n+d}{d}$ coefficient vector of an n-variate degree-d polynomial f and are required to determine if there is an arithmetic circuit of size at most s computing f. MCSP for Boolean circuits is not in P assuming the existence of cryptographically secure one-way functions [KC00]. In fact, $N^{1-o(1)}$-approximate[30] MCSP for Boolean circuits is not in P under the same assumption [AH17].[31] Analogous results about MCSP for arithmetic circuits are not known. However, drawing analogy with the Boolean world, it is plausible that $N^{1-\delta}$-approximate MCSP for general arithmetic circuits is not in P, for every constant $\delta > 0$. Here, $N = \binom{n+d}{d}$ is the size of the input coefficient vector. Such a hardness result may even be true for $d = n^{O(1)}$.

MCSP for a circuit class $\mathcal{C}$ is defined like MCSP except that we are now interested in checking if the input f has a $\mathcal{C}$-circuit of size at most s. It is known that the arithmetic analogue of MCSP is NP-

[30] Meaning multiplicative factor approximation of the minimum circuit size

[31] However, proving MCSP is NP-hard is quite demanding as that would imply $EXP \neq ZPP$ [MW17], which is a long-standing open problem.

hard for set-multilinear depth three circuits (tensors) [Hås90] and for depth three powering circuits (symmetric tensors) [Shi16]. It is also known that there is a constant $\delta \approx 0.0005$ such that $(1+\delta)$-approximate MCSP is NP-hard for set-multilinear depth three circuits [Swe18, BIJL18, SWZ19]. Similar hardness results are known about MCSP for restricted Boolean circuit classes, e.g., MCSP for DNF is NP-hard [Mas79, Czo99]. In fact, there is a $\delta > 0$ such that $(\log N)^{\delta}$-approximate MCSP is NP-hard for DNF [Fel09][32]. Approximate MCSP is a difficult problem even for AC^0 circuits: For every $\delta, \epsilon > 0$ there is a h such that $N^{1-\delta}$-approximate MCSP for depth-h circuits is not in BPP unless m-bit Blum integer[33] factorization is in $\mathrm{BPTIME}(2^{m^{\epsilon}})$ [AHM+08][34].

Learning implies lower bounds. It was shown in [FK09] that a randomized polynomial-time (worst case, improper) reconstruction algorithm for an arithmetic circuit class $\mathcal{C}$ implies the existence of a polynomial that can be computed on Boolean inputs in $BPEXP$ and cannot be computed by circuits in $\mathcal{C}$ of polynomial size[35]. Similar results hold for Boolean circuits. Thus, if we are aiming for *worst-case* polynomial-time reconstruction then we must necessarily focus on classes for which super-polynomial lower bounds are known. In fact, to our knowledge, all efficient reconstruction algorithms, in the worst or the average case, that are known till date are either for models for which non-trivial lower bounds are known (or for models that are incomplete).

Hardness of PAC learning depth three arithmetic circuits. [KS09b] showed that PAC-learning depth three arithmetic circuit cannot be done in polynomial time unless the length of a shortest nonzero vector of an n-dimensional lattice can be approximated to within a factor of $\tilde{O}(n^{1.5})$ in polynomial time by a quantum algorithm. What this means is that it is hard to PAC learn the class of Boolean functions which match the output of polynomial-sized depth three arithmetic circuits on the Boolean hypercube.

Membership queries versus random samples. Membership queries provide an interactive model of learning as compared to learning from random samples. For some circuit classes, we know of efficient learning from membership queries but not learning from random samples. For instance, there is a deterministic polynomial-time algorithm for interpolating sparse polynomials using membership queries [KS01], but the same is not known using random samples. The best known complexity for learning an s-sparse n-variate degree-d real polynomial with error ϵ from random samples from the real cube $[-1,1]^n$ is $\mathrm{poly}(n, s, 2^d, \frac{1}{\epsilon})$ [APVZ14]. If the random samples are restricted to the Boolean hypercube $\{-1,1\}^n$ then exact learning of s-sparse real polynomials can be done in $\mathrm{poly}(n, 2^s)$ time provided the input polynomial satisfies a certain property[36] [KSDK14]. Another example, in the Boolean world, is the quasi-polynomial time algorithm for PAC learning $AC^0[p]$ circuits under the uniform distribution from membership queries [CIKK16]. It is not known if the same can be achieved without membership queries (like in the case of AC^0-

[32]In contrast, a greedy algorithm solves $(\log N)$-approximate MCSP for DNF in $\mathrm{poly}(N)$ time [Joh74, Lov75, Chv79]. If the input is a DNF instead of a truth-table then we know of the following result: $s^{\frac{1}{4}-\epsilon}$ factor approximation of the minimum DNF size of an input DNF of size s is not in P, for every $\epsilon \in (0,1)$, assuming $\Sigma_2^p \nsubseteq \mathrm{DTIME}(n^{O(\log n)})$ [Uma99].

[33]An integer of the form pq for primes p and q, where $p = q = 3 \mod 4$.

[34]Similar results have also been shown for NC^1 and TC^0 circuits [AKRR03].

[35]A deterministic analogue of this result was shown in [Vol16].

[36]This property is satisfied with high probability if the coefficients of the input polynomial are perturbed slightly by a random noise. Removing this condition on the input polynomial would immediately improve the state-of-the-arts of learning $\omega(1)$-juntas.

learning [LMN93]).

Difficulty of learning homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}$ circuits. It turns out though that learning $\Sigma \wedge \Sigma\Pi^{[t]}$ circuits *in the worst-case* is quite challenging. The reason is, if we can learn homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}$ circuits efficiently then we can solve approximate MCSP for polynomial-size ABP[37] efficiently. This can be argued as follows: Suppose f is a homogeneous n-variate polynomial of degree $d = n^{O(1)} < n$ that is computable by an ABP of size $\text{poly}(n)$. Let $t \ll d$ be a number dividing d. Then, f can be computed by a homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}$ circuit[38] of size $n^{O(\frac{d}{t})}$. Now suppose we are able to learn homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}$ circuits of size σ in time $\text{poly}(\sigma^t)$, for $t = \omega(1)$, and output $\text{poly}(\sigma)$-size circuits. Then, we would succeed in solving $N^{o(1)}$-approximate MCSP for $\text{poly}(n)$-size ABP in $\text{poly}(N)$-time, where $N = \binom{n+d}{d}$. This appears to be a difficult task with our current knowledge on MCSP[39]. If such an efficient approximate MCSP for ABP is unattainable then there is no hope of learning homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}$ circuits in $\text{poly}(\sigma^t)$ time in the worst-case (where the output is a $\text{poly}(\sigma)$-size circuit), for $t = \omega(1)$.

1.5 Related work

In view of the fact that learning general arithmetic circuits is probably a hard problem, research has focused on learning interesting special classes of circuits. Here, we give a brief account of some of these results from the literature.

Low depth circuits. A deterministic polynomial-time learning algorithm for $\Sigma\Pi$ circuits or sparse polynomials was given in [KS01]. Learning $\Pi\Sigma\Pi$ circuits in randomized polynomial-time follows from the classical circuit-factorization algorithm of [KT90]. General $\Sigma\Pi\Sigma$ circuits are much harder to learn: It follows from a depth reduction result [AV08, Koi12, GKKS16, Tav13] that polynomial-time learning for $\Sigma\Pi\Sigma$ circuits implies sub-exponential time learning for general circuits. In [Shp09], a randomized quasi-$\text{poly}(n, d, |\mathbb{F}|)$-time proper[40] learning algorithm was given for $\Sigma\Pi\Sigma$ circuits with two product gates over finite fields; if the circuit is additionally multilinear then the running time is $\text{poly}(n, |\mathbb{F}|)$. The algorithm was derandomized and generalized in [KS09a] to handle $\Sigma\Pi\Sigma$ circuits with constant number of product gates. Over fields of characteristic zero, [Sin16] gave a randomized proper[40] learning algorithm for $\Sigma\Pi\Sigma$ circuits with two product gates. A randomized polynomial-time proper learning algorithm is known for multilinear $\Sigma\Pi\Sigma\Pi$ circuits with top fan-in two over any field [GKL12]. Recently, a deterministic proper learning algorithm is given in [BSV19] for multilinear $\Sigma\Pi\Sigma\Pi$ circuits with constant top fan-in over finite fields; the running time is quasi-polynomial in the size of the circuit and $|\mathbb{F}|$.

Read-once formulas and ABPs. A deterministic polynomial-time proper learning algorithm is known for read-once formulas [MV18, SV14]. Read-once oblivious algebraic branching programs (ROABPs) form an important subclass of ABPs that captures several other interesting and well-studied circuit models. There is a randomized polynomial-time proper learning algorithm for

[37] Algebraic branching programs (ABP) form a powerful circuit class – a circuit can be converted to an ABP with only a quasi-polynomial blow up in size.

[38] This circuit is obtained by homogenizing the ABP and then dividing it into pieces of length t and multiplying out each piece. This gives a $\Sigma\Pi\Sigma\Pi^{[t]}$ circuit which is converted to a $\Sigma \wedge \Sigma\Pi^{[t]}$ circuit using Fischer's formula [Fis94].

[39] See the discussion on MCSP at the start of this section.

[40] Provided the circuit satisfies a certain rank condition

ROABP [KS06,BBB$^+$00] which was derandomized in quasi-polynomial time in [FS13]. The method used for ROABP reconstruction can be adapted to give learning algorithms for set-multilinear ABPs and non-commutative ABPs [FS13].

Reconstruction under non-degeneracy conditions. Reconstruction in the worst case appears to be an extremely hard problem even for circuit models for which good lower bounds are known. It is natural to ask – Can we use the techniques used for proving lower bound for a circuit class $\mathcal{C}$ to learn *almost all* $\mathcal{C}$-circuits? The notion of "almost all $\mathcal{C}$-circuits" is formalized as random $\mathcal{C}$-circuits under some natural distribution, or preferably, as $\mathcal{C}$-circuits satisfying a set of clearly stated non-degenerate conditions such that a random $\mathcal{C}$-circuit (under any natural distribution) is non-degenerate with high probability. In [GKL11], a randomized polynomial-time proper learning algorithm was given for non-degenerate[41] multilinear formulas having fan-in two. A randomized polynomial-time proper learning algorithm for non-degenerate regular formulas having fan-in two was given in [GKQ14]. An efficient randomized reconstrution for non-degenerate homogeneous ABPs of width at most $\frac{\sqrt{n}}{2}$ is presented in [KNS19]. All the above reconstruction algorithms are implicitly connected to the corresponding lower bounds: a quasi-polynomial lower bound for multilinear formulas was already shown in [Raz09], a quasi-polynomial lower bound for regular formulas was proven in [KSS14][42] and a width lower bound of n is also known for homogeneous ABPs [Kum19]. Recently, [KS19] gave a randomized polynomial-time proper learning algorithm for non-degenerate homogeneous depth three circuits depending very explicitly on the ideas used in proving an exponential lower bound for this model [NW97]. Also, randomized polynomial-time proper learning algorithms for non-degenerate depth three powering circuits are given in [KS19, GKP18, Kay12a] which have implicit connections to the corresponding lower bound methods.

Tensor decomposition. Tensor decomposition (which is the same as reconstruction of depth three set-multilinear circuits) has garnered a lot of attention in the machine learning community and a lot of algorithms have been developed for it [Har70,LRA93,DLCC07,AGH$^+$14,ABG$^+$14,BCMV14, BKS15, GM15, HSSS16, MSS16]. Comparing to our model, symmetric tensor decomposition corresponds to learning sums of powers of linear forms. We do not know of any work that designs algorithms for sums of powers of degree-t polynomials for $t > 1$, except for the work of [GHK15]. An algorithm for learning sums of cubes of quadratics in the non-degenerate case, with the number of summands upper bounded by $\sqrt{n}$ is implicit in [GHK15]. Their approach is to reduce the problem to tensor decomposition. However, we believe such an approach cannot be made to handle larger number of summands (say $\text{poly}(n)$) even in the quadratic case as the lower bounds for sums of powers of quadratics need substantially newer ideas than the linear case as discussed in Section 1.2.3.

Lower bounds and PIT for $\Sigma \wedge \Sigma\Pi^{[t]}$ circuits. Homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}$ circuits have been well-studied in the context of lower bound and polynomial identity testing (PIT). Understanding how to prove lower bound for this model played a vital role in the proof of the exponential lower bound

[41]The papers [GKL11,GKQ14] state the results for random formulas, but it is not difficult to state the non-degeneracy conditions by taking a closer look at the algorithms.

[42]Note that here the lower bound comes later than the average case reconstruction algorithm; in fact, the ideas arising out of the reconstruction algorithm were helpful in proving the lower bound.

for homogeneous depth four circuits that emerged out of a chain of work [Kay12b,GKKS14,KSS14, FLMS15,KLSS17,KS17b]. Also, a deterministic $s^{O(t \log s)}$-time black-box PIT algorithm is known for homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}(s)$ circuits [For15b].

Improper learning for sums of powers of low degree polynomials. In this paper, we focus on proper learning (i.e., the input and output representations are from the same circuit class). However, to our knowledge, there is no known efficient learning algorithm for sums of powers of degree-t polynomials (worst case or average case) even for $t = 2$, and even in the improper setting. For the $t = 1$ case, a polynomial-time improper learning algorithm (worst case) follows from ROABP reconstruction [BBB+00, KS06] as sums of powers of linear forms (depth three powering circuits) is a subclass of ROABP. But, reconstruction for ROABP does not give a learning algorithm for sums of powers of quadratics as there is a power of a quadratic that requires an exponential-size ROABP [For15b].

To summarize, we have efficient learning algorithms (even under non-degeneracy conditions) only for some models for which good lower bounds are known (or for models that are incomplete). Moreover, barring a few exceptions like ROABP, read-once formulas and sparse polynomials, the circuit models for which efficient learning is known do have the fan-in of the sum gates very small (mostly bounded by a constant). In comparison, our strategy for translating techniques from lower bounds to learning works for a much larger additive fan-in. Such a translation is only possible for learning under non-degeneracy condition as worst-case learning is arguably much harder[43] than proving lower bound.

1.6 Roadmap of the paper

In Section 2, we state our algorithm for learning sums of powers of low degree polynomials and its analysis, and also prove that the non-degeneracy conditions are satisfied in the random case. In Section 3, we provide an algorithm for recovering the parameters of a non-degenerate mixture of zero-mean Gaussians given access to its exact $O(1)$-order moments. In Section 4, we show how our new lower bound measure APP can be used to give alternate proofs of the lower bounds for homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}$ circuits (almost matching the best known lower bounds which use the shifted partials measure). Section 5 contains some of the interesting open problems and directions for future work.

In the Appendix, Section A mentions some important facts about the adjoint algebra and a proof of the uniqueness of decomposition in our setting. In Section B, we show a reduction from the vector space decomposition[44] problem to the module decomposition problem. Section C contains a discussion about why the shifted partials measure is unlikely to satisfy the non-degeneracy conditions required for our learning framework. Finally, Sections D and E supply the missing proofs from Sections 2 and 4, respectively.

[43]See the discussion on the difficulty of learning homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}$ circuits in the worst case in Section 1.4.

[44]In fact a generalization of it.

2 Learning sums of powers of low degree polynomials

We prove Theorem 1 in this section. Our algorithm is an implementation of the lower bound to learning strategy (proposed in Section 1.2) for homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}$ circuits[45]. Section 2.1 describes the algorithm. Section 2.2 describes the analysis of the algorithm. In Section 2.3, we prove that a random $\Sigma \wedge \Sigma\Pi^{[t]}$ circuit satisfies our non-degeneracy conditions. Finally Section 2.4 lists some relations between various parameters which are needed for the analysis to work.

For simplicity of presentation, we will assume that $\mathbb{F}$ is a *finite field* of sufficiently large size and characteristic. The analysis goes through over any $\mathbb{F}$ that satisfies the restrictions on size and characteristic stated in Theorem 1 – we simply have to work with a sufficiently large subset of $\mathbb{F}$ and do a few minor changes to the algorithm and its analysis.

2.1 The algorithm

We are given black-box access to an n-variate degree-d polynomial f that is computed by a homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}(s)$ formula C, i.e.,

$$f(\mathbf{x}) = c_1 Q_1^m + \ldots + c_s Q_s^m, \tag{10}$$

where each $c_i \in \mathbb{F}^\times$, Q_i is a homogeneous polynomial of degree t, and $tm = d$. Moreover, formula C is non-degenerate (see Definition 1.1). Assume that the algorithm knows[46] d, t and s and these parameters satisfy the conditions stated in Theorem 1. The task is to output a homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}(s)$ formula for f efficiently. The parameters k, n_0 and m_0, in Algorithm 2, are chosen according to Proposition 2.6, which is stated in Section 2.4. A *random linear form* is a linear form whose coefficients are chosen independently and uniformly at random from $\mathbb{F}$. A *tuple of n random linear forms* is a tuple of n independently chosen random linear forms.

2.2 Analysis of the algorithm

We analyze the correctness and efficiency of the algorithm in this section. The three main segments of the algorithm are Steps 1-5, Step 6 and Steps 7-8 – we examine these one by one. The missing proofs of the technical statements are given in Section D of the appendix.

Steps 1-5: Constructing two sets of linear operators and obtaining the relevant vector spaces

Let σ be the size of the non-degenerate formula C that computes f. In Step 2, the algorithm computes black-box access to a basis of $U = \langle \pi_L(\partial_{\mathbf{x}}^k f) \rangle$, where L is a tuple of n random linear forms in n_0 many $\mathbf{z}$-variables. This can be done in poly(σ, s^t) time, with success probability $1 - o(1)$, due to the choice of k (see Proposition 2.6) and the following easily verifiable fact.

[45] We deviate from the strategy slightly, by introducing an intermediate *multi-gcd* step (see Algorithm 2), in order to make the analysis simpler.

[46] If s is unknown, we can simply go over s incrementally (starting from 1 and going up to the upper bound stated in Theorem 1) and run the algorithm for each s. A randomized identity test at the end of the algorithm determines if we have learnt the circuit correctly with high probability.

[47] this will be explained in the analysis of this step

Algorithm 2 Learning sums of powers of degree-t polynomials

```
**Input**: Black-box access to an $f \in \mathbb{F}[\mathbf{x}]$ that is computed by a non-degenerate homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}(s)$ formula $C$ (as in Equation (10)), i.e., $f = c_1 Q_1^m + \ldots + c_s Q_s^m$.
**Output**: A non-degenerate homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}(s)$ formula computing $f$.

            /* Constructing two sets of linear operators and obtaining the relevant vector spaces */
1: Pick a tuple of $n$ random linear forms $L = (\ell_1(\mathbf{z}), \ldots, \ell_n(\mathbf{z}))$, where $|\mathbf{z}| = n_0$. Let $\mathcal{L}_1$ be the set of operators $\pi_L(\partial_{\mathbf{x}}^k)$.
2: Compute black-box access to a basis of $U = \langle \mathcal{L}_1 \circ f \rangle = \left\langle \pi_L(\partial_{\mathbf{x}}^k f) \right\rangle$.
3: (*Multi-gcd step*) Compute black-box access to a basis of $V = \left\langle \pi_L(Q_1)^{m-k}, \ldots, \pi_L(Q_s)^{m-k} \right\rangle$ using the basis of $U$. It will follow from Observation 2.2 that
       $$V = V_1 \oplus \ldots \oplus V_s, \qquad \text{(with high probability)}$$
   where $V_i = \left\langle G_i^e \right\rangle$, $G_i = \pi_L(Q_i)$ and $e = m - k$. Get black-box access to a random $g(\mathbf{z})$ in $V$.
4: Pick a tuple of $n_0$ random linear forms $P = (p_1(\mathbf{w}), \ldots, p_{n_0}(\mathbf{w}))$, where $|\mathbf{w}| = m_0$. Let $\mathcal{L}_2$ be the set of operators $\pi_P(\partial_{\mathbf{z}}^k)$.
5: Compute black-box access to a basis of $W = \left\langle \pi_P(\partial_{\mathbf{z}}^k g) \right\rangle$. It will follow from Proposition 2.2,
       $$W = W_1 \oplus \ldots \oplus W_s, \qquad \text{(with high probability)}$$
   where $W_i := \left\langle \pi_P(\partial_{\mathbf{z}}^k G_i^e) \right\rangle = \langle \mathcal{L}_2 \circ V_i \rangle$.

            /* Decomposing the vector spaces */
6: Compute black-box access to bases of $V_1, \ldots, V_s$ by decomposing $V$ and $W$ under the action of $\mathcal{L}_2$ into indecomposable subspaces.

            /* Recovering the terms of the formula */
7: Run Steps 1-6 "$d$ times"[47] to compute black-box access to $c_1' Q_1(\mathbf{x})^e, \ldots, c_s' Q_s(\mathbf{x})^e$ for some constants $c_1', \ldots, c_s' \in \mathbb{F}^\times$.
8: Compute (dense representations of) the polynomials $\hat{c}_1 Q_1(\mathbf{x}), \ldots, \hat{c}_s Q_s(\mathbf{x})$ for some constants $\hat{c}_1, \ldots, \hat{c}_s \in \mathbb{F}^\times$. Output a homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}(s)$ formula for $f$.
```

Fact 1. *Given black-box access to an n-variate degree-d polynomial $f(\mathbf{x})$, black-box access to the polynomials in $\partial_{\mathbf{x}}^k f$ can be computed in determimistic $\mathrm{poly}((nd)^k)$ time. Given black-box access to n-variate degree-d polynomials $g_1, \ldots, g_r$, black-box access to a basis of $\langle g_1, \ldots, g_r \rangle$ can be computed in randomized $\mathrm{poly}(n, d, r)$ time with probability at least $1 - \frac{rd}{|\mathbb{F}|}$.*

Observation 2.1. *Let $U_i = \langle \pi_L(\partial_{\mathbf{x}}^k Q_i^m) \rangle$. With probability $1 - o(1)$ over the randomness of L,*

$$U = U_1 \oplus \ldots \oplus U_s \quad \textit{and} \quad \dim U_i = \binom{n_0 + k(t-1) - 1}{k(t-1)} \textit{ for all } i \in [s].$$

It follows from the above observation that $U_i = \left\langle \mathbf{z}^{k(t-1)} \cdot \pi_L(Q_i)^{m-k} \right\rangle$.

The multi-gcd step. In Step 3, the algorithm computes a basis of $V = \langle G_1^e, \ldots, G_s^e \rangle$, where $G_i = \pi_L(Q_i)$ and $e = m - k$. This step can be executed in $\mathrm{poly}(\sigma, s^t)$ time as follows:

Observation 2.2. *Let $\widetilde{U_i} := \left\langle \mathbf{z}^{2k(t-1)} \cdot G_i^e \right\rangle$. With probability $1 - o(1)$ over the randomness of L,*

$$\widetilde{U_1} + \ldots + \widetilde{U_s} = \widetilde{U_1} \oplus \ldots \oplus \widetilde{U_s}.$$

It follows from the above that $V = V_1 \oplus \ldots \oplus V_s$, where $V_i = \left\langle G_i^e \right\rangle$. The observation also helps prove the next proposition which gives a way to compute a basis of V efficiently.

Proposition 2.1. *Let $r = \binom{n_0+k(t-1)-1}{k(t-1)}$ and $f_1, \ldots, f_{sr}$ be a basis of U. Let z_1 and z_2 be two distinct variables in $\mathbf{z}$. Then, the following statements hold:*

1. *If $g(\mathbf{z}) \in V$ then there exist $a_1, \ldots a_{sr}, b_1, \ldots, b_{sr} \in \mathbb{F}$ such that*

$$\frac{a_1 f_1 + \ldots + a_{sr} f_{sr}}{z_1^{k(t-1)}} = \frac{b_1 f_1 + \ldots + b_{sr} f_{sr}}{z_2^{k(t-1)}} = g.$$

2. *If there exist $a_1, \ldots a_{sr}, b_1, \ldots, b_{sr} \in \mathbb{F}$ such that*

$$\frac{a_1 f_1 + \ldots + a_{sr} f_{sr}}{z_1^{k(t-1)}} = \frac{b_1 f_1 + \ldots + b_{sr} f_{sr}}{z_2^{k(t-1)}},$$

then $\frac{a_1 f_1 + \ldots + a_{sr} f_{sr}}{z_1^{k(t-1)}}$ is a polynomial $g(\mathbf{z})$ in V.

Let $f_1, \ldots, f_{sr}$ be the basis of U obtained in Step 2. It follows from the above proposition that a basis of the space $\mathcal{S}$ defined by all $(a_1, \ldots a_{sr}, b_1, \ldots, b_{sr}) \in \mathbb{F}^{2sr}$ satisfying

$$a_1 \cdot z_2^{k(t-1)} f_1 + \ldots + a_{sr} \cdot z_2^{k(t-1)} f_{sr} - b_1 \cdot z_1^{k(t-1)} f_1 - \ldots - b_{sr} \cdot z_1^{k(t-1)} f_{sr} = 0 \tag{11}$$

gives a basis of $z_1^{k(t-1)} \cdot V$ (and a basis of $z_2^{k(t-1)} \cdot V$). As we have black-box access to $f_1, \ldots, f_{sr}$, we can plug in $2sr$ random values to the $\mathbf{z}$-variables in Equation (11) and derive a linear system in the "variables" $a_1, \ldots a_{sr}, b_1, \ldots, b_{sr}$. A solution to this system gives a basis of $\mathcal{S}$ with probability $1 - o(1)$, thereby giving black-box access to a basis of $z_1^{k(t-1)} \cdot V$. Now, using black-box polynomial factorization[48] [KT90], we get black-box access to a basis of V. This completes the multi-gcd step.

Let $g_1, \ldots, g_s$ be the basis of V obtained in Step 3. This step also chooses black-box access to a random $g \in V$, i.e., $g = a_1 g_1 + \ldots + a_s g_s$, where $a_1, \ldots, a_s$ are picked independently and uniformly at random from $\mathbb{F}$. In Step 5, the algorithm computes black-box access to a basis of $W = \langle \pi_P(\partial^k_{\mathbf{z}}\, g)\rangle$, where P is a tuple of n_0 random linear forms in m_0 many $\mathbf{w}$-variables. By Fact 1, this can be done in $\text{poly}(\sigma)$ time with success probability $1 - o(1)$.

Proposition 2.2. *Let $W_i = \langle \pi_P(\partial^k_{\mathbf{z}}\, G_i^e)\rangle$. With probability $1 - o(1)$ over the randomness of L and P,*

$$W = W_1 \oplus \ldots \oplus W_s \quad \textit{and} \quad \dim W_i = \binom{m_0 + k(t-1) - 1}{k(t-1)} \ \textit{for all } i \in [s].$$

Step 6: Decomposing the vector spaces

In Step 6, the algorithm computes black-box access to bases of $V_1 = \langle G_1^e\rangle, \ldots, V_s = \langle G_s^e\rangle$ by decomposing the spaces V and W under the action of the set of operators $\mathcal{L}_2 = \pi_P(\partial^k_{\mathbf{z}}\ )$. We now explain how this is carried out efficiently.

Definition 2.1 (Indecomposable decomposition)**.** Let V and W be two vector spaces and $\mathcal{L}$ a set of linear operators from V to W. A decomposition of the spaces V and W as:

$$\begin{aligned} V &= V_1 \oplus \ldots \oplus V_s \\ W &= W_1 \oplus \ldots \oplus W_s \end{aligned}$$

is *indecomposable* under the action of $\mathcal{L}$ if the following hold for every $i \in [s]$,

(a) $\langle \mathcal{L} \circ V_i\rangle \subseteq W_i$

(b) There do *not* exist spaces $V_{i1}, V_{i2}, W_{i1}, W_{i2}$ such that

$$\begin{aligned} V_i &= V_{i1} \oplus V_{i2}, & W &= W_{i1} \oplus W_{i2} \ \text{ and} \\ \langle \mathcal{L} \circ V_{i1}\rangle &\subseteq W_{i1}, & \langle \mathcal{L} \circ V_{i2}\rangle &\subseteq W_{i2}. \end{aligned}$$

In our case $W = \langle \mathcal{L}_2 \circ V\rangle$ and $W_i = \langle \mathcal{L}_2 \circ V_i\rangle$ (by Proposition 2.2). Also, the decomposition $V = V_1 \oplus \ldots \oplus V_s$ and $W = W_1 \oplus \ldots \oplus W_s$ is indecomposable under the action of $\mathcal{L}_2$ as $\dim V_i = 1$ for all $i \in [s]$. It remains to show that this indecomposable decomposition is unique and it can be computed efficiently. Towards this, we take inspiration from Section A (particularly, Corollary A.1) and analyze a suitable adjoint algebra.

The adjoint algebra. Recall, $g_1, \ldots, g_s$ is the basis of V computed in Step 3. Let $h_1, \ldots, h_{sq}$ be the basis of W computed in Step 5, where $q = \binom{m_0 + k(t-1) - 1}{k(t-1)} = |\mathbf{w}^{k(t-1)}|$. With regard to the bases $g_1, \ldots, g_s$ and $h_1, \ldots, h_{sq}$, every element of $\mathcal{L}_2$ can be naturally identified with a $sq \times s$ matrix by identifying V with $\mathbb{F}^s$ and W with $\mathbb{F}^{sq}$. We will work with this matrix representation of the

[48] In fact, a simpler argument works here as the factorization is special.

elements of $\mathcal{L}_2$ which can be computed in $\text{poly}(\sigma)$ time from black-box access to $g_1, \ldots, g_s$ and $h_1, \ldots, h_{sq}$ (using Fact 1, Proposition 2.6 and solving linear systems). Let

$$\text{adj}(\mathcal{L}_2) := \left\{(D, E) \in M_s(\mathbb{F}) \times M_{sq}(\mathbb{F}) \ : \ KD = EK \ \text{ for all } \ K \in \mathcal{L}_2\right\}. \tag{12}$$

Observe that $\text{adj}(\mathcal{L}_2)$ is an $\mathbb{F}$-subalgebra of $M_s(\mathbb{F}) \times M_{sq}(\mathbb{F})$ and a basis of $\text{adj}(\mathcal{L}_2)$ can be computed in $\text{poly}(\sigma)$ time by solving a system of linear equations arising from the equation $KD = EK$ for all $K \in \mathcal{L}_2$. Define

$$\text{adj}(\mathcal{L}_2)_1 := \left\{D \in M_s(\mathbb{F}) \ : \ \text{there exists an } \ E \in M_{sq}(\mathbb{F}) \ \text{ such that } \ (D, E) \in \text{adj}(\mathcal{L}_2)\right\}. \tag{13}$$

Clearly, $\text{adj}(\mathcal{L}_2)_1$ is an $\mathbb{F}$-subalgebra of $M_s(\mathbb{F})$ and computing a basis of $\text{adj}(\mathcal{L}_2)_1$ from a basis of $\text{adj}(\mathcal{L}_2)$ is a simple task. The following proposition shows that $\text{adj}(\mathcal{L}_2)_1$ is diagonalizable.

Let $A \in \text{GL}_s(\mathbb{F})$ be the basis change matrix from $(g_1, \ldots, g_s)$ to $(G_1^e, \ldots, G_s^e)$ and

$$\mathcal{D} := \{\text{diag}(a_1, \ldots, a_s) \ : \ a_i \in \mathbb{F} \ \text{ for all } \ i \in [s]\} \ \subset \ M_s(\mathbb{F}).$$

Proposition 2.3. *$A \cdot \text{adj}(\mathcal{L}_2)_1 \cdot A^{-1} = \mathcal{D}$.*

Diagonalizing $\text{adj}(\mathcal{L}_2)_1$. Use the basis of $\text{adj}(\mathcal{L}_2)_1$ to pick a random matrix $D \in_r \text{adj}(\mathcal{L}_2)_1$. By the above proposition, and as $|\mathbb{F}| \gg s^2$, the eigenvalues of D are distinct with probability $1 - o(1)$. Compute the eigenvalues $a_1, \ldots, a_s$ by factorizing[49] the characteristic polynomial of D. Now, compute an $\tilde{A} \in \text{GL}_s(\mathbb{F})$ such that $\tilde{A} D \tilde{A}^{-1} = \text{diag}(a_1, \ldots, a_s)$ – this can be done by solving a linear system. As D has distinct eigenvalues and ADA^{-1} is also diagonal, there exist a permutation matrix $P \in \text{GL}_s(\mathbb{F})$ and a diagonal matrix $S \in \text{GL}_s(\mathbb{F})$ such that

$$\tilde{A} = PS \cdot A.$$

Computing bases of $V_1, \ldots, V_s$. Observe that

$$\begin{aligned} (G_1^e \ G_2^e \ \ldots \ G_s^e) \cdot A &= (g_1 \ g_2 \ \ldots \ g_s) \qquad \text{(by definition of } A) \\ \Rightarrow (G_1^e \ G_2^e \ \ldots \ G_s^e) \cdot S^{-1} P^{-1} &= (g_1 \ g_2 \ \ldots \ g_s) \cdot \tilde{A}^{-1}. \end{aligned}$$

Thus, by computing black-box access to the entries of the vector $(g_1 \ g_2 \ \ldots \ g_s) \cdot \tilde{A}^{-1}$, we get black-box access to $G_1^e, \ldots, G_s^e$ (up to permutation and scaling). By relabeling, we can assume that Step 6 computes black-box access to bases of $V_1 = \langle \pi_L(Q_1)^e \rangle, \ldots, V_s = \langle \pi_L(Q_s)^e \rangle$ in order.

Steps 7-8: Recovering the terms of the formula

At the end of Step 6, we have black-box access to $c_1' \pi_L(Q_1)^e, \ldots, c_s' \pi_L(Q_s)^e$, where $c_1', \ldots, c_s' \in \mathbb{F}$ and $Q_1, \ldots, Q_s \in \mathbb{F}[\mathbf{x}]$ are unknown, but $L = (\ell_1(\mathbf{z}), \ldots, \ell_n(\mathbf{z}))$ is known from Step 1. The idea now is to get black-box access to $c_1' Q_1(\mathbf{x})^e, \ldots, c_s' Q_s(\mathbf{x})^e$ by executing Steps 1-6 several times (each time by altering L slightly). Hereafter, the algorithm uses black-box polynomial factorization [KT90] to get black-box access to $\hat{c}_1 Q_1(\mathbf{x}), \ldots, \hat{c}_s Q_s(\mathbf{x})$ for some constants $\hat{c}_1, \ldots, \hat{c}_s \in \mathbb{F}^{\times}$. Then, the sparse polynomial interpolation algorithm of [KS01] gives dense representations of the

[49] This is where we need the assumption that univariate polynomial factorization over $\mathbb{F}$ can be done in randomized polynomial time.

σ-sparse polynomials $\hat{c}_1 Q_1(\mathbf{x}), \ldots, \hat{c}_s Q_s(\mathbf{x})$. Finally, we obtain a homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}(s)$ formula computing f by solving a linear system. Let us see how this idea is made to work.

Fixing the query points. The following remarks imply that the points at which the algorithm needs to query $c_1' Q_1(\mathbf{x})^e, \ldots, c_s' Q_s(\mathbf{x})^e$ in order to employ the black-box polynomial factorization algorithm and the sparse polynomial interpolation algorithm can be fixed *a priori* right after Step 6.

Remarks.

1. The sparse polynomial interpolation algorithm of [KS01] works with non-adaptive queries, i.e., each subsequent query point does not depend on answers to the previous queries.
2. The black-box polynomial factorization algorithm of [KT90] also works with non-adaptive queries. In other words, once the set of points at which we need to evaluate the irreducible factors of an input polynomial f (given as a black-box) is fixed, the algorithm uses only non-adaptive queries to f in order to compute these evaluations.

Evaluating $c_1' Q_1(\mathbf{x})^e, \ldots, c_s' Q_s(\mathbf{x})^e$ at a query point. Let $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{F}^n$ be a query point. We wish to compute $c_1' Q_1(\mathbf{a})^e, \ldots, c_s' Q_s(\mathbf{a})^e$ from black-box access to $c_1' \pi_L(Q_1)^e, \ldots, c_s' \pi_L(Q_s)^e$, where $L = (\ell_1(\mathbf{z}), \ldots, \ell_n(\mathbf{z}))$ and $\mathbf{z} = (z_1, \ldots, z_{n_0})$. Let

$$\ell_l(\mathbf{z}) = r_{l1} z_1 + \ldots + r_{l n_0} z_{n_0},$$

where $r_{l1}, \ldots, r_{l n_0} \in \mathbb{F}$ are chosen uniformly and independently at random from $\mathbb{F}$ (in Step 1) for all $l \in [n]$. Now, pick $(r_1, \ldots, r_n) \in_r \mathbb{F}^n$. For each $y \in \{1, \ldots, d\}$, define

$$\tilde{\ell}_l(y, \mathbf{z}) := (y r_l + (1-y) a_l) z_1 + r_{l2} z_2 + \ldots + r_{l n_0} z_{n_0},$$

for every $l \in [n]$. Observe that $r_{l1}' := y r_l + (1-y) a_l$ is uniformly distributed over $\mathbb{F}$ as r_l is chosen randomly from $\mathbb{F}$. Moreover, $r_{11}', r_{12}, \ldots, r_{1n_0}, \ \ldots \ , r_{n1}', r_{n2}, \ldots, r_{n n_0}$ are independent of each other as $r_1, \ldots, r_n$ are independently chosen. Hence,

$$\tilde{L}(y) := (\tilde{\ell}_1(y, \mathbf{z}), \ldots, \tilde{\ell}_n(y, \mathbf{z}))$$

is a tuple of random linear forms in the $\mathbf{z}$-variables for every $y \in \{1, \ldots, d\}$. If we execute Steps 1-6 by replacing L by $\tilde{L}(y)$ in Step 1 then we will get black-box access to

$$\tilde{c}_{\rho(1)} \cdot \pi_{\tilde{L}(y)}(Q_{\rho(1)})^e, \ \ldots \ , \tilde{c}_{\rho(s)} \cdot \pi_{\tilde{L}(y)}(Q_{\rho(s)})^e, \qquad \text{with probability } 1 - o(1), \tag{14}$$

where ρ is an unknown permutation of $[s]$ and $\tilde{c}_1, \ldots, \tilde{c}_s \in \mathbb{F}^\times$ are also unknown. We can find ρ efficiently as follows: Observe that $\tilde{L}(y)_{z_1=0} = L_{z_1=0}$. Hence, the ratio

$$\begin{aligned} \frac{c_i' \cdot \pi_{L_{z_1=0}}(Q_i)^e}{\tilde{c}_j \cdot \pi_{\tilde{L}(y)_{z_1=0}}(Q_j)^e} = \frac{c_i' \cdot [G_i^e]_{z_1=0}}{\tilde{c}_j \cdot [G_j^e]_{z_1=0}} &= \frac{c_i'}{\tilde{c}_i} \qquad \text{if } i = j, \\ &= \text{a non-constant rational function in } \mathbf{z}, \quad \text{if } i \neq j. \end{aligned}$$

The second equality is because of Condition 4 of the non-degeneracy condition (Definition 1.1), which states that there is an L such that $[G_1^e]_{z_1=0}, \ldots, [G_s^e]_{z_1=0}$ are $\mathbb{F}$-linearly independent. Thus, for a random L, $[G_1^e]_{z_1=0}, \dots, [G_s^e]_{z_1=0}$ are $\mathbb{F}$-linearly independent with probability $1 - o(1)$. Now, we can discover the permutation ρ by evaluating the ratio

$$\frac{c_i' \cdot \pi_{L_{z_1=0}}(Q_i)^e}{\tilde{c}_j \cdot \pi_{\tilde{L}(y)_{z_1=0}}(Q_j)^e}$$

at poly(d)-many random points in $\mathbb{F}^{n_0-1}$ and checking if all the evaluations are the same. This process succeeds with probability $1 - o(1)$ (as $|\mathbb{F}|$ is sufficiently large) and also gives us $\frac{c_i'}{\tilde{c}_i}$ for all $i \in [s]$. From Equation (14) and the knowledge of ρ and $\frac{c_i'}{\tilde{c}_i}$ we obtain black-box access to

$$c_1' \cdot \pi_{\tilde{L}(y)}(Q_1)^e, \; \dots \;, c_s' \cdot \pi_{\tilde{L}(y)}(Q_s)^e.$$

By setting $z_1 = 1$ and $z_2 = \dots = z_s = 0$ we get

$$p_i(y) := c_i' \cdot Q_i(yr_1 + (1-y)a_1, \; \dots \;, yr_n + (1-y)a_n)^e,$$

for every $i \in [s]$. As y is arbitrarily fixed in $[d]$, we can compute $p_i(1), \dots, p_i(d)$ for all $i \in [s]$ [50]. Treating $p_i(y)$ as a univariate polynomial in y and observing that $\deg_y p_i < d$, we can interpolate the polynomial $p_i(y)$ from the above evaluations. Notice that $p_i(0) = c_i' Q_i(\mathbf{a})^e$ for all $i \in [s]$.

Outputting a $\Sigma \wedge \Sigma\Pi^{[t]}(s)$ formula. As explained before, the black-box polynomial factorization algorithm and the sparse polynomial interpolation algorithm together give us dense representations of the polynomials $\hat{c}_1 Q_1(\mathbf{x}), \dots, \hat{c}_s Q_s(\mathbf{x})$ for some unknown $\hat{c}_1, \dots, \hat{c}_s \in \mathbb{F}$. We know that there exist $u_1, \dots, u_s \in \mathbb{F}$ such that

$$f = u_1 \cdot [\hat{c}_1 Q_1(\mathbf{x})]^m \; + \; \dots \; + \; u_s \cdot [\hat{c}_s Q_s(\mathbf{x})]^m.$$

By treating $u_1, \dots, u_s$ as formal variables, we can obtain a linear system in $u_1, \dots, u_s$ by evaluating f and $[\hat{c}_1 Q_1(\mathbf{x})]^m, \dots, [\hat{c}_s Q_s(\mathbf{x})]^m$ at s random points in $\mathbb{F}^n$. As $[\hat{c}_1 Q_1(\mathbf{x})]^m, \dots, [\hat{c}_s Q_s(\mathbf{x})]^m$ are $\mathbb{F}$-linearly independent (which follows from the non-degeneracy condition), the solution to the system gives $u_1, \dots, u_s$ that satisfy the above equation with probability $1 - o(1)$.

2.3 A random $\Sigma \wedge \Sigma\Pi^{[t]}$ circuit is non-degenerate (proof of Lemma 1.2)

We show that a homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}(s)$ formula $c_1 Q_1^m + \dots + c_s Q_s^m$ is non-degenerate with probability $1 - o(1)$ if the coefficients of the degree-t polynomials $Q_1, \dots, Q_s$ are chosen independently and uniformly at random from a set $S \subseteq \mathbb{F}$ of size at least $(ns)^{150 \cdot t}$. For this, it is sufficient to show the existence of one non-degenerate homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}(s)$ formula, as long as $|S| \gg 3ds \cdot \binom{n_0+2kt-1}{2kt}$ is sufficiently large (Proposition 2.6). This is because of Schwarz-Zippel lemma and the fact that all the non-degeneracy conditions are about vanishing of some determinant.

[50]By union bound, the total error probability remains $o(1)$ as $|\mathbb{F}|$ is sufficiently large.

Construction of a non-degenerate homogeneous $\Sigma\wedge\Sigma\Pi^{[t]}$ formula

We construct a homogeneous $\Sigma\wedge\Sigma\Pi^{[t]}(s)$ formula that satisfies non-degeneracy Conditions 1, 3 and 4 (Definition 1.1). It would easily follow from this construction that there is a homogeneous $\Sigma\wedge\Sigma\Pi^{[t]}(s)$ formula that also satisfies Condition 2 of non-degeneracy. Let $\mathbf{x} = \mathbf{y} \uplus \mathbf{z}$, where $\mathbf{z} = \{z_1, \ldots, z_{n_0}\}$. The value of n_0 is fixed in Proposition 2.6. Let L be an n-tuple of linear forms in $\mathbf{z}$-variables that defines the following affine projection: All the $\mathbf{y}$-variables map to 0, and z_u maps to z_u for all $u \in [n_0]$. Consider a homogeneous $\Sigma\wedge\Sigma\Pi^{[t]}(s)$ formula C that computes

$$f = Q_1^m + \ldots + Q_s^m, \tag{15}$$

where $Q_i = R_i(\mathbf{y},\mathbf{z}) + G_i(\mathbf{z})$ such that R_i, G_i are degree-t homogeneous polynomials, every monomial in R_i has a $\mathbf{y}$-variable, and G_i is $\mathbf{y}$-free for all $i \in [s]$. Clearly, $\pi_L(Q_i) = G_i$. We now construct $R_1, \ldots, R_s$ and $G_1, \ldots, G_s$ so that C satisfies non-degeneracy Conditions 1, 3 and 4.

Constructing $R_1, \ldots, R_s$. The number of $\mathbf{z}$-monomials of degree-$(t-1)$ is $b := \binom{n_0+t-2}{t-1}$. Let these monomials be $\gamma_1, \ldots, \gamma_b$. Consider a combinatorial design on the $\mathbf{y}$-variables, i.e., a system of s subsets of $\mathbf{y}$-variables, namely $S_1, \ldots, S_s$, such that for all $i, j \in [s]$

$$\begin{aligned} |S_i| &= kb && \text{and} \\ |S_i \cap S_j| &\le k-1 && \text{if } i \ne j. \end{aligned}$$

Such a set-system (also known as the Nisan-Wigderson design) exists[51] if $|\mathbf{y}| = n - n_0 \ge (2kb)^2$ and $s \le (kb)^k$ – these two conditions are satisfied by Proposition 2.6. Denote the kb distinct $\mathbf{y}$-variables in S_i by $\{y_{ijl} \ : \ j \in [k] \text{ and } l \in [b]\}$ and define

$$R_i := \sum_{j\in[k],\ l\in[b]} y_{ijl} \cdot \gamma_l.$$

Let the spaces $U, U_1, \ldots, U_s$ be as in Definition 1.1.

Proposition 2.4. *$U = U_1 + \ldots + U_s$ and $U_i = \left\langle \mathbf{z}^{k(t-1)} \cdot G_i^{m-k} \right\rangle$ for every $i \in [s]$.*

Constructing $G_1, \ldots, G_s$. We set $G_1, \ldots, G_s$ in such a way that $U_1 + \ldots + U_s = U_1 \oplus \ldots \oplus U_s$. Let $p := \lfloor \frac{\sqrt{n_0}}{2} \rfloor$. Consider a combinatorial design on the $\mathbf{z}$-variables, i.e., a system of s subsets of $\mathbf{z}$-variables, namely $\mathbf{z}_1, \ldots, \mathbf{z}_s$, such that for all $i, j \in [s]$

$$\begin{aligned} |\mathbf{z}_i| &= p && \text{and} \\ |\mathbf{z}_i \cap \mathbf{z}_j| &\le \left\lfloor \frac{\min\left(t(m-k), \sqrt{n_0}\right)}{10} \right\rfloor \le \left\lfloor \frac{p+1}{5} \right\rfloor, && \text{if } i \ne j. \end{aligned}$$

Such a set-system exists if

$$s \le p^{\left\lfloor \frac{\min\left(t(m-k), \sqrt{n_0}\right)}{10} \right\rfloor},$$

which is satisfied by Proposition 2.6. Let

$$G_i := \left(\sum_{z\in\mathbf{z}_i} z\right)^t, \qquad \text{for all } i \in [s]. \tag{16}$$

[51] in fact, it can be computed efficiently

Proposition 2.5. *If $m > 7k$ and $P_1, \ldots, P_s \in \mathbb{F}[\mathbf{z}]$ are such that $\deg_{\mathbf{z}} P_i \leq 2k(t-1)$ for all $i \in [s]$ and*

$$P_1 \cdot G_1^{m-k} + \ldots + P_s \cdot G_s^{m-k} = 0,$$

then $P_1 = P_2 = \ldots = P_s = 0$.

The condition $m > 7k$ is satisfied by Proposition 2.6. So, Propositions 2.4 and 2.5 imply that $U = U_1 \oplus \ldots \oplus U_s$ and $\widetilde{U_1} + \ldots + \widetilde{U_s} = \widetilde{U_1} \oplus \ldots \oplus \widetilde{U_s}$. The proof of Proposition 2.5 (in particular Observation D.1) also implies that $[G_1^{m-k}]_{z_1=0}, \ldots, [G_s^{m-k}]_{z_1=0}$ are $\mathbb{F}$-linearly independent.

In order to satisfy Condition 2, we draw an analogy between the polynomials $g_0 = G_1^e + \ldots + G_s^e$ and $f = Q_1^m + \ldots + Q_s^m$ (Equation (15)). We mimic the above construction of $Q_1, \ldots, Q_s$ and L that ensures $U = U_1 \oplus \ldots \oplus U_s$ and $\dim U_i = \binom{n_0+k(t-1)-1}{k(t-1)}$ to construct $G_1, \ldots, G_s$[52] and P such that $W = W_1 \oplus \ldots \oplus W_s$ and $\dim W_i = \binom{m_0+k(t-1)-1}{k(t-1)}$. For this, we need to satisfy $n_0 - m_0 \geq (2kc)^2$, $s \leq (kc)^k$ (where $c = \binom{m_0+t-2}{t-1}$), $e > 7k$ and

$$s \leq \left(\left\lfloor \frac{\sqrt{m_0}}{2} \right\rfloor \right)^{\left\lfloor \frac{\min\left(t(e-k), \sqrt{m_0}\right)}{10} \right\rfloor}.$$

All these relations are taken care of by Proposition 2.6.

2.4 Setting of parameters

From the statement of Theorem 1, we have $n \geq d^2$, $t \leq \sqrt{\frac{\log d}{10 \cdot \log\log d}}$, $|\mathbb{F}| \geq (ns)^{150 \cdot t}$ and

$$s \leq \min\left(n^{\frac{d}{1100 \cdot t^2}}, \exp(n^{\frac{1}{30 \cdot t^2}}) \right).$$

We leave the proof of the following proposition as an exercise.

Proposition 2.6. *Let $n_0 = \lfloor n^{\frac{1}{3 \cdot t}} \rfloor$, $m_0 = \lfloor n^{\frac{1}{15 \cdot t^2}} \rfloor$, $k = \left\lceil \frac{130 \cdot t \cdot \log s}{\log n} \right\rceil$, and (borrowing notations from Section 2.3) $b = \binom{n_0+t-2}{t-1}$ and $c = \binom{m_0+t-2}{t-1}$. Then, the following relations are satisfied:*

1. $|\mathbb{F}| \geq (ns)^{150 \cdot t} \gg d \cdot \binom{n+k-1}{k}$,
2. $|\mathbb{F}| \geq (ns)^{150 \cdot t} \gg 3ds \cdot \binom{n_0+2kt-1}{2kt}$,
3. $(nd)^k = \text{poly}(n, s^t)$,
4. $\binom{n_0+2kt-1}{2kt} = \text{poly}(n, s^t)$,
5. $n - n_0 \geq (2kb)^2$,

[52]The construction of these $G_1, \ldots, G_s$ should not be confused with the choice of G_i in Equation (16). The choices of Q_i's and G_i's till Proposition 2.5 are used to show that Condition 1, 3 and 4 of non-degeneracy are satisfied, whereas we choose the G_i's afresh to show that Condition 2 of non-degeneracy is satisfied. Finally, a union bound ensures that all the conditions of non-degeneracy are satisfied with high probability.

6. $e = m - k > 7k$,

7. $n_0 - m_0 \geq (2kc)^2$,

8. $s \leq (kc)^k$,

9. $s \leq \left(\left\lfloor \frac{\sqrt{m_0}}{2} \right\rfloor \right)^{\left\lfloor \frac{\min\left(t(e-k), \sqrt{m_0}\right)}{10} \right\rfloor}$.

Relations 1 and 2 are used for applications of the Schwartz-Zippel lemma at various places to ensure that the error probability is bounded by $o(1)$. Relations 3 and 4 guarantee that the running time of the algorithm is $\text{poly}(n, \sigma, s^t)$. Relations 5-9 are used in Section 2.3 to show that a random homogeneous $\Sigma \wedge \Sigma\Pi^{[t]}$ formula is non-degenerate with high probability.

3 Moment problem for mixtures of zero-mean Gaussians

In this section, we describe an algorithm for learning the parameters of a mixture of Gaussians in the *non-degenerate* case, given the moments of the mixture *exactly*. Since we can only estimate the moments given samples from the mixture, it is an extremely interesting problem to modify our algorithm to make it work with inexact moments (in the smoothed analysis setting) and we leave it open for future work. Our algorithm also extends naturally to the general mean case but the analysis gets more complicated and we only focus on the zero-mean case for simplicity.

Consider a mixture of Gaussians, $\mathcal{D} = \sum_{i=1}^{s} w_i \mathcal{N}(0, \Sigma_i)$, $\sum_{i=1}^{s} w_i = 1$, $w_i > 0$ for all $i \in [s]$. Let us define quadratic polynomials, $Q_1, \ldots, Q_s$, corresponding to the covariance matrices, by $Q_i(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \Sigma_i \mathbf{x}$. Also define the polynomials

$$f_m = \sum_{i=1}^{s} w_i Q_i^m, \quad f_{m-1} = \sum_{i=1}^{s} w_i Q_i^{m-1}.$$

Also let us set $n_0 = \lfloor n^{\frac{1}{6}} \rfloor$, $m_0 = \lfloor n^{\frac{1}{60}} \rfloor$, $\ell = \left\lceil \frac{260 \cdot \log s}{\log n} \right\rceil$, $m = 3\ell$, $e = 2\ell$. We will call the mixture $\mathcal{D}$ *non-degenerate* if f_m and f_{m-1} satisfy the non-degeneracy conditions in Definition 1.1 with $k = \ell$ and $k = \ell - 1$, respectively. We will need the following elementary proposition about the moment generating function of a Gaussian and mixture of Gaussians.

Proposition 3.1. *Suppose $Y \sim \mathcal{N}(\mu, \Sigma)$. Then $\mathbb{E}\left[e^{\langle \mathbf{x}, Y \rangle}\right] = e^{\langle \mu, \mathbf{x} \rangle + \frac{1}{2}\mathbf{x}^T \Sigma \mathbf{x}}$. Similary for a mixture of Gaussians, $\mathcal{D} = \sum_{i=1}^{s} w_i \mathcal{N}(\mu_i, \Sigma_i)$, the moment generating function is $\sum_{i=1}^{s} w_i e^{\langle \mu_i, \mathbf{x} \rangle + \frac{1}{2}\mathbf{x}^T \Sigma_i \mathbf{x}}$.*

The next lemma states an efficient algorithm for computing the parameters of a non-degenerate zero-mean Gaussian mixture given access to its exact moments.

Lemma 3.1. *Let $s \leq \exp\left(n^{\frac{1}{120}}\right)$. There is a randomized $\text{poly}(n, b, s)$ time algorithm (b denotes the total bit complexity of the parameters) that given black-box access to the exact $O(\log(s) / \log(n))$ order moments of a non-degenerate mixture of zero-mean Gaussians $\mathcal{D}$, recovers its parameters $(w_i, \Sigma_i)_{i=1}^{s}$.*

Remark 3. *Black-box access to the moments means given a vector $\mathbf{x} \in \mathbb{R}^n$, access to the moments of the random variable $\langle \mathbf{x}, Y \rangle$, where $Y \sim \mathcal{D}$. In other words, this means black-box access to the polynomials f_m and f_{m-1}. Of course, when $s \leq \text{poly}(n)$, our algorithm only needs access to $O(1)$ order moments of the mixture $\mathcal{D}$ in which case black-box access to the moments is immediate because we can compute all the moments explicitly. But we state our algorithm in this general form in hope of applicability in settings where black-box access to the moments might be available without an explicit access to the moments.*

Proof. (Of Lemma 3.1)

Algorithm 3 Learning mixtures of zero-mean Gaussians

```
Input: Black-box access to the exact moments of a non-degenerate mixture of zero-mean Gaussians $\mathcal{D} = \sum_{i=1}^s w_i \mathcal{N}(0, \Sigma_i)$.
Output: The parameters of the mixture $(w_i, \Sigma_i)_{i=1}^s$.

1: Use black-box access to the moments to get black-box access to the polynomials $f_m$ and $f_{m-1}$ (will be explained in the analysis).
2: Use Algorithm 2 to obtain representations $f_m = \sum_{i=1}^s w'_i (Q'_i)^m$ and $f_{m-1} = \sum_{i=1}^s \widetilde{w}_i \left(\widetilde{Q}_i\right)^{m-1}$ (with $w'_i, \widetilde{w}_i$ non-zero): we get $(w'_i, Q'_i)_{i=1}^s$ and $\left(\widetilde{w}_i, \widetilde{Q}_i\right)_{i=1}^s$.
3: Find a permutation $\sigma : [s] \to [s]$ such that $c_i = Q'_i / \widetilde{Q}_{\sigma(i)}$ is a constant (will be explained in the analysis why $\sigma$ is a permutation).
4: Let us denote $d_i = \frac{\widetilde{w}_{\sigma(i)}}{w'_i c_i^{m-1}}$. Output $\left(w'_i d_i^m, \frac{1}{d_i}\Sigma'_i\right)_{i=1}^s$, where $\Sigma'_i$ is such that $Q'_i(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \Sigma'_i \mathbf{x}$.
```

The algorithm is described in Algorithm 3. Suppose $Y \sim \mathcal{D}$. Then by Proposition 3.1, the moment generating function of $\mathcal{D}$ is given by

$$\mathbb{E}\left[e^{\langle \mathbf{x}, Y \rangle}\right] = \sum_{i=1}^s w_i e^{\frac{1}{2}\mathbf{x}^T \Sigma_i \mathbf{x}} = \sum_{i=1}^s w_i e^{Q_i(\mathbf{x})}.$$

Equating the degree $2m$ part on both sides, we get

$$\frac{m!}{(2m)!} \cdot \mathbb{E}\left[\langle \mathbf{x}, Y \rangle^{2m}\right] = \sum_{i=1}^s w_i Q_i(\mathbf{x})^m.$$

Thus given black-box access to the moments of $\mathcal{D}$, we can get black-box access to the polynomials f_m and f_{m-1}. This explains Step 1 of the algorithm. Theorem 1 guarantees us that there exist permutations $\pi_1 : [s] \to [s]$ and $\pi_2 : [s] \to [s]$, and constants $c'_1, \ldots, c'_s$ and $\widetilde{c}_1, \ldots, \widetilde{c}_s$ (all non-zero) such that

$$Q'_i = c'_i Q_{\pi_1(i)}, \quad w_{\pi_1(i)} = w'_i (c'_i)^m, \quad \widetilde{Q}_i = \widetilde{c}_i Q_{\pi_2(i)} \text{ and } w_{\pi_2(i)} = \widetilde{w}_i (\widetilde{c}_i)^{m-1}$$

for all $i \in [s]$. We also have that $\left(Q_i^m\right)_{i=1}^s$ are linearly independent (this is implied by the non-degeneracy condition) and hence the $Q'_i s$ span distinct one-dimensional spaces. Thus, there is exactly one j such that $Q'_i / \widetilde{Q}_j$ is a constant which is given by $j = \pi_2^{-1}(\pi_1(i))$. This explains Step 3 where σ is given by $\pi_2^{-1} \circ \pi_1$. Now,

$$c_i = \frac{Q'_i}{\widetilde{Q}_{\sigma(i)}} = \frac{c'_i Q_{\pi_1(i)}}{\widetilde{c}_{\sigma(i)} Q_{\pi_2(\sigma(i))}} = \frac{c'_i}{\widetilde{c}_{\sigma(i)}}.$$

Then,

$$d_i = \frac{\widetilde{w}_{\sigma(i)}}{w_i' c_i^{m-1}} = \frac{w_{\pi_2(\sigma(i))}}{\widetilde{c}_{\sigma(i)}^{m-1}} \cdot \frac{(c_i')^m}{w_{\pi_1(i)}} \cdot \frac{\widetilde{c}_{\sigma(i)}^{m-1}}{(c_i')^{m-1}} = c_i'.$$

Thus we output $\left(w_{\pi_1(i)}, \Sigma_{\pi_1(i)}\right)_{i=1}^{s}$. This completes the proof. □

Next, we instantiate the above lemma with distributional assumptions on the covariance matrices which will satisfy the non-degeneracy condition with high probability.

Corollary 3.1. *Let $s \leq \exp\left(n^{\frac{1}{120}}\right)$. There is a randomized* $\mathrm{poly}(n, b, s)$ *time algorithm (b denotes the total bit complexity of the parameters) that given black-box access to the exact $O(\log(s)/\log(n))$ order moments of a random mixture of zero-mean Gaussians $\mathcal{D}$, recovers its parameters $(w_i, \Sigma_i)_{i=1}^{s}$. Random here means that the entries of Σ_i's are chosen uniformly and indepdently at random (up to keeping the Σ_i's symmetric) from an arbitrary set $S \subset \mathbb{Q}$ of size $|S| \geq (ns)^{600}$ (and of course $w_i > 0$ for all i).*

Proof. Follows by combining Lemmas 1.2 and 3.1. □

4 Lower bound for homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}$ circuits using APP

We prove Theorem 2 in this section. The idea is to choose two parameters k and n_0 appropriately such that the measure APP_{k,n_0} of a term of a homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}(s)$ circuit is "small". We then construct an explicit polynomial $f_{n,d}$ such that $\mathsf{APP}_{k,n_0}(f_{n,d})$ is "high" which leads to a lower bound on s. It is the choice of the measure APP that is novel in this lower bound proof. The missing proofs of the technical statements can be found in Section E of the appendix.

4.1 High t case

Let $n, d, t \in \mathbb{N}$ such that $n \geq d^2$ and $\ln \frac{n}{d} \leq t \leq \frac{d}{4 \cdot e^{10} \cdot \ln d}$. We set a few parameters as follows:

- (Order of the derivatives) $k = \lfloor \delta \cdot \frac{d}{t} \rfloor$, where $\delta = \frac{1}{4e^{10}}$,
- (Number of variables after affine projection) $n_0 = \lfloor c \cdot k \rfloor$, where $c = \frac{3}{4} \cdot \frac{\ln \frac{n}{k}}{\ln \frac{d}{k}}$.

Observation 4.1. *If the parameters k, c and n_0 are chosen as above then $k \geq \lfloor \ln d \rfloor$, $c \geq \frac{3}{2}$ and $n_0 \leq \frac{d}{\ln \ln d}$.*

Observation 4.2. *Let $f \in \mathbb{F}[\mathbf{x}]$ be a homogeneous n-variate degree-d polynomial. Then,*

$$\mathsf{APP}_{k,n_0}(f) \leq \binom{d - k + n_0 - 1}{n_0 - 1}.$$

In Section 4.3, we construct an explicit family of homogeneous, multilinear polynomials $\{f_{n,d,t}\}_{n,d}$ in VNP such that $\mathsf{APP}_{k,n_0}(f_{n,d,t})$ equals the above upper bound (see Proposition 4.5).

Upper bounding the measure for a homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}$ circuit. Let C be a polynomial computed by a homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}(s)$ circuit, i.e.,

$$C = Q_{11} Q_{12} \cdots Q_{1m_1} \; + \; \ldots \; + \; Q_{s1} Q_{s2} \cdots Q_{sm_s}, \tag{17}$$

where every Q_{ij} is a homogeneous polynomial of degree at most t. By multiplying out factors if necessary, we can assume that all but one of the factors of $T_i = Q_{i1}Q_{i2}\cdots Q_{im_i}$ have degree in $[t, 2t]$. So, $m_i \leq m := \lfloor \frac{d}{t} \rfloor + 1$ for all $i \in [s]$. By subadditivity of the measure, we infer the following:

Proposition 4.1. $\mathsf{APP}_{k,n_0}(C) \leq s \cdot \binom{m}{k} \cdot \binom{n_0+2kt}{n_0}$.

Putting Propositions 4.1 and 4.5 together, we get the desired lower bound.

Proposition 4.2. *Any homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}(s)$ circuit computing $f_{n,d,t}$ must satisfy*

$$s \geq \frac{\binom{d-k+n_0-1}{n_0-1}}{\binom{m}{k} \cdot \binom{n_0+2kt}{n_0}} = \left(\frac{n}{d}\right)^{\Omega\left(\frac{d}{t\ln t}\right)}.$$

Remark. Although, in our presentation, $f_{n,d,t}$ depends on t, it is easy to get rid of t from the definition of the hard polynomial by using a simple interpolation trick (as in Lemma 14 of [KSS14]).

4.2 Low t case

Let $n, d, t \in \mathbb{N}$ such that $n \geq d^{20}$ and $1 \leq t \leq \min\left\{\frac{\ln n}{6e\cdot\ln d},\ d\right\}$. Set the parameters k, n_0 as follows:

- (Order of the derivatives) $k = \lceil \delta \cdot \frac{d}{t} \rceil$, where $\delta = \frac{1}{3e}$,
- (Number of variables after affine projection) $n_0 = \lceil n^{\frac{k}{d}} \rceil$.

The hard polynomial $f_{n,d,t}$ is defined in Section 4.3. Propositions 4.1 and 4.5 imply the following:

Proposition 4.3. *Any homogeneous $\Sigma\Pi\Sigma\Pi^{[t]}(s)$ circuit computing $f_{n,d,t}$ must satisfy*

$$s \geq \frac{\binom{d-k+n_0-1}{n_0-1}}{\binom{m}{k} \cdot \binom{n_0+2kt}{n_0}} = n^{\Omega\left(\frac{d}{t}\right)}.$$

4.3 The hard polynomial

Let the parameters n, d, t, k, n_0 be as in either Section 4.1 or Section 4.2. In this section, we describe the construction of the hard polynomial $f_{n,d,t}$. Let $n_2 := n_0(d-k)$ and $n_1 := n - n_2$. Polynomial $f_{n,d,t}$ is a homogeneous, multilinear polynomial in two sets of variables $\mathbf{y}$ and $\mathbf{u}$ such that $|\mathbf{y}| = n_1$ and $|\mathbf{u}| = n_2$. Further, $\mathbf{u} = \mathbf{u}_1 \uplus \ldots \uplus \mathbf{u}_{d-k}$, where each set $\mathbf{u}_i$ has n_0 variables $\{u_{i,1}, \ldots, u_{i,n_0}\}$.

Consider all degree-$(d-k)$ set-multilinear monomials in the $\mathbf{u}$-variables with respect to the partition $\mathbf{u} = \mathbf{u}_1 \uplus \ldots \uplus \mathbf{u}_{d-k}$. Such a set-mulilinear monomial $\beta = u_{1,j_1}u_{2,j_2}\cdots u_{d-k,j_{d-k}}$ can be naturally identified with a function

$$\begin{array}{rcl} \phi_\beta : [d-k] & \to & [n_0] \\ i & \mapsto & j_i. \end{array}$$

We say ϕ_β is *non-decreasing* if $\phi_\beta(i) \leq \phi_\beta(i+1)$ for all $i \in [d-k-1]$. Let $B := \{\beta : \phi_\beta \text{ is non-decreasing}\}$ and $\mathbf{z} = \{z_1, \ldots, z_{n_0}\}$ be a set of n_0 variables. Observe that there is a one-to-one correspondence between monomials in B and $\mathbf{z}$-monomials of degree $d-k$ which is given by the projection map

$$\begin{array}{rcl} \pi : \mathbf{u} & \to & \mathbf{z} \\ u_{i,j} & \mapsto & z_j. \end{array} \tag{18}$$

Hence, $\pi(B) = \mathbf{z}^{d-k}$ and $|B| = \binom{d-k+n_0-1}{n_0-1}$. Order the monomials in B lexicographically and call them $\left(\beta_1, \ldots, \beta_{\binom{d-k+n_0-1}{n_0-1}}\right)$. There are $\binom{n_1}{k}$ multilinear monomials in $\mathbf{y}$-variables of degree k.

Proposition 4.4. $\binom{n_1}{k} \geq \binom{d-k+n_0-1}{n_0-1}$.

Order the multilinear degree-k $\mathbf{y}$-monomials lexicographically and call the first $\binom{d-k+n_0-1}{n_0-1}$ of them $\left(\mu_1, \ldots, \mu_{\binom{d-k+n_0-1}{n_0-1}}\right)$. Define

$$f_{n,d,t}(\mathbf{y}, \mathbf{u}) := \sum_{i \in \binom{d-k+n_0-1}{n_0-1}} \mu_i \cdot \beta_i.$$

It is an easy exercise to show that the family of polynomials defined by $f_{n,d,t}$ is in VNP as the coefficient of any given monomial in $f_{n,d,t}$ can be computed efficiently.

Proposition 4.5. $\mathsf{APP}_{k,n_0}(f_{n,d,t}) = \binom{d-k+n_0-1}{n_0-1}$.

5 Conclusion and open problems

We develop a meta framework for turning lower bounds for arithmetic circuit classes into learning algorithms for the circuits classes in the *non-degenerate* case. A rudimentary form of this framework was first used in [KS19] to design learning algorithms for learning homogeneous depth three circuits in the average case. We use the framework to design learning algorithms for sums of powers of low degree polynomials. The problem of learning sums of powers of linear polynomials (aka symmetric tensor decomposition) has been extensively studied in areas across science and many algorithms have been developed for it (again in the non-degenerate case; in the worst case it is NP-hard [Hås90, Shi16]). However, even for learning sums of quadratic polynomials, we are not aware of any algorithm in the literature, except for an algorithm implicit in [GHK15] which works in a limited range of parameters. The problem of learning sums of powers of quadratics has an intimate connection to the well known problem of mixtures of Gaussians (Sections 1.3 and 3). We hope that our paper will lead to further algorithms for learning arithmetic circuits and also new connections between learning arithmetic circuits and machine learning problems, which is promising since tensor decomposition (aka learning depth three set-multilinear circuits) has found so many applications in ML. We list some of the interesting open problems below.

- **Smoothed analysis of mixtures of (general) Gaussians.** One immediate open problem is to make our algorithm resilient to noise. This is relevant to mixtures of Gaussians since given samples from the mixture, we can only estimate its moments (upto $1/\text{poly}(n)$ error using $\text{poly}(n)$ samples). We are hopeful that an appropriate modification of our algorithm will lead to polynomial time algorithm for mixtures of general Gaussians in the *smoothed* setting and when the number of components $s \leq \text{poly}(n)$.

- **Learning other arithmetic circuit classes.** It is natural to implement our framework for other arithmetic circuit classes for which we have lower bounds e.g. set-multilinear circuits, multilinear formulas, regular formulas etc. [Raz09, KSS14].

- **More connections between learning arithmetic circuits and ML.** As already mentioned, tensor decomposition finds multiple applications in ML (e.g. see [AGH+14]). It is natural to wonder if algorithms for learning more general classes of arithmetic circuits will also find applications in ML. For example, if we had learning algorithms for higher depth set-multilinear circuits (say depth-4), can this be utilized to solve problems in ML which tensor decomposition couldn't solve?

- **Combining SoS and our techniques.** One of the algorithmic techniques which is very successfully used to design algorithms for tensor decomposition is the Sum of Squares (SoS) method [BKS15, GM15, HSSS16, MSS16, RSS18]. Can SoS be also used to design learning algorithms for sums of powers of low degree polynomials (these algorithms might also be more robust to noise)? Perhaps combining SoS with our techniques might help?

- **New lower bounds using** APP**.** Can the method of affine projections of partials, perhaps also combining with shifts, be used to prove new lower bounds? May be for depth-5 circuits?

Acknowledgments

We would like to thank Youming Qiao for insightful discussions on simultaneous block-diagonalization of rectangular matrices during the workshop on *Algebraic Methods* held at the Simons Institute for the Theory of Computing in December 2018. We thank Youming particularly for his suggestion to analyze the adjoint algebra and for referring us to the paper [CIK97]. We also thank Navin Goyal for multiple helpful discussions on learning mixtures of Gaussians and related problems and for referring us to the paper [GHK15].

References

[ABG+14] Joseph Anderson, Mikhail Belkin, Navin Goyal, Luis Rademacher, and James R. Voss. The more, the merrier: the blessing of dimensionality for learning large gaussian mixtures. In *Proceedings of The 27th Conference on Learning Theory, COLT 2014, Barcelona, Spain, June 13-15, 2014*, pages 1135–1164, 2014.

[AGH+14] Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research*, 15(1):2773–2832, 2014.

[Agr05] Manindra Agrawal. Proving lower bounds via pseudo-random generators. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 92–105. Springer, 2005.

[AH17] Eric Allender and Shuichi Hirahara. New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems. In *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, pages 54:1–54:14, 2017.

[AHK12] Animashree Anandkumar, Daniel J. Hsu, and Sham M. Kakade. A method of moments for mixture models and hidden markov models. In *COLT 2012 - The 25th Annual*

Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland, pages 33.1–33.34, 2012.

[AHM+08] Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing Disjunctive Normal Form Formulas and AC^0 Circuits Given a Truth Table. *SIAM J. Comput.*, 38(1):63–84, 2008. Conference version appeared in the proceedings of CCC 2006.

[AKRR03] Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. Derandomization and Distinguishing Complexity. In *18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark*, pages 209–220, 2003.

[AM05] Dimitris Achlioptas and Frank McSherry. On spectral learning of mixtures of distributions. In *International Conference on Computational Learning Theory*, pages 458–469. Springer, 2005.

[Ang87] Dana Angluin. Queries and Concept Learning. *Machine Learning.*, 2(4):319–342, 1987.

[APVZ14] Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. Learning sparse polynomial functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 500–510, 2014.

[AV08] Manindra Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 67–75, 2008.

[BBB+00] Amos Beimel, Francesco Bergadano, Nader H. Bshouty, Eyal Kushilevitz, and Stefano Varricchio. Learning functions represented as multiplicity automata. *J. ACM*, 47(3):506–530, 2000. Conference version appeared in the proceedings of FOCS 1996.

[BCMV14] Aditya Bhaskara, Moses Charikar, Ankur Moitra, and Aravindan Vijayaraghavan. Smoothed analysis of tensor decompositions. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 594–603, 2014.

[Ber70] Elwyn R Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24:713–735, 1970.

[BIJL18] Markus Bläser, Christian Ikenmeyer, Gorav Jindal, and Vladimir Lysikov. Generalized matrix completion and algebraic natural proofs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1193–1206, 2018.

[BKS15] Boaz Barak, Jonathan A Kelner, and David Steurer. Dictionary learning and tensor decomposition via the sum-of-squares method. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 143–151, 2015.

[BS10] Mikhail Belkin and Kaushik Sinha. Polynomial learning of distribution families. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 103–112, 2010.

[BSV19] Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Reconstruction of depth-4 multilinear circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:104, 2019.

[BV08] S Charles Brubaker and Santosh S Vempala. Isotropic pca and affine-invariant clustering. In *Building Bridges*, pages 241–281. Springer, 2008.

[Chv79] Vasek Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4(3):233–235, 1979.

[CIK97] 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.

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 10:1–10:24, 2016.

[CJ10] Pierre Comon and Christian Jutten. *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

[CKW11] Xi Chen, Neeraj Kayal, and Avi Wigderson. Partial derivatives in arithmetic complexity and beyond. *Foundations and Trends in Theoretical Computer Science*, 6(1-2):1–138, 2011.

[Czo99] Sabastian Czort. The complexity of minimizing disjunctive normal form formulas. Master's thesis, University of Aarhus, 1999.

[Das99] Sanjoy Dasgupta. Learning mixtures of gaussians. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 634–644, 1999.

[DKK$^+$19] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high-dimensions without the computational intractability. *SIAM Journal on Computing*, 48(2):742–864, 2019.

[DLCC07] Lieven De Lathauwer, Josphine Castaing, and Jean-Franois Cardoso. Fourth-order cumulant-based blind identification of underdetermined mixtures. *IEEE Transactions on Signal Processing*, 55(6):2965–2973, 2007.

[DS07] Sanjoy Dasgupta and Leonard Schulman. A probabilistic analysis of em for mixtures of separated, spherical gaussians. *Journal of Machine Learning Research*, 8(Feb):203–226, 2007.

[DSY10] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM Journal on Computing*, 39(4):1279–1293, 2010.

[Ebe91] Wayne Eberly. Decompositions of algebras over R and C. *Computational Complexity*, 1:211–234, 1991.

[Fel09] Vitaly Feldman. Hardness of approximate two-level logic minimization and PAC learning with membership queries. *J. Comput. Syst. Sci.*, 75(1):13–26, 2009. Conference version appeared in the proceedings of STOC 2006.

[Fis94] Ismor Fischer. Sums of like powers of multivariate linear forms. *Mathematics Magazine*, 67(1):59–61, 1994.

[FK09] Lance Fortnow and Adam R. Klivans. Efficient learning algorithms yield circuit lower bounds. *J. Comput. Syst. Sci.*, 75(1):27–36, 2009. Conference version appeared in the proceedings of COLT 2006.

[FLMS15] Hervé Fournier, Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower bounds for depth-4 formulas computing iterated matrix multiplication. *SIAM J. Comput.*, 44(5):1173–1201, 2015. Conference version appeared in the proceedings of STOC 2014.

[For15a] Michael A Forbes. Deterministic divisibility testing via shifted partial derivatives. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 451–465. IEEE, 2015.

[For15b] Michael A. Forbes. Deterministic divisibility testing via shifted partial derivatives. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 451–465, 2015.

[FR85] Katalin Friedl and Lajos Rónyai. Polynomial time solutions of some problems in computational algebra. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 153–162, 1985.

[FS13] Michael A. Forbes and Amir Shpilka. Quasipolynomial-Time Identity Testing of Non-commutative and Read-Once Oblivious Algebraic Branching Programs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 243–252, 2013.

[GHK15] 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 2015, Portland, OR, USA, June 14-17, 2015*, pages 761–770, 2015.

[GKKS14] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Approaching the Chasm at Depth Four. *J. ACM*, 61(6):33:1–33:16, 2014. Conference version appeared in the proceedings of CCC 2013.

[GKKS16] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic circuits: A chasm at depth 3. *SIAM J. Comput.*, 45(3):1064–1079, 2016. Conference version appeared in the proceedings of FOCS 2013.

[GKL11] Ankit Gupta, Neeraj Kayal, and Satyanarayana V. Lokam. Efficient Reconstruction of Random Multilinear Formulas. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 778–787, 2011.

[GKL12] Ankit Gupta, Neeraj Kayal, and Satyanarayana V. Lokam. Reconstruction of depth-4 multilinear circuits with top fan-in 2. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 625–642, 2012.

[GKP18] Ignacio García-Marco, Pascal Koiran, and Timothée Pecatte. Polynomial equivalence problems for sum of affine powers. In *Proceedings of the 2018 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2018, New York, NY, USA, July 16-19, 2018*, pages 303–310, 2018.

[GKQ14] Ankit Gupta, Neeraj Kayal, and Youming Qiao. Random arithmetic formulas can be reconstructed efficiently. *Computational Complexity*, 23(2):207–303, 2014. Conference version appeared in the proceedings of CCC 2013.

[GM15] Rong Ge and Tengyu Ma. Decomposing overcomplete 3rd order tensors using sum-of-squares algorithms. *arXiv preprint arXiv:1504.05287*, 2015.

[Har70] R Harshman. Foundations of the parafac procedure: Model and conditions for an explanatory factor analysis. *Technical Report UCLA Working Papers in Phonetics 16, University of California, Los Angeles, Los Angeles, CA*, 1970.

[Hås90] Johan Håstad. Tensor Rank is NP-Complete. *J. Algorithms*, 11(4):644–654, 1990. Conference version appeared in the proceedings of ICALP 1989.

[HK13] Daniel J. Hsu and Sham M. Kakade. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 11–20, 2013.

[HKZ12] Daniel Hsu, Sham M Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012.

[HL18] Samuel B Hopkins and Jerry Li. Mixture models, robustness, and sum of squares proofs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1021–1034, 2018.

[HS80] Joos Heintz and Claus-Peter Schnorr. Testing polynomials which are easy to compute. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 262–272, 1980.

[HSSS16] Samuel B Hopkins, Tselil Schramm, Jonathan Shi, and David Steurer. Fast spectral algorithms from sum-of-squares proofs: tensor decomposition and planted sparse vectors. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 178–191, 2016.

[Jac89] Nathan Jacobson. *Basic Algebra 2 (Second Edition)*. Dover Books on Mathematics, 1989.

[Joh74] David S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974.

[Kay11] Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1409–1421, 2011.

[Kay12a] Neeraj Kayal. Affine projections of polynomials: extended abstract. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 643–662, 2012.

[Kay12b] Neeraj Kayal. An exponential lower bound for the sum of powers of bounded degree polynomials. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:81, 2012.

[KC00] Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 73–79, 2000.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *computational complexity*, 13(1-2):1–46, 2004.

[KK10] Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 299–308. IEEE, 2010.

[KLSS17] Neeraj Kayal, Nutan Limaye, Chandan Saha, and Srikanth Srinivasan. An Exponential Lower Bound for Homogeneous Depth Four Arithmetic Formulas. *SIAM J. Comput.*, 46(1):307–335, 2017. Conference version appeared in the proceedings of FOCS 2014.

[KNS16] Neeraj Kayal, Vineet Nair, and Chandan Saha. Separation between read-once oblivious algebraic branching programs (roabps) and multilinear depth three circuits. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, pages 46:1–46:15, 2016.

[KNS19] Neeraj Kayal, Vineet Nair, and Chandan Saha. Average-case linear matrix factorization and reconstruction of low width algebraic branching programs. *Computational Complexity*, 28(4):749–828, 2019.

[KNST17] Neeraj Kayal, Vineet Nair, Chandan Saha, and Sébastien Tavenas. Reconstruction of Full Rank Algebraic Branching Programs. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, pages 21:1–21:61, 2017.

[Koi12] Pascal Koiran. Arithmetic circuits: The chasm at depth four gets wider. *Theor. Comput. Sci.*, 448:56–65, 2012.

[KS01] Adam R. Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 216–223, 2001.

[KS06] Adam R. Klivans and Amir Shpilka. Learning restricted models of arithmetic circuits. *Theory of Computing*, 2(10):185–206, 2006. Conference version appeared in the proceedings of COLT 2003.

[KS09a] Zohar Shay Karnin and Amir Shpilka. Reconstruction of generalized depth-3 arithmetic circuits with bounded top fan-in. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 274–285, 2009.

[KS09b] Adam R. Klivans and Alexander A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. *J. Comput. Syst. Sci.*, 75(1):2–12, 2009. Conference version appeared in the proceedings of FOCS 2006.

[KS14] Mrinal Kumar and Shubhangi Saraf. The limits of depth reduction for arithmetic formulas: it's all about the top fan-in. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 136–145, 2014.

[KS17a] Pravesh K Kothari and David Steurer. Outlier-robust moment-estimation via sum-of-squares. *arXiv preprint arXiv:1711.11581*, 2017.

[KS17b] Mrinal Kumar and Shubhangi Saraf. On the Power of Homogeneous Depth 4 Arithmetic Circuits. *SIAM J. Comput.*, 46(1):336–387, 2017. Conference version appeared in the proceedings of FOCS 2014.

[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, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 413–424, 2019.

[KSB] Krull-Schmidt Theorem. https://mathstrek.blog/2015/01/17/krull-schmidt-theorem/.

[KSDK14] Murat Kocaoglu, Karthikeyan Shanmugam, Alexandros G. Dimakis, and Adam R. Klivans. Sparse polynomial learning and graph sketching. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3122–3130, 2014.

[KSS14] Neeraj Kayal, Chandan Saha, and Ramprasad Saptharishi. A super-polynomial lower bound for regular arithmetic formulas. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 146–153, 2014.

[KSS18] Pravesh K Kothari, Jacob Steinhardt, and David Steurer. Robust moment estimation and improved clustering via sum of squares. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1035–1046, 2018.

[KSV05] Ravindran Kannan, Hadi Salmasian, and Santosh Vempala. The spectral method for general mixture models. In *International Conference on Computational Learning Theory*, pages 444–457. Springer, 2005.

[KT90] 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. *J. Symb. Comput.*, 9(3):301–320, 1990.

[Kum19] Mrinal Kumar. A quadratic lower bound for homogeneous algebraic branching programs. *Computational Complexity*, 28(3):409–435, 2019. Conference version appeared in the proceedings of CCC 2017.

[LLL82] Arjen K Lenstra, Hendrik W Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant Depth Circuits, Fourier Transform, and Learnability. *J. ACM*, 40(3):607–620, 1993. Conference version appeared in the proceedings of FOCS 1989.

[Lov75] László Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383–390, 1975.

[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.

[Mas79] W. J. Masek. Some NP-complete set covering problems. Unpublished Manuscript, 1979.

[MR05] Elchanan Mossel and Sébastien Roch. Learning nonsingular phylogenies and hidden markov models. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 366–375, 2005.

[MSS16] Tengyu Ma, Jonathan Shi, and David Steurer. Polynomial-time tensor decompositions with sum-of-squares. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 438–446. IEEE, 2016.

[MV10] Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of gaussians. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 93–102, 2010.

[MV18] Daniel Minahan and Ilya Volkovich. Complete derandomization of identity testing and reconstruction of read-once formulas. *TOCT*, 10(3):10:1–10:11, 2018. Conference version appeared in the proceedings of CCC 2017.

[MW17] Cody D. Murray and R. Ryan Williams. On the (Non) NP-Hardness of Computing Circuit Complexity. *Theory of Computing*, 13(4):1–22, 2017.

[Nis91] Noam Nisan. Lower Bounds for Non-Commutative Computation (Extended Abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 410–418, 1991.

[NW97] Noam Nisan and Avi Wigderson. Lower Bounds on Arithmetic Circuits Via Partial Derivatives. *Computational Complexity*, 6(3):217–234, 1997. Conference version appeared in the proceedings of FOCS 1995.

[OSlV16] Rafael Oliveira, Amir Shpilka, and Ben lee Volk. Subexponential size hitting sets for bounded depth multilinear formulas. *computational complexity*, 25(2):455–505, 2016.

[Pea94] Karl Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185:71–110, 1894.

[PFJ03] Haim H. Permuter, Joseph M. Francos, and Ian H. Jermyn. Gaussian mixture models of texture and colour for image database retrieval. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '03, Hong Kong, April 6-10, 2003*, pages 569–572, 2003.

[Raz09] Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM*, 56(2):8:1–8:17, 2009. Conference version appeared in the proceedings of STOC 2004.

[Rón90] Lajos Rónyai. Computing the structure of finite algebras. *J. Symb. Comput.*, 9(3):355–373, 1990.

[RR95] Douglas A Reynolds and Richard C Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE transactions on speech and audio processing*, 3(1):72–83, 1995.

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005.

[RSS18] Prasad Raghavendra, Tselil Schramm, and David Steurer. High-dimensional estimation via sum-of-squares proofs. *arXiv preprint arXiv:1807.11419*, 6, 2018.

[RV17] Oded Regev and Aravindan Vijayaraghavan. On learning mixtures of well-separated gaussians. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 85–96. IEEE, 2017.

[Sap15] Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. *Github survey*, 2015.

[Sch80] Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM*, 27(4):701–717, 1980.

[Shi16] Yaroslav Shitov. How hard is the tensor rank? *arXiv*, abs/1611.01559, 2016.

[Shp09] Amir Shpilka. Interpolation of depth-3 arithmetic circuits with two multiplication gates. *SIAM J. Comput.*, 38(6):2130–2161, 2009. Conference version appeared in the proceedings of STOC 2007.

[Sin16] Gaurav Sinha. Reconstruction of real depth-3 circuits with top fan-in 2. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 31:1–31:53, 2016.

[SK01] Arora Sanjeev and Ravi Kannan. Learning mixtures of arbitrary gaussians. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 247–257, 2001.

[Str73] Volker Strassen. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973.

[SV14] Amir Shpilka and Ilya Volkovich. On reconstruction and testing of read-once formulas. *Theory of Computing*, 10:465–514, 2014. Conference version appeared in the proceedings of STOC 2008 and APPROX-RANDOM 2009.

[Swe18] Joseph Swernofsky. Tensor Rank is Hard to Approximate. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, pages 26:1–26:9, 2018.

[SWZ19] Zhao Song, David P. Woodruff, and Peilin Zhong. Relative Error Tensor Low Rank Approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2772–2789, 2019.

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

[Tav13] Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. In *Mathematical Foundations of Computer Science 2013 - 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings*, pages 813–824, 2013.

[Uma99] Christopher Umans. Hardness of Approximating Sigma$_2{}^{\mathrm{P}}$ Minimization Problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 465–474, 1999.

[Vol16] Ilya Volkovich. A Guide to Learning Arithmetic Circuits. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, pages 1540–1561, 2016.

[VW04] Santosh Vempala and Grant Wang. A spectral algorithm for learning mixture models. *Journal of Computer and System Sciences*, 68(4):841–860, 2004.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, pages 216–226, 1979.

A The adjoint algebra

Let U and W be vector spaces and $\mathcal{L}$ a set of linear operators from U to W such that $W = \langle \mathcal{L} \circ U \rangle$. Suppose U and W decompose into indecomposable subspaces as:

$$U = U_1 \oplus \ldots \oplus U_s \quad \text{and} \quad W = W_1 \oplus \ldots \oplus W_s$$

such that $W_i = \langle \mathcal{L} \circ U_i \rangle$ for all $i \in [s]$. In this section, we give a brief overview of the adjoint algebra associated with $\mathcal{L}$ and show how analyzing the adjoint provides an avenue to showing uniqueness of decomposition of the above spaces. We will explain this by assuming[53] $U = W$ and $U_i = W_i$ for all $i \in [s]$. Let $m = \dim U$. Once a basis of U is fixed, U can be identified with $\mathbb{F}^m$

[53]This assumption is without any loss of generality (see Section B).

and elements of $\mathcal{L}$ are $m \times m$ matrices in $M_m(\mathbb{F})$. Let $\mathcal{R} \subseteq M_m(\mathbb{F})$ be the $\mathbb{F}$-algebra generated by[54] $\mathcal{L} \cup \{I_m\}$, where I_m is the $m \times m$ identity matrix. As $U = \langle \mathcal{L} \circ U \rangle$ and $U_i = \langle \mathcal{L} \circ U_i \rangle$, we have $L\mathbf{u} \in U$ and $L\mathbf{u}_i \in U_i$ for all $L \in \mathcal{L}$, $\mathbf{u} \in U$ and $\mathbf{u}_i \in U_i$. This gives U an $\mathcal{R}$-module[55] structure and $U_1, \ldots, U_s$ are $\mathcal{R}$-submodules of U. We say U_i is an *indecomposable* $\mathcal{R}$-module if there are no proper $\mathcal{R}$-submodules U_{i1} and U_{i2} of U_i such that $U_i = U_{i1} \oplus U_{i2}$. A decomposition of an $\mathcal{R}$-module U as

$$U = U_1 \oplus \ldots \oplus U_s,$$

where $U_1, \ldots, U_s$ are indecomposable $\mathcal{R}$-submodules of U, is *unique* if it is the only possible decomposition of U into indecomposable $\mathcal{R}$-submodules (up to reordering of the U_i's).

A.1 Module homomorphisms

A map ϕ from an $\mathcal{R}$-module U to another $\mathcal{R}$-module V is an $\mathcal{R}$-module *homomorphism* from U to V if $\phi(R\mathbf{u} + S\mathbf{v}) = R\phi(\mathbf{u}) + S\phi(\mathbf{v})$ for all $R, S \in \mathcal{R}$ and $\mathbf{u}, \mathbf{v} \in U$. Such a ϕ is an $\mathcal{R}$-module *isomorphism* from U to V if it is a bijection. An $\mathcal{R}$-module homomorphism from U to U is called an $\mathcal{R}$-module *endomorphism* of U, and an $\mathcal{R}$-module isomorphism from U to U is called an $\mathcal{R}$-module *automorphism* of U. It turns out that the set of $\mathcal{R}$-module endomorphisms of U can be computed efficiently as follows: Recall that in our case, $U = \mathbb{F}^m$ and $\mathcal{R} \subseteq M_m(\mathbb{F})$. Define the *adjoint* of $\mathcal{R}$ as

$$\text{adj}(\mathcal{R}) := \{D \in M_m(\mathbb{F}) \ : \ LD = DL \text{ for all } L \in \mathcal{L}\}. \tag{19}$$

Observe that $\text{adj}(\mathcal{R})$ is an $\mathbb{F}$-subalgebra of $M_m(\mathbb{F})$.

Proposition A.1. *The adjoint* $\text{adj}(\mathcal{R})$ *is precisely the set of all* $\mathcal{R}$*-module endomorphism of* U.

Proof. Let ϕ be an $\mathcal{R}$-module endomorphism of U. As $\mathcal{R}$ contains the identity matrix I_m, $\mathbb{F} \subseteq \mathcal{R}$ and so ϕ is a linear transformation from U to U. Let $D_\phi \in M_m(\mathbb{F})$ be the matrix corresponding to ϕ. Since $\phi(R\mathbf{u}) = R\phi(\mathbf{u})$, we have $D_\phi R\mathbf{u} = RD_\phi\mathbf{u}$ for all $R \in \mathcal{R}$ and $\mathbf{u} \in U$. Hence, $D_\phi R = RD_\phi$ for all $R \in \mathcal{R}$ implying $D_\phi \in \text{adj}(\mathcal{R})$. On the other hand, if $D \in \text{adj}(\mathcal{R})$ then the map $\phi_D : U \to U$ defined as $\phi_D(\mathbf{u}) := D\mathbf{u}$ satisfies $\phi_D(R\mathbf{u} + S\mathbf{v}) = R\phi_D(\mathbf{u}) + S\phi_D(\mathbf{v})$ for all $R, S \in \mathcal{R}$ and $\mathbf{u}, \mathbf{v} \in U$. So, ϕ_D is an $\mathcal{R}$-module endomorphism of U. $\square$

A basis of the adjoint can be computed efficiently by solving a system of linear equations arising from the equation $LD = DL$ for all $L \in \mathcal{L}$.

A.2 Module decomposition

Let $U = \mathbb{F}^m$ be an $\mathcal{R}$-module, where $\mathcal{R} \subseteq M_m(\mathbb{F})$. By Proposition A.1, the invertible elements of $\text{adj}(\mathcal{R})$ are the $\mathcal{R}$-module automorphisms of U and these can be used to describe all possible decomposition of U into indecomposable $\mathcal{R}$-modules.

[54] An $\mathbb{F}$-algebra $\mathcal{R}$ has two binary operations $+$ and $\cdot$ defined on its elements such that $(\mathcal{R}, +)$ is a $\mathbb{F}$-vector space, $(\mathcal{R}, +, \cdot)$ is an associative ring, and for every $a, b \in \mathbb{F}$ and $B, C \in \mathcal{R}$ it holds that $(aB)C = B(aC) = a(BC)$. The $\mathbb{F}$-algebra $\mathcal{R} \subseteq M_m(\mathbb{F})$ generated by $\mathcal{L} \subseteq M_m(\mathbb{F})$ is the set of all finite $\mathbb{F}$-linear sums of finite products of elements of $\mathcal{L}$.

[55] Let $\mathcal{R}$ be an $\mathbb{F}$-algebra with a multiplicative identity I. A vector space U is an $\mathcal{R}$-module if there is a bilinear map $\circ$ from $\mathcal{R} \times U$ to U such that $I \circ \mathbf{u} = \mathbf{u}$ and $(RS) \circ \mathbf{u} = R \circ (S \circ \mathbf{u})$ for all $\mathbf{u} \in U$ and $R, S \in \mathcal{R}$. In our case, $\circ$ is simply the matrix-vector multiplication operation.

Proposition A.2. *(a) If $U = U_1 \oplus \ldots \oplus U_s$ is a decomposition of U into indecomposable $\mathcal{R}$-submodules and $D \in \mathrm{adj}(\mathcal{R})$ is invertible then*

$$U = DU_1 \oplus \ldots \oplus DU_s$$

is another decomposition of U into indecomposable $\mathcal{R}$-submodules.

(b) If $U = U'_1 \oplus \ldots \oplus U'_l$ is any other decomposition of U into indecomposable $\mathcal{R}$-submodules then $l = s$ and there is an invertible $D \in \mathrm{adj}(\mathcal{R})$ and a permutation σ of $[s]$ such that

$$U'_i = DU_{\sigma(i)} \quad \textit{for all } i \in [s].$$

Proof. The proof of (a) follows from the easy observation that DU_i is an $\mathcal{R}$-submodule of U.

To prove (b) we will make use of the Krull-Schmidt theorem for modules ([Jac89] p. 110, [KSB]).

Theorem 3 (Krull-Schmidt)**.** *Let $\mathcal{R}$ be an $\mathbb{F}$-algebra and U a finite dimensional vector space that is also an $\mathcal{R}$-module. If*

$$U = U_1 \oplus \ldots \oplus U_s \quad \textit{and} \quad U = U'_1 \oplus \ldots \oplus U'_l$$

are two decomposition of U into indecomposable $\mathcal{R}$-submodules then $l = s$ and there is a permutation σ of s such that U'_i and $U_{\sigma(i)}$ are isomorphic as $\mathcal{R}$-modules for all $i \in [s]$.

The theorem holds for any module that is both Noetherian and Artinian – a finite dimensional module is trivially Noetherian and Artinian. Applying the Krull-Schmidt theorem to our setting, we get $l = s$ and that there is a permutation σ of $[s]$ such that $U'_i \cong U_{\sigma(i)}$ as $\mathcal{R}$-modules for all $i \in [s]$. Let these ismorphisms be $\phi_1, \ldots, \phi_s$, i.e.,

$$U'_i = \phi_i(U_{\sigma(i)}) \quad \text{for all } i \in [s].$$

Define a map ϕ from U to U as follows: Let $\mathbf{u} \in U$. If $\mathbf{u} = \mathbf{u}_1 + \ldots + \mathbf{u}_s$, where $\mathbf{u}_i \in U_i$, then $\phi(\mathbf{u}) := \phi_1(\mathbf{u}_{\sigma(1)}) + \ldots + \phi_s(\mathbf{u}_{\sigma(s)})$. Observe that ϕ restricted to $U_{\sigma(i)}$ is just ϕ_i. It is easy to verify that ϕ is an $\mathcal{R}$-module automorphism of U. Hence, by Proposition A.1, there is an invertible $D \in \mathrm{adj}(\mathcal{R})$ such that $\phi(\mathbf{u}) = D\mathbf{u}$ and so $U'_i = DU_{\sigma(i)}$ for all $i \in [s]$. $\square$

Proposition A.2 implies that the invertible elements of $\mathrm{adj}(\mathcal{R})$ exactly capture the various possible decompositions of U into indecomposable $\mathcal{R}$-modules. So, analyzing the adjoint becomes vital in showing uniqueness of a module decomposition.

A.3 Uniqueness of decomposition

It turns out that showing uniqueness of module decomposition is essentially equivalent to showing that the elements of the adjoint are simultaneously block-diagonalizable. As before, let $U_1 \oplus \ldots \oplus U_s$ be a decomposition of the $\mathcal{R}$-module $U = \mathbb{F}^m$ into indecomposable $\mathcal{R}$-submodules. For simplicity, assume $\dim U_i = r$ for all $i \in [s]$. Let $(\mathbf{u}_{i1}, \ldots, \mathbf{u}_{ir})$ be a basis of U_i, and $A \in \mathrm{GL}_m(\mathbb{F})$ be the basis change matrix from the standard basis of $\mathbb{F}^m$ to $(\mathbf{u}_{11}, \mathbf{u}_{12}, \ldots, \mathbf{u}_{1r}, \ \ldots \ , \mathbf{u}_{s1}, \mathbf{u}_{s2}, \ldots, \mathbf{u}_{sr})$.

Proposition A.3. *(a) If $U = U_1 \oplus \ldots \oplus U_s$ is the unique decomposition of U into indecomposable $\mathcal{R}$-submodules, and $U_i \not\cong U_j$ as $\mathcal{R}$-modules for $i \neq j$, then $A \cdot \mathrm{adj}(\mathcal{R}) \cdot A^{-1}$ consists of block-diagonal matrices (with block size r).*

(b) *If $A \cdot \mathrm{adj}(\mathcal{R}) \cdot A^{-1}$ consists of block-diagonal matrices (with block size r) then $U = U_1 \oplus \ldots \oplus U_s$ is the unique decomposition of U into indecomposable $\mathcal{R}$-submodules.*

Proof. Let $D \in \mathrm{adj}(\mathcal{R})$ be invertible. By Proposition A.2 (a),

$$U = DU_1 \oplus \ldots \oplus DU_s$$

is another decomposition of U into indecomposable $\mathcal{R}$-submodules. If $U = U_1 \oplus \ldots \oplus U_s$ is the unique decomposition and $U_i \not\cong U_j$ as $\mathcal{R}$-modules for $i \neq j$, then $DU_i = U_i$ for all $i \in [s]$. In other words, $A \cdot D \cdot A^{-1}$ is block-diagonal for every invertible $D \in \mathrm{adj}(\mathcal{R})$. Now, a simple application of the Schwartz-Zippel lemma implies $A \cdot \mathrm{adj}(\mathcal{R}) \cdot A^{-1}$ consists of block-diagonal matrices if $|\mathbb{F}| > 3sr$. This completes the proof of part (a).

Suppose $U = U'_1 \oplus \ldots \oplus U'_s$ be another decomposition of U into indecomposable $\mathcal{R}$-submodules. By Proposition A.2 (b), there is an invertible $D \in \mathrm{adj}(\mathcal{R})$ and a permutation σ of $[s]$ such that

$$U'_i = DU_{\sigma(i)} \quad \text{for all } i \in [s].$$

If $A \cdot \mathrm{adj}(\mathcal{R}) \cdot A^{-1}$ consists of only block-diagonal matrices then

$$DU_{\sigma(i)} \subseteq U_{\sigma(i)}, \quad \text{implying} \quad U'_i \subseteq U_{\sigma(i)} \quad \text{for all } i \in [s].$$

This further implies $U'_i = U_{\sigma(i)}$ for all $i \in [s]$ as $\sum_{i \in [s]} \dim U'_i = \sum_{i \in [s]} \dim U_{\sigma(i)}$. Thus, the decomposition $U = U_1 \oplus \ldots \oplus U_s$ is unique. □

The adjoint algebra arising in our case

As briefed in Section 1.2, our learning problem is essentially reduced to the following module decomposition problem: We are given a basis of an appropriate $\mathcal{R}$-module U that decomposes as

$$U = U_1 \oplus \ldots \oplus U_s, \tag{20}$$

where U_i is an $\mathcal{R}$-submodule of U that is not guaranteed to be indecomposable and $\dim U_i = r$ for all $i \in [s]$. We are required to

- show that each U_i is an indecomposable $\mathcal{R}$-module,
- show that the above decompostion is unique,
- find the decomposition, i.e., compute bases of $U_1, \ldots U_s$.

Here, $\mathcal{R}$ is the $\mathbb{F}$-algebra generated by a set of linear operators $\mathcal{L}$ on U. Guided by Proposition A.3, we analyze the adjoint $\mathrm{adj}(\mathcal{R})$. It turns out that the "richness" of the carefully chosen set of linear operators $\mathcal{L}$ implies that

$$A \cdot \mathrm{adj}(\mathcal{R}) \cdot A^{-1} \;=\; \mathcal{D} \;:=\; \{\mathrm{diag}(a_1, \ldots, a_s) \otimes I_r \;:\; a_i \in \mathbb{F} \text{ for all } i \in [s]\},$$

where A is as defined at the beginning of this section. In other words, elements of the adjoint are simultaneously diagonalizable. The following is an easy corollary of Proposition A.3.

Corollary A.1. *If $A \cdot \mathrm{adj}(\mathcal{R}) \cdot A^{-1} = \mathcal{D}$ then the $\mathcal{R}$-modules $U_1, \ldots, U_s$ (in Equation (20)) are indecomposable and $U = U_1 \oplus \ldots \oplus U_s$ is the unique decomposition of U into indecomposable $\mathcal{R}$-submodules.*

Finally, we find the decomposition by simultaneously diagonalizing the basis elements of $\mathrm{adj}(\mathcal{R})$.

B Reducing vector space decomposition to module decomposition

In this section, we reduce the vector space decomposition problem to the module decomposition problem. In fact, our reduction works for a more general problem, which we call generalized vector space decomposition. We describe this setting below.

Suppose we have a directed graph $G = (V, E)$. At each vertex $v \in V$, we have a vector space U_v and each edge $(v, w) \in E$ carriers a set of linear maps $\mathcal{L}_{v,w}$ from U_v to U_w. A vector space decomposition of the collection of vector spaces $(U_v)_{v\in V}$ is a collection of decompositions

$$U_v = U_{v,1} \oplus \cdots \oplus U_{v,s}$$

such that $\langle \mathcal{L}_{v,w} \circ U_{v,i} \rangle \subseteq U_{w,i}$ for all $i \in [s]$ and $(v, w) \in E$. The collection of decompositions is indecomposable if there are no *finer* decompositions, i.e., there are no proper subspaces $U'_{v,i}, U''_{v,i}$ of $U_{v,i}$ (and $U'_{w,i}, U''_{w,i}$ of $U_{w,i}$) such that $U_{v,i} = U'_{v,i} \oplus U''_{v,i}$ (and $U_{w,i} = U'_{w,i} \oplus U''_{w,i}$), and $\langle \mathcal{L}_{v,w} \circ U'_{v,i} \rangle \subseteq U'_{w,i}$, $\langle \mathcal{L}_{v,w} \circ U''_{v,i} \rangle \subseteq U''_{w,i}$ for all $i \in [s]$ and $(v, w) \in E$. The generalized vector space decomposition problem is the task of computing a collection of indecomposable decompositions of the spaces $(U_v)_{v\in V}$ from the graph G. Note that the module isomorphism problem corresponds to a single loop on one vertex and the vector space decomposition problem corresponds to two vertices and a single edge between them.

There is a simple reduction from the generalized vector space decomposition problem to the module decomposition problem. Given the above instance, we consider the vector space $U = \oplus_{v\in V} U_v$. We define some special linear maps from U to U which will be central to the reduction. We note that we just need to describe the behaviour of the linear maps on each of the U_v's, as we can extend the maps linearly to the whole space U. The first set of linear maps are projections onto U_v's.

$$\mathit{\Pi}_v(u) = \begin{cases} u & \text{if } u \in U_v \\ 0 & \text{if } u \in U_{v'} \text{ for } v' \neq v. \end{cases}$$

That is, $\mathit{\Pi}_v$ is the projector onto U_v. The second set of linear maps are the natural extensions of $\mathcal{L}_{v,w}$'s to the whole space. Given $L \in \mathcal{L}_{v,w}$, we define the extension of L as

$$\text{ext}(L)(u) = \begin{cases} L(u) & \text{if } u \in U_v \\ 0 & \text{if } u \in U_{v'} \text{ for } v' \neq v. \end{cases}$$

Then, we can define $\widetilde{\mathcal{L}}_{v,w} = \{\text{ext}(L) : L \in \mathcal{L}_{v,w}\}$. Let $\mathcal{R}$ be the algebra generated by $\{I_m\} \cup \{\mathit{\Pi}_v\}_{v\in V} \cup \{\widetilde{\mathcal{L}}_{v,w}\}_{(v,w)\in E}$, where $m := \dim(U)$. Observe that U can be naturally treated as an $\mathcal{R}$-module. Now, we have the following elementary proposition which characterizes $\mathcal{R}$-submodules of U (i.e., subspaces of U that are invariant with respect to $\mathcal{R}$).

Proposition B.1. *A subspace $U' \subseteq U$ is an $\mathcal{R}$-submodule of U (i.e., $\langle \mathcal{R} \circ U' \rangle \subseteq U'$) if and only if it is of the form $\oplus_{v\in V} U'_v$ such that $U'_v \subseteq U_v$ for all $v \in V$ and $\langle \mathcal{L}_{v,w} \circ U'_v \rangle \subseteq U'_w$ for all $(v, w) \in E$.*

Proof. One direction is clear. If U' is of the form $\oplus_{v\in V} U'_v$ such that $U'_v \subseteq U_v$ for all $v \in V$ and $\langle \mathcal{L}_{v,w} \circ U'_v \rangle \subseteq U'_w$ for all $(v, w) \in E$, then U' is an $\mathcal{R}$-submodule of U. In the other direction, suppose U' is an $\mathcal{R}$-submodule of U. Let $U'_v = \mathit{\Pi}_v \circ U'$. Then, $U'_v \subseteq U'$ for all $v \in V$ since U' is an

$\mathcal{R}$-module. On the other hand, $U' \subseteq \oplus_{v\in V} U'_v$, hence $U' = \oplus_{v\in V} U'_v$. Another consequence of U' being an $\mathcal{R}$-module is that

$$\langle \widetilde{\mathcal{L}}_{v,w} \circ U'_v \rangle = \langle \widetilde{\mathcal{L}}_{v,w} \circ \Pi_v \circ U' \rangle \subseteq U'$$

Since every map in $\widetilde{\mathcal{L}}_{v,w}$ maps U to U_w, we have that,

$$\langle \widetilde{\mathcal{L}}_{v,w} \circ U'_v \rangle \subseteq U' \cap U_w = U'_w$$

which is the same as $\langle \mathcal{L}_{v,w} \circ U'_v \rangle \subseteq U'_w$. This completes the proof.

□

This yields the following corollary which characterizes decomposition of U into $\mathcal{R}$-submodules.

Corollary B.1 (Reduction to module decomposition)**.** *$U = U_1 \oplus \cdots \oplus U_s$ is a decomposition of U into $\mathcal{R}$-submodules if and only if each U_i is of the form $\oplus_{v\in V} U_{v,i}$ such that $U_v = U_{v,1} \oplus \cdots \oplus U_{v,s}$ for all $v \in V$ and $\langle \mathcal{L}_{v,w} \circ U_{v,i} \rangle \subseteq U_{w,i}$ for all $i \in [s]$, $(v,w) \in E$.*

Proof. Again one direction is clear. In the other direction, suppose $U = U_1 \oplus \cdots \oplus U_s$ is a decomposition of U into $\mathcal{R}$-submodules. This means that each of the U_i's is an $\mathcal{R}$-submodule of U. By Proposition B.1, each U_i is of the form $\oplus_{v\in V} U_{v,i}$ such that $U_{v,i} \subseteq U_v$ for all $i \in [s], v \in V$ and $\langle \mathcal{L}_{v,w} \circ U_{v,i} \rangle \subseteq U_{w,i}$ for all $i \in [s]$, $(v,w) \in E$. Now $U = U_1 \oplus \cdots \oplus U_s$ and $U_{v,i} \subseteq U_v, U_i$, hence $U_{v,1}, \ldots, U_{v,s}$ form a direct sum and

$$U_{v,1} \oplus \cdots \oplus U_{v,s} \subseteq U_v$$

for all $v \in V$. What remains to prove is that

$$U_v = U_{v,1} \oplus \cdots \oplus U_{v,s}$$

for all $v \in V$. Suppose there is some $w \in V$ such that

$$U_{w,1} \oplus \cdots \oplus U_{w,s} \subsetneq U_w.$$

Then

$$U = \oplus_{i\in[s]} U_i = \oplus_{i\in[s]} \oplus_{v\in V} U_{v,i} = \oplus_{v\in V} \oplus_{i\in[s]} U_{v,i} \subsetneq \oplus_{v\in V} U_v = U$$

which is a contradiction. Hence,

$$U_v = U_{v,1} \oplus \cdots \oplus U_{v,s}$$

for all $v \in V$. This completes the proof. □

The Krull-Schmidt theorem for module decomposition and the above reduction allows one to obtain a uniqueness theorem for generalized vector space decomposition.

Theorem 4 (Generalized vector space decomposition: uniqueness)**.** *Suppose $U_v = U_{v,1} \oplus \cdots \oplus U_{v,s}$ and $U_v = U'_{v,1} \oplus \cdots \oplus U'_{v,s'}$ are two collection of decompositions (which are further indecomposable) for the generalized vector space decomposition problem. Then $s = s'$. Furthermore, there exist linear maps $L_v : U_v \to U_v$ and a permutation $\sigma : [s] \to [s]$ such that $U'_{v,i} = L_v \circ U_{v,\sigma(i)}$ for all $i \in [s]$ and $v \in V$. Also, $L_w \circ L_{v,w} = L_{v,w} \circ L_v$ for all $L_{v,w} \in \mathcal{L}_{v,w}$ and $v, w \in V$.*

Proof. We look at the reduction to module decomposition, the algebra $\mathcal{R}$ discussed above and the vector space $U = \oplus_{v \in V} U_v$. Let us define $U_i = \oplus_{v \in V} U_{v,i}$ and $U'_j = \oplus_{v \in V} U'_{v,j}$. Then $U = U_1 \oplus \cdots \oplus U_s$ and $U = U'_1 \oplus \cdots \oplus U'_{s'}$ are two decompositions of U into indecomposable $\mathcal{R}$-submodules (because of Corollary B.1). Hence by Theorem 3, $s = s'$, and there exist a permutation $\sigma : [s] \to [s]$ and a linear map $L : U \to U$ such that $U'_i = L \circ U_{\sigma(i)}$ and $L \circ R = R \circ L$ for every $R \in \mathcal{R}$.

Now for every $v \in V$, $L \circ \Pi_v = \Pi_v \circ L$ which implies that $L \circ U_v \subseteq U_v$. We can call the restriction of L to U_v as the map $L_v : U_v \to U_v$. Now take an operator $L_{v,w} \in \mathcal{L}_{v,w}$ and $\text{ext}(M) \in \widetilde{\mathcal{L}}_{v,w}$. We have that $L \circ \text{ext}(L_{v,w}) = \text{ext}(L_{v,w}) \circ L$. This implies that $L_w \circ L_{v,w} = L_{v,w} \circ L_v$ for all $L_{v,w} \in \mathcal{L}_{v,w}$. Also,

$$U'_{v,i} = \Pi_v \circ U'_i = \Pi_v \circ L \circ U_{\sigma(i)} = L \circ \Pi_v \circ U_{\sigma(i)} = L \circ U_{v,\sigma(i)}.$$

This completes the proof. □

As a corollary, we get a uniqueness theorem for vector space decomposition.

Corollary B.2 (Vector space decomposition: uniqueness). *Suppose $\mathcal{L}$ is a set of linear maps between vector spaces U and W. Suppose $U = U_1 \oplus \cdots \oplus U_s$, $W = W_1 \oplus \cdots \oplus W_s$ and $U = U'_1 \oplus \cdots \oplus U'_{s'}$, $W = W'_1 \oplus \cdots \oplus W'_{s'}$ are two indecomposable decompositions with respect to $\mathcal{L}$. Then $s = s'$. Furthermore, there exist linear maps $D : U \to U$ and $E : W \to W$ and a permutation $\sigma : [s] \to [s]$ such that $U'_i = D \circ U_{\sigma(i)}$, $W'_i = E \circ W_{\sigma(i)}$ for all $i \in [s]$ and $E \circ L = L \circ D$ for all $L \in \mathcal{L}$.*

We also mention that via the above reduction, we get a polynomial time algorithm for generalized vector space decomposition (over finite fields, reals and complex numbers) using the polynomial time algorithm for module decomposition in [CIK97]. However, we do not use this algorithm for our learning problem since we also want the algorithm to work over rationals which is possible to do in our setting with a simpler specialized algorithm.

C Why doesn't the shifted partials measure work?

In this section, we explain why the shifted partials measure (as it is) is unlikely to satisfy the basic non-degeneracy condition given by Equation (3) in Section 1, if $n \geq d^2$. The shifted partials measure (SP), introduced in [Kay12b], is defined as follows: Let $f \in \mathbb{F}[\mathbf{x}]$ be an n-variate degree-d homogeneous polynomial and $k, \ell \in \mathbb{N}$. Then,

$$\mathsf{SP}_{k,\ell}(f) := \dim \left\langle \mathbf{x}^{\ell} \cdot \partial_{\mathbf{x}}^{k} f \right\rangle.$$

Clearly, $\mathsf{SP}_{k,\ell}(f)$ is upper bounded by $\min\left(\binom{n+k-1}{k} \cdot \binom{n+\ell-1}{\ell}, \binom{n+d-k+\ell-1}{d-k+\ell}\right)$. Suppose

$$f = c_1 Q_1^m + \ldots + c_s Q_s^m,$$

where each $c_i \in \mathbb{F}^{\times}$, Q_i is a homogeneous polynomial of degree t, and $tm = d$. Let $U(f) := \langle \mathbf{x}^{\ell} \cdot \partial_{\mathbf{x}}^{k} f \rangle$. We wish to satisfy the main non-degeneracy condition

$$U(f) = U(Q_1^m) \oplus \ldots \oplus U(Q_s^m), \tag{21}$$

for random $Q_1, \ldots, Q_s$. This imposes the restriction $\ell < d$, as otherwise $U(Q_i^m) \cap U(Q_j^m) \neq \{0\}$ for $i \neq j$. On the other hand, $\mathsf{SP}_{k,\ell}(Q_i^m)$ is upper bounded by $\binom{n+k(t-1)+\ell-1}{k(t-1)+\ell}$. If

$$s \cdot \binom{n+k(t-1)+\ell-1}{k(t-1)+\ell} \leq \min\left(\binom{n+k-1}{k} \cdot \binom{n+\ell-1}{\ell}, \binom{n+d-k+\ell-1}{d-k+\ell}\right),$$

then we may be able to satisfy the direct sum given by Equation (21). For this, we need $kt \leq d$. But, with both ℓ and kt upper bounded by d, $\binom{n+k(t-1)+\ell-1}{k(t-1)+\ell}$ cannot be less that $\binom{n+k-1}{k} \cdot \binom{n+\ell-1}{\ell}$ with growing t, if $n \geq d^2$.

Thus, it seems difficult to satisfy the direct sum condition using the shifted partials measure if $n \geq d^2$. However, if n is much smaller than d then it may be possible to achieve the same. This is what spurred us to think in the direction of reducing the number of variables to below d using affine projections. Indeed, we have shown in this work that such affine projections do work (for both lower bound and learning) even *without shifts by monomials*. But, shifts may play a crucial role if n is much smaller than d to begin with (say, if n is a constant), in which case doing affine projections does not seem to help.

D Proofs from Section 2

Proof of Observation 2.1

As C is non-degenerate, Condition 1 of Definition 1.1 implies,

$$\mathsf{APP}_{k,n_0}(f) = s \cdot \binom{n_0+k(t-1)-1}{k(t-1)}.$$

By Proposition 2.6, $|\mathbb{F}| \gg (d-k) \cdot \binom{n+k}{k}$. Arguing as in Observation 1.1, with probability $1-o(1)$,

$$\mathsf{APP}_{k,n_0}(f) = \dim U = s \cdot \binom{n_0+k(t-1)-1}{k(t-1)},$$

which implies $U = U_1 \oplus \ldots \oplus U_s$ and $\dim U_i = \binom{n_0+k(t-1)-1}{k(t-1)}$ for all $i \in [s]$.

Proof of Observation 2.2

As C is non-degenerate, Condition 3 of Definition 1.1 implies that there exists an L such that

$$\left\langle \mathbf{z}^{2k(t-1)} \cdot \pi_L(Q_1)^e \right\rangle + \ldots + \left\langle \mathbf{z}^{2k(t-1)} \cdot \pi_L(Q_s)^e \right\rangle = \left\langle \mathbf{z}^{2k(t-1)} \cdot \pi_L(Q_1)^e \right\rangle \oplus \ldots \oplus \left\langle \mathbf{z}^{2k(t-1)} \cdot \pi_L(Q_s)^e \right\rangle.$$

For any tuple of n linear forms L, the degree of a polynomial in $\mathbf{z}^{2k(t-1)} \cdot \pi_L(Q_i)^e$ is at most $2d$ as $kt \leq d$ (by Proposition 2.6). If L is a tuple of random linear forms then the polynomials in the set

$$\mathbf{z}^{2k(t-1)} \cdot \pi_L(Q_1)^e \bigcup \ldots \bigcup \mathbf{z}^{2k(t-1)} \cdot \pi_L(Q_s)^e$$

are $\mathbb{F}$-linearly independent with probability $1-o(1)$ if $|\mathbb{F}| \gg 2ds \cdot \binom{n_0+2k(t-1)-1}{2k(t-1)}$, which is ensured by Proposition 2.6.

Proof of Proposition 2.1

Recall that $U_i = \left\langle \mathbf{z}^{k(t-1)} \cdot G_i^e \right\rangle$. If $g \in V = \langle G_1^e, \ldots, G_s^e \rangle$ then $z_1^{k(t-1)} \cdot g$ and $z_2^{k(t-1)} \cdot g$ belong to $U_1 + \ldots + U_s = U$. Hence, there are $a_1, \ldots a_{sr}, b_1, \ldots, b_{sr} \in \mathbb{F}$ such that

$$\begin{aligned} a_1 f_1 + \ldots + a_{sr} f_{sr} &= z_1^{k(t-1)} \cdot g \quad \text{and} \\ b_1 f_1 + \ldots + b_{sr} f_{sr} &= z_2^{k(t-1)} \cdot g. \end{aligned}$$

On the other hand, suppose that there exist $a_1, \ldots a_{sr}, b_1, \ldots, b_{sr} \in \mathbb{F}$ such that

$$\frac{a_1 f_1 + \ldots + a_{sr} f_{sr}}{z_1^{k(t-1)}} = \frac{b_1 f_1 + \ldots + b_{sr} f_{sr}}{z_2^{k(t-1)}}. \tag{22}$$

As $f_1, \ldots, f_{sr}$ is a basis of U, there are polynomials $P_1, \ldots, P_s, P'_1, \ldots, P'_s \in \left\langle \mathbf{z}^{k(t-1)} \right\rangle$ that satisfy

$$\begin{aligned} a_1 f_1 + \ldots + a_{sr} f_{sr} &= P_1 G_1^e + \ldots + P_s G_s^e \quad \text{and} \\ b_1 f_1 + \ldots + b_{sr} f_{sr} &= P'_1 G_1^e + \ldots + P'_s G_s^e. \end{aligned}$$

From Equation (22), we have

$$\left(z_2^{k(t-1)} P_1 - z_1^{k(t-1)} P'_1 \right) \cdot G_1^e + \ldots + \left(z_2^{k(t-1)} P_s - z_1^{k(t-1)} P'_s \right) \cdot G_s^e = 0.$$

As $\left(z_2^{k(t-1)} P_i - z_1^{k(t-1)} P'_i \right) \in \left\langle \mathbf{z}^{2k(t-1)} \right\rangle$, by Observation 2.2, $z_2^{k(t-1)} P_i - z_1^{k(t-1)} P'_i = 0$ for all $i \in [s]$. Hence, $z_1^{k(t-1)}$ divides P_i and $z_2^{k(t-1)}$ divides P'_i for all $i \in [s]$. But, $\deg(P_i) = \deg(P'_i) = k(t-1)$. Therefore, there are $\hat{a}_1, \ldots, \hat{a}_s \in \mathbb{F}$ such that

$$\frac{a_1 f_1 + \ldots + a_{sr} f_{sr}}{z_1^{k(t-1)}} = \frac{b_1 f_1 + \ldots + b_{sr} f_{sr}}{z_2^{k(t-1)}} = \hat{a}_1 G_1^e + \ldots + \hat{a}_s G_s^e =: g(\mathbf{z}) \in V.$$

Proof of Proposition 2.2

As C is non-degenerate, Condition 2 of Definition 1.1 implies that there exist L and P such that

$$\begin{aligned} \left\langle \pi_P(\partial_{\mathbf{z}}^k \, (G_1^e + \ldots + G_s^e)) \right\rangle &= \left\langle \pi_P(\partial_{\mathbf{z}}^k \, G_1^e) \right\rangle \oplus \ldots \oplus \left\langle \pi_P(\partial_{\mathbf{z}}^k \, G_s^e) \right\rangle, \quad \text{and} \\ \dim \left\langle \pi_P(\partial_{\mathbf{z}}^k \, G_i^e) \right\rangle &= \binom{m_0 + k(t-1) - 1}{k(t-1)} \quad \text{for all} \quad i \in [s], \end{aligned} \tag{23}$$

where $G_i = \pi_L(Q_i)$ and $e = m - k$. If L and P are tuples of random linear forms (as in Step 1 and 4 of Algorithm 2) then the above equation holds with probability $1 - o(1)$ provided $|\mathbb{F}| \gg 2ds \cdot \binom{m_0 + k(t-1) - 1}{k(t-1)}$ (which is ensured by Proposition 2.6). Let $g_0 = G_1^e + \ldots + G_s^e$. By Equation (23),

$$\left\langle \pi_P(\partial_{\mathbf{z}}^k \, g_0) \right\rangle = W_1 \oplus \ldots \oplus W_s \quad \text{and} \quad \dim W_i = \binom{m_0 + k(t-1) - 1}{k(t-1)} \quad \text{for all} \quad i \in [s], \tag{24}$$

where $W_i = \langle \pi_P(\partial_{\mathbf{z}}^k \, G_i^e) \rangle$. Recall that $W = \langle \pi_P(\partial_{\mathbf{z}}^k \, g) \rangle$, where g is a random element of V. As $G_1^e, \ldots, G_s^e$ is a basis of V, we have $g = b_1 G_1^e + \ldots + b_s G_s^e$, where $b_1, \ldots, b_s \in_r \mathbb{F}$. The next claim completes the proof of the proposition.

Claim D.1. *If $g = b_1 G_1^e + \ldots + b_s G_s^e$ such that $b_1, \ldots, b_s \in_r \mathbb{F}^\times$, then $\left\langle \pi_P(\partial_{\mathbf{z}}^k \, g)\right\rangle = \left\langle \pi_P(\partial_{\mathbf{z}}^k \, g_0)\right\rangle$ with probability* $1 - o(1)$.

Proof. With every polynomial $\hat{g} \in V$, associate a $\binom{n_0+k-1}{k} \times \binom{m_0+et-k-1}{et-k}$ matrix $M(\hat{g})$ as follows: The rows of $M(\hat{g})$ are indexed by all monomials in $\mathbf{z}$-variables of degree k and the columns are indexed by all monomials in $\mathbf{w}$-variables of degree $et-k$. If α is a $\mathbf{z}$-monomial of degree k and β is a $\mathbf{w}$-monomial of degree $(et-k)$ then the (α, β)-th entry of $M(\hat{g})$ is the coefficient of β in $\pi_P\left(\frac{\partial^k \hat{g}}{\partial \alpha}\right)$. In other words, $M(\hat{g})$ is the *coefficient matrix* consisting of the coefficients of the polynomials in $\pi_P(\partial_{\mathbf{z}}^k \, \hat{g})$. Let $q = \binom{m_0+k(t-1)-1}{k(t-1)}$. Clearly, for every $\hat{g} \in V$,

$$\begin{aligned} \left\langle \pi_P(\partial_{\mathbf{z}}^k \, \hat{g})\right\rangle &\subseteq W_1 \oplus \ldots \oplus W_s = \left\langle \pi_P(\partial_{\mathbf{z}}^k \, g_0)\right\rangle, \quad \text{by Equation (24)} \\ \Rightarrow \text{rank}\,[M(\hat{g})] &\leq \text{rank}\,[M(g_0)] = s \cdot \binom{m_0+k(t-1)-1}{k(t-1)} = sq. \end{aligned}$$

There exist a $sq \times \binom{n_0+k-1}{k}$ matrix R and a $\binom{m_0+et-k-1}{et-k} \times sq$ matrix C such that

$$\text{rank}\,[R \cdot M(g_0) \cdot C] = \text{rank}\,[M(g_0)] = sq. \tag{25}$$

For any $\hat{g} \in V$, denote the $sq \times sq$ matrix $R \cdot M(\hat{g}) \cdot C$ by $N(\hat{g})$ and $R \cdot M(g_0) \cdot C$ by $N(g_0)$. Let $\hat{g} = y_1 G_1^e + \ldots + y_s G_s^e$ be an arbitrary element of V, where $y_1, \ldots, y_s \in \mathbb{F}$. Then

$$\begin{aligned} M(\hat{g}) &= y_1 \cdot M(G_1^e) + \ldots + y_s \cdot M(G_s^e) \\ \Rightarrow N(\hat{g}) &= y_1 \cdot N(G_1^e) + \ldots + y_s \cdot N(G_s^e). \end{aligned}$$

Treating $y_1, \ldots, y_s$ as formal variables, we can infer that $\det(N(\hat{g}))$ is a non-zero polynomial in $y_1, \ldots, y_s$ of degree at most sq. This is because, by setting $y_1 = \ldots = y_s = 1$ we get $\hat{g} = g_0$, and we already know that $\det(N(g_0)) \neq 0$ from Equation (25). Thus, if $g = b_1 G_1^e + \ldots + b_s G_s^e$ such that $b_1, \ldots, b_s \in_r \mathbb{F}^\times$, then with probability $1 - o(1)$ we have $\det(N(g)) \neq 0$ as $|\mathbb{F}| \gg sq$ (by Proposition 2.6). That is, rank $[N(g)] = sq =$ rank $[M(g)]$ which implies $\langle \pi_P(\partial_{\mathbf{z}}^k \, g)\rangle = \langle \pi_P(\partial_{\mathbf{z}}^k \, g_0)\rangle$. ☐

Proof of Proposition 2.3

Treat $\mathbf{w}^{k(t-1)}$ as an ordered set and let $B \in \text{GL}_{sq}(\mathbb{F})$ be the basis change matrix from $(h_1, \ldots, h_{sq})$ to $(\mathbf{w}^{k(t-1)} \cdot \pi_P(G_1^{e-k}), \ldots, \mathbf{w}^{k(t-1)} \cdot \pi_P(G_s^{e-k}))$. Let K be an arbitrary element of $\langle \mathcal{L}_2 \rangle$. As $V = V_1 \oplus \ldots \oplus V_s$, $W = W_1 \oplus \ldots \oplus W_s$ is an indecomposable decomposition of V and W under the action of $\mathcal{L}_2$, the matrix BKA^{-1} has the following structure: The columns of BKA^{-1} are indexed by $(G_1^e, \ldots, G_s^e)$ and the rows are indexed by $(\mathbf{w}^{k(t-1)} \cdot \pi_P(G_1^{e-k}), \ldots, \mathbf{w}^{k(t-1)} \cdot \pi_P(G_s^{e-k}))$. The G_j^e-th column of BKA^{-1} has its non-zero entries confined to the q rows indexed by $\mathbf{w}^{k(t-1)} \cdot \pi_P(G_j^{e-k})$.

By definition of the adjoint, $(D, E) \in \text{adj}(\mathcal{L}_2)$ if and only if

$$BKA^{-1} \cdot ADA^{-1} \;=\; BEB^{-1} \cdot BKA^{-1} \quad \text{ for all } K \in \langle \mathcal{L}_2 \rangle. \tag{26}$$

Expressed in the basis $(G_1^e, \ldots, G_s^e)$ of V, the element $g_0 = G_1^e + \ldots + G_s^e$ is the all-one vector $\mathbf{1} \in \mathbb{F}^s$. Let $\beta \in \mathbf{w}^{k(t-1)}$ and $j \in [s]$ be arbitrarily chosen. From the proof of Proposition 2.2, it follows that there is a $K \in \langle \mathcal{L}_2 \rangle$ such that $BKA^{-1} \cdot \mathbf{1}$ is the unit vector whose $(\beta \cdot \pi_P(G_j^{e-k}))$-th entry is one and all other entries are zero. In other words, all but the $(G_j^e,\ \beta \cdot \pi_P(G_j^{e-k}))$-th entry of BKA^{-1} is zero, and the $(G_j^e,\ \beta \cdot \pi_P(G_j^{e-k}))$-th entry is 1. As ADA^{-1} and BEB^{-1} satisfy Equation (26) for every such K (as we vary $\beta \in \mathbf{w}^{k(t-1)}$ and $j \in [s]$), both ADA^{-1} and BEB^{-1} are diagonal matrices, i.e., $A \cdot \mathrm{adj}(\mathcal{L}_2)_1 \cdot A^{-1} \subseteq \mathcal{D}$.

Using Equation (26), it is an easy exercise to show that $\mathcal{D} \subseteq A \cdot \mathrm{adj}(\mathcal{L}_2)_1 \cdot A^{-1}$. Therefore, $A \cdot \mathrm{adj}(\mathcal{L}_2)_1 \cdot A^{-1} = \mathcal{D}$.

Proof of Proposition 2.4

Clearly, $U \subseteq U_1 + \ldots + U_s$. We will prove that $U_i \subset U$ for every $i \in [s]$. Observe that

$$U_i = \left\langle \pi_L(\partial_{\mathbf{x}}^k\, Q_i^m) \right\rangle \subseteq \left\langle \mathbf{z}^{k(t-1)} \cdot \pi_L(Q_i)^{m-k} \right\rangle.$$

We will now show that $\left\langle \mathbf{z}^{k(t-1)} \cdot \pi_L(Q_i)^{m-k} \right\rangle \subset U$. Let μ be an arbitrary $\mathbf{z}$-monomial of degree $k(t-1)$. Then

$$\mu = \gamma_{l_1}^{k_1} \cdot \gamma_{l_2}^{k_2} \cdots \gamma_{l_r}^{k_r}$$

for some distinct $l_1, \ldots, l_r \in [b]$, where $k_1 + \ldots + k_r = k$. Let $\alpha_i := (y_{i1l_1} \cdot y_{12l_1} \cdots y_{ik_1l_1}) \cdot (y_{i1l_2} \cdot y_{12l_2} \cdots y_{ik_2l_2}) \cdots (y_{i1l_r} \cdot y_{12l_r} \cdots y_{ik_rl_r})$. Using the combinatorial design of the sets $S_1, \ldots S_s$, we get

$$\frac{\partial^k f}{\partial \alpha_i} = \frac{\partial^k Q_i^m}{\partial \alpha_i} = k! \cdot \binom{m}{k} \cdot \mu \cdot Q_i^{m-k}.$$

As μ is arbitrary and $\mathrm{char}(\mathbb{F}) \nmid k! \cdot \binom{m}{k}$, we have $\left\langle \mathbf{z}^{k(t-1)} \cdot \pi_L(Q_i)^{m-k} \right\rangle \subset U$. From the above equation, it is also easy to notice that $U_i = \left\langle \mathbf{z}^{k(t-1)} \cdot \pi_L(Q_i)^{m-k} \right\rangle$.

Proof of Proposition 2.5

We will show that, for every $i \in [s]$, there is a monomial in $P_i \cdot G_i^{m-k}$ that cannot be generated by any other $P_j \cdot G_j^{m-k}$ for $i \neq j$. The following observation will be useful.

Observation D.1. *Consider a product $P \cdot \ell^{\hat{d}}$, where P is a non-zero polynomial in $\mathbb{F}[\mathbf{z}]$ and $\ell = \sum_{z \in \hat{\mathbf{z}}} z$ for some $\hat{\mathbf{z}} \subseteq \mathbf{z}$. Let $\mathrm{char}(\mathbb{F}) > \deg_{\mathbf{z}}(P \cdot \ell^{\hat{d}})$. Then, for every monomial $\mu \in \hat{\mathbf{z}}^{\hat{d}}$, there is a monomial β (with non-zero coefficient) in $P \cdot \ell^{\hat{d}}$ such that μ divides β.*

Proof. Let μ be a monomial in $\hat{\mathbf{z}}^{\hat{d}}$. Write the product $P \cdot \ell^{\hat{d}}$ as $P' \cdot \ell^{d'}$, where P' is coprime to ℓ and $d' \geq \hat{d}$. For contradiction, suppose that there is no monomial in $P' \cdot \ell^{d'}$ that is divisible by μ. Then,

$$\begin{aligned} \frac{\partial^{\hat{d}}}{d\mu}(P' \cdot \ell^{d'}) &= 0 \\ \Rightarrow \frac{d'!}{(d'-\hat{d})!} \cdot P' \cdot \ell^{d'-\hat{d}} + g \cdot \ell^{d'-\hat{d}+1} &= 0, \qquad \text{for some } g \in \mathbb{F}[\mathbf{z}] \quad \text{(by chain rule)} \\ \Rightarrow \frac{d'!}{(d'-\hat{d})!} \cdot P' + g \cdot \ell &= 0. \end{aligned}$$

This gives a contradiction as $\text{char}(\mathbb{F}) > \deg_{\mathbf{z}}(P \cdot \ell^{\hat{d}})$ and P' is non-zero and not divisible by ℓ. $\square$

Let $\mathbf{z}_i = \{z_{i,1}, \ldots, z_{i,p}\}$. We do the analysis for the two cases $t(m-k) \le \sqrt{n_0}$ and $t(m-k) \ge \sqrt{n_0}$.

Suppose $t(m-k) \le \sqrt{n_0}$ so that $|\mathbf{z}_i \cap \mathbf{z}_j| \le \left\lfloor \frac{t(m-k)}{10} \right\rfloor$ for $i \neq j$. By Observation D.1, there is a monomial β_i in $P_i \cdot \left(z_{i,1} + \ldots + z_{i,p}\right)^{t(m-k)}$ that is divisible by $z_{i,1} \cdot z_{i,2} \cdots z_{i,\lfloor \frac{t(m-k)}{2} \rfloor}$ as $p = \lfloor \frac{\sqrt{n_0}}{2} \rfloor \ge \lfloor \frac{t(m-k)}{2} \rfloor$. If β_i is generated by some other term $P_j \cdot \left(z_{j,1} + \ldots + z_{j,p}\right)^{t(m-k)}$ then there is a monomial in P_j that is divisible by at least $\lfloor \frac{t(m-k)}{2} \rfloor - \lfloor \frac{t(m-k)}{10} \rfloor$ distinct $\mathbf{z}$-variables, as $|\mathbf{z}_i \cap \mathbf{z}_j| \le \left\lfloor \frac{t(m-k)}{10} \right\rfloor$. But this is not possible as $2k(t-1) < \lfloor \frac{t(m-k)}{2} \rfloor - \lfloor \frac{t(m-k)}{10} \rfloor$ for $m > 7k$.

Suppose $t(m-k) \ge \sqrt{n_0}$, in which case $\frac{t(m-k)}{p} \ge 2$. By Observation D.1, there is a monomial β_i in $P_i \cdot \left(z_{i,1} + \ldots + z_{i,p}\right)^{t(m-k)}$ that is divisible by

$$z_{i,1}^{\lfloor \frac{t(m-k)}{p} \rfloor} \cdot z_{i,2}^{\lfloor \frac{t(m-k)}{p} \rfloor} \cdots z_{i,p}^{\lfloor \frac{t(m-k)}{p} \rfloor}.$$

If β_i is generated by some other term $P_j \cdot \left(z_{j,1} + \ldots + z_{j,p}\right)^{t(m-k)}$ then there is a monomial in P_j that is divisible by at least $p - \frac{p+1}{5} = \frac{4p-1}{5}$ distinct $\mathbf{z}$-variables each with multiplicity $\lfloor \frac{t(m-k)}{p} \rfloor$ (as $|\mathbf{z}_i \cap \mathbf{z}_j| \le \frac{p+1}{5}$). But this is not possible as $2k(t-1) < \frac{4p-1}{5} \cdot \lfloor \frac{t(m-k)}{p} \rfloor$ for $m > 7k$.

E Proofs from Section 4

Proof of Observation 4.1

As $k = \lfloor \delta \cdot \frac{d}{t} \rfloor$ and $t \le \frac{\delta \cdot d}{\ln d}$, we have $k \ge \lfloor \ln d \rfloor$. By definition,

$$c = \frac{3}{4} \cdot \frac{\ln \frac{n}{k}}{\ln \frac{d}{k}} = \frac{3}{4} \cdot \left[\frac{\ln \frac{n}{d}}{\ln \frac{d}{k}} + 1 \right] \ge \frac{3}{4} \cdot 2 = \frac{3}{2} \quad (\text{as } n \ge d^2).$$

By choice, $n_0 = \lfloor c \cdot k \rfloor$. So,

$$\begin{aligned} n_0 \le ck &\le \frac{3}{4} \cdot \left[\frac{\ln \frac{n}{d}}{\ln \frac{t}{\delta}} + 1 \right] \cdot \frac{\delta d}{t} && (\text{as } k \le \frac{\delta d}{t}) \\ &\le \frac{3}{4} \cdot \left[\frac{\ln \frac{n}{d}}{\ln \frac{\ln \frac{n}{d}}{\delta}} + 1 \right] \cdot \frac{\delta d}{\ln \frac{n}{d}} && (\text{as } \ln \frac{n}{d} \le t) \\ &= \frac{3}{4} \cdot \left[\frac{1}{\ln \ln \frac{n}{d} + \ln \frac{1}{\delta}} + \frac{1}{\ln \frac{n}{d}} \right] \cdot \delta d \\ &\le \frac{3}{4} \cdot \frac{2}{\ln \ln \frac{n}{d}} \cdot \delta d \le \frac{d}{\ln \ln d}. \end{aligned}$$

Proof of Observation 4.2

Recall from Equation (6),

$$\mathsf{APP}_{k,n_0}(f) = \max_{L} \dim \left\langle \pi_L(\partial_{\mathbf{x}}^k f) \right\rangle,$$

where $L = (\ell_1(\mathbf{z}), \ldots, \ell_n(\mathbf{z}))$ is an n-tuple of linear forms in $\mathbb{F}[\mathbf{z}]$ and $|\mathbf{z}| = n_0$. An element of $\pi_L(\partial_{\mathbf{x}}^k f)$ is a homogeneous polynomial of degree $d-k$ in $\mathbf{z}$-variables, such a polynomial can have at most $\binom{d-k+n_0-1}{n_0-1}$ many $\mathbf{z}$-monomials. Hence, $\mathsf{APP}_{k,n_0}(f) \le \binom{d-k+n_0-1}{n_0-1}$.

Proof of Proposition 4.1

For $i \in [s]$, let $T_i = Q_{i1}Q_{i2}\cdots Q_{im_i}$ be a term of the formula C given by Equation (17), where degree of every Q_{ij} is in $[t, 2t]$. Observe that for any n-tuple of linear forms $L = (\ell_1(\mathbf{z}), \ldots, \ell_n(\mathbf{z}))$,

$$\left\langle \pi_L(\partial_{\mathbf{x}}^k T_i) \right\rangle \subseteq \left\langle \mathbf{z}^{\le 2tk} \cdot \bigcup_{S \in \binom{[m_i]}{k}} \left\{ \prod_{j \in [m_i] \setminus S} Q_{ij} \right\} \right\rangle.$$

Hence, $\mathsf{APP}_{k,n_0}(T_i) \le \binom{m}{k} \cdot \binom{n_0+2kt}{n_0}$ as $m_i \le m$. By subadditivity of the APP measure, we have

$$\mathsf{APP}_{k,n_0}(T_1 + \ldots + T_s) \le s \cdot \binom{m}{k} \cdot \binom{n_0 + 2kt}{n_0}.$$

Proof of Proposition 4.2

Suppose $C = f_{n,d,t}$ in Equation (17). Then, by Proposition 4.1, $\mathsf{APP}_{k,n_0}(f_{n,d,t}) \le s \cdot \binom{m}{k} \cdot \binom{n_0+2kt}{n_0}$. On the other hand, by Proposition 4.5, $\mathsf{APP}_{k,n_0}(f_{n,d,t}) = \binom{d-k+n_0-1}{n_0-1}$. Therefore,

$$
\begin{aligned}
s \ge \frac{\binom{d-k+n_0-1}{n_0-1}}{\binom{m}{k}\cdot\binom{n_0+2kt}{n_0}} &= \frac{n_0}{d-k+n_0}\cdot\frac{\binom{d-k+n_0}{n_0}}{\binom{m}{k}\cdot\binom{n_0+2kt}{n_0}} \\
&\ge \frac{k}{d}\cdot\frac{\binom{d}{n_0}}{\binom{m}{k}\cdot\binom{n_0+2kt}{n_0}}, && \text{(as } n_0 \ge k\text{)} \\
&\ge \frac{k}{d}\cdot\frac{\left(\frac{d}{n_0}\right)^{n_0}}{\left(\frac{em}{k}\right)^k\cdot\left(\frac{e\cdot(n_0+2kt)}{n_0}\right)^{n_0}} \\
&= \frac{k}{d}\cdot\frac{1}{\left(\frac{em}{k}\right)^k}\cdot\left[\frac{d}{e\cdot(n_0+2kt)}\right]^{n_0} \\
&\ge \frac{k}{d}\cdot\frac{1}{\left(\frac{em}{k}\right)^k}\cdot\left[\frac{d}{4ekt}\right]^{n_0}, && \text{(as } n_0 \le 2kt\text{)} \\
&\ge \frac{k}{d}\cdot\frac{1}{(4.01\cdot e^{11})^k}\cdot e^{9n_0}, && \text{(plugging in the values of } k, m \text{ and } \delta\text{)} \\
&\ge \frac{k}{de^9}\cdot\frac{1}{e^{12.4\cdot k}}\cdot e^{9\cdot\frac{3}{4}\cdot\left[\frac{\ln\frac{n}{d}}{\ln\frac{d}{k}}+1\right]\cdot k} && \text{(plugging in the values of } n_0, c\text{, putting } 4.01 < e^{1.4}\text{)} \\
&\ge \frac{k}{de^9}\cdot\frac{1}{e^{5.65\cdot k}}\cdot e^{6.75\cdot\left[\frac{\ln\frac{n}{d}}{\ln\frac{d}{k}}\right]\cdot k} \\
&\ge \frac{k}{de^9}\cdot e^{1.1\cdot\left[\frac{\ln\frac{n}{d}}{\ln\frac{d}{k}}\right]\cdot k} && \text{(as } \tfrac{n}{d} \ge \tfrac{d}{k}\text{)} \\
&\ge \frac{1}{e^9}\cdot e^{0.1\cdot\left[\frac{\ln\frac{n}{d}}{\ln\frac{d}{k}}\right]\cdot k} && \text{(as } \tfrac{\ln\frac{n}{d}}{\ln\frac{d}{k}}\cdot k \ge \ln\tfrac{d}{k}\text{)} \\
&= \left(\frac{n}{d}\right)^{\Omega\left(\frac{d}{t\ln t}\right)} && \text{(as } \tfrac{d}{k} = \Theta(t)\text{).}
\end{aligned}
$$

Proof of Proposition 4.3

As in the proof of Proposition 4.2, we have

$$\begin{aligned}
s \geq \frac{\binom{d-k+n_0-1}{n_0-1}}{\binom{m}{k}\cdot\binom{n_0+2kt}{n_0}} &= \frac{n_0}{d-k+n_0}\cdot\frac{\binom{d-k+n_0}{n_0}}{\binom{m}{k}\cdot\binom{n_0+2kt}{n_0}} \\
&\geq \frac{1}{2}\cdot\frac{\binom{n_0+d-k}{n_0}}{2^{O(\frac{d}{t})}\cdot\binom{n_0+2kt}{n_0}}, && \text{(as } n_0 \geq d,\ m = \Theta(d/t) \text{ and } k = \Theta(d/t)\text{)} \\
&= \frac{1}{2^{O(\frac{d}{t})}}\cdot\left(1+\frac{n_0}{d-k}\right)\cdots\left(1+\frac{n_0}{d-k-(d-k-2kt-1)}\right) \\
&\geq \frac{1}{2^{O(\frac{d}{t})}}\cdot\left(\frac{n_0}{d}\right)^{d-k-2kt} \\
&\geq \frac{1}{2^{O(\frac{d}{t})}}\cdot n^{\frac{k}{2d}\cdot(d-k-2kt)} && \text{(as } \sqrt{n_0} \geq n^{\frac{k}{2d}} \geq d\text{)} \\
&= n^{\Omega(\frac{d}{t})} && \text{(plugging in the value of } k\text{).}
\end{aligned}$$

Proof of Proposition 4.4

High t case. In this case $n_0 = \lfloor c\cdot k\rfloor$, where $c = \frac{3}{4}\cdot\frac{\ln\frac{n}{k}}{\ln\frac{d}{k}}$. On one hand,

$$\begin{aligned}
\binom{d-k+n_0-1}{n_0-1} \leq \binom{d+n_0}{n_0} &\leq \left(e\cdot\left(\frac{d}{n_0}+1\right)\right)^{n_0} \\
&\leq \left(e\cdot\left(\frac{d}{ck}+1\right)\right)^{ck} && \text{(as } n_0 \leq ck\text{)} \\
&\leq \left(e\cdot\frac{d}{k}\right)^{ck} && \text{(as } c \geq \frac{3}{2}\text{)} \\
&= e^{ck}\cdot\left(\frac{n}{k}\right)^{0.75\cdot k} && \text{(plugging in the value of } c\text{)} \\
&\leq \left(\frac{n}{k}\right)^{0.76\cdot k} && \text{(as } e^{ck} \ll \left(\frac{n}{k}\right)^{0.01\cdot k}\text{).}
\end{aligned}$$

On the other hand,

$$\begin{aligned}
\binom{n_1}{k} = \binom{n-n_0(d-k)}{k} &\geq \left(\frac{n-n_0(d-k)}{k}\right)^k \\
&\geq \left(\frac{n}{k}-c(d-k)\right)^k && \text{(as } n_0 \leq ck\text{)} \\
&\geq \left(\frac{n}{k}\right)^{0.76\cdot k} && \text{(as } cd \ll \frac{n}{k}\text{).}
\end{aligned}$$

Low t case. In this case $n_0 = \lceil n^{\frac{k}{d}} \rceil$. Verify that $n_0 \geq d$.

$$\begin{aligned}
\binom{d-k+n_0-1}{n_0-1} \leq \binom{d+n_0}{n_0} &\leq \left(e \cdot \frac{n_0+d}{d}\right)^d \\
&\leq \left(\frac{2e \cdot n_0}{d}\right)^d \quad \text{(as } n_0 \geq d\text{)} \\
&\leq \left(\frac{n_1}{k}\right)^k \qquad \text{(verify after putting the values of } n_0 \text{ and } n_1\text{)} \\
&\leq \binom{n_1}{k}
\end{aligned}$$

Proof of Proposition 4.5

Observe that $\partial_{\mathbf{y}}^k f_{n,d,t} = B$ and $\pi(B) = \mathbf{z}^{d-k}$. So, $\mathsf{APP}_{k,n_0}(f_{n,d,t}) \geq \binom{d-k+n_0-1}{n_0-1}$. On the other hand, by Observation 4.2, $\mathsf{APP}_{k,n_0}(f_{n,d,t}) \leq \binom{d-k+n_0-1}{n_0-1}$. Hence, we get the equality.