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

Ankit Garg
Microsoft Research
Bengaluru, India
Email: garga@microsoft.com

Neeraj Kayal
Microsoft Research
Bengaluru, India
Email: neeraka@microsoft.com

Chandan Saha
Indian Institute of Science
Bengaluru, India
Email: chandan@iisc.ac.in

***Abstract*—We develop algorithms for writing a polynomial as sums of powers of low degree polynomials in the non-degenerate case. This problem generalizes symmetric tensor decomposition which is widely studied, having many applications in machine learning. Our algorithm for this more general problem 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 of Kayal and Saha. Here, we make it more general and detailed, and potentially applicable to learning other circuit models.

An exponential lower bound on the representation above 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 such 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 circuit of the above kind. The APP measure could be of independent interest for proving other lower bounds.

***Keywords*-Arithmetic circuits; Reconstruction; Mixtures of Gaussians;**

I. 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 [2]–[4] on this topic.

[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 [1].

[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.

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00087

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) [5], [6]. Furthermore, non-trivial (deterministic) PIT algorithms also imply lower bounds [5], [7], [8]. While these concrete connections are not always present for restricted circuit models, several PIT algorithms have been inspired by corresponding lower bounds, e.g., [9]–[12].
- **Lower bounds and reconstruction.** It is known that worst case reconstruction of a circuit model implies lower bound for the same model [13], [14]. In the other direction, several reconstruction algorithms are inspired by lower bounds for the corresponding models [10], [15]–[20].
- **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 [2], 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 [10], [21]–[23]. Of course, deterministic PIT algorithms can also be sometimes used to get deterministic reconstruction algorithms when randomized ones are known [10], [15].

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 [20] on building a new connection between lower bounds and reconstruction. We build on the work of [20] 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 I-A). 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. 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) [24], [25]. Regardless of the NP-hardness, one can design algorithms for this model under reasonable assumptions and such algorithms are widely used in machine learning. One such algorithm, attributed to Jennrich [26], [27], 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,

[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 [20], this framework is present in a rudimentary form.

[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 [20], [28]–[30]. Other algorithms include [31], [32].

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 the full version of the paper.

Let us briefly describe the roadmap for the rest of this section now. In Section I-A, we describe our main results about learning sums of powers of low degree polynomials in the non-degenerate case. In Section I-B, 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.

A. 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

$$f = c_1 Q_1^m + \ldots + c_s Q_s^m, \tag{1}$$

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.

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 the full version of the paper. 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 I-B4.
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 the full version 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 I-B3) and is relevant to the problem of mixtures of Gaussians.
4) *Uniqueness of Q_i's.* A corollary of the analysis of our algorithm is that for a non-degenerate $f =$

[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.

[13] Univariate polynomials over finite fields can be factored in randomized polynomial time [33], and over $\mathbb{Q}$, they can be factored in deterministic polynomial time [34].

[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.

$c_1Q_1^m + \ldots + c_sQ_s^m$ (which holds if the Q_i's are chosen randomly), this representation is of the smallest size and also unique. That is, if $f = \widetilde{c}_1\widetilde{Q}_1^m + \ldots + \widetilde{c}_{\widetilde{s}}\widetilde{Q}_{\widetilde{s}}^m$ 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 $\widetilde{Q}_i = d_iQ_{\pi(i)}$ 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 I.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 polynomial $f = c_1Q_1^m + \ldots + c_sQ_s^m$, 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* $\mathrm{poly}((ns)^t)$.

B. 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 [35], [36]. The influence of lower bound techniques on learning is also apparent in the case of ROABP reconstruction [15], [37]. 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 I-B1, we will see how lower bounds are typically proven in arithmetic complexity. Section I-B2 describes our meta framework of turning lower bounds into learning algorithms. In Section I-B3, we discuss how implement the framework for learning sums of powers of low degree polynomials. Finally in Section I-B4, we state the non-degeneracy conditions we require explicitly.

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

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

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 $\mathcal{C}$.

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.,

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

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 [41], [42].
- 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 I.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

[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 [38]–[40].

[16] Here, $\langle S \rangle$ denotes the $\mathbb{F}$-linear span of a set of vectors S. Also $\mathcal{L} \circ T$ denotes the set of vectors obtained by applying each linear map in $\mathcal{L}$ to T.

[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., $\partial_{\mathbf{x}}^k = \partial_{\mathbf{x}}^{k-1} \circ \partial_{\mathbf{x}}$.

(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

$$\begin{aligned} U &:= \langle \mathcal{L}_1 \circ f \rangle = \langle \mathcal{L}_1 \circ T_1 \rangle \oplus \ldots \oplus \langle \mathcal{L}_1 \circ T_s \rangle \quad \text{and} \\ V &:= \langle \mathcal{L}_2 \circ \mathcal{L}_1 \circ f \rangle \qquad (4) \\ &= \langle \mathcal{L}_2 \circ (\mathcal{L}_1 \circ T_1) \rangle \oplus \ldots \oplus \langle \mathcal{L}_2 \circ (\mathcal{L}_1 \circ T_s) \rangle. \qquad (5) \end{aligned}$$

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 $\mathcal{L}$.

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 [43] based on the algorithms developed for decomposition of algebras (e.g., see [44]–[46]). 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 the full version 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 [43] 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 (5). 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, see full version) and it extends to vector space decomposition via our reduction (see full version). 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,

$$\begin{aligned} U &:= \langle \mathcal{L}_1 \circ f \rangle = \langle \mathcal{L}_1 \circ T_1 \rangle \oplus \ldots \oplus \langle \mathcal{L}_1 \circ T_s \rangle \quad \text{and} \\ V &:= \langle \mathcal{L}_2 \circ \mathcal{L}_1 \circ f \rangle \qquad (6) \\ &= \langle \mathcal{L}_2 \circ (\mathcal{L}_1 \circ T_1) \rangle \oplus \ldots \oplus \langle \mathcal{L}_2 \circ (\mathcal{L}_1 \circ T_s) \rangle. \qquad (7) \end{aligned}$$

2) (7) 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 $U_i = \langle \mathcal{L}_1 \circ T'_i \rangle$).

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 is the lower bound. So, at first we need to understand how lower bounds are proven for this model [47]–[49]. Let us first consider the setting of sums of powers of linear forms[21], i.e., $t = 1$ [50], [51]. 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

[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.
[20] The symmetric version is known as *module decomposition* in the literature.
[21] Also known in the literature as symmetric tensor decomposition or the Waring rank problem.

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 $\mathcal{L} = \partial^k_{\mathbf{x}}$, 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 $\alpha \in \mathbb{Z}^n_{\geq}$ with $|\alpha| := \sum_{i=1}^n \alpha_i = k$, then Q^{m-k} divides $\partial^\alpha_{\mathbf{x}} Q^m$ for $k \leq m$. Hence, any $\partial^\alpha_{\mathbf{x}} Q^m$ is of the form $Q^{m-k}R$, where R is homogeneous of degree $k(t-1)$. Now, the main observation in [47] 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,

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

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 $\beta = \ell$, $\mathbf{x}^\beta \partial^\alpha_{\mathbf{x}} Q^m$ 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 [47] and other more general models as well [48], [49], [52]–[55]. 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 (7) (see full version) 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]

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 $\partial^k_{\mathbf{x}} f$ 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,

$$\text{(The measure)} \quad \mathsf{APP}_{k,n_0}(f) := \max_L \dim \left\langle \pi_L(\partial^k_{\mathbf{x}} f) \right\rangle, \tag{8}$$

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,

$$\text{(Linearity of the measure)} \tag{9}$$

$$\mathsf{APP}_{k,n_0}(f+g) \leq \mathsf{APP}_{k,n_0}(f) + \mathsf{APP}_{k,n_0}(g). \tag{10}$$

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 [41], [42].

Observation I.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,*

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

Remark 1. Similarity with the skewed partials measure. *The* APP *measure is akin to the skewed partials* (SkP) *measure introduced in [56] – SkP is a special case*

[22] If we use this lower bound with the recipe in Section I-B2, 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.

[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.

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 $\partial^{\alpha}_{\mathbf{x}} Q^m$, 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.

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 the full version) 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 [49], [53]. 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 (7). 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 I.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 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:

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

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 $(D, E) \in \mathrm{adj}(\mathcal{L}_2)$, 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 (see full version). We show this is the case in our setting. We deviate from the framework in Section I-B2 a little bit to simplify the analysis. Due to this, the vector spaces in our case turn out to be $V = \left\langle \pi_L(Q_1)^{m-k}, \ldots, \pi_L(Q_s)^{m-k} \right\rangle$ and $W = W_1 \oplus \ldots \oplus W_s$, $W_i := \left\langle \pi_P(\partial^k_{\mathbf{z}}\ \pi_L(Q_i)^{m-k}) \right\rangle$, and $\mathcal{L}_2 = \pi_P(\partial^k_{\mathbf{z}}\ )$ are linear maps from V to W. Here, L is a random projection onto n_0 variables $\mathbf{z}$, and P is a random projection onto m_0 variables $\mathbf{w}$.

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

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 full version.

Next, we explicitly state the non-degeneracy conditions we require. The details of our algorithm and analysis can be found in the full version. 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.

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

$$f = c_1 Q_1^m + \ldots + c_s Q_s^m. \tag{11}$$

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 $k = \left\lceil \frac{130 \cdot t \cdot \log s}{\log n} \right\rceil$:

- $U := \left\langle \pi_L(\partial_{\mathbf{x}}^k f) \right\rangle$ and $U_i := \left\langle \pi_L(\partial_{\mathbf{x}}^k Q_i^m) \right\rangle$,
- $G_i := \pi_L(Q_i)$ 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 $2k(t-1)$,
- $W := \left\langle \pi_P(\partial_{\mathbf{z}}^k g_0) \right\rangle$ and $W_i := \left\langle \pi_P(\partial_{\mathbf{z}}^k G_i^e) \right\rangle$.

Observe that

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

implying

$$\dim U_i \leq \binom{n_0 + k(t-1) - 1}{k(t-1)}$$

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

implying

$$\dim W_i \leq \binom{m_0 + k(t-1) - 1}{k(t-1)}.$$

Definition I.1 (Non-degeneracy)**.** The circuit C, given by Equation (11), 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 the full version, which says that if we choose the Q_i's randomly in Equation (11), then the circuit is non-degenerate with high probability.

Lemma I.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 I.1 are satisfied.*

C. Full version of the paper

The full version of the paper can be found at https://arxiv.org/abs/2004.06898.

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 [43]. We thank Navin Goyal for multiple helpful discussions on learning mixtures of Gaussians and related problems and for referring us to the paper [57]. We would also like to thank Ravi Kannan for pointing a bug in the statement and proof of a Corollary in an earlier version of the full paper.

References

[1] V. Strassen, "Vermeidung von divisionen." *Journal für die reine und angewandte Mathematik*, vol. 264, pp. 184–202, 1973.

[2] A. Shpilka and A. Yehudayoff, "Arithmetic circuits: A survey of recent results and open questions," *Foundations and Trends in Theoretical Computer Science*, vol. 5, no. 3-4, pp. 207–388, 2010.

[3] X. Chen, N. Kayal, and A. Wigderson, "Partial derivatives in arithmetic complexity and beyond," *Foundations and Trends in Theoretical Computer Science*, vol. 6, no. 1-2, pp. 1–138, 2011.

[4] R. Saptharishi, "A survey of lower bounds in arithmetic circuit complexity," *Github survey*, 2015.

[5] V. Kabanets and R. Impagliazzo, "Derandomizing polynomial identity tests means proving circuit lower bounds," *computational complexity*, vol. 13, no. 1-2, pp. 1–46, 2004.

[6] Z. Dvir, A. Shpilka, and A. Yehudayoff, "Hardness-randomness tradeoffs for bounded depth arithmetic circuits," *SIAM Journal on Computing*, vol. 39, no. 4, pp. 1279–1293, 2010.

[7] J. Heintz and C.-P. Schnorr, "Testing polynomials which are easy to compute," in *Proceedings of the twelfth annual ACM symposium on Theory of computing*, 1980, pp. 262–272.

[8] M. Agrawal, "Proving lower bounds via pseudo-random generators," in *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer, 2005, pp. 92–105.

[9] R. Raz and A. Shpilka, "Deterministic polynomial identity testing in non-commutative models," *Computational Complexity*, vol. 14, no. 1, pp. 1–19, 2005.

[10] M. A. Forbes and A. 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*, 2013, pp. 243–252.

[11] R. Oliveira, A. Shpilka, and B. lee Volk, "Subexponential size hitting sets for bounded depth multilinear formulas," *computational complexity*, vol. 25, no. 2, pp. 455–505, 2016.

[12] M. A. Forbes, "Deterministic divisibility testing via shifted partial derivatives," in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE, 2015, pp. 451–465.

[13] L. Fortnow and A. R. Klivans, "Efficient learning algorithms yield circuit lower bounds," *J. Comput. Syst. Sci.*, vol. 75, no. 1, pp. 27–36, 2009, conference version appeared in the proceedings of COLT 2006.

[14] I. 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*, 2016, pp. 1540–1561.

[15] A. R. Klivans and A. Shpilka, "Learning restricted models of arithmetic circuits," *Theory of Computing*, vol. 2, no. 10, pp. 185–206, 2006, conference version appeared in the proceedings of COLT 2003.

[16] A. Gupta, N. Kayal, and S. 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*, 2011, pp. 778–787.

[17] A. Gupta, N. Kayal, and Y. Qiao, "Random arithmetic formulas can be reconstructed efficiently," *Computational Complexity*, vol. 23, no. 2, pp. 207–303, 2014, conference version appeared in the proceedings of CCC 2013.

[18] N. Kayal, V. Nair, C. Saha, and S. Tavenas, "Reconstruction of Full Rank Algebraic Branching Programs," in *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, 2017, pp. 21:1–21:61.

[19] N. Kayal, V. Nair, and C. Saha, "Average-case linear matrix factorization and reconstruction of low width algebraic branching programs," *Computational Complexity*, vol. 28, no. 4, pp. 749–828, 2019.

[20] N. Kayal and C. 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.*, 2019, pp. 413–424.

[21] A. R. Klivans and D. 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*, 2001, pp. 216–223.

[22] A. Shpilka, "Interpolation of depth-3 arithmetic circuits with two multiplication gates," *SIAM J. Comput.*, vol. 38, no. 6, pp. 2130–2161, 2009, conference version appeared in the proceedings of STOC 2007.

[23] Z. S. Karnin and A. 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*, 2009, pp. 274–285.

[24] J. Håstad, "Tensor Rank is NP-Complete," *J. Algorithms*, vol. 11, no. 4, pp. 644–654, 1990, conference version appeared in the proceedings of ICALP 1989.

[25] Y. Shitov, "How hard is the tensor rank?" *arXiv*, vol. abs/1611.01559, 2016. [Online]. Available: https://arxiv.org/abs/1611.01559

[26] 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.

[27] S. E. Leurgans, R. T. Ross, and R. B. Abel, "A decomposition for three-way arrays," *SIAM Journal on Matrix Analysis and Applications*, vol. 14, no. 4, pp. 1064–1083, 1993.

[28] L. De Lathauwer, J. Castaing, and J.-F. Cardoso, "Fourth-order cumulant-based blind identification of underdetermined mixtures," *IEEE Transactions on Signal Processing*, vol. 55, no. 6, pp. 2965–2973, 2007.

[29] J. Anderson, M. Belkin, N. Goyal, L. Rademacher, and J. 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*, 2014, pp. 1135–1164.

[30] A. Bhaskara, M. Charikar, A. Moitra, and A. Vijayaraghavan, "Smoothed analysis of tensor decompositions," in *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, 2014, pp. 594–603.

[31] N. 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*, 2011, pp. 1409–1421.

[32] I. García-Marco, P. Koiran, and T. 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*, 2018, pp. 303–310.

[33] E. R. Berlekamp, "Factoring polynomials over large finite fields," *Mathematics of Computation*, vol. 24, pp. 713–735, 1970.

[34] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, no. 4, pp. 515–534, 1982.

[35] N. Linial, Y. Mansour, and N. Nisan, "Constant Depth Circuits, Fourier Transform, and Learnability," *J. ACM*, vol. 40, no. 3, pp. 607–620, 1993, conference version appeared in the proceedings of FOCS 1989.

[36] M. L. Carmosino, R. Impagliazzo, V. Kabanets, and A. Kolokolova, "Learning Algorithms from Natural Proofs," in *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, 2016, pp. 10:1–10:24.

[37] A. Beimel, F. Bergadano, N. H. Bshouty, E. Kushilevitz, and S. Varricchio, "Learning functions represented as multiplicity automata," *J. ACM*, vol. 47, no. 3, pp. 506–530, 2000, conference version appeared in the proceedings of FOCS 1996.

[38] M. 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*, 2008, pp. 67–75.

[39] P. Koiran, "Arithmetic circuits: The chasm at depth four gets wider," *Theor. Comput. Sci.*, vol. 448, pp. 56–65, 2012.

[40] S. 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*, 2013, pp. 813–824.

[41] J. T. Schwartz, "Fast Probabilistic Algorithms for Verification of Polynomial Identities," *J. ACM*, vol. 27, no. 4, pp. 701–717, 1980.

[42] R. 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*, 1979, pp. 216–226.

[43] A. L. Chistov, G. Ivanyos, and M. 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*, 1997, pp. 68–74.

[44] K. Friedl and L. 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*, 1985, pp. 153–162.

[45] L. Rónyai, "Computing the structure of finite algebras," *J. Symb. Comput.*, vol. 9, no. 3, pp. 355–373, 1990.

[46] W. Eberly, "Decompositions of algebras over R and C," *Computational Complexity*, vol. 1, pp. 211–234, 1991.

[47] N. Kayal, "An exponential lower bound for the sum of powers of bounded degree polynomials," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 19, p. 81, 2012.

[48] A. Gupta, P. Kamath, N. Kayal, and R. Saptharishi, "Approaching the Chasm at Depth Four," *J. ACM*, vol. 61, no. 6, pp. 33:1–33:16, 2014, conference version appeared in the proceedings of CCC 2013.

[49] N. Kayal, C. Saha, and R. 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*, 2014, pp. 146–153.

[50] N. 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*, 1991, pp. 410–418.

[51] N. Nisan and A. Wigderson, "Lower Bounds on Arithmetic Circuits Via Partial Derivatives," *Computational Complexity*, vol. 6, no. 3, pp. 217–234, 1997, conference version appeared in the proceedings of FOCS 1995.

[52] M. Kumar and S. 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*, 2014, pp. 136–145.

[53] H. Fournier, N. Limaye, G. Malod, and S. Srinivasan, "Lower bounds for depth-4 formulas computing iterated matrix multiplication," *SIAM J. Comput.*, vol. 44, no. 5, pp. 1173–1201, 2015, conference version appeared in the proceedings of STOC 2014.

[54] N. Kayal, N. Limaye, C. Saha, and S. Srinivasan, "An Exponential Lower Bound for Homogeneous Depth Four Arithmetic Formulas," *SIAM J. Comput.*, vol. 46, no. 1, pp. 307–335, 2017, conference version appeared in the proceedings of FOCS 2014.

[55] M. Kumar and S. Saraf, "On the Power of Homogeneous Depth 4 Arithmetic Circuits," *SIAM J. Comput.*, vol. 46, no. 1, pp. 336–387, 2017, conference version appeared in the proceedings of FOCS 2014.

[56] N. Kayal, V. Nair, and C. 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*, 2016, pp. 46:1–46:15.

[57] R. Ge, Q. Huang, and S. 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*, 2015, pp. 761–770.