Tensor Reconstruction Beyond Constant Rank

Shir Peleg
Blavatnik School of Computer Science, Tel Aviv University, Israel

Amir Shpilka
Blavatnik School of Computer Science, Tel Aviv University, Israel

Ben Lee Volk
Efi Arazi School of Computer Science, Reichman University, Herlizya, Israel

Abstract

We give reconstruction algorithms for subclasses of depth-3 arithmetic circuits. In particular, we obtain the first efficient algorithm for finding tensor rank, and an optimal tensor decomposition as a sum of rank-one tensors, when given black-box access to a tensor of super-constant rank. Specifically, we obtain the following results:

1. A deterministic algorithm that reconstructs polynomials computed by $\Sigma^{[k]}\bigwedge^{[d]}\Sigma$ circuits in time $\mathsf{poly}(n,d,c)\cdot\mathsf{poly}(k)^{k^{k^{10}}}$,
2. A randomized algorithm that reconstructs polynomials computed by multilinear $\Sigma^{[k]}\prod^{[d]}\Sigma$ circuits in time $\mathsf{poly}(n,d,c)\cdot k^{k^{k^{k^{O(k)}}}}$,
3. A randomized algorithm that reconstructs polynomials computed by set-multilinear $\Sigma^{[k]}\prod^{[d]}\Sigma$ circuits in time $\mathsf{poly}(n,d,c)\cdot k^{k^{k^{k^{O(k)}}}}$,

where $c=\log q$ if $\mathbb{F}=\mathbb{F}_q$ is a finite field, and c equals the maximum bit complexity of any coefficient of f if $\mathbb{F}$ is infinite.

Prior to our work, polynomial time algorithms for the case when the rank, k, is constant, were given by Bhargava, Saraf and Volkovich [5].

Another contribution of this work is correcting an error from a paper of Karnin and Shpilka [20] (with some loss in parameters) that also affected Theorem 1.6 of [5]. Consequently, the results of [20, 5] continue to hold, with a slightly worse setting of parameters. For fixing the error we systematically study the relation between syntactic and semantic notions of rank of $\Sigma\Pi\Sigma$ circuits, and the corresponding partitions of such circuits.

We obtain our improved running time by introducing a technique for learning rank preserving coordinate-subspaces. Both [20] and [5] tried all choices of finding the "correct" coordinates, which, due to the size of the set, led to having a fast growing function of k at the exponent of n. We manage to find these spaces in time that is still growing fast with k, yet it is only a fixed polynomial in n.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Algebraic circuits, reconstruction, tensor decomposition, tensor rank

Digital Object Identifier 10.4230/LIPIcs.ITCS.2024.87

Related Version *Full Version*: `https://arxiv.org/abs/2209.04177`

Funding *Shir Peleg*: The research leading to these results has received funding from the Israel Science Foundation (grant number 514/20) and from the Len Blavatnik and the Blavatnik Family foundation.
Amir Shpilka: The research leading to these results has received funding from the Israel Science Foundation (grant number 514/20) and from the Len Blavatnik and the Blavatnik Family foundation.
Ben Lee Volk: The research leading to these results has received funding from the Israel Science Foundation (grant number 843/23).

1 Introduction

Reconstruction of algebraic circuits is a natural algorithmic problem that asks, given a black box access to a polynomial f from some circuit class $\mathcal{C}$, to efficiently output an algebraic circuit computing f. Algebraic circuits are computational devices that compute multivariate polynomials using basic arithmetic operations, much like boolean circuits compute boolean functions using boolean bit operations. Thus, the reconstruction problem is a natural algebraic analog for well studied boolean learning problems [7].

It is often desired that the output of the algorithm will also be a circuit from the class $\mathcal{C}$ (which is called *proper learning*). Requiring the learning algorithm to be efficient imposes an obvious upper bound on the size of the output, but it is also desirable to output a circuit as small as possible, ideally the smallest possible circuit from the class $\mathcal{C}$ that computes f.

Reconstruction, however, is also a hard algorithmic problem. Results such as the NP hardness of computing or even approximating tensor rank [18, 29, 6, 33] force us to carefully manage our expectations regarding what's possible to compute efficiently, since it turns out that even for weak classes $\mathcal{C}$ (such as depth-3 set-multilinear circuits) it's unlikely to find an efficient algorithm that outputs the smallest possible circuit. Furthermore, reconstruction appears to be an even harder problem than black box Polynomial Identity Testing (PIT), the problem of determining whether the black box f computes the identically zero polynomial. While PIT can be efficiently solved using randomness, efficient deterministic algorithms are known only for a handful of restricted circuit classes (we note, though, that in the reconstruction problem even giving a randomized algorithm is a non-trivial task). For a survey on algebraic circuits, PIT and reconstruction, see [30].

Nevertheless, for some restricted classes, or when the constraints are sufficiently relaxed, it is possible to give many non-trivial efficient reconstruction algorithms. For example, many works have dealt with *random* algebraic circuits (see, e.g., [16, 22, 13], among others). In this setting, we think of the black box as being chosen randomly from the class $\mathcal{C}$ under some natural distribution on circuits from $\mathcal{C}$, and we require the algorithm to reconstruct f with high probability over the chosen circuit (and perhaps over the random coins of the algorithm as well). Random circuits often avoid the degeneracies and pathologies that are associated with the clever cancellations that facilitate sophisticated algebraic algorithms, and are thus easier to handle and argue about.

Another line of study, more relevant to our work, has to do with reconstruction of small depth algebraic circuits. The simplest non-trivial case is depth-2 circuits, for which the reconstruction problem is pretty well understood and can be done efficiently [3, 24]. However, even slightly larger depths, like depth-3 and depth-4 circuits, already pose a much greater challenge. This is perhaps explained by a sequence of depth reduction results [2, 25, 34, 15] that show that low depth circuits are expressive enough to non-trivially simulate any algebraic circuit of polynomial size (and arbitrary depth). Thus, most attention has focused on restricted classes of depth-3 and depth-4 circuits [20, 17, 31, 32, 4, 5].

1.1 Circuit Classes

A depth-3 circuit with top fan-in (that is, the in-degree of the top sum gate) k computes a polynomial of the form $\sum_{i=1}^{k} \prod_{j=1}^{d_i} \ell_{i,j}(\mathbf{x})$, where each $\ell_{i,j}$ is a linear function in the input variables $\mathbf{x}$. We denote this class $\Sigma^k\Pi\Sigma$. When k is constant, this is a subclass of general depth-3 circuits that has been extensively studied (see Section 4.6 of [30]).

The circuit is called *multilinear* if every gate in the circuit computes a multilinear polynomial. An even stronger restriction is *set-multilinearity.* A polynomial f is set-multilinear if the set of variables $\mathbf{x}$ can be partitioned into disjoint sets $\mathbf{x}_1, \ldots, \mathbf{x}_d$ such that every monomial appearing in f is a product of variables $x_{1,i_1} x_{2,i_2} \cdots x_{d,i_d}$ such that x_{j,i_j}

is in $\mathbf{x}_j$. That is, a degree-d set-multilinear polynomial is simply a d-dimensional tensor. Depth-3 set-multilinear circuits, which are circuits in which every gate computes a multilinear polynomial, are a natural model for computing tensors. Each product of linear functions $\prod_{j=1}^{d} \ell_j(\mathbf{x}_j)$ corresponds to a rank one tensor, and thus we see that f can be computed by a set-multilinear circuit of top fan-in k if and only if its rank is at most k.

Finally, the most restricted model we study is *depth-3 powering circuits*. In this model, multiplication gates are replaced by powering gates. Such gates get as input a single linear function and their output is that function raised to some power. We denote the class of depth-3 powering circuit by $\Sigma^k \wedge \Sigma$. This is a natural computational model for computing *symmetric tensors*, where again the top fan-in corresponds to the rank.

Karnin and Shpilka [20] presented polynomial time reconstruction algorithms for $\Sigma^k \wedge \Sigma$ multilinear circuits for $k = O(1)$ over fields of size at most polynomial in n. More recently, Bhargava, Saraf, and Volkovich [5] presented proper reconstruction algorithms for circuit models discussed above. The running times of their algorithms are polynomial in n, the number of variables, and the degree d, assuming k is constant, but not when k is any growing function of n or d. Unlike [20], their algorithms work even over infinite fields. The exact running time is a polynomial whose exponent is a somewhat complicated expression that involves some quickly growing function of k. We describe their results more precisely vis-à-vis our results in Subsection 1.2.

In particular, given a constant upper bound on the rank, they obtain efficient algorithms that given a tensor (or a symmetric tensor) can exactly compute its rank, and also obtain a decomposition as a sum of rank-one tensors. Since for large enough ranks the problem of computing the tensor rank becomes NP hard, it's natural to wonder at which point the intractability kicks in. That is, is there an efficient polynomial time algorithm that can compute the rank and obtain a decomposition even when the upper bound k is super-constant?

In this paper we obtain faster algorithms that remain polynomial time algorithms (in n and d) even when k is slightly super-constant. Our running times are of the form $\mathsf{poly}(n, d, T(k))$ where T is some quickly growing function of k. Like the algorithms of Karnin and Shpilka [20] and Bhargava, Saraf, and Volkovich [5], our learning algorithms are proper and return the smallest possible representation of f in the relevant circuit model. In particular, they imply efficient randomized algorithms for computing tensor rank even when the rank is slightly super-constant.

Another contribution of this work is correcting an error that appeared in previous work. This error originated in [20] and affected Theorem 1.6 of [5] as well. Explaining the nature of the error requires some technical details that we present in Subsubsection 1.3.4. Our correction recovers the affected results of [20, 5], albeit with a slight change in the parameters that implies a somewhat worse dependence on the parameter k.

Our algorithms require the field $\mathbb{F}$ to be large enough. The precise meaning of what "large enough" means depends on each case. The largeness assumption can always be guaranteed without loss of generality by considering field extensions, if necessary (in which case the output will also be a circuit over the extension field). In certain cases, we also assume that the characteristic of the field is large enough.

1.2 Our Results

We start by describing our results for depth-3 powering circuits.

▶ **Theorem 1.** *There exists a randomized algorithm that, given a black box access to a polynomial f with n variables and degree d that is computed by a $\Sigma^k \wedge \Sigma$ circuit, reconstructs f in time $\mathsf{poly}(n, d, c) \cdot \mathsf{poly}(k)^{k^{k^{10}}}$, where $c = \log q$ if $\mathbb{F} = \mathbb{F}_q$ is a finite field, and c equals the maximum bit complexity of any coefficient of f if $\mathbb{F}$ is infinite.*

In [5], the authors give an algorithm for a similar task that runs in time $\mathsf{poly}((dk)^{k^{k^{10}}}, n, c)$. Note that unlike the algorithm in [5], when $d = \mathsf{poly}(n)$ our algorithms runs in polynomial time even when k is a slightly super-constant function of n (e.g., $k = (\log\log n/\log\log\log n - O(1))^{1/10}$). As in [5], we can derandomize the algorithm from Theorem 1 over $\mathbb{R}$ or $\mathbb{C}$ and obtain a deterministic algorithm that runs in roughly the same time. Theorem 1 is proved in Section 3.

We also provide reconstruction algorithms for multilinear depth-3 circuits with top fan-in k.

▶ **Theorem 2.** *There exists a randomized algorithm that, given a black box access to a polynomial f with n variables and degree d, which is computed by a $\Sigma^k\Pi\Sigma$ multilinear circuit, reconstructs f in time $\mathsf{poly}(n, d, c) \cdot k^{k^{k^{O(k)}}}$, where $c = \log q$ if $\mathbb{F} = \mathbb{F}_q$ is a finite field, and c equals the maximum bit complexity of any coefficient of f if $\mathbb{F}$ is infinite.*

Note that again, the algorithm in Theorem 2 runs in polynomial time for small enough (but super-constant) k, whereas the corresponding algorithm of [5] had running time of roughly $n^{T(k)}$ for some quickly growing function $T(k)$.

Finally, we also present a reconstruction algorithm for set-multilinear depth-3 circuits. Note that even though this class is a subclass of the previous model of multilinear circuits, as long as we insist on proper learning, reconstruction algorithms for a more general class don't imply reconstruction algorithms for its subclasses.

▶ **Theorem 3.** *There exists a randomized algorithm that, given a black box access to a polynomial $f(\mathbf{x}_1, \ldots \mathbf{x}_d)$ such that $|\mathbf{x}_i| \leq n$ for every $i \in [d]$, such that f is computed by a depth-3 set-multilinear circuit with top fan-in k, reconstructs f in time $\mathsf{poly}(n, d, c) \cdot k^{k^{k^{O(k)}}}$, where $c = \log q$ if $\mathbb{F} = \mathbb{F}_q$ is a finite field, and c equals the maximum bit complexity of any coefficient of f if $\mathbb{F}$ is infinite.*

Unlike the algorithms from [5] and our algorithm from Theorem 1, we don't know how to derandomize the algorithms from Theorem 2 and Theorem 3, even over $\mathbb{R}$ or $\mathbb{C}$. This remains an interesting open problem.

In this version, due to space constraints, many of the details are omitted. The full proofs appear in the full version of this paper (linked under "related version" above).

1.3 Proof Technique

While our proof follows general outline of the proofs in [20, 5], improving the running time and correcting the errors (as explain in Subsubsection 1.3.4) requires significant changes in parts of the argument.

There are two main factors contributing to the doubly or triply exponential dependence on k in the time complexity of the algorithms in [5].

The first is the fact that their algorithms solve systems of polynomial equations. This is required in order to find brute force solutions for the reconstruction problem over various projections of f to a few variables, making the number of variables in the polynomial system of equations rather small (that is, only a function of k, and not of n). They then calculate the running time using the best known algorithms for solving such systems of polynomial equations. The exact running time depends on the field, and it is typically singly or doubly exponential time in the number of variables.

Our main observation is that in all of these cases, it is also possible to modify the algorithms so that the *degree* of the polynomial system of equations and the *number of equations* are also only functions of k (and not of n or d, the number of variables and degree of the original polynomial f).

The second reason their algorithms run in time $n^{T(k)}$ is a construction of an object called "rank preserving subspace", introduced in [20], which is a subset of the coordinates that preserves certain properties of the polynomial, as we explain in Subsubsection 1.3.2. The dimension of this subspace depends on k, but finding it involves enumerating over all possible subsets of coordinates of the relevant size. As we soon explain, overcoming this difficulty requires a substantial amount of work.

1.3.1 $\Sigma^k \wedge \Sigma$ circuits

In the case of Theorem 1, getting the degree of the polynomial system of equations to be small is done in a simple way, by simply learning a high order derivative of f instead of f itself. Of course, when we take derivatives of f we have to make sure that we don't lose too much information so that the learning problem for the derivative becomes trivial but useless. We find a small sets of vectors $S \subseteq \mathbb{F}^n$ such that given any black box access to a polynomial computed by a depth-3 powering circuit, $f = \sum_{i=1}^{k} c_i \ell_i^d$, there exists $\mathbf{u} \in S$ such that in the iterated directional derivative $g := \partial^{d-2k-1} f / \partial \mathbf{u}^{d-2k-1}$ none of the linear functions "disappear": that is, g itself is a depth-3 powering circuit with top fan-in k and exactly the same k linear functions $\ell_1, \ldots, \ell_k$ appearing in the circuit, perhaps with different coefficients (that are easily computable functions of the original coefficients). We can then learn g using the algorithm of [5], except that g is a polynomial of degree $2k+1$, so that the dependence on d of our algorithm is much more tame.

1.3.2 Multilinear and Set-Multilinear $\Sigma^k \Pi \Sigma$ Circuits

The proofs of Theorem 2 and Theorem 3 can be broken down to two parts, the first handles low degree polynomials and the second high degree polynomials. The analysis of both parts in [5] incurs factors of the form $n^{T(k)}$, which we would like to eliminate. While the proof of the low degree case follows the general outline of [5], the proof of the high degree case is significantly more challenging and requires new ideas. As we describe later, the proof of the high degree case in [5], for multilinear $\Sigma^k \Pi \Sigma$ circuits, contains an error originating in [20]. We are able to correct this error (see Theorem 21), but even this correction doesn't suffice for improving the running time and a new approach is needed. Their result for set-multilinear circuits were not affected by this error as they use a different proof technique in the high degree case.

The low degree case

when the degree d is small, the number of linear functions in the circuit, which is bounded by kd, is also small so that we can allow ourselves to try and learn the circuit for f in an almost brute-force manner by solving a system of polynomial equations. Following [5] we first find, in polynomial time, an invertible linear transformation A so that $g := f(A\mathbf{x})$ depends on a few *variables* (and not merely linear functions). We then obtain a low-degree polynomial in a small number of variables, so that we can allow ourselves to learn the new circuit by solving a set of polynomial equations whose variables are the coefficients of the purported small circuit.

We then wish to output $g(A^{-1}\mathbf{x})$. The problem is that this circuit may not be multilinear. To solve this, [5] introduce an additional set of $\approx \mathsf{poly}(n)$ low degree polynomial equations to guarantee that the output circuit is multilinear. This results in a running time of about $n^{T(k)}$ for this part alone. We observe however that this set is highly redundant in the sense that, by dimension arguments, many of these equations are linearly dependent. By finding a basis to the polynomial system of equations and solving that basis alone, we're able to reduce the running time to $n \cdot T'(k)$ for some different function $T'(k)$.

Our algorithm for low-degree set-multilinear circuits is very similar but a bit simpler. By slightly tweaking the polynomial system of equations that describes the circuit, we can learn $f(A\mathbf{x})$ as a set-multilinear circuit. Further, in this case it's possible to find A such that $g(A^{-1}\mathbf{x})$ will automatically be set-multilinear, so that the challenge described in the previous paragraph doesn't exists in this setting.

The high degree case

this is the more complicated and tedious part of the argument. We start by explaining the high level approach of [20] and [5].

The *(syntactic) rank* of a $\Sigma^k\Pi\Sigma$ circuit is defined to be the dimension of the span of the linear functions appearing in its multiplication gates, after factoring out the greatest common divisors of these gates (that is, the linear functions appearing in all of them). For more details see Definition 16. This is a well studied notion originating in the work of Dvir and Shpilka [10] and used in many later works [19, 20, 26, 28, 23]. The rank function allows one to define the *distance* between two circuits C_1 and C_2 as the rank of their sum.

The algorithm of [5] for learning multilinear $\Sigma^k\Pi\Sigma$ circuits relies on a structural property of such circuits claimed by Karnin and Shpilka [20]. Karnin and Shpilka [20] partition the k multiplication gates in the circuit to *clusters*, so that each cluster has a low rank and each two distinct clusters have a large distance. In [20], it is claimed that for some choice of parameters, this partition is unique and depends only on the polynomial computed by the circuit and not on the circuit itself (this is where the error is, and this is what is being corrected in Theorem 21). Thus, the authors of [5] try to obtain black box access to each of the clusters. Then, factoring out their greatest common divisors they can reconstruct them as, by multilinearity, the remaining part is a low degree polynomial.

Obtaining black box access to the clusters is most of the technical work in the proof of Theorem 1.6 of [5]. In a high level, using their uniqueness result, Karnin and Shpilka [20] claimed to prove the existence of a small "rank preserving subset" of the variables B, such that after randomly fixing the variables outside of B, the remaining circuit $C|_B$ has the property that its clusters are in one-to-one correspondence with the original clusters restricted to B. The circuit $C|_B$ can again be reconstructed using the low-degree case, as it only involves a small number of variables, and thus we can get direct access to its clusters. Using a clever algorithm, Bhargava, Saraf and Volkovich [5] are able to obtain evaluations of the original clusters using evaluations of the restricted clusters.

Regardless of the correctness issue that we discuss soon, a big bottleneck of this argument is that one needs to iterate over all subsets B of $[n]$ up to a certain size bound (that depends only on k). Clearly such a procedure requires running time of the form $n^{T(k)}$.

Thus, we would like to obtain an algorithm that explicitly constructs a set B. One natural approach is to start with the empty set and add one variable at a time. This can be done by reconstructing the polynomial f restricted to the current set B, and its clusters, and checking whether adding a variable to B changes one of the parameters. If so then we add the variable and repeat the process. We continue as long as either the number of clusters or the rank of a cluster increases. The challenge with this approach is that the uniqueness guarantee of Theorem 21 does not suffice. Note that if f has a $\Sigma^k\Pi\Sigma$ circuit C, f restricted to B as a natural $\Sigma^k\Pi\Sigma$ circuit obtained by restricting the circuit C to B. However, our low-degree algorithm learns *some*, and potentially different, $\Sigma^k\Pi\Sigma$ circuit that computes the restriction of f to B. While Theorem 21 guarantees both circuits would have the same number of clusters, computing the same polynomials, we don't have the guarantee that the *rank* of each cluster is the same in the different circuits, as the rank may depend on the circuit. Thus, as we gradually increase B, it seems hard to compare rank of clusters between different representations.

To circumvent that we introduce *semantic* versions of ranks and distances, which are properties of a *polynomial* and not of a circuit computing it. We then develop a theory that studies the semantic and syntactic notions of rank, and the relations between them. In fact, our version of the semantic rank was already introduced by Karnin and Shpilka in [19, 20], in the context of learning so-called $\Sigma\Pi\Sigma(k, d, \rho)$ circuits. Their notion of "rank" for such circuits is a certain hybrid between syntactic and semantic rank. Since in our case the distinction is important, we try to mention explicitly whether we mean syntactic or semantic rank.

1.3.3 Semantic Notions of Rank

The semantic rank of a polynomial f is defined as follows: first write $f = \prod_i \ell_i \cdot h$, where the ℓ_i's are linear functions and h has no linear factors. Then define the semantic rank of f to be the minimal number r such that h depends on r linear functions.

This number is well defined and doesn't depend on any representation of f as a $\Sigma^k\Pi\Sigma$ circuit. Working with semantic rank has advantages and disadvantages. On the one hand, it is now possible to prove stronger uniqueness properties regarding the clusters, since if two clusters compute the same polynomial then they also have the same rank. Indeed we prove such a uniqueness statement for some parameters. On the other hand, analyzing the semantic rank and its behavior under various operations (such as restricting the circuit to a subset of the variables, or increasing the set B using the approach mentioned above) is significantly more difficult. Thus, we also prove various connections between semantic and syntactic ranks and we are able to show that if f is computed by an $\Sigma^k\Pi\Sigma$ circuit C, then the semantic and syntactic ranks of C are not too far apart.

Recall that our main challenge is to explicitly construct a cluster-preserving subset B of the variables, whose existence for syntactic ranks was proved by [20] (see Subsubsection 1.3.4 for a discussion of this result). In the context of semantic rank, proving such an analogous statement is significantly more challenging. In fact, while the proof of [20] is existential (and then the algorithm of [5] essentially enumerates over all possible subspaces), our proof is algorithmic. In essence, our algorithm follows the outline described above: it starts with the empty set and on each iteration adds a few variables to B until the cluster structure "stabilizes", i.e., their number and their ranks stay the same. Proving that this algorithm works requires a significant amount of technical work.

1.3.4 The Errors in Previous Work and Our Corrections

Explaining the nature of the erroneous statements appearing in [5, 20] requires giving some more technical details.

As mentioned earlier, one of the main components in the reconstruction algorithm for multilinear $\Sigma^k\Pi\Sigma$ circuits given in [5] is the uniqueness of clusters property for such circuits, which is claimed by Karnin and Shpilka [20]. Note that the rank and distance measures for $\Sigma^k\Pi\Sigma$ circuits are *syntactic* and inherently tied to a circuit. Karnin and Shpilka [20] define a *clustering* algorithm that, given a circuit, partitions the k multiplication gates into several sets such that the rank of the subcircuit corresponding to each set is small, and the distance between every pair of subcircuits is large.

In Corollary 6.8 of [5] it is claimed, based on [20], that these clusters are unique, even among different circuits that compute the same polynomial. That is, if C and C' are two circuits computing the same polynomial f, the clustering algorithm of [20] would return the same clusters (perhaps up to a permutation). Such a claim can indeed be read from Theorem 5.3 of [20]. However, in our judgment, the paper [20] does not contain a valid mathematical proof for such a statement.

Karnin and Shpilka associate with each partition into clusters two parameters, κ and r. The parameter r upper bounds the rank of each cluster, and the parameter κ controls the distance: their clustering algorithm guarantees that each pair of clusters has distance at least κr. Consequentially, their clustering algorithm receives κ as an additional input, and outputs a clustering with parameters κ and r for some value of r that can be upper bounded as a function of κ and k.

The proof of Theorem 5.3 of [20] assumes without justification that, given two different circuits C and C' computing the same polynomial, the clustering algorithm with parameters κ would return partitions with the same value of the parameter r, which is crucially used in their proof.

In this work we provide a corrected proof of Theorem 5.3 of [20] (Theorem 21). While the corrected version is not identical to the original statement word-for-word (as our parameter κ is much larger than originally stated as a function of k), it suffices for fixing the arguments in [20] and [5], with the straightforward corresponding changes in parameters throughout.

We wish to stress again that Theorems 1.1 and 1.4 of [5], that give algorithms for learning depth-3 set-multilinear and depth-3 powering circuits, respectively, are not affected by the error in [20].

1.4 Open Problems

One natural problem our work raises is the question of how large the top fan-in k needs to be before reconstruction problem becomes intractable. The NP-hardness results for tensor rank imply that clearly when $k = \mathsf{poly}(n)$ we shouldn't expect to find exact proper learning algorithm, whereas we show that the intractability barrier is not at the regime when k is constant. It remains an interesting problem to bridge the gap.

Another interesting problem is derandomizing our algorithms from Theorem 2 and Theorem 3.

2 Preliminaries

All proof from this section are omitted and appear in the full version of the paper.

The following notation will be very useful throughout our paper.

▶ **Definition 4.** *For $\mathbf{a} \in \mathbb{F}^n$, $B \subseteq [n]$, and a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ we define $f|_{B,\mathbf{a}}$ the polynomial obtained by fixing $x_j = \mathbf{a}_j$ for every $j \notin B$.*

2.1 Black Box Access to Directional Derivatives

▶ **Lemma 5.** *Let $\mathbb{F}$ be a field of size at least $d+1$ and let $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial of degree d. Given a black box access to f, for every $e \leq d$ and for every variable $x \in \{x_1, \ldots, x_n\}$, we can simulate a black box access to $g := \partial^e f / \partial x^e$ using at most $d+1$ queries to f.*

Lemma 5 can be generalized to directional derivatives.

▶ **Lemma 6.** *Let $\mathbb{F}$ be a field of size at least $d+1$ and let $f(x_1, \dots, x_n) \in \mathbb{F}[x_1, \dots, x_n]$ be a polynomial of degree d. Given a black box access to f, for every $e \le d$ and for every $0 \neq \mathbf{u} \in \mathbb{F}^n$, we can simulate a black box access to $g := \frac{\partial^e f}{\partial \mathbf{u}^e}$ using at most $d+1$ queries to f.*

2.2 Essential Variables

Let f be an n-variate polynomial. We say that f depends on m essential variables if there exists an invertible linear transformation A such that $f(A\mathbf{x})$ depends on m variables. An interesting fact is that it's possible, given a black box access to f, to compute a linear transformation A such that $g := f(A\mathbf{x})$ depends only on $x_1, \dots, x_m$.

▶ **Lemma 7** ([21, 8])**.** *Let $f \in \mathbb{F}[\mathbf{x}]$ be an n-variate polynomial of degree d with m essential variables, where $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > d$. Suppose f is computed by a circuit of size s. Then, there's an efficient randomized algorithm that, given black box access to f, runs in time $\mathsf{poly}(n, d, s)$ and computes an invertible linear transformation A such that $f(A\mathbf{x})$ depends on the first m variables $x_1, \dots, x_m$.*

Bhargava, Saraf and Volkovich [5] derandomize this lemma when f is computed by a $\Sigma^k \wedge \Sigma$ circuit, a depth-3 set-multilinear circuit of top fan-in k or a depth-3 multilinear circuit of top fan-in k. However the time required for their derandomization involves factors of $n^{O(k)}$ and thus we want to obtain an improved running time.

For a class of polynomials $\mathcal{C}$, we denote by $\Sigma^t \mathcal{C}$ the class of polynomials of the form $\alpha_1 f_1 + \alpha_2 f_2 + \cdots + \alpha_t f_t$ with $\alpha_i \in \mathbb{F}$ and $f_i \in \mathcal{C}$ for every i.

▶ **Lemma 8.** *Let $\mathcal{C}$ be a class of polynomials and let $f_1, \dots, f_t \in \mathcal{C}$. Let $\mathcal{H}$ be a hitting set for $\Sigma^t \mathcal{C}$. Denote by $f_i|_{\mathcal{H}}$ the vector (of length $|\mathcal{H}|$) $(f_i(\beta))_{\beta \in \mathcal{H}}$. Then for any $\alpha_1, \dots, \alpha_t \in \mathbb{F}$,*

$$\sum_{i=1}^{t} \alpha_i f_i = 0 \iff \sum_{i=1}^{t} \alpha_i f_i|_{\mathcal{H}} = 0.$$

In particular, the polynomials $f_1, \dots, f_t$ are linearly independent if and only if the vectors $f_1|_{\mathcal{H}}, \dots, f_t|_{\mathcal{H}}$ are linearly independent.

Lemma 8 gives an efficient way to test for dependency of polynomials assuming the existence of a small and efficiently constructible hitting set for $\Sigma^t \mathcal{C}$.

A derandomized version of Lemma 7 is given below.

▶ **Lemma 9.** *Let $\mathcal{C}$ be a class of polynomials closed under taking first order partial derivatives. Denote by $|\mathcal{H}|$ a hitting set for $\Sigma^{t+1}\mathcal{C}$. Then, there's a deterministic algorithm that, given a black box access to a degree-d polynomial $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ that has t essential variables such that $f \in \mathcal{C}$, runs in time $\mathsf{poly}(n, d, |\mathcal{H}|)$ and outputs an invertible matrix $A \in \mathbb{F}^{n \times n}$ such that $f(A\mathbf{x})$ depends only the first t variables.*

We note that the models we consider in this work are all closed under first order partial derivatives.

2.3 Hitting Sets for Depth-3 Circuits

While there exist hitting sets of size $n^{O(\log\log n)}$ for depth-3 powering circuits [11] and quasi-polynomial size hitting sets for depth-3 set-multilinear circuits [12, 11, 1], we insist on obtaining polynomial size hitting sets for these models when k is slightly super-constant (when k is constant there are hitting sets of size $n^{\mathsf{poly}(k)}$ for general depth-3 circuits of top fan-in k, see, e.g., Section 4.6.2 of [30] and [27]). Guo and Gurjar constructed such explicit polynomial size hitting sets for read-once algebraic branching programs (roABPs) of super-constant width.

▶ **Theorem 10** ([14]). *There's an explicit hitting set of size* $\mathsf{poly}(n,d)$ *for the class of n-variate, individual degree d polynomials computed by any-order roABPs of width w, assuming there's a constant $\varepsilon > 0$ such that $w = 2^{O(\log^{1-\varepsilon}(nd))}$.*

▶ **Corollary 11.** *There's an explicit hitting set of size* $\mathsf{poly}(n,d)$ *for the class of set-multilinear polynomials computed by depth-3 set-multilinear circuits of degree d and top fan-in $k = 2^{O(\log^{1-\varepsilon}(nd))}$.*

We can also reduce depth-3 powering circuits to roABPs.

▶ **Lemma 12** ([12]). *Suppose f is computed by a depth-3 powering circuit of top fan-in k and degree d. Then f is computable by an any-order roABP of width $O(dk)$, size* $\mathsf{poly}(dk,n)$ *and degree d.*

We only use Lemma 12 when k is very small and the degree d is commensurate with k. Thus, since the parameters of Theorem 10 are quite comfortable, we can deduce.

▶ **Corollary 13.** *Suppose d and k are both $2^{O(\log^{1-\varepsilon}(n))}$. Then, there's an explicit hitting set of size* $\mathsf{poly}(n)$ *for the class of set-multilinear polynomials computed by a $\Sigma^k\wedge\Sigma$ circuits of degree d and top fan-in k.*

For general depth-3 circuits with top fan-in k, the known results are slightly weaker.

▶ **Lemma 14** ([27]). *There exists an explicit hitting set for the class of n-variate polynomials computed by multilinear $\Sigma^k\Pi\Sigma$ circuits of degree d of size $n^{(O(k^2\log k)}$.*

We remark that the hitting set presented in [30] is of size $n^{O(R(k,r))}$ where $R(k,r)$ is the so-called rank bound for $\Sigma^k\Pi\Sigma$ circuits, which (for some fields, as explained in [30]) depends on d. However for *multilinear* circuits the above result is a corollary of Corollary 6.9 of [10] and the rank bounds of [28].

We further note that had we used Lemma 14, our algorithm wouldn't run in polynomial time for super-constant k, which is one of the reasons we use a randomized PIT algorithm for this class in our reconstruction algorithm. However, this is not the major obstacle for derandomization: derandomizing our algorithm in polynomial time would require a deterministic PIT for much larger classes than multilinear $\Sigma^k\Pi\Sigma$ circuits. It's an interesting open problem to obtain a derandomization for our algorithm even modulo Lemma 14.

2.4 Solving a System of Polynomial Equations

Let $\mathbb{F}$ be a field and let $\mathrm{Sys}_{\mathbb{F}}(n,m,d)$ denote the randomized time complexity of finding a solution to a polynomial system of m equations in n variables of degree d. A detailed analysis of this function for various fields $\mathbb{F}$ appears in Section 3.8 of the arXiv version of [5]. For our purposes, it is enough to note that for every field $\mathbb{F}$, $\mathrm{Sys}_{\mathbb{F}}(n,m,d) = \mathsf{poly}(nmd)^{n^n}$, if we allow solutions from an algebraic extension of $\mathbb{F}$. Further, for $\mathbb{F} = \mathbb{R}$, $\mathbb{C}$ or $\mathbb{F}_q$, extensions are not needed, and if $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$ then the algorithm is in fact deterministic.

2.5 Resultants

Let $f(x), g(x)$ be two polynomials of degrees m and ℓ in the variable x, respectively. Suppose $m, \ell > 0$, and write

$$\begin{aligned}
&f(x) = c_m x^m + x_{m-1}x^{m-1} + \cdots + c_0\\
&g(x) = d_\ell x^\ell + d_{\ell-1}x^{\ell-1} + \cdots + d_0.
\end{aligned}$$

The *Sylvester matrix* of the polynomials f and g with respect to the variable x is the following $(m+\ell)\times(m+\ell)$ matrix:

$$\begin{pmatrix}
c_m & & & & d_\ell & & & \\
c_{m-1} & c_m & & & d_{\ell-1} & & & \\
c_{m-2} & c_{m-1} & \ddots & & d_{\ell-2} & d_{\ell-1} & \ddots & \\
\vdots & & \ddots & c_m & \vdots & & \ddots & d_\ell \\
 & \vdots & & c_{m-1} & & \vdots & & d_{\ell-1} \\
c_0 & & & & d_0 & & & \\
 & c_0 & & \vdots & & d_0 & & \vdots \\
 & & \ddots & & & & \ddots & \\
 & & & c_0 & & & & d_0
\end{pmatrix}$$

The determinant of this matrix is called the *resultant* of f and g with respect to the variable x and is denoted $\mathrm{Res}_x(f, g)$.

In our case we often think of $f, g \in \mathbb{F}[x_1, \ldots, x_n]$ interchangeably as n-variate polynomials or as univariate polynomials in some variable, say x_1, over the ring $\mathbb{F}[x_2, \ldots, x_n]$, in which case the resultant is a polynomial in $x_2, \ldots, x_n$. The main property of resultant we use is that, assuming the degree in x_1 of both f and g is positive, f and g have a common factor in $\mathbb{F}[x_2, \ldots, x_n]$ if and only if $\mathrm{Res}_{x_1}(f, g) = 0$ (see, e.g., Proposition 3 in Chapter 3, Section 6 of [9]).

3 A Reconstruction Algorithm for Depth-3 Powering Circuits of Super-Constant Top Fan-in

The full version of this paper contains the proof of the following theorem.

▶ **Theorem 15.** *Suppose $|\mathbb{F}| > kn+1$ and $\mathrm{char}(\mathbb{F}) > d$ or $\mathrm{char}(\mathbb{F}) = 0$. There's a randomized algorithm that, given a black box access to a polynomial f computed by a $\Sigma^k \wedge \Sigma$ circuit, reconstructs f and runs in time $\mathsf{poly}(n, d, c) \cdot \mathsf{poly}(k)^{k^{k^{10}}}$, where $c = \log q$ if $\mathbb{F} = \mathbb{F}_q$ is a finite field, and c equals the maximum bit complexity of any coefficient of f if $\mathbb{F}$ is infinite.*

The details are omitted from this abridged version.

4 Syntactic Rank of Depth-3 Circuits

In the following two sections, we define syntactic and semantic notions of ranks of polynomials computed by $\Sigma^k\Pi\Sigma$ circuits. Note that *syntactic* ranks are inherently tied to *circuits* computing the polynomials, whereas semantic ranks are independent of the representation or computation of the polynomials.

For a circuit C we denote by $[C]$ the polynomial computed by C. For two $\Sigma^k\Pi\Sigma$ circuits C, C', we define their *syntactic sum*, $C + C'$, to be the depth-3 circuit whose top gate sums all multiplication gates in C and C'. Observe that $C + C'$ is a $\Sigma^{2k}\Pi\Sigma$ circuit.

We start by defining *syntactic* notions of rank and distance for $\Sigma^k\Pi\Sigma$ circuits.

▶ **Definition 16** (Syntactic Rank and Distance). *Let $C = \sum_{i=1}^k M_i = \sum_{i=1}^k \prod_{j=1}^{d_i} \ell_{i,j}$ be a $\Sigma^k\Pi\Sigma$ circuit. Define the following notions:*

1. $\deg(C) = \max\{\deg[M_i] : 1 \leq i \leq k\}$.
2. $\gcd(C)$ *is the set of linear functions appearing in all of $M_1, \ldots, M_k$ (up to multiplication by a constant). I.e.,* $\gcd(C) = \gcd(M_1, \ldots, M_k)$.
3. $sim(C) := \frac{C}{\gcd(C)} = \sum_{i=1}^k \frac{M_i}{\gcd(C)} \in \Sigma^k\Pi\Sigma$ *is called the* simplification *of C. C is called* simple *if* $\gcd(C) = 1$.
4. *We say that C is* minimal *if for every $\emptyset \neq S \subsetneq [k]$, $\sum_{i\in S} M_i \neq 0$.*
5. *Let $\mathcal{L}_i$ be the collection of linear polynomials appearing in $\frac{M_i}{\gcd(C)}$, we define $\Delta_{syn}(C) :=$* $\dim(span\{\mathcal{L}_1, \ldots, \mathcal{L}_k\})$.
6. *Let C' be a $\Sigma^k\Pi\Sigma$ circuit. We define $dist(C, C') = \Delta_{syn}(C + C')$.*

The usefulness of syntactic rank is expressed in the following well known *rank bound* for multilinear depth-3 circuits.

▶ **Theorem 17** ([10, 23, 26, 28]). *There's a monotone function $R(k, d)$ such that any simple and minimal $\Sigma^k\Pi\Sigma$ circuit C that computes the zero polynomial and such that* $\deg(C) \leq d$, *satisfies $\Delta_{syn}(C) \leq R(k, d)$. Further, $R(k, d) \leq 4k^2 \log(2d)$.*

If C is multilinear there's a similar function $R_M(k)$ depending only on k: any simple and minimal, multilinear $\Sigma^k\Pi\Sigma$ circuit C, computing the zero polynomial satisfies $\Delta_{syn}(C) \leq R_M(k)$. One can take $R_M(k) \leq 10k^2 \log k$.

4.1 Syntactic Partitions of $\Sigma^k\Pi\Sigma$ Circuits

In this section we study syntactic partitions of $\Sigma^k\Pi\Sigma$ circuits. In Subsection 5.2 we shall discuss *semantic* partitions and compare the two notions.

▶ **Definition 18** (Syntactic Partition, Definition 3.3 of [20]). *Let $C = \sum_{i=1}^k \prod_{j=1}^{d_i} \ell_{i,j} = \sum_{i=1}^k M_i$ be a $\Sigma^k\Pi\Sigma$ circuit. Let $I = \{A_1, \ldots, A_s\}$ be a partition of $[k]$. For each $i \in [s]$ let $C_i = \sum_{j\in A_i} M_j$. We say that $\{C_i\}_{i\in[s]}$ is a (τ, r)-syntactic partition of C if:*

- *For every $i \in [s]$, $\Delta_{syn}(C_i) \leq r$.*
- *For every $i \neq j \in [s]$, $dist(C_i, C_j) \geq \tau r$.*

The main corollary proved in the full version of the paper is:

▶ **Corollary 19** (Uniqueness of syntactic partitions with the same number of clusters). *Let $\tau \geq 10$. Let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit. Let $(C_1, \ldots, C_s)$ and $(D_1, \ldots, D_s)$ be (τ, r_C) and (τ, r_D)-syntactic partitions of the multiplication gates in C, respectively. Then, there is a permutation π on $[s]$ such that for every $i \in [s]$, $C_i = D_{\pi(i)}$.*

4.1.1 Algorithms for Computing Partitions

An algorithm for computing (τ, r)-syntactic partitions was provided by Karnin and Shpilka [20].

▶ **Lemma 20** (Syntactic Clustering Algorithm; See Algorithm 1 and Lemma 5.1 of [20]). *Let $n, k, r_{init}, \tau \in \mathbb{N}$. There exists an algorithm that given τ and an n-variate multilinear $\Sigma^k\Pi\Sigma$ circuit C as input, outputs $r \in \mathbb{N}$ such that*

$$R_M(2k) \leq r \leq k^{(k-2)\cdot\lceil\log_k(\tau)\rceil} \cdot R_M(2k) \leq (k\tau)^{k-2} \cdot R_M(2k)$$

and a (τ, r)-syntactic partition of $[k]$, in time $O(\log(\tau) \cdot n^3k^4)$. Further, with an additional running time of $2^{O(k^2)} \cdot \mathsf{poly}(n)$, we can guarantee that this syntactic partition has the lowest value of r among all τ syntactic partitions of C.

We remark that the "further" part isn't explicitly stated in [20]. However, it is easy to modify their algorithm in order to guarantee this property. For example, after running their algorithm one can run a brute force search over all partitions of $[k]$ and search for a τ-partition with a lower value of r. In the applications of Lemma 20, the additional running time incurred by this step is either irrelevant or anyway subsumed by larger factors of k originating from other elements in the proof.

4.2 Existence of a Unique Syntactic Partition

In the full version of the paper, we prove that for every multilinear polynomial $f \in \Sigma^k\Pi\Sigma$ there is a parameter τ, which is bounded by some function of k, such that any two τ partitions of any two $\Sigma^k\Pi\Sigma$ circuits computing f define, up to a permutation, the same clusters.

▶ **Theorem 21.** *For every multilinear polynomial $f \in \Sigma^k\Pi\Sigma$ there is $\tau = O(k^{k+2})^{k^{2k+1}}$ such that the following holds: Let C, D be any two $\Sigma^k\Pi\Sigma$ circuits computing f. Let $C = \sum_{i=1}^{s} C_i$ and $D = \sum_{i=1}^{s'} D_i$ be the τ-partitions of C and D, respectively, that Lemma 20 guarantees. Then $s = s'$ and there is a permutation $\pi : [s] \to [s]$ such that $[C_i] = [D_{\pi(i)}]$. Furthermore, for every i, $\Delta_{syn}(C_i)/k - 2R_M(2k) \leq \Delta_{syn}(D_{\pi(i)}) \leq k \cdot \Delta_{syn}(C_i) + 2kR_M(2k)$.*

5 Semantic Rank of Depth-3 Circuits

In the following section, we define the semantic rank of polynomials computed by $\Sigma^k\Pi\Sigma$ circuits. Note that while *syntactic* ranks is inherently tied to a *circuit* C computing the polynomial, the *semantic* rank is independent of the representation or computation of the polynomial. We omit all proofs from this version.

We say that a polynomial $g \in \mathbb{F}[x_1, \ldots, x_n]$ depends on r linear functions if there exist r linear functions $\ell_1, \ldots, \ell_r$ and a polynomial $h \in \mathbb{F}[y_1, \ldots, y_r]$ such that $g(\mathbf{x}) = h(\ell_1(\mathbf{x}), \ldots, \ell_r(\mathbf{x}))$.

▶ **Definition 22** (Semantic Rank). *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial. Define $Lin(f)$ to be the product of the linear factors of f. Let $r \in \mathbb{N}$ be the minimal integer such that $f/Lin(f)$ is a polynomial of exactly r linear functions. We define $\Delta_{sem}(f) = r$.*

Recall that the number of linear functions that a polynomial depends on equals the rank of its *partial derivative matrix*.

▶ **Definition 23.** *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial. Define M_f to be a matrix whose i-th row contains the coefficients of $\partial f/\partial x_i$.*

Note that if f depends on exactly r linear functions and $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > \deg(f)$, then $\mathrm{rank}(M_f) = r$.

▶ Remark 24. Note that under the definition above, it may be the case that f is non-zero and yet $\Delta_{\text{sem}}(f)$ equals 0. This happens when f is a product of linear functions. In what follows we will often implicitly assume that $\Delta_{\text{sem}}(f) \geq 1$. This doesn't affect our results but somewhat simplifies the presentation. One may also arbitrarily define the semantic rank of f to be 1 when f is a non-zero product of linear functions.

5.1 Semantic vs. Syntactic Rank

We now prove several claims that relate the syntactic and semantic notions of rank for polynomials computed by multilinear $\Sigma^k\Pi\Sigma$ circuits. We start by observing that the semantic rank is at most the syntactic rank.

▶ **Observation 25.** *Suppose f is a polynomial in multilinear $\Sigma^k\Pi\Sigma$. Then, every multilinear $\Sigma^k\Pi\Sigma$ circuit C computing f satisfies $\Delta_{syn}(C) \geq \Delta_{sem}(f)$.*

We want to upper bound the syntactic rank as a function of the semantic rank (naturally, this only makes sense for *minimal* circuits, as other circuits can have artificially large syntactic rank). Our argument is essentially identical to Lemma 2.20 of [20].

▶ **Lemma 26** (Lemma 4.2 of [19]). *Let C be a simple and minimal $\Sigma\Pi\Sigma(k,d,\rho)$ circuit computing the zero polynomial. Suppose*

$$C = \sum_{i=1}^{k} \left(\prod_{j=1}^{d_i} \ell_{i,j} \right) \cdot h_i(\tilde{\ell}_{i_1}, \ldots, \tilde{\ell}_{i,\rho_i})$$

and let $\tilde{R} = \sum_{i=1}^{k} \rho_i$. Then $\Delta_{syn}(C) \leq R(k,d) + \tilde{R}$.

Here $R(k,d) = 4k^2 \log(2d)$ is the rank bound for (not necessarily multilinear) $\Sigma^k\Pi\Sigma$ circuits (recall Theorem 17). Note that trivially $\tilde{R} \leq k\rho$, but in Lemma 28 we shall use the stricter upper bound stated in the lemma.

The proof of Lemma 26 in [19] is also not very complicated. Given a $\Sigma\Pi\Sigma(k,d,\rho)$ circuit C as in the statement of the lemma, one fixes randomly the linear functions $\tilde{\ell}_{i_1}, \ldots, \tilde{\ell}_{i,\rho_i}$, for $i \in [k]$, to obtain a simple and minimal $\Sigma^k\Pi\Sigma$ circuit of degree at most d, and applies the rank bound for such circuits. Note that fixing those linear functions might make the circuit non-multilinear even if the original circuit was multilinear, which means we have to use the rank bound $R(k,d)$ for non-multilinear $\Sigma^k\Pi\Sigma$ circuits. This incurs a dependence on the degree d. However, it is also convenient to have a form of Lemma 26 with no dependence on d. This is possible since C is multilinear. A similar observation was made by Dvir and Shpilka [10] for $\Sigma^k\Pi\Sigma$ circuits. Since C is multilinear, all linear functions appearing in each multiplication gate are variable disjoint, and hence linearly independent, which implies that $\Delta_{\text{syn}}(C) \geq d$. Together with the upper bound in Lemma 26, this implies the following corollary.

▶ **Corollary 27** (Rank bound for multilinear $\Sigma\Pi\Sigma(k,d,\rho)$ circuits with no dependence on d). *Let C be a simple and minimal $\Sigma\Pi\Sigma(k,d,\rho)$ circuit computing the zero polynomial. Then,*

$$\Delta_{syn}(C) \leq 40 \cdot (k^2 \log k + k^2\rho).$$

The following lemma uses the notation of Theorem 17.

▶ **Lemma 28** (Small semantic-rank implies small syntactic-rank, similar to Lemma 2.20 in [20]). *Let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit computing a polynomial f. Suppose that $\Delta_{sem}(f) \leq r$. Then $\Delta_{syn}(C) \leq r + R(k+1, \Delta_{syn}(C))$. In particular, $\Delta_{syn}(C) \leq 2^7 rk^2 \log k$.*

5.2 Semantic Partitions of $\Sigma^k\Pi\Sigma$ Circuits

We next define semantic partitions of $\Sigma^k\Pi\Sigma$ circuits, that correspond to semantic rank in the same manner that syntactic partitions correspond to syntactic rank (recall Definition 18).

▶ **Definition 29** (Semantic Partition). *Let f be a multilinear polynomial. We say that $(f_1, \dots, f_s)$ is a (τ, r)* semantic *partition of f if $f = \sum_{i=1}^{s} f_i$, and*

- *For every $i \in [s]$, $\Delta_{sem}(f_i) \leq r$.*
- *For every $i \neq j \in [s]$, $\Delta_{sem}(f_i + f_j) \geq \tau r$.*

We further say that the partition is realizable *if there exists a $\Sigma^k\Pi\Sigma$ circuit C computing f and a partition of its multiplication gates $(C_1, \dots, C_s)$ such that $[C_i] = f_i$. From now on, we only consider realizable partitions.*

We also often use the term "τ-partition" (either syntactic or semantic) where it is implied that the partition is a (τ, r)-partition for some value of r.

▶ **Corollary 30.** *Let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit. Every (τ, r)-semantic partition of $[C]$ is also a (τ', r')-syntactic partition of C with $r' = 2^7 k^2 \log k \cdot r$ and $\tau' = \tau/(2^7 k^2 \log k)$.*

▶ **Corollary 31.** *Let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit. Let $(C_1, \dots, C_s)$ be a (τ, r)-syntactic partition of C. Then $([C_1], \dots, [C_s])$ is a (τ', r)-semantic partition of $[C]$ with $\tau' = \tau/(2^7 k^2 \log k)$.*

5.2.1 Uniqueness Properties of Semantic Partitions

In the full version of this paper, we prove:

▷ Claim 32 (Lower rank implies finer partition). Let $\tau > 40k^2 \log k + k^2$. Let C, D be two minimal multilinear $\Sigma^k\Pi\Sigma$ circuits computing the same polynomial f. Let $C = \sum_{i=1}^{s} C_i$ be a partition of the multiplication gates in C and similarly $D = \sum_{i=1}^{s'} D_i$ a partition of the gates in D. Let $f_i = [C_i]$ and suppose that $(f_1, \dots, f_s)$ is a (τ, r_1)-semantic partition of f. Similarly, let $g_i = [D_i]$ and suppose that $(g_1, \dots, g_{s'})$ is a (τ, r_2)-semantic partition of f. Assume $r_1 \geq r_2$.

Then, for every $i \in [s]$ there is a subset $S_i \subseteq [s']$ such that

$$f_i = \sum_{j \in S_i} g_j$$

and the subsets $S_1, \dots, S_s$ form a partition of $[s']$.

▶ **Corollary 33.** *Let C and D be as in Claim 32. If $s = s'$ then there is a permutation π of $[s]$ such that $f_i = g_{\pi(i)}$.*

We also prove:

▶ **Corollary 34** (Uniqueness of maximal partition regardless of representation). *Let $\tau > R_M(2k) + 2^8 k^2 \log k$. Let C, D be two minimal multilinear $\Sigma^k\Pi\Sigma$ circuits computing the same polynomial f. Let $(C_1, \dots, C_s)$ be a τ-semantic partition of the multiplication gates in C of minimal semantic rank. Similarly let $(D_1, \dots, D_{s'})$ be a τ-semantic partition of D of minimal semantic rank. Then $s = s'$ and there is a permutation π such that for every $i \in [s]$, $[C_i] = [D_{\pi(i)}]$.*

5.2.2 An Algorithm for Computing Partitions

We first note that computing the semantic rank of a polynomial f in $\Sigma^k\Pi\Sigma$ can be done in randomized polynomial time given black box access to f.

▶ **Lemma 35.** *There exists a randomized polynomial time algorithm that, given black box access to a polynomial $f \in \Sigma^k\Pi\Sigma$ computes $\Delta_{sem}(f)$.*

In the full version of this paper, we describe a semantic clustering algorithm that outputs a (τ, r) partition of a circuit C. Recall that Lemma 20 implies that the Karnin-Shpilka *syntactic* clustering algorithm returns *syntactic* clusters. This allows us to obtain some guarantees on the output of the algorithm.

▷ Claim 36. There's an algorithm that for every τ, runs in time at most $2^{k^2} \cdot \mathsf{poly}(n)$ and outputs a (τ, r) partition where

$$r \le R_M(2k) \cdot k^{\lceil \log_k(\tau \cdot 2^7 k^2 \log k)\rceil \cdot (k-2)} \le R_M(2k) \cdot 2^{7k} k^{4k} \tau^{k-2}.$$

▶ Remark 37. Note that Corollary 34 shows that the semantic partition with the minimal semantic rank is unique regardless of the representation. Hence the output of the semantic clustering algorithm does not depend on the circuit C but only on the polynomial f it computes.

5.2.3 Semantic Partitions under Restrictions

Corollary 34 proves that any maximal semantic partition is unique. However, in our reconstruction algorithm we shall consider restrictions of the unknown polynomial to subsets of the variables. Hence, we will need a stronger property.

The next claim shows that for every multilinear polynomial, $f \in \Sigma^k\Pi\Sigma$, there exists a (τ_1, r)-semantic partition with the special property that its rank bound, r, is upper bounded as a function of τ_0, which is much smaller than τ_1.

▷ Claim 38. For every function $\varphi : \mathbb{N} \to \mathbb{N}$, every $\tau_{\min} \in \mathbb{N}$ and for every multilinear $f \in \Sigma^k\Pi\Sigma$ there is $\tau_{\min} \le \tau_0 \le \tau(k) = R_M(2k)^{\varphi(k)^k}$ such that there is a (τ_1, r)-semantic partition of f with:

- $\tau_1 = \tau_0^{\varphi(k)}$.
- $r \le R_M(2k) 2^{7k} k^{4k} \tau_0^{k-2}$.

6 Learning Low Degree Polynomials

As in [5], we start by providing an algorithm, which is efficient when the degree d is very small. All the details are omitted from this version.

▶ **Corollary 39.** *Let $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ be a set-multilinear polynomial computed by a degree d, set-multilinear depth-3 circuit. Suppose $\mathbf{x} = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_d$ and $|\mathbf{x}_i| \le n$ for all i. Then, there is a randomized algorithm that given n, k, d and black-box access to f outputs a set-multilinear depth-3 circuit with top fan-in k that computes f, in time $\mathsf{poly}\left(n, c, (dk)^{O(d^2k^3)^{d^2k^2}}\right)$.*

▶ **Lemma 40.** *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial computed by set-multilinear $\Sigma^k\Pi\Sigma$ circuit C with $\Delta_{sem}(C) \le r$. Then, there is a randomized algorithm that given k, r and black-box access to f outputs a set-multilinear $\Sigma^k\Pi\Sigma$ circuit computing f, where $k' \le k$ is the smallest possible fan-in, in time $\mathsf{poly}\left(n, c, (rk)^{O(r^3k^2)^{r^2k^2}}\right)$.*

7 Efficient Construction of Cluster Preserving Sets

In order to reconstruct general multilinear $\Sigma^k\Pi\Sigma$ circuits, we would again like to follow the steps of [5]. However, as some of our definitions are different, and we replace some brute force steps with algorithmically efficient steps, we're required to make substantial changes in the algorithm. In particular we replace their use of the notion of "rank preserving subspaces" with an explicit construction of a subset B of the variables that, in some sense, preserves the structure of semantic clusters of f.

In the full version, we give an algorithm that attempts to construct a set B together with a vector $\mathbf{a}$ such that the clusters of $f|_{B,\mathbf{a}}$ (with respect to a certain semantic τ-partition), found by the algorithm of Lemma 35, are in one-to-one correspondence with the clusters that the same algorithm would have outputted on f. Our algorithm receives τ as a parameter.

We now explain what guarantees we get on the outputs $(B, \mathbf{a})$ of the algorithm.

▷ Claim 41. The algorithm for finding a cluster preserving set, when given the parameter τ guaranteed in Claim 42 as input, runs in time $\mathsf{poly}(n) \cdot k^{k^{k^{k^{\mathsf{poly}(k)}}}}$ and returns a set B of size at most $k^{k^{O(k)}}$.

The following important claim shows that if $(B, \mathbf{a})$ is the output of the algorithm, then for the "correct" choice of τ, $f|_{B,\mathbf{a}}$ preserves the clusters (with respect to a τ-semantic partition) of f.

▷ Claim 42. Let $f \in \Sigma^k\Pi\Sigma$ be a multilinear polynomial and let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ computing f. There exists a non-zero polynomial Γ_C of degree at most $n^{k^{k^{O(k)}}}$ such that if $B, \mathbf{a}$ are the outputs of the algorithm on f, and $\Gamma_C(\mathbf{a}) \neq 0$, then the following holds: Consider the semantic partition of f, $f = \sum_{i=1}^{s} f_i$, given by Claim 38 with $\varphi(k) = k^2$ and $\tau_{\min} = R_M(2k) \cdot 2^{7k+20} \cdot k^{4k+4}$. Let τ_0, τ_1, r be its parameters as promised by the claim and let $\tau = \tau_0^k$. Let D be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit computing $f|_{B,\mathbf{a}}$. Then, the output of the semantic clustering algorithm on D with parameter τ, denoted by $[D] = \sum_{i=1}^{s'} g_i$, satisfies:

1. $s' = s$.
2. There is a permutation π on $[s]$ such that $g_{\pi(i)} = (f_i)|_{B,\mathbf{a}}$.
3. $\Delta_{\text{sem}}(g_{\pi(i)}) = \Delta_{\text{sem}}(f_i)$.

In particular, the g_i's also form a (τ, r) partition.

Proof for both of these claims appear in the full version.

8 Reconstruction Algorithm for Multilinear $\Sigma^k\Pi\Sigma$ Circuits

In this section we provide our algorithm for learning multilinear $\Sigma^k\Pi\Sigma$ circuits.

We start by explaining how the results of the previous sections imply that we can get black box access to the clusters on arbitrary points. In the previous section, we picked a random $\mathbf{a}$ in and argued about the clusters of $f|_{B,\mathbf{a}}$. We'd like to obtain similar claims about the clusters of $f|_{B,\mathbf{b}}$, assuming $\mathbf{b}$ doesn't satisfy certain degeneracy conditions. We say that an output $(B, \mathbf{a})$ of the algorithm from Section 7 is *good* if it satisfies $\Gamma_C(\mathbf{a}) \neq 0$, where Γ_C is the polynomial defined in Claim 42.

▷ Claim 43. Let $f \in \Sigma^k\Pi\Sigma$ be a multilinear polynomial and let C be a minimal $\Sigma^k\Pi\Sigma$ computing f. Let $(B, \mathbf{a})$ be a good output on f.

Consider a partition of f, $f = \sum_{i=1}^{s} f_i$, as given by Claim 38 with $\varphi(k) = k^2$ and $\tau_{\min}$ as in Claim 42. Let τ_0, τ_1, r be its parameters as promised by the claim. Denote $\tau = \tau_0^k$.

Then, there exists a polynomial $\Theta_{B,C}$ of degree at most $2n^7$ such that $\Theta_{B,C}(\mathbf{a}) \neq 0$, and the following property holds: for every $\mathbf{b} \in \mathbb{F}^n$ such that $\Theta_{B,C}(\mathbf{b}) \neq 0$ and circuit D computing $f|_{B,\mathbf{b}}$, it holds that the output of the semantic clustering algorithm, when given D and τ as input, which we denote $[D] = \sum_{i=1}^{s'} g_i$, satisfies:

1. $s' = s$,
2. $g_i = (f_i)|_{B,\mathbf{b}}$, up to reordering of the indices,
3. $\Delta_{\text{sem}}(g_i) = \Delta_{\text{sem}}(f_i)$.

In particular, the g_i's also form a (τ, r) partition.

8.1 Cluster Evaluation

In the full version of the paper we explain how to evaluate the clusters at arbitrary points. Recall that what we have is access to the clusters $f_i|_{B,\mathbf{a}}$ so we'd like to replace $\mathbf{a}$ by an arbitrary point $\mathbf{b} \in \mathbb{F}^n$. We do it in several stages as in [5]. Basically, we replace all uses of their Lemma 6.14 by our Claim 43, to prove:

▶ **Lemma 44** (Similar to Lemma 6.19 in [5]). *Let $f \in \Sigma^k\Pi\Sigma$ be a multilinear polynomial and let C be a $\Sigma^k\Pi\Sigma$ circuit computing f. Let $(B, \mathbf{a})$ be good outputs on f. Let $f|_{B,\mathbf{a}} = \sum_{i=1}^{s} f_i|_{B,\mathbf{a}}$ be the output of the semantic clustering algorithm on $f|_{B,\mathbf{a}}$.*

Then, there exists an algorithm that, given any $\mathbf{b} \in \mathbb{F}^n$, runs in time $2^{k^2} \cdot \mathsf{poly}(n)$ and outputs $(f_1|_{B,\mathbf{b}}, \ldots, f_s|_{B,\mathbf{b}})$.

8.2 The Reconstruction Algorithm

In the full version of the paper, we give our reconstruction algorithm for multilinear $\Sigma^k\Pi\Sigma$ circuits, to prove:

▶ **Theorem 45.** *Suppose $|\mathbb{F}| > n^{k^{k^{O(k)}}}$. There's a randomized algorithm that, given black box access to a polynomial f computed by a multilinear $\Sigma^k\Pi\Sigma$ circuit. with high probability, returns a multilinear $\Sigma^k\Pi\Sigma$ circuit $\tilde{C}$ computing f in time $\mathsf{poly}(n) \cdot k^{k^{k^{\mathsf{poly}(k)}}}$.*

8.3 Proper Learning of Depth-3 Set-Multilinear Circuits

With small changes, we can modify the algorithm in order to prove Theorem 3. The details appear in the full version.

References

1 Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM Journal of Computing*, 44(3):669–697, 2015. `doi:10.1137/140975103`.

2 Manindra Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, pages 67–75, 2008. `doi:10.1109/FOCS.2008.32`.

3 Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynominal interpolation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC 1988)*, pages 301–309. ACM, 1988. `doi:10.1145/62212.62241`.

4 Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Reconstruction of depth-4 multilinear circuits. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2020)*, pages 2144–2160. SIAM, 2020. doi:10.1137/1.9781611975994.132.

5 Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Reconstruction algorithms for low-rank tensors and depth-3 multilinear circuits. In *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing (STOC 2021)*, pages 809–822. ACM, 2021. doi:10.1145/3406325.3451096.

6 Markus Bläser, Christian Ikenmeyer, Gorav Jindal, and Vladimir Lysikov. Generalized matrix completion and algebraic natural proofs. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC 2018)*, pages 1193–1206. ACM, 2018. doi:10.1145/3188745.3188832.

7 Nader H. Bshouty. Exact learning from membership queries: Some techniques, results and new directions. In *Algorithmic Learning Theory – 24th International Conference, ALT 2013*, volume 8139 of *Lecture Notes in Computer Science*, pages 33–52. Springer, 2013. doi:10.1007/978-3-642-40935-6_4.

8 Enrico Carlini. Reducing the number of variables of a polynomial. In *Algebraic Geometry and Geometric Modeling*, pages 237–247, 2006. doi:10.1007/978-3-540-33275-6_15.

9 David A. Cox, John B. Little, and Donal O'Shea. *Ideals, Varieties and Algorithms.* Undergraduate texts in mathematics. Springer, 2007. doi:10.1007/978-0-387-35651-8.

10 Zeev Dvir and Amir Shpilka. Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *SIAM J. Comput.*, 36(5):1404–1434, 2007. Preliminary version in the *37th Annual ACM Symposium on Theory of Computing (STOC 2005)*. doi:10.1137/05063605X.

11 Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 867–875, 2014. doi:10.1145/2591796.2591816.

12 Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*, pages 243–252, 2013. Full version at arXiv:1209.2408. doi:10.1109/FOCS.2013.34.

13 Ankit Garg, Neeraj Kayal, and Chandan Saha. Learning sums of powers of low-degree polynomials in the non-degenerate case. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2020)*, pages 889–899. IEEE, 2020. doi:10.1109/FOCS46700.2020.00087.

14 Zeyu Guo and Rohit Gurjar. Improved explicit hitting-sets for roabps. In *Proceedings of the 24th International Workshop on Randomization and Computation (RANDOM 2020)*, volume 176 of *LIPIcs*, pages 4:1–4:16, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.4.

15 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic circuits: A chasm at depth 3. *SIAM J. Comput.*, 45(3):1064–1079, 2016. doi:10.1137/140957123.

16 Ankit Gupta, Neeraj Kayal, and Satyanarayana V. Lokam. Efficient reconstruction of random multilinear formulas. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011)*, pages 778–787. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.70.

17 Ankit Gupta, Neeraj Kayal, and Satyanarayana V. Lokam. Reconstruction of depth-4 multilinear circuits with top fan-in 2. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC 2012)*, pages 625–642. ACM, 2012. doi:10.1145/2213977.2214035.

18 Johan Håstad. Tensor rank is np-complete. *J. Algorithms*, 11(4):644–654, 1990. doi:10.1016/0196-6774(90)90014-6.

19 Zohar Shay Karnin and Amir Shpilka. Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity, CCC 2008, 23-26 June 2008, College Park, Maryland, USA*, pages 280–291. IEEE Computer Society, 2008. doi:10.1109/CCC.2008.15.

20 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)*, pages 274–285. IEEE Computer Society, 2009. doi:10.1109/CCC.2009.18.

21 Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, pages 1409–1421. SIAM, 2011. doi:10.1137/1.9781611973082.108.

22 Neeraj Kayal, Vineet Nair, and Chandan Saha. Average-case linear matrix factorization and reconstruction of low width algebraic branching programs. *Comput. Complex.*, 28(4):749–828, 2019. doi:10.1007/s00037-019-00189-0.

23 Neeraj Kayal and Shubhangi Saraf. Blackbox polynomial identity testing for depth-3 circuits. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, 2009. doi:10.1109/FOCS.2009.67.

24 Adam Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC 2001)*, pages 216–223, 2001. doi:10.1145/380752.380801.

25 Pascal Koiran. Arithmetic circuits: The chasm at depth four gets wider. *Theoretical Computer Science*, 448:56–65, 2012. doi:10.1016/j.tcs.2012.03.041.

26 Nitin Saxena and C. Seshadhri. An almost optimal rank bound for depth-3 identities. *SIAM J. Comput.*, 40(1):200–224, 2011. doi:10.1137/090770679.

27 Nitin Saxena and C. Seshadhri. Blackbox identity testing for bounded top-fanin depth-3 circuits: The field doesn't matter. *SIAM J. Comput.*, 41(5):1285–1298, 2012. Preliminary version in the *43rd Annual ACM Symposium on Theory of Computing (STOC 2011)*. doi:10.1137/10848232.

28 Nitin Saxena and C. Seshadhri. From sylvester-gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits. *J. ACM*, 60(5):33:1–33:33, 2013. doi:10.1145/2528403.

29 Yaroslav Shitov. How hard is the tensor rank? *arXiv preprint*, 2016. arXiv:1611.01559.

30 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5:207–388, March 2010. doi:10.1561/0400000039.

31 Gaurav Sinha. Reconstruction of real depth-3 circuits with top fan-in 2. In *Proceedings of the 31st Annual Computational Complexity Conference (CCC 2016)*, volume 50 of *LIPIcs*, pages 31:1–31:53, 2016. doi:10.4230/LIPIcs.CCC.2016.31.

32 Gaurav Sinha. Efficient reconstruction of depth three arithmetic circuits with top fan-in two. In Mark Braverman, editor, *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ICTS 2022)*, volume 215 of *LIPIcs*, pages 118:1–118:33. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.118.

33 Joseph Swernofsky. Tensor rank is hard to approximate. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018*, volume 116 of *LIPIcs*, pages 26:1–26:9, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.26.

34 Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Inf. Comput.*, 240:2–11, 2015. Preliminary version in the *38th International Symposium on the Mathematical Foundations of Computer Science (MFCS 2013)*. doi:10.1016/j.ic.2014.09.004.