Electronic Colloquium on Computational Complexity, Report No. 115 (2015)

A Guide to Learning Arithmetic Circuits

Ilya Volkovich [*]

Abstract

An *arithmetic circuit* is a directed acyclic graph in which the operations are $\{+, \times\}$. In this paper, we exhibit several connections between learning algorithms for arithmetic circuits and other problems. In particular, we show that:

1. Efficient learning algorithms for arithmetic circuit classes imply explicit exponential lower bounds.
2. General circuits and formulas can be learned efficiently with membership and equivalence queries iff they can be learned efficiently with membership queries only.
3. Low-query, learning algorithms for certain classes of circuits imply explicit rigid matrices.
4. Learning algorithms for multilinear depth-3 and depth-4 circuits must compute square roots.

1 Introduction

Revealing a hidden function from a set of examples is a natural and basic problem referred to as learning. The purpose of this paper is to serve as a guide for learning arithmetic circuits by pointing out approaches one must avoid or take with care. We achieve this goal by showing a relation between learning and other problems related to arithmetic circuits.

1.1 Arithmetic Circuits

Arithmetic circuits is the standard computational model for computing polynomials. A circuit C in the variables $X = \{x_1, \ldots, x_n\}$ over the field $\mathbb{F}$ is a labelled directed acyclic graph. The inputs (nodes of in-degree zero) are labelled by variables from X or by constants from the field. The internal nodes are labelled by $+$ or $\times$, computing the sum and product, resp., of the polynomials on the tails of incoming edges (subtraction is obtained using the constant -1). A *formula* is a circuit whose nodes have out-degree one (namely, a tree). The output of a circuit (formula) is the polynomial computed at the output node. The *size* of a circuit (formula) is the number of gates in it. The *depth* of the circuit (formula) is the length of a longest path between the output node and an input node.

A circuit is *multilinear* if the inputs to its multiplication gates are variable-disjoint. A circuit is *set-multilinear* if there exists a partition of the variables $X_1 \cup \ldots \cup X_\ell$ such that the inputs to all the multiplication gates cannot contain variables for the same set. Particular examples of set-multilinear polynomials are the Permanent and the Determinant. Finally, we shall say that C is an *(n, s, d)-circuit* if it is an n-variate arithmetic circuit of size s and of degree at most d. Sometimes we shall think of an arithmetic circuit and of the polynomial that it computes as the same objects.

[*]Department of EECS, CSE Division, University of Michigan, Ann Arbor, MI. Email: `ilyavol@umich.edu`.

ISSN 1433-8092

1.2 Arithmetic Circuit Classes

In this paper we will usually refer to a class of arithmetic circuits $\mathcal{C}$. It should be thought or as either the class of general circuits or formulas, of as some restricted class. In particular, we will be interested in the following classes:

- s-sparse polynomial is a polynomial that has at most s (non-zero) monomials.
- Depth-3 $\Sigma\Pi\Sigma(k)$ circuits computes polynomials of the form $C(\bar{x}) = \sum_{i=1}^{k} \prod_{j=1}^{d_i} L_{ij}(\bar{x})$ where L_{ij}-s are linear forms.
- Depth-4 $\Sigma\Pi\Sigma\Pi(k)$ circuits computes polynomials of the form $C(\bar{x}) = \sum_{i=1}^{k} \prod_{j=1}^{d_i} P_{ij}(\bar{x})$ where P_{ij}-s are sparse polynomials.
- Read-once Oblivious Algebraic Branching Programs (ABPs) compute polynomials of the form $C(\bar{x}) = \bar{v}^t \cdot D_n(x_{\sigma(n)}) \cdot D_{n-1}(x_{\sigma(n-1)}) \cdot \ldots \cdot D_1(x_{\sigma(1)}) \cdot \bar{u}$ when each $D_i(z)$ is a matrix and $\sigma : [n] \to [n]$ is permutation.

1.3 Exact Learning

Angluin's Exact Learning model consists of a (computationally-bounded) learner and an (all-powerful) teacher. The learner's goal is to output a target function f from a given function class $\mathcal{C}$. To do so, the learner is allowed to query the value $f(\bar{x})$ on any input $\bar{x}$ (membership query). In addition, the learner is also allowed to propose a hypothesis $\hat{f}$ and ask the teacher whether it is equivalent to f (equivalence query). If this is indeed the case, the learner has achieved his goal. Otherwise, the teacher presents the learner with a counterexample $\bar{a}$ for which $f(\bar{a}) \neq \hat{f}(\bar{a})$. We say that a function class $\mathcal{C}$ is *exactly learnable* if there exists a learner which given any $f \in \mathcal{C}$, in time polynomial in n and $|f|$ (the size of f in the representation scheme) outputs a hypothesis $\hat{f}$ such that $f(\bar{x}) = \hat{f}(\bar{x})$ for all $\bar{x}$, using membership and equivalence queries.

1.4 Polynomial Identity Testing

Let $\mathcal{C}$ be a class of arithmetic circuits defined over some field $\mathbb{F}$. The polynomial identity testing problem (PIT for short) for $\mathcal{C}$ is the question of deciding whether a given circuit form $\mathcal{C}$ computes the identically zero polynomial. This question can be considered both in the white-box and the black-box models. In the white-box model, the circuit is given to us as an input. In the black-box model, we can only access the polynomial computed by the circuit using (membership) queries. It is easy to see that a black-box PIT algorithm is equivalent to a "hitting set". That is, a set of points $\mathcal{H}$ such that for every $C \in \mathcal{C}$: $C \equiv 0$ iff $C|_{\mathcal{H}} \equiv 0$.

The importance of this fundamental problem stems from its many applications. For example, the deterministic primality testing algorithm of [AKS04] and the fast parallel algorithm for perfect matching of [MVV87] are based on solving PIT problems.

PIT has a well-known randomized algorithm [Sch80, Zip79, DL78] by evaluating the circuit at a random point. However, we are interested in the problem of obtaining efficient *deterministic* algorithms for it. This question has received a lot of attention recently but its deterministic complexity is still far from being well-understood. For a more detailed discussion, we refer the reader to the excellent survey [SY10].

1.5 Polynomial Reconstruction

The *reconstruction* problem for arithmetic circuits can be seen as an algebraic analog of the learning problem. Given a black-box (i.e. oracle), access to a degree d polynomial $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ computable by an arithmetic circuit of size s from a circuit class $\mathcal{C}$, output a circuit from $C \in \mathcal{C}$ that computes P. More generally, the algorithm is allowed to output a circuit of size $\mathrm{poly}(s)$ and a formal degree at most $\mathrm{poly}(d)$. Similar to the learning scenario, an efficient reconstruction algorithm must run in time polynomial in n, s, d.

A key difference between learning and reconstruction is that in the reconstruction problem we seek to output a circuit that computes the target polynomial as a formal sum of monomials and not just as a function. This poses an implicit requirement that the underlying field $\mathbb{F}$ is larger than the degree of the polynomial it is trying to reconstruct. Otherwise, the algorithm is allowed to query the polynomial on a sufficiently (polynomially) large extension field. Particular examples include reconstruction algorithms for sparse polynomials [KS01, LV03], $\Sigma\Pi\Sigma(k)$ circuits [KS09a] and read-once formulas [BHH95, SV14].

In this regard, a natural question is the dependence of the running time on the field size. Let $\log |\mathbb{F}|$ denote the bit-complexity of the coefficients in the underlying polynomials. For finite fields, the bit-complexity corresponds with the log of the field size. Ideally, we would like the running time of the algorithm to be polynomial in the bit-complexity, that is $\mathrm{poly}(\log |\mathbb{F}|)$. Unfortunately, this is not always the case (see e.g. [KS09a]). For a more detailed discussion, we refer the reader to the excellent survey [SY10].

1.6 Elimination of Equivalence Queries in the Arithmetic Setting

Unlike the Boolean setting, where the equivalence queries are essential (see [Ang87]), in the arithmetic setting, an efficient black-box PIT algorithm can be used to eliminate them. More specifically, let $\mathcal{H}$ be a hitting set for $\mathcal{C}$ and let $C \in \mathcal{C}$ be the target circuit. Given an equivalence query $\hat{C}$, the learning algorithm tests if $\hat{C}|_{\mathcal{H}} \equiv C|_{\mathcal{H}}$, using membership queries to C. If $\hat{C}(\bar{a}) \neq C(\bar{a})$ for some $\bar{a} \in \mathcal{H}$, the algorithm answers the query with $\bar{a}$. Otherwise, the algorithm stops and outputs $\hat{C}$.

Of course, the above works if $\mathcal{C}$ is closed under subtraction, that is $C - \hat{C} \in \mathcal{C}$, since by definition $C \equiv \hat{C}$ iff $C|_{\mathcal{H}} \equiv \hat{C}|_{\mathcal{H}}$. This is typically the case when the learning algorithm is proper. For example, Beimel et al. [BBB$^+$00] gave a proper exact learning algorithm for read-once oblivious ABPs. Later, Forbes & Shpilka [FS13] devised an efficient black-box PIT algorithm for that class of circuits. As read-once oblivious ABPs are closed under subtraction, this results in a learning algorithm for read-once oblivious ABPs that uses membership queries only. In the general case, one can compare $\hat{C}(\bar{a})$ and $C(\bar{a})$ at a random point $\bar{a}$, though making the algorithm randomized.

1.7 Circuit Lower Bounds

Attaining explicit lower bounds for both arithmetic and Boolean circuits is one of the biggest challenges in the theory of computation and has been the focus of much research. It is not hard to prove lower bounds for polynomials of high degree d. Therefore, the goal is to prove lower bounds for polynomials of low degree, typically $d = \mathrm{poly}(n)$.

So far, the best known lower bounds for arithmetic circuits are $\Omega(n \log d)$, due to Baur & Strassen [BS83]. When restricting ourselves to multilinear circuits, we can get better bounds: $\Omega(n^{4/3}/\log^2 n)$ for multilinear circuits due to Raz et al. [RSY08], and $n^{\Omega(\log n)}$ for multilinear formulas due to Raz [Raz09]. It is an interesting open question to improve any of these bounds.

1.8 Matrix Rigidity

The notion of matrix rigidity was introduced by Valiant in [Val77]. For $r, s \in \mathbb{N}$, we say that a matrix S is *s-sparse* if each row of the matrix contains at most s non-zero entries. A matrix $A \in \mathbb{F}^{m \times n}$ is (r,s)-*rigid* if A cannot be written as a sum of two matrices $A = L + S$ such that $\mathrm{rank}(L) \leq r$ and S is s-sparse. In the same paper, Valiant showed that if a matrix A with $m = O(n)$ is $(\Omega(n), n^{\varepsilon})$-rigid th then the the linear map $A \cdot \bar{x}$, induced by A, cannot be computed by a circuit of size $\mathcal{O}(n)$ and depth $\mathcal{O}(\log n)$.

While it is easy to see that a random matrix satisfies the above conditions, a big body of research was invested in attempts to construct rigid matrices explicitly. Yet, the best known result is due to Saraf & Yekhanin [SY11] that gives a construction of $(n/2, s)$-rigid matrices with $m = \exp(s) \cdot n$. For a further discussion of the work on matrix rigidity, we refer the reader to the excellent survey [Lok09].

1.9 Square Root Extraction

Given $\beta \in \mathbb{F}$, output $\alpha \in \mathbb{F}$ such that $\alpha^2 = \beta$. In addition to being a natural problem, square root extraction plays an import role in cryptography [GM84, Rab05]. The best known *deterministic* root extraction algorithms over the finite fields have polynomial dependence on $|\mathbb{F}|$, i.e. exponential in the bit-complexity of the coefficients (see e.g. [Sho91, GG99, GKL04, Kay07])[1]. In other words, the naive brute-force algorithm is essentially optimal. While in the *randomized* setting, this dependence is polynomial in $\log |\mathbb{F}|$. In particular, there is no known efficient deterministic root extraction algorithm when $\mathbb{F}$ is large. One reason for that is the two-way connection between modular square root extraction and finding quadric non-residues. Over fields with characteristic 0 (e.g. $\mathbb{Q}$), both the *deterministic* and the *randomized* complexities are polynomial in $\log |\mathbb{F}|$ (see e.g. [LLL82]).

1.10 Our Results

We begin with our main result: exact learning algorithms for arithmetic circuit classes imply explicit exponential lower bounds. The result is obtained by combining and extending the techniques of [KKO13] with the techniques of [HS80, Agr05].

Theorem 1. *Let $\mathcal{C}$ be an arithmetic circuit class over the field $\mathbb{F}$. If $\mathcal{C}$ is exactly learnable then for every $n \in \mathbb{N}$, there exists an explicit*[2] *multilinear polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ that requires circuits of size $2^{\Omega(n)}$ from $\mathcal{C}$.*

Using the connection between lowers bounds and PIT algorithms [KI04], as well as depth reduction techniques [AV08], we are able to convert efficient learning algorithms for certain (relatively general) circuit classes into efficient PIT algorithms.

Theorem 2. *Let $\mathcal{C} \in \{$general circuits, general formulas, $\Sigma\Pi\Sigma\Pi$ circuits$\}$ be an arithmetic circuit class over the field $\mathbb{F}$. If $\mathcal{C}$ is exactly learnable then there exists a quasi-polynomial time black-box PIT algorithm for the class of general arithmetic circuits.*

In the case when the learning algorithm is proper or, more generally, outputs circuits of polynomial degree as its hypotheses, we can eliminate the equivalence queries by implementing the procedure outlined in Section 1.6.

[1] In fact, for fields of size p^e this dependence is polynomial in p.

[2] Computable in time $2^{\mathcal{O}(n)}$.

Theorem 3. *Let $\mathcal{C} \in \{$general circuits, general formulas, $\Sigma\Pi\Sigma\Pi$ circuits$\}$ be an arithmetic circuit class over the field $\mathbb{F}$. If $\mathcal{C}$ is exactly learnable with hypotheses being general circuits of polynomial degree then $\mathcal{C}$ is learnable with membership queries only in quasi-polynomial time. In addition, if the algorithm is proper, then $\mathcal{C}$ can be reconstructed in quasi-polynomial time.*

We see this theorem as evidence that reconstruction is, perhaps, a more natural framework for learning arithmetic circuits.

Zeev Dvir pointed out that the same techniques can be used to show that "low-query" exact learning algorithms for a certain class of multilinear polynomials give rise to explicit rigid matrices.

Theorem 4. *Let $r, s \in \mathbb{N}$. Let $\mathcal{C}_{(r,s)}$ be a class of multilinear polynomials of the form:*

$$\mathcal{C}_{(r,s)} = \left\{ P(\bar{x}, \bar{y}) = \bar{x}^t \cdot L \cdot \bar{y} + \bar{x}^t \cdot S \cdot \bar{y} \;\middle|\; L, S \in \mathbb{F}^{m \times n},\ \operatorname{rank}(L) \leq r,\ S \text{ is } s\text{-sparse.} \right\}$$

If $\mathcal{C}_{(r,s)}$ is exactly learnable with $< mn$ queries, then there exist explicit (r, s)-rigid matrices.

Combined with [Val77], this result has the same flavor as the results of [FK09, KKO13] and Theorem 1: learning algorithms imply lower bounds. Using the same techniques, one could show that devising a learning algorithm with $< nm$ queries for the class of linear maps computable by circuits of size $\mathcal{O}(n)$ and depth $\mathcal{O}(\log n)$ would entail the same lower bounds. We would like to point out that while the learning algorithms can output the polynomials in any representation, the bound on the number of queries is tight since any $m \times n$ matrix is learnable using mn membership queries.

On a different note, we study the question of the dependence of the running time of the learning/reconstruction algorithm on the field size. In [KS09a], a deterministic reconstruction algorithm for $\Sigma\Pi\Sigma(k)$ circuits was given. The running time of the algorithm has polynomial dependence on field the size. In addition, in [GKL12], a randomized reconstruction algorithm for multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits was given with polylogarithmic dependence on the field size. Recently, this algorithm was derandomized in [Vol15] preserving the polylogarithmic dependence on the field size. The only additional assumption was that the algorithm is given a square root oracle[3]. As we can implement such an oracle in deterministic poly($|\mathbb{F}|$) time (see Section 1.9), this results in a deterministic reconstruction algorithm with a polynomial dependence on $|\mathbb{F}|$. To the best of our knowledge, there is no known deterministic reconstruction algorithm for multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits with a sub-polynomial dependence on $|\mathbb{F}|$. Moreover, this still holds true for multilinear $\Sigma\Pi\Sigma(2)$ circuits and even set-multilinear $\Sigma\Pi\Sigma(2)$ circuits[4]. Our next theorem suggests an explanation for this state of affairs. We show that any proper learning/reconstruction algorithm for these classes of circuits must compute square roots.

Theorem 5. *Let $k \in \mathbb{N}$. If the class of (set)-multilinear $\Sigma\Pi\Sigma(2)$ or $\Sigma\Pi\Sigma\Pi(2)$ circuits over the field $\mathbb{F}$ is exactly learnable with hypotheses being multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits of polynomial size then the learning algorithm must compute square roots over $\mathbb{F}$.*

Given the state of our current techniques (see Section 1.9), the theorem suggests that any reconstruction algorithm for these classes of circuits must be either randomized or have a polynomial dependence on the field size. This partially answers an open question of [GKL12] that asks whether it is possible to remove the polynomial dependence on the field size from the reconstruction algorithm of [KS09a] even for $\Sigma\Pi\Sigma(2)$ circuits.

[3]An algorithm that can extract square roots of field elements efficiently.

[4]The learning algorithm of [BBB$^+$00, KS06] for set-multilinear $\Sigma\Pi\Sigma$ circuits outputs read-once oblivious ABPs as hypotheses.

1.11 Techniques

In this Section, we outline the techniques we use to prove our results.

1.11.1 From Exact Learning to Lower Bounds

We combine and extend the techniques of [KKO13] with the techniques of [HS80, Agr05]. Given an exact learner $\mathcal{A}$, the lower bound of [KKO13] is obtained by running $\mathcal{A}$ and diagonalizing against its queries, thus producing a function that disagrees with the output of $\mathcal{A}$ on at least one point. Applying the same idea naively in the arithmetic setting may result in a polynomial of a high degree.

In order to maintain the low degree of the polynomial, we use a version of the evaluation matrix that was defined in [HS80, Agr05]. Given an assignment $\bar{a} \in \mathbb{F}^n$, the corresponding row $M(\bar{a})$ of the evaluation matrix is the 2^n-length vector containing the evaluations of all the multilinear monomials on $\bar{a}$ (see Definition 2.4). We use the evaluation matrix to ensure that the values of the polynomial on diagonalizing points remain in a low-rank subspace, when we view the multilinear polynomials as a vector space over $\mathbb{F}$. Specifically, throughout the execution the algorithm keeps an evaluation matrix $M(S)$ with independent rows S, that represents the values of the "hard-to-be" polynomial P.

Given a membership query $\bar{a}$ of $\mathcal{A}$, the algorithm first checks if the value of the polynomial P on $\bar{a}$ is predefined by the answers to the previous queries. This is done by checking the linear dependence between the row vector $M(\bar{a})$ and the rows of $M(S)$. If the value is predefined, the algorithm computes $P(\bar{a})$ an returns it to $\mathcal{A}$. Otherwise, the algorithm declares $P(\bar{a}) = 0$ and adds the row vector $M(\bar{a})$ to $M(S)$. Given an equivalence query h of $\mathcal{A}$, the algorithms finds a point $\bar{a} \in \mathbb{F}^n$ such that the row vector $M(\bar{a})$ is independent of the rows of $M(S)$. Given such $\bar{a}$, the algorithm diagonalizes against h by declaring $P(\bar{a}) = h(\bar{a}) - 1$ and returning $\bar{a}$ to $\mathcal{A}$ as a counterexample.

We note that the rank of the matrix $M(S)$ equals to the number q of queries $\mathcal{A}$. Therefore, as long as $q < 2^n$, there always exists $\bar{a} \in \mathbb{F}^n$ such that row vector $M(a)$ is independent of the rows $M(S)$. Moreover, we show that such $\bar{a}$ actually exists in $\{0,1\}^n$. This allows the algorithm to be independent of the field size and, in particular, handle infinite fields. For more details, see Section 3.

Using the connection between lowers bounds and PIT algorithms [KI04], as well as depth reduction techniques [AV08], we are able to convert efficient exact learning algorithms into efficient black-box PIT algorithms. Later on, we use the PIT algorithms to eliminate the equivalence queries.

1.11.2 From Exact Learning to Square Roots

For every $\alpha \in \mathbb{F}$, we construct a family of polynomials $\{P_\alpha^n(\bar{x})\}_{n\in\mathbb{N}}$ computable by set-multilinear $\Sigma\Pi\Sigma(2)$ circuits, that can be evaluated efficiently given α^2 (without the knowledge of α). However, for a constant k and sufficiently large n, every (set)-multilinear $\Sigma\Pi\Sigma(k)$ or even polynomial-size $\Sigma\Pi\Sigma\Pi(k)$ circuit that computes P_α^n must have a factor of the form $x_j \pm \alpha$ in one of its multiplication gates. Consequently, any proper exact learning/reconstruction algorithm $\mathcal{A}$ for these classes of circuits must extract square roots. Moreover, we show how to use such an $\mathcal{A}$ as an oracle, in order to actually compute a square root given $\alpha^2 = \beta \in \mathbb{F}$.

1.12 Related Work

Fortnow & Klivans [FK09] initiated a body of work dedicated to the study of the relations between learning algorithms and circuit lower bounds. In the same paper it was shown that an efficient exact learning algorithm for a Boolean circuit class $\mathcal{C}$ implies that the complexity class $\mathsf{EXP}^{\mathsf{NP}}$ requires circuits of super-polynomial size from $\mathcal{C}$. Following a line of improvements, Klivans et al. [KKO13] replaced $\mathsf{EXP}^{\mathsf{NP}}$ by $\mathsf{DTIME}(n^{\omega(1)})$. In the arithmetic setting, the best result [FK09] gives super-polynomial lower bounds against the complexity class $\mathsf{ZPEXP}^{\mathsf{RP}}$.

Our improvement, therefore, is two-fold: First, we obtain a quantitative improvement replacing $\mathsf{ZPEXP}^{\mathsf{RP}}$ by a subclass EXP and getting an exponential lower bound as opposed to super-polynomial. Second, our lower bound is more structured and constructive since we give an explicit multilinear polynomial, while the result of [FK09] is existential.

Bshouty et al. [BHH95] gave deterministic exact learning and randomized reconstruction algorithms for read-once formulas with addition, multiplication and division gates. Their algorithm is assumed to be given a square root oracle. They also show that any reconstruction algorithm for read-once formulas with these operations must compute square roots. Recently, Shpilka & Volkovich [SV14] gave a deterministic reconstruction algorithm for read-once formulas with addition and multiplication gates (no division) which has a polynomial dependence on $\log(|\mathbb{F}|)$. This demonstrates that the need for square root extraction is not necessarily if we choose to deal with polynomials rather than rational functions. However, using the ideas of Section 5 with the techniques of [AvMV15, SV15], one could show that any reconstruction algorithm for read-twice formulas or even sums of read-once formulas with addition and multiplication gates must compute square roots.

Several other results [Val84, KV94, KS09b, GH11] exhibit a two-way connection between learning and cryptography: basing the hardness of learning on cryptography and constructing cryptographic primitives based on the hardness of learning.

1.13 Organization

The paper is organized as follows: We begin by some basic definitions and notation in Section 2, where we also introduce our main tool. In Section 3, we prove our main result (Theorem 1): exact learning algorithm for arithmetic circuit classes imply explicit exponential lower bounds, as well as Theorems 2 and 3. Next in Section 4, we show how the techniques can be modified to prove Theorem 4. In Section 5, we show that proper learning algorithm for certain circuit classes must compute square roots, thus proving Theorem 5. We conclude with discussion & open questions in Section 6. In Appendix A, we show that PAC and Exact Learning models are essentially equivalent for Arithmetic Circuits. For that reason we are not, considering the PAC model in our paper.

2 Preliminaries

For a positive integer n, we denote $[n] = \{1, \ldots, n\}$. For a polynomial $P(x_1, \ldots, x_n)$, a variable x_i and a field element α, we denote with $P|_{x_i=\alpha}$ the polynomial resulting from setting $x_i = \alpha$. Given a subset $I \subseteq [n]$ and an assignment $\bar{a} \in \mathbb{F}^n$, we define $P|_{\bar{x}_I=\bar{a}_I}$ to be the polynomial resulting from setting $x_i = a_i$ for every $i \in I$. We often denote variables interchangeably by their index or by their label: i versus x_i. For $\bar{v} \in \mathbb{N}^n$ we define $\bar{a}^{\bar{v}} \triangleq \prod_{i=1}^{n} a_i^{v_i}$, when $0^0 \triangleq 1$.

2.1 Matrices

In this section we give definitions related to matrices and prepare the ground for our main tool. For more details about matrices, we refer the reader to [HJ91].

Lemma 2.1. *Let $A \in \mathbb{F}^{n\times m}$ and $\bar{v} \in \mathbb{F}^n$. There exists an algorithm that given A and $\bar{v}$ solves the system $A \cdot \bar{b} = \bar{v}$ using* $\operatorname{poly}(n,m)$ *field operations. That is, the algorithm returns a solution if one exists, or $\bar{b} = \perp$ otherwise.*

Definition 2.2 (Tensor Product)**.** *Let $A \in \mathbb{F}^{m\times n}$ and $B \in \mathbb{F}^{k\times \ell}$. Then the* tensor product of A and B, *$A \otimes B \in \mathbb{F}^{mk\times n\ell}$ is defined as:*

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & & \vdots \\ a_{1m}B & \dots & a_{mn}B \end{bmatrix}.$$

Fact 2.3. $\operatorname{rank}(A \otimes B) = \operatorname{rank}(A) \cdot \operatorname{rank}(B)$.

We now introduce our main tool - the evaluation matrix. This tool has been previous used in [HS80] and [Agr05] to show that efficient black-box PIT algorithms imply circuit lower bounds. In this paper we will focus on evaluation matrices for multilinear polynomials. Let $n \in \mathbb{N}$ and $\bar{a} \in \mathbb{F}^n$. A row of such matrix is a (row) vector of length 2^n listing the evaluations of all n-variate multilinear monomials to $\bar{a}$. Formally:

Definition 2.4 (Evaluation Matrix)**.** *Let $\bar{a} \in \mathbb{F}^n$ and $\bar{v} \subseteq \{0,1\}^n$. We define $M(\bar{a}) \in \mathbb{F}^{1\times 2^n}$ as $M(\bar{a})_{1,\bar{v}} = \bar{a}^{\bar{v}}$. We now extend the definition to sets of assignments.*
Let $S \subseteq \mathbb{F}^n$. We define $M(S) \in \mathbb{F}^{|S|\times 2^n}$ as $M(S)_{\bar{a}} = M(\bar{a})$ for $\bar{a} \in S$, when $M(\emptyset) \in \mathbb{F}^{1\times 2^n} \triangleq \bar{0}$.
We say that $\bar{a}$ is independent of S *if* $\operatorname{rank}(M(S \cup \{\bar{a}\})) > \operatorname{rank}(M(S))$.

To provide additional intuition, we list several important and useful properties of $M(S)$ which we will apply in our proofs.

Lemma 2.5. *Let $n \in \mathbb{N}$ and $S \subseteq \mathbb{F}^n$. Then we have the following:*

1. *Let $\bar{a} \in \mathbb{F}^n$ and $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ be a multilinear polynomial. Denote by $\bar{c}$ the vectors of coefficients of P. Then $P(\bar{a}) = M(\bar{a}) \cdot \bar{c}$.*

2. *$\bar{a}$ is independent of S iff $M(S)^t \cdot \bar{b} = M(\bar{a})^t$ has no solution.*

3. *Suppose $S = S_1 \times S_2 \times \dots \times S_n$ when $S_1, S_2, \dots S_n \subseteq \mathbb{F}$ are of size $|S_i| = 2$ for $i \in [n]$. Then $M(S) = M(S_1) \otimes M(S_2) \otimes \dots \otimes M(S_n)$.*

4. $\operatorname{rank}\left(M\left(\{0,1\}^n\right)\right) = 2^n$.

5. *Suppose $|S| < 2^n$. Then there exists $\bar{a} \in \{0,1\}^n$ such that $\bar{a}$* is independent of S. *Moreover, given S such $\bar{a}$ can be found using* $\operatorname{poly}(n, |S|)$ *field operations.*

Proof. 1. By inspection.

2. $\text{rank}(M(S \cup \{\bar{a}\})) > \text{rank}(M(S))$ iff the (row) vector $M(\bar{a})$ can not be we written as a linear combination of the rows of $M(S)$.

3. By inspection.

4. $M(\{0,1\}) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. By the previous part: $\text{rank}(M(\{0,1\}^n)) = \text{rank}(M(\{0,1\}))^n = 2^n$.

5. Since $\text{rank}(M(S)) < 2^n$ and $\text{rank}(M(\{0,1\}^n)) = 2^n$, the existence of such $\bar{a} \in \{0,1\}^n$ follows from the exchange property of the rank. Therefore, we can find such $\bar{a}$, by going over all the assignments in $\{0,1\}^n$ and checking if $M(S)^t \cdot \bar{b} = M(\bar{a})^t$ has a solution (using Lemma 2.1).

□

3 From Learning to Lower Bounds

In this section we prove our main result (Theorem 1): exact learning algorithms for arithmetic circuit classes imply explicit exponential lower bounds, as well as Theorems 2 and 3. We begin with our main result.

```
Input: $n \in \mathbb{N}, \bar{\alpha} \in \mathbb{F}^n$, exact learner $\mathcal{A}$ via oracle
Output: $P(\bar{\alpha})$.

1  $S \leftarrow \emptyset$, $u$ - a vector indexed by elements of $S$;
2  Run $\mathcal{A}$ with $n, s = 2^{n/4\ell}, d = n$.
   /* Answering a membership query. */
3  Given a membership query point $\bar{a} \in \mathbb{F}^n$ by $\mathcal{A}$:
4  Compute $\bar{b}$ such that $M(S)^t \cdot \bar{b} = M(\bar{a})^t$ (using Lemma 2.1);
5  if $\bar{b} = \perp$ then ($\bar{a}$ is independent of $S$)
6      $S \leftarrow S \cup \{\bar{a}\}$; $\bar{u}(\bar{a}) \leftarrow 0$;
7      Answer $\bar{u}(\bar{a})$ to $\mathcal{A}$;
8  else
9      Answer $\bar{b}^t \cdot \bar{u}$ to $\mathcal{A}$;
   /* Answering an equivalence query. */
10 Given an equivalence query hypothesis $h(\bar{x}) : \mathbb{F}^n \to \mathbb{F}$ by $\mathcal{A}$:
11 Find an assignment $\bar{a} \in \{0,1\}^n$ independent of $S$ (using by Lemma 2.5);
12 $S \leftarrow S \cup \{\bar{a}\}$; $\bar{u}(\bar{a}) \leftarrow h(\bar{a}) - 1$;
13 Answer $\bar{a}$ to $\mathcal{A}$;
   /* Completing the matrix to full rank. */
14 while $|S| < 2^n$ do
15     Find an assignment $\bar{a} \in \{0,1\}^n$ independent of $S$ (using by Lemma 2.5);
16     $S \leftarrow S \cup \{\bar{a}\}$; $\bar{u}(\bar{a}) \leftarrow 0$;
17 Compute $\bar{c}$ such that $M(S) \cdot \bar{c} = \bar{u}$;
18 Output $M(\bar{\alpha}) \cdot \bar{c}$;
```

Algorithm 1: Hard Polynomial from Exact Learner

Lemma 3.1. *Let $\mathcal{C}$ be an arithmetic circuit class over the field $\mathbb{F}$. Suppose there exist an exact leaner $\mathcal{A}$ for $\mathcal{C}$ that learns (n, s, d)-circuits from $\mathcal{C}$ in time s^ℓ when $n, d \leq s$. Then for every $n \in \mathbb{N}$ in time $2^{\mathcal{O}(n)}$ Algorithm 1 computes a multilinear polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ that requires circuits of size $2^{\Omega(n)}$ from $\mathcal{C}$, using $\mathcal{A}$ as an oracle.*

Proof. First, observe that throughout the execution of the algorithm the rows of $M(S)$ are linearly independent. Indeed, in the beginning S is empty. Going forward, the algorithm adds elements to S and thus rows to $M(S)$ (in Lines 6, 12 and 14) making sure the rank is increasing (Lemma 2.5). In addition, the algorithm adds elements to S only when $|S| < 2^n$. Indeed, elements are added following a query from $\mathcal{A}$. Yet, the number of queries it at most $s^\ell \leq 2^{n/4}$. Consequently, in Line 17, $|S| = 2^n$ and $M(S)$ is a full rank $2^n \times 2^n$ matrix. Note the that algorithm computes the same (final) set S and vector $\bar{u}$ in every execution. Let us denote them by S_f and u_f, respectively. Therefore, the exists a unique vector $\bar{c} \in \mathbb{F}^{2^n}$ such that $M(S_f) \cdot \bar{c} = \bar{u}_f$.

Next, we claim consistency with the queries to $\mathcal{A}$. Let $\bar{a} \in \mathbb{F}^n$ be a membership query point by $\mathcal{A}$. Consider S and $\bar{u}$ at the time of the query. If $\bar{a}$ was independent of S, then $\bar{a} \in S_f$ and $\bar{u}_f(\bar{a}) = 0$. Therefore, $M(S_f) \cdot \bar{c} = \bar{u}_f$ implies that

$$P(\bar{a}) = M(\bar{a}) \cdot \bar{c} = \bar{u}_f(\bar{a}) = 0.$$

Otherwise, $M(S)^t \cdot \bar{b} = M(\bar{a})^t$ for some $\bar{b} \in \mathbb{F}^{|S|}$. As $S \subseteq S_f$, we can extend the vector $\bar{b}$ to $\bar{b}_e \in \mathbb{F}^{2^n}$ by adding zeros in the entries indexed by $S_f \setminus S$. Observe that

$$M(S_f)^t \cdot \bar{b}_e = M(S)^t \cdot \bar{b} \text{ and } \bar{b}_e^t \cdot \bar{u}_f = \bar{b}^t \cdot \bar{u}.$$

Therefore, we obtain:

$$P(\bar{a}) = M(\bar{a}) \cdot \bar{c} = \bar{b}^t \cdot M(S) \cdot \bar{c} = \bar{b}_e^t \cdot M(S_f) \cdot \bar{c} = \bar{b}_e^t \cdot \bar{u}_f = \bar{b}^t \cdot \bar{u}.$$

Let $h(\bar{x}) : \mathbb{F}^n \to \mathbb{F}$ be a hypothesis by $\mathcal{A}$ and let $\bar{a} \in \{0,1\}^n$ be the assignment computed in Line 11. Repeating the previous reasoning we get that:

$$P(\bar{a}) = M(\bar{a}) \cdot \bar{c} = \bar{u}_f(\bar{a}) = h(\bar{a}) - 1 \neq h(\bar{a}).$$

Finally, suppose P was computable by a circuit of size s from $\mathcal{C}$. By the definition, $\mathcal{A}$ should output a hypothesis h such that $h \equiv P$. However, as we saw previously, for every hypothesis h by $\mathcal{A}$ there exists an assignment $\bar{a} \in \{0,1\}^n$ such that $P(\bar{a}) \neq h(\bar{a})$ leading to a contradiction.

For the running time, by Lemmas 2.1 and 2.5, the algorithm uses $2^{\mathcal{O}(n)}$ field operations. □

Theorem 1 follows as a corollary of the Lemma. Next, we use the connection between lower bounds and PIT algorithms, and depth reduction techniques to obtain an efficient PIT algorithm, thus proving Theorem 2. We require the following results.

Lemma 3.2 ([KI04]). *Let $\mathbb{F}$ be a field. Suppose that for every $n \in \mathbb{N}$ there exists an explicit multilinear polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ that requires general circuits of size $2^{\Omega(n)}$. Then there exists an algorithm that given $n, d, s \in \mathbb{N}$ in time $(nd)^{\mathcal{O}(\log s)}$ outputs a hitting set $\mathcal{H}$ for general (n, s, d)-circuits.*

Lemma 3.3 ([AV08])**.** *Let $\mathbb{F}$ be a field. Suppose that a multilinear polynomial $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ requires general formulas or $\Sigma\Pi\Sigma\Pi$ circuits of size $2^{\Omega(n)}$. Then P also requires general circuits of size $2^{\Omega(n)}$.*

Theorem 2 follows as corollary from the following Lemma.

Lemma 3.4. *Suppose that the class $\mathcal{C} \in \{$general circuits, general formulas, $\Sigma\Pi\Sigma\Pi$ circuits$\}$ is exactly learnable. Then there exists an algorithm that given $n, d, s \in \mathbb{N}$ in time $(nd)^{\mathcal{O}(\log s)}$ outputs a hitting set $\mathcal{H}$ for general (n, s, d)-circuits.*

Proof. By Theorem 1, for every $n \in \mathbb{N}$ there exists an explicit multilinear polynomial $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ that requires circuits of size $2^{\Omega(n)}$ from $\mathcal{C}$. Since $\mathcal{C} \in \{$general circuits, general formulas, $\Sigma\Pi\Sigma\Pi$ circuits$\}$, by Lemma 3.3 each such $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ actually requires general circuits of size $2^{\Omega(n)}$. Finally, by Lemma 3.2 there exists an algorithm that given $n, d, s \in \mathbb{N}$ in time $(nd)^{\mathcal{O}(\log s)}$ outputs a hitting set $\mathcal{H}$ for general (n, s, d)-circuits. □

We prove Theorem 3 by implementing the procedure outlined in Section 1.6.

Proof of Theorem 3. Let $\mathcal{A}$ be an exact learner for $\mathcal{C}$ and let $C \in \mathcal{C}$ be a (n, s, d)-circuit from $\mathcal{C}$ given via oracle access. We run $\mathcal{A}$ on C. Given a membership query $\bar{a} \in \mathbb{F}^n$, we simply answer by $C(\bar{a})$. Now, suppose that we are given an equivalence query hypothesis $\hat{C}$. By the properties of $\mathcal{A}$, $\hat{C}$ is a $(n, \text{poly}(s), \text{poly}(d))$-circuit and so is $C - \hat{C}$. Let $\mathcal{H}$ be the hitting set from Lemma 3.4 with the appropriate parameters. We test if $\hat{C}|_{\mathcal{H}} \equiv C|_{\mathcal{H}}$ using membership queries to C. If $\hat{C}(\bar{a}) \neq C(\bar{a})$ for some $\bar{a} \in \mathcal{H}$, then we answer the query with $\bar{a}$. Otherwise, the we stop and output $\hat{C}$. □

4 From Learning to Rigidity

In this section we discuss the proof Theorem 4. In similar fashion to Definition 2.4, we can define a slightly different evaluation matrix.

Definition 4.1. *Let $\bar{a} \in \mathbb{F}^m$, $\bar{b} \in \mathbb{F}^n$ and $i \in [m], j \in [n]$. Define $\hat{M}(\bar{a}, \bar{b}) \in \mathbb{F}^{1 \times mn}$ as $\hat{M}(\bar{a}, \bar{b})_{1,(i,j)} = \bar{a}_i \cdot \bar{b}_j$. With the extension: for $S \subseteq \mathbb{F}^n \times \mathbb{F}^m$, $\hat{M}(S) \in \mathbb{F}^{|S| \times mn}$ when $\hat{M}(S)_{(\bar{a},\bar{b})} = \hat{M}(\bar{a}, \bar{b})$ for $(\bar{a}, \bar{b}) \in S$.*

In the spirit of Lemma 2.5, we observe that:

1. Given $\bar{a} \in \mathbb{F}^m$, $\bar{b} \in \mathbb{F}^n$ and a matrix $A \in \mathbb{F}^{m \times n}$ we have that $\bar{a}^t \cdot A \cdot \bar{b} = \hat{M}(\bar{a}, \bar{b}) \cdot v(A)$, when $v(A)$ is a vector of length mn containing the entries of A indexed by (i, j).
2. Let $S = \{(e_i, e_j)\}_{i \in [m], j \in [n]}$ when e_i represents the i-th vector of the standard basis. Then $\text{rank}(\hat{M}(S)) = mn$.

Given the above, one can use an argument similar to the one in Lemma 3.1 and Algorithm 1 to show that if $\mathcal{C}_{(r,s)}$ is exactly learnable with $q < mn$ queries, then the rank of the corresponding evaluation matrix will be q. Therefore, the algorithm could diagonalize against the output hypothesis (Lines 11 and 12), thus producing a vector which corresponds to a (r, s)-rigid matrix. We leave the formalization of the proof of Theorem 4 as an exercise for the reader.

5 From Learning to Square Roots

In this section, we establish a connection between learning and square root extraction, thus proving Theorem 5.

We begin by formally presenting the models of depth-4 and depth-3 multilinear circuits and some related definitions. Similar definitions were given in [DS06, SV11, KMSV13].

Definition 5.1. *A depth-4 $\Sigma\Pi\Sigma\Pi(k)$ circuit C has four layers of alternating Σ and Π gates (the top Σ gate is at level one) and it computes a polynomial of the form*

$$C(\bar{x}) = \sum_{i=1}^{k} F_i(\bar{x}) = \sum_{i=1}^{k} \prod_{j=1}^{d_i} P_{ij}(\bar{x})$$

where the $P_{ij}(\bar{x})$-s are polynomials computed by the last two layers of $\Sigma\Pi$ gates of the circuit and are the inputs to the Π gates at the second level.

A *multilinear* $\Sigma\Pi\Sigma\Pi(k)$ circuit is a $\Sigma\Pi\Sigma\Pi(k)$ circuit in which each multiplication gate F_i computes a multilinear polynomial. The requirement that the F_i-s compute multilinear polynomials implies that for each $i \in [n]$ the polynomials $\{P_{ij}\}_{j\in[d_i]}$ are variable-disjoint. In other words, each F_i induces a partition of the input variables. Yet in general, different multiplication gates may induce different partitions. A *set-multilinear* $\Sigma\Pi\Sigma\Pi(k)$ circuit is a multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit in which all the F_i-s induce the same partition.

Note that if the circuit is of size s then each P_{ij} is s-sparse. A depth-3 $\Sigma\Pi\Sigma(k)$ circuit can be a seen a partial case of $\Sigma\Pi\Sigma\Pi(k)$ circuit in which the polynomials P_{ij} are linear forms (in particular, n-sparse).

We say that a circuit is *minimal* if no proper subcircuit of C computes the zero polynomial. We say that the circuit C is *simple* if $\gcd(F_1, \dots, F_k) = 1$. Saraf & Volkovich showed that the multiplication gates of a multilinear depth-4 computing the zero polynomial must compute sparse polynomials. This is referred to as the "Sparsity Bound" for multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits. Previously, Dvir & Shpilka [DS06] proved a similar statement for $\Sigma\Pi\Sigma(k)$ circuits. (See [DS06] for more details).

Lemma 5.2 ([SV11]). *There exists an non-decreasing function $\varphi(k, s)$ such that if $C(\bar{x}) = \sum\limits_{i=1}^{k} F_i(\bar{x})$ is a simple and minimal, multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit of size s computing the zero polynomial, then for each $i \in [k]$ it holds that $\|F_i\| \le \varphi(k, s)$ and $\varphi(k, s) \le s^{5k^2}$.*

We now move the proof of Theorem 5.

Definition 5.3. *For $n \in \mathbb{N}$ and $\alpha \in \mathbb{F}$ we define the polynomials $\Phi_n^{\alpha}(x_1, \dots, x_n) \triangleq \prod\limits_{i=1}^{n} (x_i + \alpha)$, $P_n^{\alpha}(\bar{x}) \triangleq \Phi_n^{\alpha}(\bar{x}) + \Phi_n^{-\alpha}(\bar{x})$, $Q_n^{\alpha}(\bar{x}) \triangleq \alpha \cdot \Phi_n^{\alpha}(\bar{x}) - \alpha \cdot \Phi_n^{-\alpha}(\bar{x})$ and $B_\alpha(z) \triangleq \begin{bmatrix} z & 1 \\ \alpha^2 & z \end{bmatrix}$.*

The following lemma ties the above together:

Lemma 5.4. *Let $n \in \mathbb{N}$ and $\alpha \in \mathbb{F}$. Then*

$$B_\alpha(x_n) \cdot B_\alpha(x_{n-1}) \cdot \ldots \cdot B_\alpha(x_1) \cdot \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} P_n^{\alpha} \\ Q_n^{\alpha} \end{bmatrix}.$$

Before giving the proof, we remark that the lemma actually shows that the polynomials P_n^α and Q_n^α are computable by read-once oblivious ABPs. This is not a coincidence. The results of [BBB⁺00, KS06] show that set-multilinear $\Sigma\Pi\Sigma$ circuits can be simulated by read-once oblivious ABPs of small width.

Proof. Fix $\alpha \in \mathbb{F}$. The proof is by induction on n. For $n = 1$ we get $B_\alpha(x_1) = \begin{bmatrix} 2x_1 \\ 2\alpha^2 \end{bmatrix} = \begin{bmatrix} P_1^\alpha \\ Q_1^\alpha \end{bmatrix}$. Suppose $n \geq 2$. By the induction hypothesis:

$$B_\alpha(x_{n-1}) \cdot \ldots \cdot B_\alpha(x_1) \cdot \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} P_{n-1}^\alpha \\ Q_{n-1}^\alpha \end{bmatrix}.$$

Therefore:

$$B_\alpha(x_n) \cdot B_\alpha(x_{n-1}) \cdot \ldots \cdot B_\alpha(x_1) \cdot \begin{bmatrix} 2 \\ 0 \end{bmatrix} = B_\alpha(x_n) \cdot \begin{bmatrix} P_{n-1}^\alpha \\ Q_{n-1}^\alpha \end{bmatrix} = \begin{bmatrix} x_n \cdot P_{n-1}^\alpha + Q_{n-1}^\alpha \\ \alpha^2 \cdot P_{n-1}^\alpha + x_n \cdot Q_{n-1}^\alpha \end{bmatrix} = \begin{bmatrix} P_n^\alpha \\ Q_n^\alpha \end{bmatrix}.$$

□

Corollary 5.5. *Let $n \in \mathbb{N}$ and $\alpha^2 = \beta \in \mathbb{F}$. Then the polynomial $P_n^\alpha(\bar{x})$ is computable by a set-multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit of size $4n$. In addition, given an assignment $\bar{a} \in \mathbb{F}^n$, $P_n^\alpha(\bar{a})$ can be evaluated using $\mathcal{O}(n)$ field operations given β (without the knowledge of α).*

Before giving the proof of the main claim of the section, we require the following result that gives an efficient deterministic factorization algorithm for multilinear sparse polynomials.

Lemma 5.6 ([SV10])**.** *Given a multilinear s-sparse polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ there is a deterministic algorithm that outputs the* irreducible *factors, $h_1, \ldots, h_k$ of P using* poly(n, s) *field operations. Furthermore, each h_i is s-sparse.*

```
Input: $\beta \in \mathbb{F}$, exact learner $\mathcal{A}$ via oracle
Output: $\alpha$ such that $\alpha^2 = \beta$

1 Run $\mathcal{A}$ with $n = 10\ell(k+2)^2 \cdot \log(\ell(k+2)^2), s = 4n$ on $P_n^\alpha$. /* $2^n > \varphi\left(k+2, (8n)^\ell\right)$ */
2 Given a membership query point $\bar{a} \in \mathbb{F}^n$ by $\mathcal{A}$:
3 Use Corollary 5.5 to evaluate $P_n^\alpha(\bar{a})$;
4 Given an equivalence query hypothesis $\hat{C}$:
5 Test if $\hat{C}|_{\{0,1\}^n} \equiv P_n^\alpha|_{\{0,1\}^n}$ /* Following the procedure outlined in Section 1.6.
     */
6 Let $C = \sum_{i=1}^k F_i$ be the output of $\mathcal{A}$;
7 Use Lemma 5.6 to factor all the $F_i$-s into irreducible factors;
8 For each factor of the form $x_j + \gamma$: if $\gamma^2 = \beta$ then Output $\gamma$;
```

Algorithm 2: Square Root Extraction from Exact Learner for multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits

Lemma 5.7. *Suppose there exists an exact leaner $\mathcal{A}$ that learns multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits of size s over $\mathbb{F}$ in time $T(s, |\mathbb{F}|)$ when $n \leq s$, with hypotheses being multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits of size at most s^ℓ. Then given $\beta \in \mathbb{F}$ in time $T(\text{poly}(\ell, k), |\mathbb{F}|) \cdot 2^{\text{poly}(\ell,k)}$ Algorithm 2 outputs $\alpha \in \mathbb{F}$ such that $\alpha^2 = \beta$, using $\mathcal{A}$ as an oracle.*

Proof. By Corollary 5.5, $\mathcal{A}$ will output a multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit of size at most $(4n)^\ell$ computing P_n^α. Therefore, the correctness of the algorithm is based on the following claim: There exist i, j such that $x_j \pm \alpha$ is an irreducible factor of F_i. Assume for the contradiction that this is not the case. Consider the following $\Sigma\Pi\Sigma\Pi(k+2)$ circuit: $C' \triangleq \Phi_n^\alpha + \Phi_n^{-\alpha} - F_1 - \ldots - F_k$. By construction, $C' \equiv 0$. Let C'' be the minimal zero subcircuit of C' that contains Φ_n^α. As $\Phi_n^\alpha + \Phi_n^{-\alpha} \not\equiv 0$, C'' must contain at least one (non-zero) F_i. We now claim that C'' is simple. Indeed, the only irreducible factors of Φ_n^α are of the form $x_j + \alpha$. However, by assumption none of them is a factor of F_i. Finally, observe that size of C'' is at most $(4n)^\ell + (4n) \leq (8n)^\ell$. By Lemma 5.2, Φ_n^α must be $\varphi\left(k+2, (8n)^\ell\right)$-sparse. That is, $2^n \leq \varphi\left(k+2, (8n)^\ell\right)$, in contradiction to the choice of n. $\square$

Theorem 5 follows as a corollary.

6 Discussion & Open Questions

In this paper we show several consequences from efficient learnability of various classes of arithmetic circuits. In particular, we show that efficient exact learning algorithms for general circuits and formulas imply efficient black-box PIT algorithms for these circuits. The proof goes via constructing explicit lower bounds and applying the hardness-randomness paradigm of [KI04], which is so far limited to general circuit/formulas only. Can one give a more direct proof of this implication? In particular, a proof that will cover a larger family of arithmetic circuit classes? Further more, as outlined in Section 1.6, efficient black-box PIT algorithms can be used to eliminate equivalence queries. As a consequence, one might be able to show that a "rich enough" arithmetic circuit class is effiently learnable iff it is efficiently reconstructible. We note that there is a long standing open problem to transform explicit lower bounds for a class $\mathcal{C}$ into an efficient PIT algorithm for that class (see e.g. [SY10]). As we show that efficient exact learning algorithms imply explicit lower bounds, our question might be easier.

Another natural question is: can one devise or explain the lack of progress for reconstruction algorithms for multilinear formulas with bounded read? We note that there no efficient reconstruction algorithms even for a sum of two read-once formulas. We would like to point out that using the ideas of Section 5 with the techniques of [AvMV15, SV15], one could show that any reconstruction algorithm for read-twice formulas or even sums of read-once formulas must compute square roots.

Finally, can one prove other consequences from efficient learnability of arithmetic circuits? In particular, can one show that certain learning algorithms imply integer factorization?

Acknowledgment

The author would like to thank Zeev Dvir for pointing out that the techniques used in the paper could show that efficient learning algorithm imply explicit rigid matrices, which resulted in Theorem 3. The author would also like to thank Michael Forbes and Rocco Servedio for useful conversations.

References

[Agr05] M. Agrawal. Proving lower bounds via pseudo-random generators. In *Proceedings of the 25th FSTTCS*, volume 3821 of *LNCS*, pages 92–105, 2005.

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

[Ang87] D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.

[AV08] M. 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)*, pages 67–75, 2008.

[AvMV15] M. Anderson, D. van Melkebeek, and I. Volkovich. Deterministic polynomial identity tests for multilinear bounded-read formulae. *Computational Complexity*, 2015.

[BBB$^+$00] A. Beimel, F. Bergadano, N. H. Bshouty, E. Kushilevitz, and S. Varricchio. Learning functions represented as multiplicity automata. *J. ACM*, 47(3):506–530, 2000.

[BF90] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *STACS*, pages 37–48, 1990.

[BHH95] N. H. Bshouty, T. R. Hancock, and L. Hellerstein. Learning arithmetic read-once formulas. *SIAM J. on Computing*, 24(4):706–735, 1995.

[Blu94] A. Blum. Separating distribution-free and mistake-bound learning models over the boolean domain. *SIAM J. Comput.*, 23(5):990–1000, 1994.

[BS83] W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical Comp. Sci.*, 22:317–330, 1983.

[DL78] R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978.

[DS06] Z. Dvir and A. Shpilka. Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. *SIAM J. on Computing*, 36(5):1404–1434, 2006.

[FK09] L. Fortnow and A. R. Klivans. Efficient learning algorithms yield circuit lower bounds. *J. Comput. Syst. Sci.*, 75(1):27–36, 2009.

[FS13] M. Forbes and A. 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)*, pages 243–252, 2013. Full version at http://eccc.hpi-web.de/report/2012/115.

[GG99] J. von zur Gathen and J. Gerhard. *Modern computer algebra.* Cambridge University Press, 1999.

[GH11] C. Gentry and Sh. Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 107–109, 2011.

[GKL04] S. Gao, E. Kaltofen, and A. G. B. Lauder. Deterministic distinct-degree factorization of polynomials over finite fields. *J. Symb. Comput.*, 38(6):1461–1470, 2004.

[GKL12] A. Gupta, N. Kayal, and S. V. Lokam. Reconstruction of depth-4 multilinear circuits with top fanin 2. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 625–642, 2012. Full version at http://eccc.hpi-web.de/report/2011/153.

[GM84] Sh. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[HJ91] R.A. Horn and C.R. Johnson. *Topics in Matrix Analysis.* Cambridge University Press, 1991.

[HS80] J. Heintz and C. P. Schnorr. Testing polynomials which are easy to compute (extended abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC)*, pages 262–272, 1980.

[Kay07] N. Kayal. *Derandomizing some number-theoretic and algebraic algorithms.* PhD thesis, Indian Institute of Technology, Kanpur, India, 2007.

[KI04] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KKO13] A. Klivans, P. Kothari, and I. Oliveira. Constructing hard functions from learning algorithms. In *Proceedings of the 28th Annual IEEE Conference on Computational Complexity (CCC)*, pages 86–97, 2013.

[KMSV13] Z. S. Karnin, P. Mukhopadhyay, A. Shpilka, and I. Volkovich. Deterministic identity testing of depth 4 multilinear circuits with bounded top fan-in. *SIAM J. on Computing*, 42(6):2114–2131, 2013.

[KS01] A. Klivans and D. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–223, 2001.

[KS06] A. Klivans and A. Shpilka. Learning restricted models of arithmetic circuits. *Theory of computing*, 2(10):185–206, 2006.

[KS09a] 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)*, pages 274–285, 2009. Full version at http://www.cs.technion.ac.il/ shpilka/publications/KarninShpilka09.pdf.

[KS09b] A. R. Klivans and A. A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. *J. Comput. Syst. Sci.*, 75(1):2–12, 2009.

[KV94] M. J. Kearns and L. G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994.

[LLL82] A.K. Lenstra, H.W. Lenstr, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen,*, 261(4):515–534, 1982.

[Lok09] S. V. Lokam. Complexity lower bounds using linear algebra. *Foundations and Trends in Theoretical Computer Science*, 4(1-2):1–155, 2009.

[LV03] R. J. Lipton and N. K. Vishnoi. Deterministic identity testing for multivariate polynomials. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 756–760, 2003.

[MVV87] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[Rab05] M. O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005.

[Raz09] R. Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM*, 56(2), 2009.

[RSY08] R. Raz, A. Shpilka, and A. Yehudayoff. A lower bound for the size of syntactically multilinear arithmetic circuits. *SIAM J. on Computing*, 38(4):1624–1647, 2008.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[Sho91] V. Shoup. A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic. In *ISSAC*, pages 14–21, 1991.

[SV10] A. Shpilka and I. Volkovich. On the relation between polynomial identity testing and finding variable disjoint factors. In *Automata, Languages and Programming, 37th International Colloquium (ICALP)*, pages 408–419, 2010. Full version at http://eccc.hpi-web.de/report/2010/036.

[SV11] S. Saraf and I. Volkovich. Blackbox identity testing for depth-4 multilinear circuits. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 421–430, 2011. Full version at http://eccc.hpi-web.de/report/2011/046.

[SV14] A. Shpilka and I. Volkovich. On reconstruction and testing of read-once formulas. *Theory of Computing*, 10:465–514, 2014.

[SV15] A. Shpilka and I. Volkovich. Read-once polynomial identity testing. *Computational Complexity*, 2015.

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

[SY11] S. Saraf and S. Yekhanin. Noisy interpolation of sparse polynomials, and applications. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity CCC*, pages 86–92, 2011.

[Val77] L. G. Valiant. Graph-theoretic arguments in low-level complexity. In *Lecture notes in Computer Science*, volume 53, pages 162–176. Springer, 1977.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Vol15] I. Volkovich. On some computations on sparse polynomials. *Manuscript*, 2015. (submitted).

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 216–226, 1979.

A PAC and Exact Learning are essentially equivalent for Arithmetic Circuits

In this section we give a formal proof to folklore result. We begin by formally defining the model.

In Valiant's PAC Learning model, we have a (computationally bounded) learner that is given a set of samples of the form $(\bar{x}, f(\bar{x}))$ from some fixed function $f \in \mathcal{C}$, where $\bar{x}$ is chosen according to some unknown distribution D. Given $\varepsilon > 0$ and $\delta > 0$, the learner's goal is to output a hypothesis "functionally close" to f w.h.p. Formally, we say that a function class $\mathcal{C}$ is PAC *learnable* if there exists a learner which given any $f \in \mathcal{C}$, $\varepsilon > 0$ and $\delta > 0$ in time polynomial in $n, 1/\varepsilon, 1/\delta, |f|$ outputs with probability $1 - \varepsilon$ a hypothesis $\hat{f}$ such that $\hat{f}$ is a $1 - \delta$ close to f under D. In a more general model, the learner is allowed membership queries (as in the exact learning model). In this case, we say that $\mathcal{C}$ is PAC *learnable* with *membership queries*.

We show that both PAC and Exact learning models are, essentially, equivalent in the arithmetic setting. In fact we show that if an arithmetic circuit class $\mathcal{C}$ is PAC learnable with *membership queries* then $\mathcal{C}$ is learnable with membership queries only. In addition, if the learning algorithm is proper and $\mathbb{F}$ is sufficiently large then the algorithm actually outputs a circuit from $\mathcal{C}$. The celebrated result of [Blu94] shows that if one-way functions exist, then this is not the case for Boolean functions.

It is also known [Ang87] that both randomized and exact learners can be used to obtain a PAC learner with membership queries.

Definition A.1 (Distance). *Let $f, g : \mathbb{F}^n \to \mathbb{F}$ be functions. We define their (relative)* distance *as $\Delta(f, g) \triangleq \Pr_{\bar{a} \in \mathbb{F}^n}[f(\bar{a}) \neq g(\bar{a})]$. For $\delta > 0$ we say that f is* δ-far *from g if $\Delta(f, g) > \delta$; otherwise we say that f is* δ-close *to g.*

We give the Schwartz-Zippel Lemma which provides a bound on the number of zeros of a polynomial.

Lemma A.2 ([Zip79, Sch80]). *Let $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a non-zero polynomial of degree at most d and let $V \subseteq \mathbb{F}$. Then $\Pr_{\bar{a} \in V^n}[P(\bar{a}) = 0] \leq \frac{d}{|V|}$.*

The following is an important property of low-degree polynomials: Self-Correctability.

Lemma A.3 ([BF90]). *Let $n, d \in \mathbb{N}$ and $\mathbb{F}$ be a field of size $|\mathbb{F}| > d + 2$. Let $f : \mathbb{F}^n \to \mathbb{F}$ be a function that is $1/d$-close to a degree d polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$. Then there exists a randomized algorithm* CORRECT *that uses* $\mathrm{poly}(n, d)$ *field operations and oracle calls to f such that for any $\bar{a} \in \mathbb{F}^n$: $\Pr[\mathsf{CORRECT}^f(\bar{a}) \neq P(\bar{a})] \leq 2^{-10n}$.*

The following is obtained by standard techniques:

Observation A.4. *Let $f : \mathbb{F}^n \to \mathbb{F}$ be a function that is $1/d$-close to a degree d polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and let C be a Boolean circuit of size s computing f. Then there exists a circuit $\hat{C}$ of size $\mathrm{poly}(s, n, d)$ that computes P. Moreover, there is randomized algorithm that given C output $\hat{C}$ with high probability in time $\mathrm{poly}(s, n, d)$.*

We now give our main claim:

Lemma A.5. *Let $\mathcal{C}$ be an arithmetic circuit class over the field $\mathbb{F}$. If $\mathcal{C}$ is* PAC *learnable with membership queries then, $\mathcal{C}$ is learnable with membership queries only. In addition, if the learning algorithm is proper and $\mathbb{F} = \mathrm{poly}(d)$ is sufficiently large, then $\mathcal{C}$ can be reconstructed efficiently.*

Proof. Let $\mathcal{A}$ be an exact learner for $\mathcal{C}$ and let $C \in \mathcal{C}$ be a (n, s, d)-circuit given via oracle access. We run $\mathcal{A}$ on C with $\delta = 1/d$ and $\varepsilon = 1/n$. Whenever $\mathcal{A}$ asks for a random sample point, we pick $\bar{a} \in \mathbb{F}^n$ at random and return $(\bar{a}, C(\bar{a}))$. Let $h(\bar{x}) : \mathbb{F}^n \to \mathbb{F}$ be the output of $\mathcal{A}$. We have that $\Delta(h, C) \leq 1/d$. We can compute a Boolean Circuit C' for h. Applying Observation A.4, in randomized time $\mathrm{poly}(n, s, d)$, we can compute a circuit $\hat{C}$ such that $\hat{C} \equiv C$.

If $\mathcal{A}$ is proper, h is a $(n, \mathrm{poly}(s), \mathrm{poly}(d))$-circuit from $\mathcal{C}$. By Lemma A.2, if $h \not\equiv C$ then $\Delta(h, C) \geq 1 - \mathrm{poly}(d)/|\mathbb{F}|$ which is larger than $1 > d$ when $\mathbb{F} = \mathrm{poly}(d)$. Therefore, $h \equiv C$. $\square$