Learning Read-Constant Polynomials of Constant Degree Modulo Composites

Arkadev Chattopadhyay[1], Ricard Gavaldà[2],
Kristoffer Arnsfelt Hansen[3], and Denis Thérien[4]

[1] Tata Institute of Fundamental Research, arkadev.c@tifr.res.in
[2] Universitat Politècnica de Catalunya, gavalda@lsi.upc.edu
[3] Aarhus University, arnsfelt@cs.au.dk
[4] McGill University, denis@cs.mcgill.ca

Abstract. Boolean functions that have constant degree polynomial representation over a fixed finite ring form a natural and strict subclass of the complexity class ACC^0. They are also precisely the functions computable efficiently by *programs* over fixed and finite nilpotent groups. This class is not known to be learnable in any reasonable learning model. In this paper, we provide a deterministic polynomial time algorithm for learning Boolean functions represented by polynomials of constant degree over arbitrary finite rings from membership queries, with the additional constraint that each variable in the target polynomial appears in a constant number of monomials. Our algorithm extends to superconstant but low degree polynomials and still runs in quasipolynomial time.

1 Introduction

Understanding the computational power of computation over rings of the form $\mathbb{Z}_m$, for an arbitrary composite number m, is a fundamental open problem. A concrete and natural setting in which to explore this power is the model of representing Boolean functions by low degree polynomials over such rings, in the following sense [5]: An assignment to the variables is a 1 of the Boolean function if and only if the polynomial on it evaluates to an element of a prespecified accepting subset of the ring. The class of Boolean functions that are so computable by constant degree polynomials forms a strict subclass of the complexity class ACC^0 [21, 22]. They are also precisely the functions computable efficiently by *programs* over fixed and finite nilpotent groups [21, 22, 7].

When the modulus is a prime number and the ring thus turns into a finite field, our knowledge of representations is far better than the general case. For instance, it is known that degree $\Omega(n)$ is required in order to represent the Boolean function MOD_q by polynomials over the field $\mathbb{Z}_p$, when p is a prime and q has a prime factor different from p. The stronger result that MOD_q remains hard to even approximate well by such polynomials of low degree is a key insight in the celebrated lower bound of Razborov [24] and Smolensky [26] on the size of bounded-depth circuits.

In contrast, we do not even know the exact degree of the Parity function for polynomials over $\mathbb{Z}_m$, as soon as m is an odd number having two distinct prime factors. In a beautiful work, Barrington, Beigel, and Rudich [5] showed that composite moduli give non-trivial advantage to polynomials as compared to prime moduli. More precisely, they showed that the degree of the OR and the AND function over $\mathbb{Z}_m$ is $O\big(n^{1/t}\big)$ if m has t distinct prime factors. On the other hand, it is well known that if m is a fixed prime, then this degree is $\Omega(n)$. This surprising construction of Barrington *et al.* has found diverse applications. Indeed, Efremenko [15] recently built efficient locally decodable codes from it. Also, Gopalan [17] shows that several previously known constructions of explicit Ramsey graphs can all be derived from this construction.

The best known lower bounds on the composite degree of any Boolean function is $\Omega(\log n)$ (see for example [18, 27, 10] and the survey [14]). Proving anything better is a tantalizingly open problem. In this work, we look at low degree polynomials through the lens of computational learning theory. The motivation and hope is that this approach will lead to new insights into the structure of these polynomials, thus benefiting both the fields of learning theory and complexity theory.

Given that we know degree lower bounds of $\Omega(\log n)$, it is reasonable to hope that we can learn functions represented by constant degree polynomials. We take on this task in this paper in the setting where the learner is allowed to ask membership queries. The main difficulty that one faces is essentially the same that confronts one when proving lower bounds on the degree: while computation by the target polynomial takes place in the entire ring $\mathbb{Z}_m$, the information revealed to the learner is just Boolean. That is, we learn only whether the polynomial when evaluated on the chosen point yields an element of the unknown accepting set. Although several equivalent low degree representations may exist for the target concept, it is a non-trivial fact (Corollary 4) that polynomially many such queries are able to isolate a unique function in the concept class that agrees with the answers of the teacher. The computational challenge is to recognize this unique function in the sense of predicting it on an arbitrary, previously unseen, input.

Our Result We consider the concept class of functions that have a representation by a constant degree polynomial in which every variable appears in a constant number of monomials. We show that this class is exactly learnable in polynomial time from the values of the target function at all input assignments of Hamming weight bounded by another constant. These values can be obtained, in particular, from membership queries. Additionally, our learning algorithm is proper in the sense that it outputs a constant degree polynomial equivalent to the target polynomial with respect to the Boolean function they compute. It is worth remarking that there are very few instances in which concepts are known to be properly learnable, especially when there is no guarantee of a unique representation.

Overview of Our Techniques Our learning algorithm uses some novel ideas exploiting the following structural property of low degree polynomials first discovered in the work of Péladeau and Thérien [21] (see the translation [22]): for every constant degree polynomial P over any fixed finite commutative ring with identity, there exists a "magic set" of variables of constant cardinality such that every value in the range of P can be attained by setting only a subset of variables from the magic set to 1 and all other variables to 0. This property is very convenient and in particular, implies that every Boolean function that can be represented by a constant degree polynomial gets uniquely determined by the values it takes on points of constant Hamming weight. It is worthwhile to note that although the function gets fixed by knowing its behavior on all low weight points, it is not clear how to efficiently determine the value of this function on any other input point of the Boolean cube. This is the essential challenge that the learning algorithm has to overcome.

To be more specific, using this magic set we define an equivalence relation among monomials of the same degree. We show that there always exists a polynomial representing the same function that the teacher holds, in which all monomials belonging to the same equivalence class have identical coefficients. The number of equivalence classes is upper bounded by a constant and there is a very efficient test of equivalence. These properties allow us to enumerate all possible values of coefficients and then choose any that satisfies the polynomially many points of constant weight.

Relations to Existing Work Polynomials have been intensely studied in learning theory. When the learner can use evaluation queries returning the precise value of the polynomial over the base ring or field, polynomials of degree d over arbitrary finite rings are easy to learn from roughly n^d evaluation queries, by learning in order the coefficients of monomials of degree 0, 1, 2, etc. With evaluation *and* equivalence[5] queries one can learn polynomials of arbitrary degree over finite fields (and, improperly, over finite rings) [25, 9, 13].

In this paper, we concentrate on learning Boolean functions represented by programs over polynomials, that is, when evaluation queries do not return a field or ring element, but only its membership to the accepting set. For all finite rings (and many infinite ones), degree-1 polynomials whose accepting set is a singleton can be learned in the PAC model by a variation of the subspace-learning algorithm in [19] (see also [16]); the constant-degree case can be reduced to the degree-1 case by standard techniques.

For the field $\mathbb{Z}_p$ in particular, a standard use of Fermat's little theorem shows that for every polynomial of degree d with an arbitrary accepting set there is a polynomial of degree $d(p-1)$ whose range is $\{0,1\}$ computing the same Boolean function. This means that we can learn Boolean functions computed by constant

[5] In an equivalence query, the learning algorithm emits some representation of a Boolean function and receives as answer either a Boolean assignment where it differs from the target function, or "yes" if no such assignment exists.

degree polynomials over $\mathbb{Z}_p$ both in the PAC model, as above, and exactly from membership queries.

In this paper we make progress, for the first time to our best knowledge, in the analogous problem for the non-field case, i.e., learning Boolean functions represented by constant-degree polynomials over rings. The degree-1 case was solved in [16] by a technique that does not seem to extend to higher degrees. The emphasis in [16] was the classification of families of Boolean functions computed by *programs* over finite monoids (cf. [4, 8, 7]), with respect to their learnability in different models. In this setting, polynomials of constant degree over finite rings are equivalent in power to programs over nilpotent groups (as shown in [21]) with degree-1 polynomials corresponding to programs over Abelian groups. The class of functions computed by such programs is a natural subclass of functions computable by programs over solvable groups. Starting with the famous and surprising work of Barrington [4] that showed the class of functions computed by polynomial length programs over finite non-solvable groups is exactly the complexity class NC^1, programs over groups, or monoids in general, have been used (see for example [8, 7]) to characterize natural subclasses of NC^1.

2 Preliminaries

2.1 Polynomials over Finite Rings

Let $\mathcal{R}$ be a commutative finite ring with unit, and let $P(x_1, \ldots, x_n)$ be a polynomial over $\mathcal{R}$. We say P is a *read-k* polynomial if every variable in P appears in at most k monomials of P.

Consider a family of polynomials $\mathcal{P} = \{P_i\}_{i=1}^{\infty}$, where P_i is a polynomial in i variables. We say the family $\mathcal{P}$ is *read-constant*, if there exist a k such that every $P_i \in \mathcal{P}$ is read-k. Similarly, we say that $\mathcal{P}$ is *constant degree* if there exists d such that every $P_i \in \mathcal{P}$ is of degree at most d.

In this work, we will restrict our attention to variables ranging over the set $\{0,1\} \subseteq \mathcal{R}$, and as a consequence we can without loss of generality restrict our attention to *multilinear* polynomials. Formally we consider the ring of polynomials $\mathcal{R}[x_1, \ldots, x_n]/N$, where N is the ideal generated by the set of polynomials $\{x_i^2 - x_i \mid i = 1, \ldots, n\}$. Any function $\{0,1\}^n \to \mathcal{R}$ is uniquely expressed by such a polynomial. Define the *range* of P as $\mathrm{range}(P) = \{r \in \mathcal{R} \mid \exists x \in \{0,1\}^n : P(x) = r\}$.

Equipping a polynomial P with an *accepting set* $A \subseteq \mathcal{R}$, we say that the pair (P, A) computes a Boolean function $f : \{0,1\}^n \to \{0,1\}$ if it holds that $P(x) \in A$ if and only if $f(x) = 1$, for all $x \in \{0,1\}^n$.

Given a set of indices $J \subseteq [n]$, we let $\chi_J \in \{0,1\}^n$ denote the characteristic vector of J. Conversely, for $w \in \{0,1\}^n$, define $I_w = \{i \in [n] \mid w_i = 1\}$. Thus $\chi_{I_w} = w$ and $I_{\chi_J} = J$. For $u, v \in \{0,1\}^n$, let $u \vee v \in \{0,1\}^n$ be defined by $I_{u \vee v} = I_u \cup I_v$.

Consider now a degree-d polynomial, $P(x) = \sum_{I \subseteq [n], |I| \leq d} c_I \prod_{i \in I} x_i$. For a subset $S \subseteq [n]$ we define the polynomial P_S of monomials from S by, $P_S(x) =$

$\sum_{I \subseteq S, |I| \leq d} c_I \prod_{i \in I} x_i$. For disjoint subsets $S, T \subseteq [n]$ define the polynomial $P_{S \times T}$ consisting of cross terms between S and T:

$$P_{S \times T}(x) = \sum_{I, J \neq \emptyset; I \subseteq S, J \subseteq T; |I|+|J| \leq d} c_{I \cup J} \prod_{i \in I \cup J} x_i \ .$$

We associate to P the graph G_P defined as follows. The set of vertices of G_P is $\{1, \ldots, n\}$ and the set of edges is $E(G_P) = \{(i, j) \mid x_i \text{ and } x_j \text{ appear together in some monomial of } P \text{ with nonzero coefficient}\}$ This will allow us to speak of the *distance* between variables of P, namely as distances in the graph G_P.

2.2 Structural Properties of Polynomials

Using an inductive Ramsey-theoretic argument, the following important structural result about constant degree polynomials over finite rings was proved by Péladeau and Thérien [21, 22].

Theorem 1 (Péladeau and Thérien). *Let $\mathcal{R}$ be a finite commutative ring with unity and let d be any number. Then there exists a constant $c = c(\mathcal{R}, d)$ with the following property: For any multilinear polynomial P over $\mathcal{R}$ of degree at most d and for any $r \in \text{range}(P)$ there exists $w \in \{0,1\}^n$ with $|I_w| \leq c$ such that $P(w) = r$.*

Remark 2.
- The theorem as stated above is actually only implicitly given in the proof of Lemma 2 of [21].
- In Sect. 4 we shall present with full proof a quantitative strengthening of the theorem based on a result of Tardos and Barrington [27].

Two easy consequences of this theorem are given below. Our learning algorithm will be heavily based on these results.

Corollary 3. *There exists a constant $s = s(\mathcal{R}, d)$, such that for every multilinear polynomial P over $\mathcal{R}$ of degree at most d, there exists a set $J \subseteq \{1, \ldots, n\}$ with the following properties: (1). $|J| \leq s$. (2). For every $r \in \text{range}(P)$ there exists $w \in \{0,1\}^n$ with $I_w \subseteq J$ such that $P(w) = r$.*

Proof. Let $s = |\mathcal{R}| c(\mathcal{R}, d)$, with $c(\mathcal{R}, d)$ as given by Theorem 1. We can then simply take J to be the union of $|\text{range}(P)|$ sets I_w provided by Theorem 1 for each $r \in \text{range}(P)$. □

For a given polynomial P we will refer to the set J as guaranteed above to exist as the *magic set* of variables for P.

Corollary 4. *There exists a constant $c' = c'(\mathcal{R}, d)$ with the following property: Let P and Q be polynomials of degree at most d with accepting sets A and B, respectively. If the Boolean functions computed by the pairs (P, A) and (Q, B) agree on all inputs $w \in \{0,1\}^n$ with $|I_w| \leq c'$, then the two Boolean functions are identical.*

Proof. We take $c' = c(\mathcal{R} \times \mathcal{R}, d)$ as given by Theorem 1. Now, write $P(x) = \sum c_I \prod_{i \in I} x_i$ and $Q(x) = \sum d_I \prod_{i \in I} x_i$. Consider the polynomial $(P \times Q)$ over $\mathcal{R} \times \mathcal{R}$ given by $(P \times Q)(x) = \sum (c_I, d_I) \prod_{i \in I} x_i$ If (P, A) and (Q, B) do not compute the same Boolean function there is $(r, s) \in \text{range}(P \times Q)$ such that either $r \in A$ and $s \notin B$ or $r \notin A$ and $s \in B$. Then by Theorem 1 and the choice of c' this would be witnessed by a $w \in \{0,1\}^n$ with $|I_w| \leq c'$ such that $(P \times Q)(w) = (r, s)$. □

Remark 5. The corollary above shows that a Boolean function representable by a low degree polynomial is completely identified by the function's values on points of the cube having low Hamming weight. This fact follows easily, using standard techniques like Möbius inversion, when the function is "exactly" representable by a low degree polynomial. Corollary 4 shows that this surprisingly remains true for the "weaker" notion of representation via an accepting set.

3 Learning with Membership Queries

In this section we will present our algorithm for learning read-constant, constant-degree polynomials. For convenience we choose to present the algorithm as a nondeterministic algorithm, that when terminating with success always outputs a correct polynomial. Afterwards we will be able to convert this nondeterministic algorithm into a deterministic algorithm simply by enumerating over all possible sequences of guesses of the algorithm, arguing that there are only polynomially many such sequences.

To ensure that the nondeterministic algorithm always produces a correct output we use a consistency check procedure, described as Algorithm 1.

Algorithm 1: Consistent$_{\mathcal{R},d}(Q, A, f)$

Input: Polynomial Q with accepting set $A \subseteq R$. Membership query access to Boolean function f.

Output: Decides if the pair (Q, A) computes the function f.

1: Query f on all $w \in \{0,1\}^n$ with $|I_w| \leq c'(\mathcal{R}, d)$

2: Return true if and only if for each queried w, $f(w) = 1$ if and only if $Q(w) \in A$

The correctness of the procedure is immediate from Corollary 4.

3.1 Equivalence Relations Between Monomials

For our algorithm we need the following somewhat technical definition of parameterized equivalence relations of monomials. Intuitively, they serve the following purpose: we want to learn an unknown polynomial, singling it out from exponentially many possibilities. One way to reduce this huge search space is to deduce, from membership queries, that some of the $n^{O(d)}$ coefficients must be the same, as they can only have a constant ($|R|$) number of values. Equivalence among two monomials, as defined below, is intended to suggest that they define isomorphic subpolynomials of the target polynomial.

For example, if a target polynomial contains terms $2x_1$, $3x_2$, x_1x_2, $2x_3$, $3x_4$, x_3x_4 we would like to say that monomials x_1x_2 and x_3x_4 are equivalent. This is because not only the coefficients of these monomials are the same, but also the sub-monomials of x_1x_2 (x_1, x_2) have the same coefficients respectively as the sub-monomials of x_3x_4 (x_3, x_4). Hence, when searching for coefficients for these monomials, we can discard all settings of coefficients where they differ.

The idea of the learning algorithm, to be explained in more detail in the next section, is to first implement equivalence tests among all monomials and then, on the basis of this information, actually find the values of all coefficients exploring a polynomial search space rather than an exponential one.

For any monomial M, let $I_M = \{i \mid x_i \text{ appears in } M\}$. Conversely, for any set of indices I let M_I denote the monomial $\Pi_{i\in I}x_i$. Since we consider only multilinear polynomials, we may identify a monomial with the set of variables it contains. Thus, we often write e.g. M instead of I_M. Given a polynomial P, an accepting set A, and a set J of indices in $\{1,\ldots,n\}$, we define a parameterized equivalence relation $\sim_{d,J}$ on tuples $(M,\preceq)$, where $M \subseteq [n], M \cap J = \emptyset$, $|M| = d$, and $\preceq$ is a total ordering[6] on $[n]$, by induction on d. We say $(M,\preceq_1) \sim_{d,J} (M',\preceq_2)$ if the following is satisfied:

1. For every assignment w, such that $I_w \subseteq J$, we have $P(w \vee \chi_M) \in A$ if and only if $P(w \vee \chi_{M'}) \in A$.
2. Let $M_1,\ldots,M_d$ (and $M'_1,\ldots,M'_d$) be the subsets of M (M') of size $d-1$ listed in the lexicographic order w.r.t $\preceq_1$ ($\preceq_2$). Then $(M_i,\preceq_1) \sim_{d-1,J} (M'_i,\preceq_2)$, for all $i \leq d$.

For every pair of monomials M and M', each of degree d, with $I_M, I_{M'}$ disjoint from J, we say that $M \sim_{d,J} M'$ if there exist $\preceq_1$ and $\preceq_2$ such that $(I_M,\preceq_1) \sim_{d,J} (I_{M'},\preceq_2)$.

Observe that $\sim_{d,J}$ is an equivalence relation, as suggested by the notation. Observe also that it depends only on the Boolean function computed by (P,A), and not on (P,A) itself. Finally, note that the number of equivalence classes is constant, independent of n, for constant d, $\mathcal{R}$, and $|J|$.

3.2 Idea of the Learning Algorithm

Let P be a polynomial with accepting set A. Let $\mathcal{M}_{\text{range}(P)}$ be the subgroup of the additive group of $\mathcal{R}$ generated by range(P) (i.e. the $\mathbb{Z}$-module generated by range(P)). Consider next the following equivalence relation on monomials of a polynomial P:

For tuples $(M,\preceq_1)$ and $(M',\preceq_2)$, where $M, M' \subseteq [n]$, $|M| = |M'| = d$, and $\preceq_1$ and $\preceq_2$ are total orderings, we say $(M,\preceq_1)\widehat{\sim}_d(M',\preceq_2)$ if the following are satisfied:

[6] Alternatively one could fix the same ordering, say $1,\ldots,n$ for all monomials. However we find it natural to identify monomials that are identical up to a permutation of the variables.

1. For every $r \in \mathcal{M}_{\text{range}(P)}$, $r + P(\chi_M) \in A$ if and only if $r + P(\chi_{M'}) \in A$.
2. Let $M_1, \ldots, M_d$ (and $M'_1, \ldots, M'_d$) be the subsets of M (M') of size $d-1$ listed in the lexicographic order w.r.t $\preceq_1$ ($\preceq_2$). Then $(M_i, \preceq_1) \widehat{\sim}_{d-1} (M'_i, \preceq_2)$, for all $i \leq d$.

For every pair of monomials M and M', each of degree d, we say that $M \widehat{\sim}_d M'$ if there exist $\preceq_1$ and $\preceq_2$ such that $(I_M, \preceq_1) \widehat{\sim}_d (I_{M'}, \preceq_2)$.

Assume now (by induction) that for all monomials M_1 and M_2 of degree $s < r$ we have $c_{M_1} = c_{M_2}$, whenever $M_1 \widehat{\sim}_s M_2$. Next consider monomials M and M' with $M \widehat{\sim}_r M'$, and let P' be the polynomial obtained from P by replacing the coefficient of M' with the coefficient of M. By our (inductive) assumption we have $P(\chi_M) = P'(\chi_{M'})$. Let $x \in \{0,1\}^n$ be arbitrary. Since $P(x), P(\chi_{M'}) \in \text{range}(P)$ we have $r = P(x) - P(\chi_{M'}) \in \mathcal{M}_{\text{range}(P)}$. We then have $r + P(\chi_M) \in A$ if and only if $r + P(\chi_{M'}) \in A$. But $r + P(\chi_{M'}) = (P(x) - P(\chi_{M'})) + P(\chi_{M'}) = P(x)$ and $r + P(\chi_M) = (P(x) - P(\chi_{M'})) + P(\chi_M) = P(x) - P(\chi_{M'}) + P'(\chi_{M'})$. This is equal to $P'(x)$, if $I_{M'} \subseteq I_x$. Hence $P(x) \in A$ if and only if $P'(x) \in A$ for this case. Otherwise, if $I_{M'} \not\subseteq I_x$, we have $M'(x) = 0$ and $P(x) = P'(x)$.

As a consequence, there is some P' such that every two $\widehat{\sim}_d$-equivalent monomials have the same coefficient in P' and it agrees with P in the following sense: for every input x, $P'(x)$ hits the accepting set if and only if $P(x)$ does so. Thus, if we were able to actually *implement* testing of the above equivalence relation, we would have a simple learning algorithm as follows: First compute all equivalence classes. Then enumerate all candidate polynomials obtained by trying all possible coefficients for these equivalence classes, and test for correctness using the **Consistent** procedure. We do not know how to accomplish this. However using the notion of a magic set, we *are* in fact able to implement (possibly a refinement of) this equivalence relation on P restricted to *all but a constant number of variables,* when P is constant-degree and read-constant.

3.3 Properties of Polynomials Equipped with a Magic Set

Before stating our learning algorithm, we establish a number of properties to be used later for polynomials P equipped with a magic set J.

Lemma 6. *Let $P(x) = \sum_{I \subseteq [n], |I| \leq d} c_I \prod_{i \in I} x_i$ be any polynomial over $\mathcal{R}$, with a magic set J. Let N be the set of indices that (viewed as vertices in the graph G_P) are at distance at least 2 from J in G_P. Then, $r + \sum_{I \subseteq N, 0 < |I| \leq d} \lambda_I c_I \in \text{range}(P)$, for all $r \in \text{range}(P)$ and all $\lambda_I \in \{0, \ldots, |\mathcal{R}| - 1\}$.*

Proof. We prove the statement by induction, first by induction on the degree d, and then by further induction on the monomials of degree d. The base case $d = 0$ trivially holds. Assume as induction hypothesis that $r + \sum_{I \subseteq N, 0 < |I| < d} \lambda_I c_I \in \text{range}(P)$ for all r and λ_I, and consider the case $d+1$. Enumerate all $\binom{|N|}{d}$ subsets of N of cardinality d, and let I^i denote the ith set in this enumeration. We shall now further induct on k to show that $r + \sum_{I \subseteq N, 0 < |I| < d} \lambda_I c_I + \sum_{i=1}^{k} \lambda_{I^i} c_{I^i} \in \text{range}(P)$ for every $r \in \text{range}(P)$. The $k = 0$ base case trivially holds. For the

inductive step, define $r_0 = r + \sum_{I \subseteq N, 0<|I|<d} \lambda_I c_I + \sum_{i=1}^{k} \lambda_{I^i} c_{I^i}$; we then want to show that $r_0 + \lambda c_{I^{k+1}} \in \text{range}(P)$ for every $\lambda \geq 0$. For $\lambda = 0$, there is nothing to show. So we assume $\lambda > 0$. By induction hypothesis and the magical set property of J, there exists u_0 with $I_{u_0} \subseteq J$ such that $r_0 = P(u_0)$. By the definition of N, no monomial in P can intersect both $I_{u_0} \subseteq J$ and $I^{k+1} \subseteq N$. Then clearly, $r_1 = P(u_0 \vee \chi_{I^{k+1}}) = r_0 + \sum_{I \subset I^{k+1}} c_I + c_{I^{k+1}} \in \text{range}(P)$. Hence, there is u_1 with $I_{u_1} \subseteq J$ such that $P(u_1) = r_1$. Continuing in this way for λ times we see that $r_\lambda = r_0 + \lambda \left(\sum_{I \subset I^{k+1}} c_I\right) + \lambda c_{I^{k+1}} \in \text{range}(P)$. Applying once more our outer induction hypothesis, we conclude $r_\lambda + (|\mathcal{R}| - \lambda)(\sum_{I \subset I^{k+1}} c_I) = r_0 + \lambda c_{I^{k+1}} \in \text{range}(P)$. This completes the inner and outer induction. □

In words, the lemma says that we remain in the range of P if we take any element in the range of P and add to it *any* linear combination of coefficients of monomials involving only variables at distance at least 2 from a fixed magic set.

For a polynomial P with accepting set A we can always obtain equivalent polynomial in which the constant term is 0 by *shifting* the accepting set according to the constant term. Thus in the following assume the constant term of P is 0. Let J be a magic set of P. Let N be the set of indices that are at distance 2 or more from J in G_P. Let P_N be the polynomial obtained from P by fixing to 0 every variable indexed in the set $[n] \setminus N$.

The crucial insight required for limiting the amount of nondeterministic guesses in our learning algorithm is expressed in the following lemma.

Lemma 7. *Let P be any polynomial of degree d with accepting set A and a magic set J, let N be as in Lemma 6, and $r \leq d$. Assume that for all monomials M_1 and M_2 in P_N of degree $s < r$ we have $c_{M_1} = c_{M_2}$ whenever $M_1 \sim_{s,J} M_2$. Consider now monomials M and M' of degree r in P_N such that $M \sim_{r,J} M'$. Let P' be the polynomial obtained from P by replacing the coefficient of M' with the coefficient of M. Then the pairs (P, A) and (P', A) compute the same Boolean function, and J is also a magic set for P'.*

Proof. Since $M \sim_{r,J} M'$, we have $(I_M, \preceq_M) \sim_{r,J} (I_{M'}, \preceq_{M'})$ for some $\preceq_M$ and $\preceq_{M'}$. Let us enumerate lexicographically the subsets of I_M and $I_{M'}$ according to $\preceq_M$ and $\preceq_{M'}$. Let M_i and ${M'}_i$ be the monomials corresponding to the ith such subsets and let d_i denote their degree. By definition we have $M_i \sim_{d_i,J} {M'}_i$, and so by assumption the coefficients of M_i and ${M'}_i$ are the same. We thus have that $P(\chi_M) = P'(\chi_{M'})$.

To prove that P and P' with accepting set A compute the same Boolean function, let $x \in \{0,1\}^n$ be arbitrary. Obviously, $P(x) \in \text{range}(P)$. By Lemma 6 we then have that also $P(x) - P(\chi_{M'}) \in \text{range}(P)$, because $-P(\chi_{M'})$ is obviously a linear combination of coefficients involving the variables in M'. It follows that there is u with $I_u \subseteq J$ such that $P(u) = P(x) - P(\chi_{M'})$. Since $M \sim_{r,J} M'$ we have $P(u \vee \chi_M) = P(u) + P(\chi_M) \in A$ if and only if $P(u \vee \chi_{M'}) = P(u) + P(\chi_{M'}) \in A$. But $P(u) + P(\chi_M) = P(x) - P(\chi_{M'}) + P(\chi_M) = P(x) - P(\chi_{M'}) + P'(\chi_{M'}) = P'(x)$ and $P(u) + P(\chi_{M'}) = P(x)$. Hence we can conclude that $P(x) \in A$ if and only if $P'(x) \in A$.

Finally, consider any assignment x. If x does not set to 1 all the variables in M', then $P'(x) = P(x)$ and so the element $P'(x)$ is achieved by some assignment setting only variables in J. If, on the other hand, x sets all the variables in M', then $P'(x) = P(x) - c_{M'} + c_M$. By Lemma 6 again, $P'(x)$ is in $range(P)$, hence it can be also achieved by appropriately setting only variables in J. This implies that J remains a magic set of P'. □

The following theorem says that every polynomial is equivalent to a structurally simpler one, in a sense that we will be able to exploit for learning:

Theorem 8. *Let P be a polynomial of degree d, A an accepting set, and J a magic set for P. Then there is a polynomial P', also of degree d, such that (P, A) and (P', A) compute the same Boolean function and the following holds for P': for every pair of monomials M_1, M_2 of the same degree not involving any variable at distance less than 2 from J, if $M_1 \sim_{d,J} M_2$ then $c_{M_1} = c_{M_2}$. Furthermore, J is a magic set for P' too and if P is read-k then P' is read-k too.*

Proof. Consider an ordering $M_1, M_2, \ldots, M_m$ of all monomials of degree up to d having only variables that are at distance at least 2 from J, with the property that $deg(M_i) \leq deg(M_{i+1})$. We will iteratively build a sequence of polynomials $P_0, P_1, \ldots, P_m$, starting with $P_0 = P$, so that

1. all (P_i, A) compute the same Boolean function,
2. J is a magic set for all P_i,
3. for every i, and every $j, k \leq i$, if M_j and M_k have the same degree and if $M_j \sim_{d,J} M_k$ then $c_{M_j} = c_{M_k}$ in P_i.

It is clear then that P_m satisfies the properties required for P' in the theorem. The construction can be regarded as "fixing" the coefficients in P one by one to satisfy the desired property.

Initially, the properties hold for $i = 0$ trivially. Assume inductively that they hold up to $i - 1$. That is, P_0 up to P_{i-1} with A all compute the same Boolean function, J is a magic set for all of them, and in P_{i-1} we have $c_{M_j} = c_{M_k}$ whenever $j, k \leq i - 1$ and $M_j \sim_{d,J} M_k$. Consider now monomial M_i. If for some $j < i$ we have M_j of the same degree as M_i and $M_j \sim_{d,J} M_i$, then in P_i we do the following: if either c_{M_i} or c_{M_j} is zero, then we set both to zero. Otherwise, we set c_{M_i} to the value of c_{M_j} in P_{i-1}. All other coefficients of P_i remain the same as in P_{i-1}; note that there is no conflict in defining so because $\sim_{d,J}$ is an equivalence relation and, for $j, k < i$, M_j and M_k already have the same coefficient if equivalent. Otherwise, if there is no such j, we define $P_i = P_{i-1}$.

It is clear now that property 3 holds for M_i as well. Because all submonomials of M_i and M_j appear before them in our enumeration of monomials, the assumption in the statement of Lemma 7 is satisfied, and then the lemma guarantees that P_i computes the same Boolean function as P_{i-1} and J is a magic set of P_i. This completes the induction.

The read-k property is preserved along the process since no zero coefficients get ever replaced with non-zero coefficients, hence the number of monomials in which a variable appears does not increase. □

3.4 The Learning Algorithm

We are now finally in position to state our algorithm.

Algorithm 2: Learn-Poly$_{\mathcal{R},d,k}(f)$

Input: Membership query access to Boolean function f.
Output: Returns pair (Q, A) computing the function f, or **fail**.

1: Nondeterministically guess the following:
A magic set $J \subseteq [n]$, $|J| \leq s(\mathcal{R}, d)$ and polynomial Q_J.
The set $K \subseteq [n]$ at distance 1 in G_P from J and polynomials Q_K and $Q_{K \times J}$.
The set $L \subseteq [n]$ at distance 2 in G_P from J, and polynomial $Q_{K \times L}$.
An accepting set $A \subseteq \mathcal{R}$ for Q.
2: Let $N = [n] \setminus (J \cup K)$.
3: Query f on all inputs $(w \vee x)$ where $I_w \subseteq J$, $I_x \subseteq N$, and $|I_x| \leq d$.
4: Compute the equivalence classes of $\sim_{r,J}$, for all $r = 1, \ldots, d$, over monomials M_I with $I \subseteq N$ and $|I| \leq d$.
5: Nondeterministically guess an element of $\mathcal{R}$ for each equivalence class.
6: Construct polynomial $Q = Q_{J \cup K} + Q_{K \times L} + \sum_{I \subseteq N, |I| \leq d} c_I \cdot M_I$, where $c_I \in \mathcal{R}$ is the element guessed for the equivalence class of monomial M_I.
7: If **Consistent**$_{\mathcal{R},d}(Q, A, f)$, output (Q, A), otherwise output **fail**.

Theorem 9. *Let $\mathcal{R}$ be a fixed commutative finite ring with unit. Let $F_{R,d,k}$ be the class of Boolean functions that can be computed by degree d, read-k polynomials over $\mathcal{R}$. Then* **Learn-poly**$_{\mathcal{R},d,k}$ *is a nondeterministic exact learning algorithm for $F_{\mathcal{R},d,k}$ running in time n^c, for some $c = c(\mathcal{R}, d, k)$.*

Proof. Fix a target function, some pair (P, A) computing it, and a smallest magic set J for this pair. Apply Theorem 8 to (P, A) and J and get an equivalent polynomial P' for whom equivalent monomials have the same coefficient, and P' is read-k.

Then at least one of the possible computation paths of **Learn-poly** will succeed, namely the one which nondeterministically guesses the correct values relative to P' and J in steps 1 and 5. Conversely, all paths with incorrect guesses which result in a candidate polynomial nonequivalent to the target function will be rejected by the **Consistent** procedure. □

It is clear that a deterministic algorithm can be derived from the one above by enumerating all possible guesses of **Learn-Poly**. We provide a brief analysis of the running time of this deterministic algorithm in terms of parameters $s(\mathcal{R}, d)$ and $c'(\mathcal{R}, d)$. The algorithm runs over all possible choices of a magic set J of size $s = s(\mathcal{R}, d)$, the set K, of size $ks(d-1)$, of variables that are at distance 1 from the set J and set L, of size $k^2 s(d-1)^2$, at distance 2 from J. For estimating the size of K and L, we have used the additional fact that the target function is represented by a read k-polynomial. Thus the total number of such choices is at most $n^{k^2 s d^2}$. For each such choice of J, K, L, the algorithm considers all possible degree d polynomials of variables indexed in $J \cup K$ and $K \times L$. Thus, it

has to consider at most $|\mathcal{R}|^{d(k^2sd^2)^d}$ many polynomials. Further, for each choice of J, K, L it does equivalence testing for monomials that are free of variables indexed by $J \cup K$. There are at most dn^d such monomials of degree d and the test for each involves 2^s assignments of variables in J. Thus, equivalence testing takes $O\big(dn^d \cdot 2^s\big)$ time. It is not hard to see that degree d monomials having only variables in N (at distance at least 2 from J) split up into at most $|\mathcal{R}|^{2^d}$ equivalence classes; this is because the class of one such monomial M is completely determined by the coefficients in P of all of its 2^d submonomials. Thus, one has to consider all possible ways of coloring equivalence classes with elements of $\mathcal{R}$ giving rise to $|\mathcal{R}|^{d|\mathcal{R}|^{2^d}}$ such choices. Finally having guessed an entire candidate polynomial, the algorithm invokes procedure **Consistent** that verifies the consistency of the polynomial with all assignments of weight at most $c' = c'(\mathcal{R}, d)$. This requires $O\big(n^{c'+1}\big)$ time. Summing these up, the total running time is

$$O(2^{|\mathcal{R}|}) \times O(n^{O(k^2sd^2)}) \times O(|\mathcal{R}|^{d(k^2sd^2)^d}) \times O(dn^d 2^s) \times O(|\mathcal{R}|^{d|\mathcal{R}|^{2^d}}) \times O\big(n^{c'+1}\big). \tag{1}$$

Let us also note the query complexity of the algorithm. It asks membership queries in two places: in step 3, with assignments of Hamming weight at most $s(\mathcal{R}, d)+d$, and in step 7, with assignments of Hamming weight at most $c'(\mathcal{R}, d)$. Thus, it asks $n^{O(\max\{s+d,c'\})}$ many membership queries in total.

Recall that for a fixed ring $\mathcal{R}$ and constant d, both $s(\mathcal{R}, d)$ and $c'(\mathcal{R}, d)$ are constants. We thus have:

Corollary 10. *There is a polynomial-time, Membership-query, exact learning deterministic algorithm for each class $F_{\mathcal{R},d,k}$, when $\mathcal{R}$, d, and k are fixed.*

4 Extension to Higher Degree

For a Boolean function f on n variables, define $\Delta(f, \mathcal{R})$ to be the minimal degree of a polynomial over $\mathcal{R}$ computing f. Consider a family of Boolean functions $f = \{f_n\}_{n=1}^{\infty}$, one for each input length. Define $\Delta(f, \mathcal{R}, n) = \Delta(f_n, \mathcal{R})$. Define $\Lambda(f, \mathcal{R}, d)$ as the maximal n such that $\Delta(f, \mathcal{R}, n) \leq d$.

The notion of the degree of the Boolean AND function allows the following quantitative version of Theorem 1. Recall from Theorem 1 that $c(\mathcal{R}, d)$ is the smallest number such that every value in the range of a degree-d polynomial over $\mathcal{R}$ can be obtained by an assignment of weight at most $c(\mathcal{R}, d)$.

Proposition 11. $c(\mathcal{R}, d) \leq \Lambda(\text{AND}, \mathcal{R}, d)$

Proof. Let P be a multilinear polynomial of degree d over $\mathcal{R}$ in n variables. Let $r \in \text{range}(P)$. We will find $w \in \{0,1\}^n$ with $|I_w| \leq \Lambda(\text{AND}, \mathcal{R}, d)$ such that $P(w) = r$. If $P(0) = r$, we are done. Otherwise, pick $w \in \{0,1\}^n$ such that $|I_w|$ is minimal with $P(w) = r$. Let P' be the polynomial obtained from P by fixing to 0 all variables not in I_w. By minimality of $|I_w|$, we have that P' computes the AND function with accepting set $\{r\}$ on $|I_w|$ variables. It follows that $|I_w| \leq \Lambda(\text{AND}, \mathcal{R}, d)$. □

As a consequence, we obtain the following bounds for $s(\mathcal{R}, d)$-the size of the magic set for polynomials over $\mathcal{R}$ of degree d, and $c'(\mathcal{R}, d)$-the Hamming weight of assignments that uniquely identify a Boolean function represented by such a polynomial, in terms of $\Lambda(\mathrm{AND}, \mathcal{R}, d)$ as well, following the proofs of Corollary 3 and Corollary 4.

Corollary 12. *$s(\mathcal{R}, d) \leq |\mathcal{R}|\Lambda(\mathrm{AND}, \mathcal{R}, d)$ and $c'(\mathcal{R}, d) \leq \Lambda(\mathrm{AND}, \mathcal{R} \times \mathcal{R}, d)$.*

Thus, lower bounds for the degree of the AND function implies upper bounds on the above quantities. The degree of the AND function has been intensively studied over the ring $\mathbb{Z}_m$ [5, 27]. Let in the following $m = p_1^{k_1} \cdots p_r^{k_r}$ have r distinct prime factors, and let $q_{\min} = \min(p_1^{k_1}, \ldots, p_r^{k_r})$ and $q_{\max} = \max(p_1^{k_1}, \ldots, p_r^{k_r})$. With these definitions, Tardos and Barrington [27] obtained the following lower bound, which is currently the best known.

Theorem 13 (Tardos and Barrington).
$\Delta(\mathrm{AND}, \mathbb{Z}_m, n) \geq ((1/(q_{\min} - 1) - o(1)) \log n)^{1/(r-1)}$.
Equivalently, $\Lambda(\mathrm{AND}, \mathbb{Z}_m, d) \leq 2^{(q_{\min} - 1 + o(1))d^{r-1}}$

Now, we want to apply the learning results in the previous section to rings of the form $\mathcal{R} = \mathbb{Z}_m^l$. In fact, we do not loose any generality if we assume $\mathcal{R}$ is of this form: The variables in our polynomials only take $\{0, 1\}$ values, and therefore we only use the additive structure of $\mathcal{R}$. This is an Abelian group, therefore of the form $\mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_l}$, which we can easily simulate by $\mathbb{Z}_m^l$ for m the least common multiple of the m_i.

The next lemma transfers the above results to Z_m^l using standard methods.

Lemma 14. *Let m' be $p_1 \cdots p_r$. Then:*

$$\Delta(\mathrm{AND}, \mathbb{Z}_{m'}, n) \leq l(q_{\max} - 1)\Delta(\mathrm{AND}, \mathbb{Z}_m^l, n).$$

Equivalently, $\Lambda(\mathrm{AND}, \mathbb{Z}_m^l, d) \leq \Lambda(\mathrm{AND}, \mathbb{Z}_{m'}, l(q_{\max} - 1)d)$.

Proof. Let P be a polynomial over $\mathbb{Z}_m^l$ of degree d computing the AND function. Without loss of generality, the accepting set is $\{\mathbf{0}\}$: Otherwise, replace P with $P - P(1, \ldots, 1)$; it is easy to check that this polynomial with accepting set $\{\mathbf{0}\}$ also computes AND and has the same degree as P. Let $P_1, \ldots, P_l$ be the l coordinate polynomials. Consider a fixed j, and the polynomials $P_1, \ldots, P_l$ modulo $p_j^{k_j}$. By well-known arguments (see e.g [27]) we can find polynomials $Q_1^j, \ldots, Q_l^j$ of degree at most $(p_j^{k_j} - 1)d$ such that $Q_i^j(x) \equiv 0 \pmod{p_j}$ if and only if $P_i(x) \equiv 0 \pmod{p_j^{k_j}}$, and furthermore $(Q_i^j(x) \mod p_j) \in \{0, 1\}$ for all x.

Define $Q^j(x) = 1 - \prod_{i=1}^{l}(1 - Q_i^j(x))$. We then have, for every x, that $Q^j(x) \equiv 0 \pmod{p_j}$ if and only if for every i $P_i(x) \equiv 0 \pmod{p_j^{k_j}}$. Note the degree of Q^j is at most $l(p_j^{k_j} - 1)d$.

Considering all such polynomials, $Q^1, \ldots, Q^l$, from the Chinese Remainder Theorem we may find a polynomial Q, of degree at most $l(q_{\max} - 1)d$ such that $Q(x) \equiv 0 \pmod{m'}$ if and only if $P(x) = 0$ for all x. □

Combining Proposition 11, Theorem 13, Lemma 14, and Corollary 12 we obtain the following concrete bounds (following the proofs of Corollaries 3 and 4):

Proposition 15. *Consider positive integers m and l, where $m = p_1^{k_1} \cdots p_r^{k_r}$. Let $q_{\max}$ be the largest prime power dividing m and let $p_{\min}$ be the smallest prime factor of m. Then:*

$$c(\mathbb{Z}_m^l, d) \leq \Lambda(\mathrm{AND}, \mathbb{Z}_m^l, d) \leq 2^{(p_{\min}-1+o(1))(l(q_{\max}-1)d)^{r-1}} .$$
$$s(\mathbb{Z}_m^l, d) \leq |\mathbb{Z}_m^l| c(\mathbb{Z}_m^l, d) \leq m^l 2^{(p_{\min}-1+o(1))(l(q_{\max}-1)d)^{r-1}} .$$
$$c'(\mathbb{Z}_m^l, d) \leq c(\mathbb{Z}_m^{2l}, d) \leq 2^{(p_{\min}-1+o(1))(2l(q_{\max}-1)d)^{r-1}} .$$

Proof. We first remark that the second and third chain of inequalities in the above proposition follow almost directly from the first. We prove the first chain now. The first inequality in this chain relating c and Λ is direct from Proposition 11. For the second inequality, note that Lemma 14 says $\Lambda(\mathrm{AND}, \mathbb{Z}_m^l, d) \leq \Lambda(\mathrm{AND}, \mathbb{Z}_{m'}, l(q_{\max}-1)d)$, where $m' = p_1 \cdots p_r$. Hence, applying Theorem 13 by substituting m with m', we are done. □

Using Proposition 15, we see that for each $\mathcal{R} = \mathbb{Z}_m^\ell$ with a fixed m and ℓ, there exists a constant γ such that $s(\mathcal{R}, d), c'(\mathcal{R}, d) \leq \gamma^{d^{r-1}}$, where r is the number of distinct prime factors of m. Hence, recalling the running time of the deterministic version of **Learn-Poly** from expression (1), we get the following:

Theorem 16. *Let m and ℓ be any fixed positive numbers. The class of Boolean functions representable by read-k polynomials of degree d over $\mathbb{Z}_m^\ell$ are exactly learnable from membership queries by a deterministic algorithm of running time $O\big(n^{k^2\gamma^{d^r}d^2} \times \gamma^{k^{2d}\gamma^{d^r}d^{2d}} \times \gamma^{\gamma^{2^d}}\big)$, where $\gamma = \gamma(m, \ell)$ is a constant and r is the number of distinct prime factors of m.*

Theorem 16 gives us a range of super-constant k and d for which we get sub-exponential running time. For instance, if we choose $k = o(\log\log n)$, and $d = o(\log\log\log n)$, the running time is $n^{(\log n)^{o(1)}}$.

5 Future Work

While the progress we make is limited from a learning theory perspective, the combinatorics involved is unexpectedly delicate, and suggests some further questions in understanding the structure of polynomials over rings of the form $\mathbb{Z}_m$.

The obvious next question is to remove the read-constant restriction in our result. Read-constant restrictions have been used, on several occasions, both in complexity theory and in learning theory. For example in complexity theory, Barrington and Straubing [6] proved superlinear bounds on the length of read-constant branching programs of bounded-width. Very recently, several works have been concerned with constructing pseudorandom generators for read-once branching programs of small width [11, 12, 20].

In learning theory, read-constant conditions have been sometimes shown to be unavoidable for efficient learning. For example, read-once Boolean formulas are known to be learnable efficiently from evaluation and equivalence queries [3]. Read-twice DNF formulas [23] are properly learnable with these queries, while read-thrice DNF are not assuming NP $\neq$ coNP [2]. In another direction, we have already mentioned that polynomials of constant degree over finite rings are equivalent in power to programs over nilpotent groups [21, 22]. The proof of equivalence, however, does not preserve any restrictions on the number of times a variable is read, in either direction. Therefore, our result does not seem to imply anything in the direction of learning programs over nilpotent groups.

In other cases, read-constant conditions for learning a target concept class can be removed at the expense of moving to a larger hypothesis class, which bypasses some computational bottleneck. For example, Aizenstein *et al.* [1] showed that read-k, satisfy-j DNF formulas[7] are learnable (as DNF formulas). Without the read-k condition, satisfy-j DNF formulas are not known to be learnable as DNF, but they can be learned as Multiplicity Automata, as pointed out in [9]. Analogously, it is possible that constant degree polynomials over finite rings can be learned (in some reasonable learning model) by not insisting that the output is itself a constant degree polynomial.

Acknowledgments A. Chattopadhyay was at the University of Toronto when this work was written up, partially supported by a Natural Sciences and Engineering Research Council (NSERC) postdoctoral fellowship, an Ontario Ministry of Research and Innovation fellowship and research grants of Prof. T. Pitassi. R. Gavaldà is partially funded by the Spanish MICINN projects TIN-2008-06582-C03-01 (SESAAME) and TIN2011-27479-C04-03 (BASMATI), by the Generalitat de Catalunya SGR2009-1428 (LARCA), and by the EU PASCAL2 Network of Excellence (FP7-ICT-216886). We thank the anonymous reviewers for their useful comments that improved the presentation. In particular, Remark 5 was pointed out by a reviewer.

References

1. H. Aizenstein, A. Blum, R. Khardon, E. Kushilevitz, L. Pitt, and D. Roth. On learning read-k-satisfy-j dnf. *SIAM J. Comput.*, 27(6):1515–1530, 1998.
2. H. Aizenstein, T. Hegedüs, L. Hellerstein, and L. Pitt. Complexity theoretic hardness results for query learning. *Comput. Complexity*, 7(1):19–53, 1998.
3. D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *J. ACM*, 40(1):185–210, 1993.
4. D. A. M. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.
5. D. A. M. Barrington, R. Beigel, and S. Rudich. Representing boolean functions as polynomials modulo composite numbers. *Comput. Complexity*, 4:367–382, 1994.

[7] A DNF formula is read-k if every variable appears at most k times; a DNF formula is satisfy-j if no assignment satisfies more than j terms simultaneously

6. D. A. M. Barrington and H. Straubing. Superlinear lower bounds for bounded-width branching programs. *J. Comput. Syst. Sci.*, 50(3):374–381, 1995.
7. D. A. M. Barrington, H. Straubing, and D. Thérien. Non-uniform automata over groups. *Inform. and Comput.*, 89(2):109–132, 1990.
8. D. A. M. Barrington and D. Thérien. Finite monoids and the finite structure of NC^1. *J. ACM*, 35(4):941–952, 1988.
9. A. Beimel, F. Bergadano, N. Bshouty, E. Kushilevitz, and S. Varricchio. Learning functions represented as multiplicity automata. *J. ACM*, 47:506–530, 2000.
10. J. Bourgain. Estimation of certain exponential sums arising in complexity theory. *C. R. Acad. Sci. Paris*, 340(9):627–631, 2005.
11. M. Braverman, A. Rao, R. Raz, and A. Yehudayoff. Pseudorandom generators for regular branching programs. In *51th Annual IEEE Symposium on Foundations of Computer Science*, pages 40–47. IEEE Computer Society, 2010.
12. J. Brody and E. Verbin. The coin problem, and pseudorandomness for branching programs. In *51th Annual IEEE Symposium on Foundations of Computer Science*, pages 30–39. IEEE Computer Society, 2010.
13. N. Bshouty, C. Tamon, and D. Wilson. Learning matrix functions over rings. *Algorithmica*, 22:91–111, 1998.
14. A. Chattopadhyay. Multilinear polynomials modulo composites. *Bulletin of the European Association on Theoretical Computer Science, Computational Complexity Column*, 100, February 2010.
15. K. Efremenko. 3-query locally decodable codes of subexponential length. In *41st Annual Symposium on Theory of Computing*, pages 39–44. ACM Press, 2009.
16. R. Gavaldà and D. Thérien. An algebraic perspective on boolean function learning. In *Algorithmic Learning Theory, 20th International Conference, ALT 2009*, LNCS, pages 201–215. Springer, 2009.
17. P. Gopalan. Constructing ramsey graphs from boolean function representations. In *21st Annual IEEE Conference on Computational Complexity*, pages 115–128. IEEE Computer Society, 2006.
18. V. Grolmusz. On the weak mod m representation of boolean functions. *Chicago J. Theoret. Comput. Sci.*, 1995.
19. D. P. Helmbold, R. H. Sloan, and M. K. Warmuth. Learning nested differences of intersection-closed concept classes. *Machine Learning*, 5:165–196, 1990.
20. M. Koucký, P. Nimbhorkar, and P. Pudlák. Pseudorandom generators for group products. In *43rd Annual ACM Symposium on Theory of Computing*. ACM Press, 2011. (to appear).
21. P. Péladeau and D. Thérien. Sur les langages reconnus par des groupes nilpotents. *C. R. Math. Acad. Sci. Paris, t. 306, Série I*, 306(2):93–95, 1988.
22. P. Péladeau and D. Thérien. On the languages recognized by nilpotent groups (a translation of "Sur les langages reconnus par des groupes nilpotents" [21]). *Electronic Colloquium on Computational Complexity (ECCC)*, 8(40), 2001.
23. K. Pillaipakkamnatt and V. V. Raghavan. Read-twice dnf formulas are properly learnable. *Inf. Comput.*, 122(2):236–267, 1995.
24. A. A. Razborov. Lower bounds on the size of bounded-depth networks over a complete basis with logical addition. *Math. Notes of the Acad. of Sci. of USSR*, 41(3):333–338, 1987.
25. R. E. Schapire and L. Sellie. Learning sparse multivariate polynomials over a field with queries and counterexamples. *J. Comput. Syst. Sci.*, 52(2):201–213, 1996.
26. R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *19th Annual ACM Symposium on Theory of Computing*, pages 77–82. ACM Press, 1987.

27. G. Tardos and D. A. M. Barrington. A lower bound on the MOD-6 degree of the OR function. *Comput. Complexity*, 7(2):99–108, 1998.