Simple Learning Algorithms for Decision Trees and Multivariate Polynomials

Nader H. Bshouty
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada

Yishay Mansour
Department of Computer Science
Tel-Aviv University
Tel-Aviv, Israel

Abstract

There were two techniques in the literature for learning multivariate polynomials and decision trees. These are learning decision trees under the uniform distribution via the Fourier Spectrum [Kushilevitz, Mansour 93 and Jackson 94] and learning decision trees and multivariate polynomials under any distribution via Lattice Theory [Bshouty 94, Schapire and Sellie 93]. These two approaches are used for proving the learnability of many other interesting classes such as CDNF (poly size DNF and CNF) under any distribution and DNF and AC^0 under the uniform distribution.

In this paper we develop a new approach. Learning decision trees and multivariate polynomials via multivariate interpolation. This is an algebraic approach that is a modification of the approaches used in [Roth, Benedek 89 and Bshouty 95] and is based on the divide and conquer technique for learning [Bshouty 95]. This new approach yelds simple learning algorithms for multivariate polynomials and decision trees over fields under *any* constant bounded product distribution. The output hypothesis is a (single) multivariate polynomial that is an ϵ-approximation of the target under *any* constant bounded product distribution. Previous approaches do not achieve this property.

The new approach also solves learnability of many other problems that cannot be solved using previous approaches. Problems include PAC-learning with membership queries of j-disjoint DNF (intersection of any j terms is empty) under any constant bounded product distribution, PAC-learning with membership queries of multivariate polynomial with bounded degree over any field under the constant bounded product distribution. Learning multivariate polynomials with bounded term size. This in particular gives a learning algorithm for $O(\log n)$-depth decision tree from membership queries only and a new learning algorithm of *any* multivariate polynomial over sufficiently large fields from membership queries only. We also show that all the results we have for learning from membership queries only are the best that can be achieved. We also prove learnability of Xor of decision trees and sum of decision trees with leaves that contain values from some field.

1 Introduction

Two techniques were used in the literature for PAC-learning decision tree with membership queries. Kushilevitz and Mansour gave in [KM93] a technique for learning decision trees under the uniform distribution via the Fourier Spectrum. Jackson in [J94] extended the result to learning DNF under the uniform distribution. The output hypothesis is a majority of parities. Bshouty gave in [Bs94] a technique for learning decision trees under any distribution via the Monotone Theory. Schapire and Sellie gave in [SS93] a Lattice based algorithm for learning multivariate polynomials over the binary field under any distribution. In the former the output hypothesis for the decision tree is depth 3 formulas. Jackson [J95] generalizes his DNF learning algorithm from uniform distribution to any fixed constant bounded product distribution. This is a distribution where there is a constant $0 < c < 1/2$ that is independent of the number of variables n such that for any variable x_i the probability that $x_i = 1$ is bounded between c and $1 - c$.

Both techniques, the Fourier Spectrum and the Lattice based algorithms give also learnability of many other classes such as learning decision trees over parities (nodes contains parities) under constant bounded product distributions and learning CDNF (poly size DNF that has poly size CNF) under any distribution.

In this paper we use a new approach to learning decision trees and multivariate polynomials via multivariate interpolation. This new approach leads to simple learning algorithms for decision trees over constant bounded product distributions that output hypotheses that is a multivariate polynomial (parity of terms). The algorithm we develop gives a single hypothesis that approximate the target against *any* constant bounded product distribution. In fact the hypothesis is a good hypothesis under any distribution that *supports small terms*. This is a distribution D that satisfies $\Pr_D[T = 1] = 1/\omega(poly(n))$ for any term T of size $\omega(\log n)$ where n is the number of variables.

The new approach also solves learnability of many other problems, some of which cannot be solved using the previous known approaches. These problems include

1. PAC-learning with membership queries of multivariate polynomials over the binary field with nonmonotone terms under any distribution that support small terms. In particular disjoint DNF (the disjunction of every two terms is 0), j-disjoint DNF (the disjunction of any j terms is 0) for constant j, decision trees and any polynomial number of Xor of them are PAC-learnable with membership queries under any distribution that supports small terms. The output hypotheses of the learning algorithm is a multivariate polynomial.

It is known that multivariate polynomials (with monotone terms) are PAC-learnable with membership queries under any distribution [SS93]. Our contribution is to show the same when the terms are not nesasary monotone. It is also known that any DNF is PAC-learnable with membership queries under constant bounded product distribution [J94]. In [J94] the output hypothesis is a majority of parities. Our contribution for j-disjoint DNF is to use an output hypothesis that is a parity of terms and to show that the output hypothesis is an ϵ approximation of the target against any constant bounded distribution. Also, the learnability of a Xor of these classes cannot be achieved using previous known approaches.

We also have an extension of (1.) to any field. We show

2. PAC-learning with membership queries of decision trees with leaves from some field $\mathcal{F}$ and any polynomial sum of them under any distribution that support small terms.

We also study the learnability of multivariate polynomials from membership queries only. We show

3. Learning multivariate polynomials over n variables with maximal degree $d < c|\mathcal{F}|$ for each variable, where c is constant, and with terms of size

$$k = O\left(\frac{|\mathcal{F}|}{d}(\log n + \log d)\right)$$

from membership queries only. This result implies learning decision trees of depth $O(\log n)$ with leaves from a field $\mathcal{F}$ from membership queries only. We also show that the above term size is tight, i.e., there is no membership query algorithm that learns multivariate polynomials over n variables with maximal degree d that contain terms of size greater than k.

The first result in (3.) is a generalization of the result in [Bs95]. In [Bs95] the learning algorithm uses membership and equivalence queries. The second result is a generalization of the result in [KM93] for learning boolean decision tree from membership queries. Result 3. also gives

4. An algorithm for learning *any* multivariate polynomial over fields of size $q = n/(d(\log n + \log d))$ from membership queries only. We also show that the above field size q is tight, i.e., if $|\mathcal{F}| = \omega(q)$ then no polynomial time learning algorithm exists.

This result is a generalization of the results in [BT88, CDG+91, Z90] for learning multivariate polynomials under any field. Previous algorithms for learning multivariate polynomial over finite fields $\mathcal{F}$ require asking membership queries with assignments in some extension of the field $\mathcal{F}$ [CDG+91]. In [CDG+91] it is shown that an extension n of the field is sufficient to interpolate any multivariate polynomial (when membership queries with assignments from an extension field are allowed). Our result in (4.) improves this extension bound to $1 + \log(n)/\log|\mathcal{F}| \le 1 + \log n$.

We also prove

5. The following problems for multivariate polynomials are equivalent (1) Distinguishing a multivariate polynomial from zero. (2) Distinguishing any two multivariate polynomials. (3) Deciding whether a multivariate polynomial depend on some variable. (4) Learning a multivariate polynomial.

2 Multivariate Interpolation

Let

$$f = \sum_{\alpha\in\mathcal{I}} a_\alpha x_1^{\alpha_1}\cdots x_n^{\alpha_n}$$

be a multivariate polynomial over the field $\mathcal{F}$ where $a_\alpha \in \mathcal{F}$ and $\alpha_1, \ldots, \alpha_n$ are integers. We will denote the class of all multivariate polynomials over the field $\mathcal{F}$ and over the variables $x_1, \ldots, x_n$ by $\mathcal{F}[x_1, \ldots, x_n]$. The number of terms of f is denoted by $|f|$. We have $|f| = |\mathcal{I}|$ when all a_α are not zero. When $f = 0$ then $|f| = 0$ and when $f = c \in \mathcal{F}\backslash\{0\}$ then $|f| = 1$. Let d be the maximal degree of variables in f, i.e., $\mathcal{I} \subseteq [d]^n$ where $[d] = \{0, 1, \ldots, d\}$. Suppose $\mathcal{F}' = \{\gamma_0, \ldots, \gamma_d\} \subseteq \mathcal{F}$ are $d+1$ distinct field constants where $\gamma_0 = 0$ is the zero of the field. A univariate polynomial $f(x_1) \in \mathcal{F}[x_1]$ over the field $\mathcal{F}$ of degree at most d can be interpolated from membership queries as follows. Suppose

$$f(x_1) = \Delta^{(d)}(f)x_1^d + \cdots + \Delta^{(1)}(f)x_1 + \Delta^{(0)}(f)$$

where $\Delta^{(i)}(f)$ is the coefficient of x^i in f in its polynomial representation. Then

$$\left\{\begin{array}{lcl} f(\gamma_0) & = & \Delta^{(d)}(f)\gamma_0^d + \cdots + \Delta^{(1)}(f)\gamma_0 + \Delta^{(0)}(f) \\ f(\gamma_1) & = & \Delta^{(d)}(f)\gamma_1^d + \cdots + \Delta^{(1)}(f)\gamma_1 + \Delta^{(0)}(f) \\ \vdots & \vdots & \vdots \\ f(\gamma_d) & = & \Delta^{(d)}(f)\gamma_d^d + \cdots + \Delta^{(1)}(f)\gamma_d + \Delta^{(0)}(f). \end{array}\right.$$

This is a linear system of equations and can be solved as follows.

$$\Delta^{(i)}(f) = \frac{det\begin{vmatrix} \gamma_0^d & \cdots & \gamma_0^{i+1} & f(\gamma_0) & \gamma_0^{i-1} & \cdots & \gamma_0 & 1 \\ \gamma_1^d & \cdots & \gamma_1^{i+1} & f(\gamma_1) & \gamma_1^{i-1} & \cdots & \gamma_1 & 1 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\ \gamma_d^d & \cdots & \gamma_d^{i+1} & f(\gamma_d) & \gamma_d^{i-1} & \cdots & \gamma_d & 1 \end{vmatrix}}{det|V(\gamma_0, \ldots, \gamma_d)|} \qquad (1)$$

where $V(\gamma_0, \ldots, \gamma_d)$ is the Vandermonde matrix.

If f is a multivariate polynomial then f can be written as

$$f(x_1, \ldots, x_n) = \Delta^{(d)}(f)x_1^d + \cdots + \Delta^{(1)}(f)x_1 + \Delta^{(0)}(f)$$

where $\Delta^{(i)}(f)$ is a multivariate polynomial over the variables $x_2, \ldots, x_n$. We can still use (1) to find $\Delta^{(i)}(f)$. Just replace each $f(\gamma_i)$ with $f(\gamma_i, x_2, \ldots, x_n)$. Notice that from the first equation in the system, since $\gamma_0 = 0$, we have

$$\Delta^{(0)}(f) = f(0, x_1, \ldots, x_n). \qquad (2)$$

¿From (1) a membership query for $\Delta^{(i)}$ can be simulated using $d+1$ membership queries to f. From (2), a membership query to $\Delta^{(0)}$ can be simulated using one membership query to f.

For boolean function we have

$$\Delta^{(0)}(f) = f(0, x_1, \ldots, x_n), \quad \Delta^{(1)}(f) = f(1, x_1, \ldots, x_n) + f(0, x_1, \ldots, x_n). \qquad (3)$$

$\Delta^{(0)}(f)$ gives the terms in f that do not contain x_1 and $\Delta^{(1)}(f)$ gives the terms that contains x_1 with $x_1 = 1$. Therefore for boolean functions we have $f = \Delta^{(0)}(f) + \Delta^{(1)}(f)x_1$.

We now extend the Δ operators as follows: for $\mathbf{i} = (i_1, \ldots, i_k) \in [d]^k$

$$\Delta^{\mathbf{i}} = \Delta^{i_k}\Delta^{i_{k-1}} \cdots \Delta^{i_1}.$$

Here Δ always operates on the variable with the smallest index. So Δ^{i_1} operates on x_1 in f to give a function f' that depends on $x_2, \ldots, x_n$. Then Δ^{i_2} operates on x_2 in f' and so on. We will also write $x^{\mathbf{i}}$ for the term $x_1^{i_1}x_2^{i_2} \cdots x_k^{i_k}$. The weight of $\mathbf{i}$, denoted by $wt(\mathbf{i})$, is the number of nonzero entries in $\mathbf{i}$.

The operator $\Delta^i(f)$ gives the coefficient of x_1^i in f when represented in $\mathcal{F}[x_2, \ldots, x_n]$ $[x_1]$, the operator $\Delta^{\mathbf{i}}(f)$ gives the coefficient of $x^{\mathbf{i}}$ when f is represented in

$$\mathcal{F}[x_{k+1}, \ldots, x_n][x_1, \ldots, x_k].$$

Suppose $\mathcal{I} \subseteq [d]^k$ be such that $\Delta^{\mathbf{i}} f \neq 0$ for all $\mathbf{i} \in \mathcal{I}$ and $\Delta^{\mathbf{i}} f = 0$ for all $\mathbf{i} \notin \mathcal{I}$, that is, $x^{\mathbf{i}}$ for $\mathbf{i} \in \mathcal{I}$ are the k-suffixes of all terms of f. Here the k-suffix of a term $x_1^{i_1} \cdots x_n^{i_n}$ is $x_1^{i_1} \cdots x_k^{i_k}$. Since $\mathbf{i} \in \mathcal{I}$ if and only if $x^{\mathbf{i}}$ is a k-suffix of some term in f, it is clear that $|\mathcal{I}| \leq |f|$ and we must have

$$f = \sum_{\mathbf{i} \in \mathcal{I}} (\Delta^{\mathbf{i}} f) x^{\mathbf{i}}.$$

We now will show how to simulate membership queries for $(\Delta^{\mathbf{i}} f)(x_{k+1}, \ldots, x_n)$, $\mathbf{i} \in \mathcal{I}$, using a polynomial number (in n and $|f|$) of membership queries to f. Suppose we want to find $(\Delta^{\mathbf{i}} f)(c)$ for some $c \in \mathcal{F}^{n-k}$ using membership query to f. We take r assignments $\hat{\gamma}_1, \ldots, \hat{\gamma}_r \in \mathcal{F}^k$ and ask membership queries for $(\hat{\gamma}_i, c)$ for all $i = 1, \ldots, r$. If $f(\hat{\gamma}_i, c) = \omega_i$ then

$$\left\{ \begin{array}{lcl} \sum_{\mathbf{i} \in \mathcal{I}} (\Delta^{\mathbf{i}} f)(c) \hat{\gamma}_1^{\mathbf{i}} & = & \omega_1 \\ \vdots & \vdots & \vdots \\ \sum_{\mathbf{i} \in \mathcal{I}} (\Delta^{\mathbf{i}} f)(c) \hat{\gamma}_r^{\mathbf{i}} & = & \omega_r \end{array} \right. .$$

Now if $\mathcal{I} = \{\mathbf{i}_1, \ldots, \mathbf{i}_r\}$ and $det|\mathbf{M}[\hat{\gamma}_j; \mathbf{i}_j]| \neq 0$ for

$$\mathbf{M}[\hat{\gamma}_j, \mathbf{i}_j] = \left[\begin{array}{ccc} \hat{\gamma}_1^{\mathbf{i}_1} & \cdots & \hat{\gamma}_1^{\mathbf{i}_r} \\ \vdots & \vdots & \vdots \\ \hat{\gamma}_r^{\mathbf{i}_1} & \cdots & \hat{\gamma}_r^{\mathbf{i}_r} \end{array} \right]$$

then the above linear system of equations can be solved in time $poly(r) = poly(|\mathcal{I}|) \leq poly(|f|)$. The solution gives $(\Delta^{\mathbf{i}} f)(c)$. The existence of $\hat{\gamma}_i$ where the above determinant is not zero will be proven in the next section.

3 Simple Algorithm for the Boolean Domain

Let $MUL(n,t)$ be the set of all multivariate polynomials over the binary field over n variables with t terms where each term is of size $O(\log n)$. This class includes decision trees of depth $O(\log n)$ and j-disjoint $O(\log n)$-DNF.

We first show how to zero-test elements in $MUL(n,t)$. Let $f \in MUL(n,t)$. Choose a term $T = x_{i_1} \cdots x_{i_k}$ of maximal size in f. Randomly and uniformly choose values in $\{0,1\}$ for the variables not in T. The projection will not be zero because the term T will stay alive in the projection. Since the projection is a nonzero function with $k = O(\log n)$ variables there are values for $x_{i_1}, \ldots, x_{i_k}$ that gives value 1 for the function. This show that for a random and uniform assignment a, $f(a) = 1$ with probability at least $1/2^k = 1/poly(n)$. So to zero test a function in $f \in MUL(n,t)$ randomly and uniformly choose polynomial number of assignments a_i. If $f(a_i)$ is zero for all the assignments then with probability close to 1 we have $f \equiv 0$.

For decision tree of depth $O(\log n)$ we can deterministicly zero-test by choosing the assignments to be the $(n, O(\log n))$-universal set. For details see [Bs93].

We now show how to reduce zero-test to learning. Let $f_0 = f$. Since we can zero-test we can find the minimal i_1 such that $f_0|_{x_1 \leftarrow 0,\ldots,\ldots,x_{i_1} \leftarrow 0} \equiv 0$. This implies that $f_{x_1 \leftarrow 0,\ldots,\ldots,x_{i_1-1} \leftarrow 0} = x_{i_1} f_1(x_{i_1+1}, \ldots, x_n)$ for some multivariate polynomial $f_1 \in MUL_{\mathcal{F}}(n,k,t,d)$. Now do the same for $f_1 = f_{x_1 \leftarrow 0,\ldots,\ldots,x_{i_1-1} \leftarrow 0, x_{i_1} \leftarrow 1}$. At some stage we get $f_0 = x_{i_1} x_{i_2} \cdots x_{i_k} f_k$. If $f_k \equiv 1$ then $x_{i_1} \cdots x_{i_k}$ is a term in f. Now define $\hat{f} = f + x_{i_1} \cdots x_{i_k}$. This removes this term from f. Then do the same for $\hat{f}$ to get a new term. This way we collect all the terms of f.

Now let MUL be the set of all boolean multivariate polynomials with polynomial number of nonmonotone terms. This class includes the class of decision trees and j-disjoint DNF.

Let $f \in MUL$. To PAC-learn f we randomly choose an assignment a and define $f' = f(x + a)$. This make sure that (large) terms in f of size k will have in average $k/2$ positive literals. We now take a zero-restriction. That is we substitute $x_i \leftarrow 0$ in f' with probability 1/2. This make sure that with high probability the projection f'' is in $MUL(n, poly(n))$. We then use the previous algorithm to learn f''. The terms of f'' are terms in f'. We continue to take zero-restrictions and collect small terms of f' until we get a good approximation of f'. A complete analysis of this algorithm is given in section 6.

4 From Zero-testing to Learning for any Field

In this section we show how to use the results from the previous section to learn multivariate polynomials.

Let $MUL_{\mathcal{F}}(n,k,t,d)$ be the set of all multivariate polynomial over the field $\mathcal{F}$ over n variables with t terms where each term is of size k and the maximal degree of each variable is at most d. We would like to answer the following 3 questions. Let $f \in MUL_{\mathcal{F}}(n,k,t,d)$.

1. Is there a polynomial time algorithm that uses membership queries to f and decides whether $f \equiv 0$?

2. Given $i \leq n$. Is there a polynomial time algorithm that uses membership queries to f and decides whether f depends on x_i?

3. Given $\{\mathbf{i}_1, \ldots, \mathbf{i}_r\} \subseteq [d]^n$ where $wt(\mathbf{i}_j) \le k$ for all j and $r \le t$. Is there an algorithm that runs in polynomial time and finds $\hat{\gamma}_1, \ldots, \hat{\gamma}_r \in \mathcal{F}^k$ such that

$$\begin{vmatrix} \hat{\gamma}_1^{\mathbf{i}_1} & \cdots & \hat{\gamma}_1^{\mathbf{i}_r} \\ \vdots & \vdots & \vdots \\ \hat{\gamma}_r^{\mathbf{i}_1} & \cdots & \hat{\gamma}_r^{\mathbf{i}_r} \end{vmatrix} \neq 0.$$

4. Is there a polynomial time algorithm that uses membership queries to f and identifies f.

When we say polynomial time we usually mean polynomial time in n, k, t and d but all the results of this section are also true for any time complexity T except that to solve 4 we get a blow up of $poly(n,t)$ in the complexity.

We show that 1,2 and 4 are equivalent and $1 \Rightarrow 3$. Obviously $2 \Rightarrow 1$, $4 \Rightarrow 1$ and $4 \Rightarrow 2$. We will show $1 \Rightarrow 2$, $1 \Rightarrow 3$, and $1+2+3 \Rightarrow 4$.

To prove $1 \Rightarrow 2$ notice that $f \in MUL_{\mathcal{F}}(n,k,t,d)$ is independent of x_i if and only if $g = f|_{x_i \leftarrow 1} - f|_{x_i \leftarrow 0} \equiv 0$. Since g is the coefficient of x_i in f we have $g \in MUL_{\mathcal{F}}(n,k,t,d)$. Therefore we can zero-test g in polynomial time.

To prove $1 \Rightarrow 3$, let $\hat{\gamma}_1, \ldots, \hat{\gamma}_s$ be a zero-test for functions in $MUL_{\mathcal{F}}(n,k,t,d)$, that is, run the algorithm that zero-test for the input 0 and take all the membership queries in the algorithm $\hat{\gamma}_1, \ldots, \hat{\gamma}_s$. We now have $f \in MUL_{\mathcal{F}}(n,k,t,d)$ is not 0 if and only if $f(\hat{\gamma}_i) = 0$ for all $i = 1, \ldots, s$. Consider the $s \times r$ matrix with rows $[\hat{\gamma}_j^{\mathbf{i}_1}, \ldots, \hat{\gamma}_j^{\mathbf{i}_r}]$. If this matrix have rank r then we choose r linearly independent rows. If the rank is less than r then its columns are dependent and therefore there are constants c_i, $i = 1, \ldots, r$ such that

$$\sum_{i=1}^{r} c_i \hat{\gamma}_j^{\mathbf{i}_i} = 0 \quad \text{for} \quad j = 1, \ldots, s.$$

This shows that the multivariate polynomial $\sum_{i=1}^{r} c_i x^{\mathbf{i}_i}$ is 0 for all $\hat{\gamma}_1, \ldots, \hat{\gamma}_s$. Since $\sum_{i=1}^{r} c_i x^{\mathbf{i}_i} \in MUL_{\mathcal{F}}(n,k,t,d)$ we get a contradiction.

Now we show that $1+2+3 \Rightarrow 4$. This will use results from the previous section. The algorithm first checks whether f depends on x_1, and if yes it generates a tree with a root labeled with x_1 that has d children. The ith child is the tree for $\Delta^i(f)$. If the function is independent of x_1 it builds a tree with one child for the root. The child is $\Delta^0(f)$. We then recursively build the tree for the children. The previous section shows how to simulate membership queries at each level in polynomial time. This algorithm obviously works. It correctness follows immediately from the previous section and (1)-(3).

The complexity of the algorithm is the size of the tree times the membership query simulation. Since the size of the tree at each level is bounded by the number terms in f

and since the depth of the tree is bounded by n the algorithm have complexity the same as zero testing with a blow up of $poly(n,t)$ queries and time. In the full paper we will give a more precise analysis of the complexity.

Now that we have reduced the problem to zero testing we will investigate in the next section the complexity of zero testing of $MUL_{\mathcal{F}}(n,k,t,d)$.

5 Zero-test of $MUL_{\mathcal{F}}(n,k,t,d)$

In this section we will study the zero testing of $MUL_{\mathcal{F}}(n,k,\star,d)$ when the number of terms is unknown and might be exponentially large. The time complexity for the zero testing should be polynomial in n and d (we have $k<n$ so it is also polynomial in k). We will show the following

Theorem 1. We have $MUL_{\mathcal{F}}(n,k,\star,d)$, where $d\le c|\mathcal{F}|$, is zero testable in randomized polynomial time in n and d for some constant c if and only if

$$k=O\left(\frac{|\mathcal{F}|}{d}(\log n+\log d)\right).$$

The algorithm for the zero testing is simply to randomly and uniformly choose $poly(n,d)$ points a_i from $\mathcal{F}^n$ and find $f(a_i)$. If for all the points a_i, f is zero then with high probability $f\equiv 0$.

The condition $d<c|\mathcal{F}|$ for some constant d is a nessasary condition for efficient learning. If d is close to 1 (say $d=|\mathcal{F}|-1$) then $k=O(\log n/\log|\mathcal{F}|)<1$ for $|\mathcal{F}|=\omega(n)$. This bound is also tight.

This theorem implies

Theorem 2. The class $MUL_{\mathcal{F}}(n,k,t,d)$ where $d<c|\mathcal{F}|$ for some constant c is learnable in randomized polynomial time (in n, d and t) from membership queries only if and only if

$$k=O\left(\frac{|\mathcal{F}|}{d}(\log n+\log d)\right).$$

The proofs are given in the appendix.

6 PAC-learning multivariate polynomials with membership queries

In this section we give an algorithm that PAC-learns with membership queries any multivariate polynomial with nonmonotone terms under distributions that support small terms. We will first consider the boolean multivariate polynomials and later in this section show how to generalize it to any field. For the analysis of the correctness of the algorithm we first need to formalize the notion of distributions that support small terms. The following is one way to define this notion.

Definition 1. Let $\mathcal{D}_{c,t,\epsilon}$ be the set of distributions that satisfy the following: For every $D \in \mathcal{D}_{c,t,\epsilon}$ and any DNF h with t terms of size greater than $c\log(t/\epsilon)$ we have

$$\Pr_D[f=1] \le \epsilon.$$

Notice that all the constant bounded product distributions D where $1-c' \le \Pr_D[x_i = 1] \le c'$ for all i are in $\mathcal{D}_{1/\log(1/c'),t,\epsilon}$.

A very rough analysis is given here to show that polynomial time algorithm exists. In the full paper a more careful analysis will be done to get the best possible constants.

Let $f = T_1 + \cdots + T_t$ be a multivariate polynomial. Suppose $|T_1| \le |T_2| \le \cdots \le |T_t|$. Our algorithm starts by choosing a random assignment a and defines $f'(x) = f(x+a)$. All terms that are of size s (in f') will contain on average $s/2$ positive literals. Therefore by Chernoff bound with high probability all the terms of size more than $32c\log(t/\epsilon)$ will contain at least $8c\log(t/\epsilon)$ positive literals. Also all terms of size $4c\log(t/\epsilon)$ with high probability will contain at least $c\log(t/\epsilon)$ positive literals. Now we split the function f' into 3 functions f_1, f_2 and f_3. The function $f_1 = T_1 + \cdots + T_{t_1}$ will contain all terms that are of size at most $4c\log(t/\epsilon)$. The function $f_2 = T_{t_1+1} + \cdots + T_{t_2}$ will contain all terms of size between $4c\log(t/\epsilon)$ and $32c\log(t/\epsilon)$ and the function $f_3 = T_{t_2+1} + \cdots + T_t$ will contain all terms of size more than $32c\log(t/\epsilon)$.

Now change f_1 to a multivariate polynomial with monotone terms. Since the size of each term in f_1 is at most $2c\log(t/\epsilon)$ the number of monotone terms in f_1 will be at most $t(t/\epsilon)^{2c}$. Now do the same for f_2 to get a multivariate polynomial with at most $t(t/\epsilon)^{20c}$ terms.

Our algorithm will find all the terms in f_1, some of the terms in f_2 and none of the terms in f_3. Therefore we will need the following claim.

Claim 1. Let $g = f_1 + h$ where h is a multivariate polynomial that contains some of the terms in f_2. Then for any $D \in \mathcal{D}_{c,t,\epsilon}$ we have

$$\Pr_D[g \ne f] \le \epsilon.$$

Proof. The error is

$$\Pr_D[(f_1+h)+f=1] = \Pr_D[h+f_2+f_3=1].$$

Let

$$f_2 = \hat{T}_{t_1+1}\tilde{T}_{t_1+1} + \cdots + \hat{T}_{t_2}\tilde{T}_{t_2}$$

where $T_i = \hat{T}_{t_i}\tilde{T}_{t_i}$, $\hat{T}_{t_i}$ is the part of the term that contains positive literals and $\tilde{T}_{t_i}$ is the part that contains the negative literals. When we change the terms of f_2 to monotone terms every monotone term in f_2 will contain one of the terms $\hat{T}_i$, $t_1+1 \le t \le t_2$. Therefore we can write $f_2 = \hat{T}_{t_1+1}f_{2,1} + \cdots + \hat{T}_{t_2}f_{2,t_2-t_1}$ where $f_{2,i}$ are multivariate polynomial with monotone terms. Since h is a multivariate polynomial that contains some of the terms in f_2 we have $f_2+h = \hat{T}_{t_1+1}h_{2,1}+\cdots+\hat{T}_{t_2}h_{2,t_2-t_1}$. Since $|\hat{T}_i| \ge c\log(t/\epsilon)$ for $t_1+1 \le i \le t_2$ and $|T_i| \ge c\log(t/\epsilon)$ for $i \ge t_2+1$, by the definition of distribution that support small terms we have

$$\begin{aligned} \Pr[(h+f_2)+f_3=1] &\le \Pr[\hat{T}_{t_1+1} \vee \cdots \vee \hat{T}_{t_2} \vee T_{t_2+1} \vee \cdots \vee T_t] \\ &\le \epsilon.\Box \end{aligned}$$

The algorithm will proceed as follows. We randomly and uniformly choose a zero restriction p of f'. That is, we substitute $x_i \leftarrow 0$ with probability $1/2$ for each x_i. This will on average leave $n/2$ variables alive in f'. Since terms in f_3 have at least $8c\log(t/\epsilon)$ positive literals with high probability we will have $f_3(p)=0$. In f_1+f_2 some of the terms will vanish and some will stay alive. Since the projection $f'(p)$ is in $MUL_{\mathcal{F}}(n, 32c\log(t/\epsilon), t^{32c+1}, 1)$ we can use the algorithm from the previous section to learn the terms from membership queries only in time $(t/\epsilon)^{cO(1)}$. After we learn $g_1 = f'(p)$ we define the function $f'' = f' + g_1$. This function have fewer terms. Now we do the same for f''. That is, we take another zero restriction and collect terms g_2 that stay alive after the projection. We do that until $f' + g_1 + g_2 + \cdots + g_r$ is ϵ-close to 0.

Notice that if M_i is a term in f_1 then a zero restriction will keep M_i alive with probability $1/(t/\epsilon)^{4c}$ so on average we need $(t/\epsilon)^{4c}$ projections to catch this term and $t(t/\epsilon)^{6c}$ projections to catch all the $t(t/\epsilon)^{2c}$ terms in f_1. Notice also that since each term in f_3 has $8c\log t$ positive literals each projection will make $f_3 = 0$ with probability at least $1 - t/(t/\epsilon)^{8c}$ and therefore the probability that $f_3 = 0$ in all the projections is at least $1 - 1/(t/\epsilon)^{2c}$. Therefore we can make the probability of success to be greater than $1/4$. This completes the description and the correctness of the algorithm.

The above analysis algorithm can also be used to learn functions of the form $f = \lambda_1 T_1 + \cdots + \lambda_t T_t$ where $\lambda_i \in \mathcal{F}$, T_i are boolean terms and $+$ is the addition of a field $\mathcal{F}$. This gives the learnability of decision trees with leaves that contain elements from the field $\mathcal{F}$.

References

[A88] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[BT88] M. Ben-Or, P. Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing.* pages 301–309, May 1988.

[Bl92] A. Blum. Learning boolean functions in an infinite attribute space. *Machine Learning* 9(4), pages 373–386. 1992.

[Bs93] N. H. Bshouty. Exact learning via the monotone theory. In *Proceedings of the 34th Symposium on Foundations of Computer Science.* pages 302–311, November 1993.

[B195] N. H. Bshouty. Simple Learning Algorithms Using Divide and Conquer. In *Proceedings of the Annual ACM Workshop on Computational Learning Theory.* 1995.

[B295] N. H. Bshouty. A Note on Learning Multivariate Polynomials under the Uniform Distribution. In *Proceedings of the Annual ACM Workshop on Computational Learning Theory.* 1995.

[CDG+91] M. Clausen, A. Dress, J. Grabmeier, M. Karpinski. On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields. *Theoretical Computer Science.* 84. pages 151–164, 1991.

[GKS90] D. Yu. Grigoriev, M. Karpinski, M. F. Singers. Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. *SIAM J. of Comp.* 19(6). pages 1059–1063, 1990.

[J94] J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. In *Proceeding of the 35th Annual Symposium on Foundations of Computer Science*, 1994.

[J95] J. Jackson. Private communication.

[KM93] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM J. Computing*, 22(6):1331–1348, 1993.

[Ma92] Y. Mansour. Randomized interpolation and approximation of sparse polynomials. In *Automata, Languages and Programming: 19th International Colloquim.* pages 261–272, July 1992.

[RB89] M. Ron Roth and G. Benedek. Interpolation and approximation of sparse multivariate polynomials over gf(2). *SIAM J. Computing*, 20(2):291–314, 1991.

[SS93] R. E. Schapire, L. M. Sellie. Learning sparse multivariate polynomial over a field with queries and counterexamples. In *Proceedings of the Sixth Annual ACM Workshop on Computational Learning Theory.* July, 1993.

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

[Z90] R. Zippel. Interpolating polynomials from their values. *Journal of Symbolic Computation*, 9. pages. 375–403. 1990.

7 Appendix.

Proof of Theorem 1 Upper Bound. Let $\gamma_1,\ldots,\gamma_d \in \mathcal{F}$ and define the function

$$f_{k,d} = \prod_{j=1}^{k}\prod_{i=1}^{d}(x_j - \gamma_i).$$

Denote by $Z(k,d)$ the number of zeros of $f_{k,d}$. It is easy to see that Z satisfies the recursive equation

$$Z(k,d) = d|\mathcal{F}|^{k-1} + (|\mathcal{F}| - d)Z(k-1,d)$$

and $Z(1,d) = d$. It is also easy to see that

$$Z(k,d) = |\mathcal{F}|^k - (|\mathcal{F}| - d)^k.$$

Now let $\phi(n,k,d)$ the maximal possible number of roots of a multivariate polynomial in $MUL_{\mathcal{F}}(n,k,\star,d)$. We will show the following facts

1. $\phi(n,k,d) \le |\mathcal{F}|^{n-k}\phi(k,k,d)$.

2. $\phi(k,k,d) \le Z(k,d)$.

Both facts implies that if $f \not\equiv 0$ we randomly uniformly choose an assignment $a \in \mathcal{F}^n$ we get

$$\begin{aligned}
\Pr_a[f(a) \ne 0] &\ge 1 - \frac{\phi(n,k,d)}{|\mathcal{F}|^n} \\
&\ge 1 - \frac{\phi(k,k,d)}{|\mathcal{F}|^k} \\
&\ge 1 - \frac{Z(k,d)}{|\mathcal{F}|^k} \\
&\ge \left(1 - \frac{d}{|\mathcal{F}|}\right)^k \\
&\ge e^{-O\left(\frac{dk}{|\mathcal{F}|}\right)} \qquad d \le c|\mathcal{F}| \\
&\ge \frac{1}{poly(n,d)}.
\end{aligned}$$

Therefore the expected running time to detect that f is not 0 is $poly(n,d)$.

It remain to prove conditions (1) and (2). To prove (1) let $f \in MUL_{\mathcal{F}}(n,k,\star,d)$ with maximal number of roots. Let m be a term in f with a maximal number of variables. Suppose, without loss of generality, $m = x_1^{i_1} \cdots x_k^{i_k}$. For any substitution $a_{k+1}, \ldots, a_n$ of the variables $x_{k+1}, \ldots, x_n$ the term m will stay alive in the projection $g = f|_{x_i \leftarrow a_i, i=k+1,\ldots,n}$ because it is maximal in f. Since g has at most $\phi(k,k,d)$ roots the result (1) follows.

To prove (2) let $f \in MUL_{\mathcal{F}}(k,k,\star,d)$. Write f as polynomial in $\mathcal{F}[x_2,\ldots,x_d][x_1]$,

$$f = f_d x_1^d + f_{d-1} x_1^{d-1} + \cdots + f_0.$$

Let t be the number of roots of f_d. Since $f_d \in MUL_{\mathcal{F}}(k-1,k-1,\star,d)$ we have

$$t \le \phi(k-1,k-1,d).$$

For $|\mathcal{F}|^{k-1} - t$ assignments a for $x_2,\ldots,x_d$ we have $f_d(a) \ne 0$. For those assignments we get a polynomial in x_1 of degree d that has at most d roots for x_1. For t assignments a for $x_2,\ldots,x_k$ we have f_d is zero and then the possible values of x_1 (to get a root for f) is bounded by $|\mathcal{F}|$. This implies

$$\begin{aligned}\phi(k,k,d) &\le d(|\mathcal{F}|^{k-1} - t) + t|\mathcal{F}| \\ &= d|\mathcal{F}|^{k-1} + (|\mathcal{F}| - d)t \\ &\le d|\mathcal{F}|^{k-1} + (|\mathcal{F}| - d)\phi(k-1,k-1,d).\end{aligned}$$

Now the result follows by induction on k.$\square$

Proof of Theorem 2 Lower Bound In the lower bound we want to show that for any $poly(d,n)$ assignments there is a nonzero polynomial f in $MUL_{\mathcal{F}}(n,k,\star,d)$ for $k = \frac{|\mathcal{F}|}{d}(\log n + \log d)$ that is zero on all the assignments. This will show that any algorithm than run in $poly(d,n)$ will not be able to distinguish between f and 0.

Let $q = poly(d,n)$ and $A = \{a_1,\ldots,a_q\}$ be a set of assignments. Let $a_{i,j}$ be the jth entry of a_i. Cosider $a_{i,1}$, $i = 1,\ldots,q$. By the Pingen hole principle there are at least $\lceil |\mathcal{F}|/q \rceil$ that have the same value $\gamma_{1,1} \in \mathcal{F}$. Take now the other $q\left(1 - \frac{1}{|\mathcal{F}|}\right)$ assignments that satisfy $a_{i,1} \ne \gamma_{1,1}$ and apply the Pingen Hole d times for the first entry and d times for the second and so forth until we cover all the assignments in A. We get $\gamma_{i,j}$, $i = 1,2,\ldots,p$, $j = 1,\ldots,d$ such that for every assignment a_l there is i_l, j_l where $a_{l,i_l} = \gamma_{i_l,j_l}$. Since each $\gamma_{i,j}$ reduce the number of assignments by a factor of $q\left(1 - \frac{1}{|\mathcal{F}|}\right)$ we have

$$q\left(1 - \frac{1}{|\mathcal{F}|}\right)^{pd} < 1.$$

This implies that

$$p \ge \frac{|\mathcal{F}|}{d} \log q = \frac{|\mathcal{F}|}{d}(\log n + \log d).$$

Now define the polynomial

$$\prod_{i=1}^{p}\prod_{j=1}^{d}(x_i - \gamma_{i,j}).$$

This polynomial is of degree d for each variable and have terms of size k that is zero on all the assignments of A.$\square$