
A Note on Learning Multivariate Polynomials under the Uniform Distribution

– Extended Abstract –

Nader H. Bshouty

Abstract

We present a PAC-learning algorithm with membership queries for learning any multivariate polynomial over any finite field \mathcal{F} under the uniform distribution. The algorithm runs in time ¹

$$t^{O(|\mathcal{F}| \log |\mathcal{F}|)} \log n$$

queries where t is the number of terms in the polynomial, n is the number of variables and $|\mathcal{F}|$ is the field size. This complexity is polynomial for any fix finite field \mathcal{F} . The output hypothesis is a multivariate polynomial with less than or equal to t terms.

We also show that $O(\log n)$ -multivariate polynomials (each term contains at most $O(\log n)$ distinct variables) are exactly learnable from membership and equivalence queries in time $n^{O(\log |\mathcal{F}|)}$.

1 Introduction

In this paper we study the learnability of multivariate polynomials with the form

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

over any finite field \mathcal{F} where $a_{\alpha} \in \mathcal{F}$ and α_i are integers. When the field is infinite, multivariate polynomials can be interpolated from membership queries only (see [BT88, CDG+91, GKS90, Ma92, RB89, Z90]). When the field is finite, membership queries alone cannot identify even one-term multivariate polynomials. Schapire and Sellie show in [SS93] that *multilinear* polynomials (when $\alpha \in \{0, 1\}^n$) are learnable from membership and equivalence queries. This

¹This is the number of examples. The time is polynomial in the number of examples and n .

Permission to make digital/hard copies of all or part of this material without fee is granted provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication and its date appear, and notice is given that copyright is by permission of the Association for Computing Machinery, Inc. (ACM). To copy otherwise, to republish, to post on servers or to redistribute to lists, requires specific permission and/or fee.
COLT'95 Santa Cruz, CA USA © 1995 ACM 0-89723-5/95/0007..\$3.50

implies learning any multivariate polynomial over the binary field $GF(2) = \{0, 1\}$ from membership and equivalence queries and implies PAC-learning multivariate polynomials over the binary field with membership queries under any distribution. Learning multivariate polynomials over finite fields in any learning model is still open.

In this paper we show that any multivariate polynomial over a *fixed* field \mathcal{F} is PAC-learnable with membership queries in polynomial time under the uniform distribution. We also show that t -multivariate polynomials, i.e., multivariate polynomials where each term contains at most t distinct variables, are learnable from membership and equivalence queries in time $|\mathcal{F}|^t$. This complexity is polynomial for any finite field and $t = O(\log n / \log |\mathcal{F}|)$, and, polynomial for any fixed field \mathcal{F} and $t = O(\log n)$.

Our approach is different from [SS93]. In [SS93] the algorithm is based on the structure of lattices that satisfy certain properties and learning multilinear polynomial over any field is obtained by taking the lattice $\{0, 1\}^n$. Our approach is algebraic and is based on generalizing interpolation from one variable polynomials to multivariable polynomials. This approach is similar to the one in [RB89] for learning multivariate polynomials over the binary field under the uniform distribution. In our algorithm the learning algorithm at some stage has a set of terms T where each term in T is a subterm of some term in the target. The learning algorithm then assumes that the terms it has are the actual terms of the target and interpolates to find the coefficients of the terms. It then asks equivalence queries with that polynomial. A counterexample will be used to find a larger T , i.e., either add a new variable to an existing term in T or add one variable that is a variable in a new term in the target.

Our paper is organized as follows. In section 2 we give some preliminary results and in section 3 we give the algorithms.

2 Preliminaries

Let

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

be a multivariate polynomial over the ring \mathcal{R} where $a_{\alpha} \in \mathcal{R}$ and $\alpha_1, \dots, \alpha_n$ are integers. We will denote the class of all multivariate polynomials over the ring \mathcal{R} and over the vari-

ables x_1, \dots, x_n by $\mathcal{R}[x_1, \dots, x_n]$. The class $\mathcal{R}[x_1, \dots, x_n]$ is itself a ring so we can also consider multivariate polynomials in $\mathcal{R}[x_1, \dots, x_n][y_1, \dots, y_m]$. These are multivariate polynomials over the variables y_1, \dots, y_m where the coefficients are multivariate polynomials over the variables x_1, \dots, x_n . This ring is isomorphic to the ring

$$\mathcal{R}[x_1, \dots, x_n, y_1, \dots, y_m]$$

of multivariate polynomials over the variables $x_1, \dots, x_n, y_1, \dots, y_m$.

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 $f = c \in \mathcal{F} \setminus \{0\}$ then $|f| = 1$. Since we will consider only finite fields $\mathcal{R} = \mathcal{F}$ in this paper we will write $\kappa = |\mathcal{F}|$ for the size of the field and $\alpha \in [\kappa]^n$ (here $[n] = \{0, \dots, n-1\}$). The latter is because $x^\kappa = x$ over the field \mathcal{F} . Suppose $\mathcal{F} = \{\gamma_0, \dots, \gamma_{\kappa-1}\}$ 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} can be interpolated from membership queries as follows. Suppose

$$f(x_1) = \Delta^{(\kappa-1)}(f)x_1^{\kappa-1} + \dots + \Delta^{(1)}(f)x_1 + \Delta^{(0)}(f)$$

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

$$\begin{cases} f(\gamma_0) & = \\ & \Delta^{(\kappa-1)}(f)\gamma_0^{\kappa-1} + \dots + \Delta^{(1)}(f)\gamma_0 + \Delta^{(0)}(f) \\ f(\gamma_1) & = \\ & \Delta^{(\kappa-1)}(f)\gamma_1^{\kappa-1} + \dots + \Delta^{(1)}(f)\gamma_1 + \Delta^{(0)}(f) \\ \vdots & \ddots \\ f(\gamma_{\kappa-1}) & = \\ & \Delta^{(\kappa-1)}(f)\gamma_{\kappa-1}^{\kappa-1} + \dots + \Delta^{(1)}(f)\gamma_{\kappa-1} + \Delta^{(0)}(f). \end{cases}$$

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

$$\Delta^{(i)}(f) = \frac{\det |M_i|}{\det |V(\gamma_0, \dots, \gamma_{\kappa-1})|} \quad (1)$$

where $V(\gamma_0, \dots, \gamma_{\kappa-1})$ is the Vandermonde matrix and M_i is

$$\begin{bmatrix} \gamma_0^{\kappa-1} & \dots & \gamma_0^{i+1} & f(\gamma_0) & \gamma_0^{i-1} & \dots & \gamma_0 & 1 \\ \gamma_1^{\kappa-1} & \dots & \gamma_1^{i+1} & f(\gamma_1) & \gamma_1^{i-1} & \dots & \gamma_1 & 1 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\ \gamma_{\kappa-1}^{\kappa-1} & \dots & \gamma_{\kappa-1}^{i+1} & f(\gamma_{\kappa-1}) & \gamma_{\kappa-1}^{i-1} & \dots & \gamma_{\kappa-1} & 1 \end{bmatrix}$$

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

$$f(x_1, \dots, x_n) = \Delta_1^{(\kappa-1)}(f)x_1^{\kappa-1} + \dots + \Delta_1^{(1)}(f)x_1 + \Delta_1^{(0)}(f)$$

where $\Delta_1^{(i)}(f)$ is a multivariate polynomial over the variables x_2, \dots, x_n . We can still use (1) to find $\Delta_1^{(i)}(f)$. Just replace each $f(\gamma_i)$ with $f(\gamma_i, x_2, \dots, x_n)$. This can be done for any variable x_j . Therefore $\Delta_j^{(i)}$ is a function operator that returns the coefficient of x_j^i when f is represented in $\mathcal{F}[x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_n][x_j]$. Notice that from the first equation in the system, since $\gamma_0 = 0$, we have

$$\Delta_j^{(0)}(f) = f(x_1, \dots, x_{j-1}, 0, x_{j+1}, \dots, x_n). \quad (2)$$

>From (1) a membership query for $\Delta_j^{(i)}$ can be simulated using κ membership queries to f . From (2), a membership query to $\Delta_j^{(0)}$ can be simulated using one membership query to f .

We now extend the Δ operators as follows: for $\mathbf{i} = (i_1, \dots, i_k)$ and $\mathbf{j} = (j_1, \dots, j_k)$ for distinct j_1, \dots, j_k

$$\Delta_{\mathbf{j}}^{\mathbf{i}} = \Delta_{j_k}^{i_k} \Delta_{j_{k-1}}^{i_{k-1}} \dots \Delta_{j_1}^{i_1}.$$

We will also write $x_{\mathbf{j}}^{\mathbf{i}}$ for the term $x_{j_k}^{i_k} x_{j_{k-1}}^{i_{k-1}} \dots x_{j_1}^{i_1}$. The weight of \mathbf{i} , denoted by $wt(\mathbf{i})$, is the number of nonzero entries in \mathbf{i} . Let \mathcal{I}_k be the set of all (\mathbf{i}, \mathbf{j}) for all possible \mathbf{i} and \mathbf{j} of length k .

Since the operator $\Delta_j^{(i)}(f)$ gives the coefficient of x_j^i in f when represented in $\mathcal{F}[x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_n][x_j]$, the operator $\Delta_{\mathbf{j}}^{\mathbf{i}}(f)$ gives the coefficient of $x_{\mathbf{j}}^{\mathbf{i}}$ when f is represented in

$$\mathcal{F}[\{x_i | i \neq i_l\}][\{x_i | i = i_l\}].$$

Also by (1) and (2), membership queries for $\Delta_{\mathbf{j}}^{\mathbf{i}}$ can be simulated by $\kappa^{wt(\mathbf{i})}$ membership queries for f . This gives the following properties of the Δ operator.

Lemma 1. *Let f be a multivariate polynomial. Then*

- 1) $\Delta_{\mathbf{j}}^{\mathbf{i}}(f)$ is the coefficient of $x_{\mathbf{j}}^{\mathbf{i}}$ in f .
- 2) A membership query to $\Delta_{\mathbf{j}}^{\mathbf{i}}(f)$ can be simulated by $\kappa^{wt(\mathbf{i})}$ membership queries to f .
- 3)

$$|f| = \sum_{(\mathbf{i}, \mathbf{j}) \in \mathcal{I}_k} |\Delta_{\mathbf{j}}^{\mathbf{i}}(f)|.$$

We will be interested in representing f using different levels of Δ , i.e.,

$$f = \sum_{(\mathbf{i}, \mathbf{j}) \in \mathcal{I}} \Delta_{\mathbf{j}}^{\mathbf{i}}(f) x_{\mathbf{j}}^{\mathbf{i}}$$

where \mathcal{I} is a set of different (\mathbf{i}, \mathbf{j}) with different lengths. To have the above well defined we must make sure that each coefficient in f occurs exactly once in the sum. Therefore we add the condition that for every term t in f there is exactly one $(\mathbf{i}, \mathbf{j}) \in \mathcal{I}$ such that $x_{\mathbf{j}}^{\mathbf{i}}$ is a subterm of t . Here $x_{\mathbf{j}}^{\mathbf{i}}$ is a subterm of t if the exponent of x_{j_l} in t is exactly i_l for every l . There might be other (\mathbf{i}, \mathbf{j}) in \mathcal{I} such that no term in f is a superterm of $x_{\mathbf{j}}^{\mathbf{i}}$. In this case $\Delta_{\mathbf{j}}^{\mathbf{i}}(f) = 0$. If \mathcal{I} satisfies the above then we say that \mathcal{I} is good for f . Property 3 in lemma 1 also holds for this representation when \mathcal{I} is good for f .

We now are ready to give the algorithm.

3 The Algorithm

3.1 k -multivariate polynomial

We will first present the algorithm that learns k -multivariate polynomials from membership and equivalence queries. In the algorithm we will have a set

- (1) \mathcal{I} that is good for f and for every $(i, j) \in \mathcal{I}$ we have $wt(i) \leq$

Therefore

$$f = \sum_{(i,j) \in \mathcal{I}} \Delta_j^i(f) x_j^i.$$

Let $\mathcal{I}' \subset \mathcal{I}$ be the elements that satisfies $\Delta_j^i(f) \neq 0$. Since \mathcal{I} is good we have

$$|\mathcal{I}'| \leq |f|.$$

Let $\|\mathcal{I}'\|$ be the sum of the lengths of all vectors in \mathcal{I}' . We will prove that

$$(2) \quad |\mathcal{I}| \leq \kappa \|\mathcal{I}'\|.$$

Since $|\mathcal{I}'| \leq |f|$ and each term contains at most k variables, we have

$$\|\mathcal{I}'\| \leq k|f|.$$

This, in particular, implies that

$$|\mathcal{I}| \leq k\kappa|f|.$$

We will also show that

$$(3) \quad \text{in each iteration of the algorithm, } \|\mathcal{I}'\|$$

will be increased by at least 1.

Since in each iteration the algorithm asks one equivalence query, the number of equivalence queries asked by the algorithm is at most $\|\mathcal{I}'\| \leq k|f|$.

Now we will show what the learning algorithm will do at each iteration and show how to use equivalence and membership queries to satisfy the above three conditions.

The algorithm asks membership queries to find $h_{i,j} = \Delta_j^i(f)(\bar{0})$ for all $(i, j) \in \mathcal{I}$. By lemma 1, a membership query to $\Delta_j^i(f)(\bar{0})$ can be simulated by $\kappa^{wt(i)}$ membership queries and since by 1, $wt(i) \leq k$, a membership query for $\Delta_j^i(f)$ can be simulated by at most κ^k membership queries for f .

The algorithm then asks an equivalence query with the hypothesis

$$h = \sum_{(i,j) \in \mathcal{I}} h_{i,j} x_j^i.$$

Notice that $h_{i,j} = \Delta_j^i(f)(\bar{0}) = 0$ for $(i, j) \notin \mathcal{I}'$. Since $|\mathcal{I}'| \leq |f|$, our hypothesis will never have a number of terms greater than the target. Therefore our learning algorithm is a proper learning algorithm.

Suppose $x^{(0)}$ is the counterexample. Since the target f and the hypothesis h agree on x_j^i and on all terms that satisfy

$\Delta_j^i(f) = 0$, the counterexample $x^{(0)}$ will be a counterexample for one of the Δ_j^i for some $(i, j) \in \mathcal{I}'$. We use membership queries for $\Delta_j^i(f)$ for all $(i, j) \in \mathcal{I}$ to find the one that satisfies $\Delta_{j_0}^{i_0}(f)(x^{(0)}) \neq \Delta_{j_0}^{i_0}(f)(\bar{0})$. We then know that $(i_0, j_0) \in \mathcal{I}'$. We now use membership queries to find a variable that $\Delta_{j_0}^{i_0}(f)$ depends on. We do that by flipping nonzero entries in $x^{(0)}$ to 0 until flipping one variable, say x_q , changes the value of $\Delta_{j_0}^{i_0}(f)$. Finding some x_q that $\Delta_{j_0}^{i_0}(f)$ depends on can be done with $\log n$ membership queries (flip one-half of the distinct entries then one-quarter, etc.). This shows that $\Delta_{j_0}^{i_0}(f)$ depends on x_q . Therefore x_q appears in $\Delta_{j_0}^{i_0}(f)$. We then remove the element (i_0, j_0) from \mathcal{I} and add

$$((i_0, 0), (j_0, q)), ((i_0, 1), (j_0, q)), \dots, ((i_0, \kappa - 1), (j_0, q)).$$

This is because we know that $x_j^i x_q^w$ is a subterm for some of the w 's but we do not know which w 's. Therefore we add them all. If $x_j^i x_q^w$ is not a subterm of one of the terms in f its coefficient will be 0 and it will never occur in the hypothesis. Let $\hat{\mathcal{I}}$ be the new set \mathcal{I} .

Now we show that $\hat{\mathcal{I}}$ satisfies the above conditions.

For condition 1 we have the following. First since \mathcal{I} is good for f , by the definition of good, the new set $\hat{\mathcal{I}}$ is also good for f because we are adding all possible exponents for x_q . Since $\Delta_{j_0}^{i_0}(f)$ depends on some variable and since the target f is a k -multivariate polynomial we must have had $wt(i_0) < k$. Therefore $wt((i_0, l)) \leq wt(i_0) + 1 \leq k$. This shows that $\hat{\mathcal{I}}$ satisfies condition 1.

Notice that we are replacing a term in \mathcal{I}' with κ new terms such that at least one of them is a subterm of one of the terms in f . This implies that $\|\mathcal{I}'\|$ is increased by at least one and therefore condition 3 is still true for $\hat{\mathcal{I}}$. Also condition 2 is true because

$$\begin{aligned} \|\hat{\mathcal{I}}\| &= |\mathcal{I}| + \kappa - 1 \leq \kappa \|\mathcal{I}'\| + \kappa \\ &\leq \kappa(\|\mathcal{I}'\| + 1) \leq \kappa \|\hat{\mathcal{I}}'\|. \end{aligned}$$

Now that conditions 1-3 are true we can analyse the complexity of the algorithm. The number of equivalence queries is less than $\|\mathcal{I}'\| \leq k|f|$. To simulate membership queries for the Δ 's we need at most κ^k membership queries. Since $|\mathcal{I}| \leq k\kappa|f|$, the number of membership queries needed to find which Δ the counterexample belongs to is $k^2|f|^2\kappa^{k+1}$. To find the relevant variables x_q we need $k|f|\kappa^k \log n$ membership queries.

Theorem 1. *The class of k -multivariate polynomials with t terms over the field \mathcal{F} is properly learnable from*

$$kt^2|\mathcal{F}|^{k+1} \log n$$

membership queries and

$$kt$$

equivalence queries.

3.2 PAC-learning

Now we show how to change the above algorithm to a PAC-learning algorithm with membership queries for any multivariate polynomial over the field \mathcal{F} under the uniform distribution.

Suppose

$$f = \sum_{\alpha \in \mathcal{I}} a_{\alpha} x_{\mathbf{j}}^{\alpha}$$

where $\mathbf{j} = (1, 2, \dots, n)$, is a multivariate polynomial with $t = |f| = |\mathcal{I}|$ terms. Let h be f where we remove all terms $x_{\mathbf{j}}^{\alpha}$ with

$$wt(\alpha) \geq w = \lceil \kappa(\ln t + \ln(2/\epsilon)) \rceil.$$

Now for a random uniform x we have

$$\begin{aligned} \Pr[h(x) \neq f(x)] &\leq \Pr \left[\bigvee_{wt(\alpha) > w} (x_{\mathbf{j}}^{\alpha} \neq 0) \right] \\ &\leq t \Pr[x_{\mathbf{j}}^{\alpha} \neq 0] \quad (wt(\alpha) > w) \\ &< t \left(1 - \frac{1}{\kappa}\right)^w \\ &\leq \frac{\epsilon}{2}. \end{aligned}$$

Therefore it is enough to find an $\epsilon/2$ approximation for h or just the terms in f of size less than or equal to w .

Notice that in the PAC model we can, for each $\Delta_{\mathbf{j}}^i(f)$, with probability at least $1 - \delta$, decide whether $\Delta_{\mathbf{j}}^i(f)$ is ξ -approximation of some constant γ in the field. If not then there must be two points, x_0 and x_1 , for which $\Delta_{\mathbf{j}}^i(f)(x_0) \neq \Delta_{\mathbf{j}}^i(f)(x_1)$. We use these two points to find a new variable x_q that $\Delta_{\mathbf{j}}^i(f)$ depends on. We use this to run the algorithm in the previous subsection. If a term is growing more than w we remove it from the hypothesis. If $\Delta_{\mathbf{j}}^i(f)$ is, with probability at least $1 - (\delta/t)$, an $\epsilon/(2t)$ -approximation of a constant γ then we replace it with γ . Since with probability $1 - (\delta/t)$ each coefficient is an $\epsilon/(2t)$ -approximation to one of the coefficients of h we have, with probability $1 - \delta$, a hypothesis that is an $\epsilon/2$ -approximation of h and therefore an ϵ -approximation of the target f .

The complexity of the algorithm is polynomial in $1/\epsilon$ and $\log(1/\delta)$ and

$$t^{O(|\mathcal{F}| \log |\mathcal{F}|)} \log n.$$

Theorem 2. *There is a PAC-learning algorithm with membership queries for learning multivariate polynomials over the field \mathcal{F} that runs in time*

$$t^{O(|\mathcal{F}| \log |\mathcal{F}|)} \log n.$$

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.

[B192] A. Blum. Learning boolean functions in an infinite attribute space. *Machine Learning* 9(4), pages 373–386. 1992.

[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.

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

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

[SS93] R. E. Shapire, 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.

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