Electronic Colloquium on Computational Complexity - Reports Series 1996 - available via:
ECCC
TR96-008
FTP: ftp.eccc.uni-trier.de:/pub/eccc/
WWW: http://www.eccc.uni-trier.de/eccc/
Email: ftpmail@ftp.eccc.uni-trier.de with subject 'help eccc'

Learning Multivariate Polynomials from Substitution and Equivalence Queries

F. Bergadano*, N.H. Bshouty+ and S. Varricchio°

*Università di Torino
+University of Calgary
°Università di L' Aquila

January 22, 1996

Abstract

It has been shown in previous recent work that multiplicity automata are predictable from multiplicity and equivalence queries. In this paper we generalize related notions in a matrix representation and obtain a basis for the solution of a number of open problems in learnability theory. Membership queries are generalized to "substitution" queries for learning non-boolean functions and provide the value of the target function for a given input. In particular, using substitution and equivalence queries, we prove the learnability of the boolean XOR of terms, XOR decision trees, decision trees with integer variables and less than conditions, multivariate polynomials over a finite field and rational functions over a fixed finite field. We also provide results for the case of infinite or large fields.

Keywords: Computational Learning Theory, Multivariate Polynomials, Learning over Large Fields, Multiplicity Automata, Decision Trees, Learning Boolean Functions.

1 Introduction

In recent years many open problems for the learnability of classes from membership and equivalence queries have been solved. These classes include CDNF [4] (polynomial size DNF that have polynomial size CNF), decision trees [4, 2], binary multivariate polynomials [6] and disjoint DNF [6]. The outstanding problem of the learnability of DNF is still an open one.

In this paper we use the technique of Bergadano and Varricchio for learning multiplicity automata as well as new modifications of their algorithm to solve some other open problems introduced in [4, 6].

In [6] Shapire and Sellie proved the learnability of multilinear multivariate polynomials from substitution and equivalence queries. This, in particular, proves the learnability of the XOR of monotone terms from membership and equivalence queries. Since then the learnability of the XOR of terms and multivariate polynomial over any field were open. In this paper we solve these two open problems and show,

- The class of the XOR of terms is learnable from membership and equivalence queries.
- The class of multivariate polynomials is learnable from substitution and equivalence queries.

We then investigate the learnability of decision trees over different bases (in the nodes) and address some open problems introduced in [4]. We prove

- The class of boolean decision trees with the domain $[n] = \{1, \ldots, n\}$ over the basis of $x_i \in P$ for any $P \subseteq [n]$ is learnable from membership and equivalence queries.

This is a generalization of the learnability of decision trees and solves the open problem introduced by Bshouty [4]. We then define a new measure called the AU-dimension of a boolean function and show that

- The class of depth $O(\log n)$ decision tree over the basis of fuctions of constant AU-dimension are learnable from membership and equivalence queries.

Constant AU-dimension functions includes the class of terms and XOR of variables. In particular we obtain the learnability of depth $O(\log n)$-decision

trees over XOR, as defined by Kushilevitz and Mansour in [5], and depth $O(\log n)$-decision trees over terms defined by Bshouty [4]. Notice that the latter is a generalization of the learnability of $O(\log n)$-term DNF.

The approach we use in this paper is algebraic. We generalize related notions in a matrix representation to show that certain matrix representable functions have polynomial size multiplicity automata. Then we show that the above classes have polynomial matrix representations and apply the Bergadano-Varricchio algorithm [2] for them. We then show how to change the Bergadano-Varricchio algorithm for learning multiplicity automata to an algorithm that runs over a large field with a query time that is independent of the field size. This improves the complexity of Bergadano-Verricchio algorithm for finite fields and generalizes it to work over infinite fields.

Using our new algorithm we prove,

- The class of functions

$$\sum_{\alpha \in \mathcal{I}} p_{\alpha_1}(x_1) \cdots p_{\alpha_n}(x_n),$$

  where p_{α_j} are polynomials of degree bounded by d, is learnable from substitution and equivalence queries in polynomial time in d, the number of variables n and $|\mathcal{I}|$.

Finally we show that for large enough fields it is possible to achieve learnability from membership queries only. We prove the following.

- If the field is infinite or finite of size greater than nd then the above classes can be learned with a randomized polynomial time algorithm from substitution queries only.

The paper is organized as follows. In section 2 we introduce the matrix representation theory for multiplicity automata and define the AU-dimension of a function. In section 3 we prove the results for multivariate polynomials and in section 4 we prove the results for decision trees over various bases.

2 Matrix Representation of $\mathcal{K}$-Automata

Let $\mathcal{K}$ be a field and let $\mathcal{A} = \{A_0, \ldots, A_{r-1}\} \subseteq \mathcal{K}^{m \times m}$ be r, $m \times m$ matrices over the field $\mathcal{K}$. Let $Z_r^* = \bigcup_{i=0}^{\infty} Z_r^i$ where Z_r is the ring of integer modulo

r. Let $AU(Z_r, \mathcal{K})$ be the set of all functions $f : Z_r^* \to \mathcal{K}$ where there exists an integer m, a row vector $\lambda \in \mathcal{K}^m$, A column vector $\gamma \in \mathcal{K}^m$ and a set of matrices $\mathcal{A} = \{A_0, \ldots, A_{r-1}\} \subseteq \mathcal{K}^{m \times m}$ such that

$$f(a_1, \ldots, a_s) = \lambda A_{a_1} \cdots A_{a_s} \gamma$$

for all $(a_1, \ldots, a_s) \in Z_r^*$. For a function f in $AU(Z_r, \mathcal{K})$ we write $\mathbf{AUdim}(f)$ for the minimal possible m. The triple $(\lambda, \mathcal{A}, \gamma)$ is called an *AU-basis* for f.

For a function $f : Z_r^* \to \mathcal{K}$ we define $f_n : Z_r^n \to \mathcal{K}$ to be $f_n(x) = f(x)$ if $x \in Z_r^n$ and $f_n(x) = 0$ otherwise.

The following lemma is proven in [3] using the K-Automata representation. We give another proof of the theorem for completeness.

Lemma 1. [3]: *Let $f, g \in AU(Z_r, \mathcal{K})$ and $c \in \mathcal{K}$. then*

1. $\boldsymbol{AUdim}(c) = 1$.
2. $\boldsymbol{AUdim}(cg) = \boldsymbol{AUdim}(g)$.
3. $\boldsymbol{AUdim}(f + g) \leq \boldsymbol{AUdim}(f) + \boldsymbol{AUdim}(g)$.
4. $\boldsymbol{AUdim}(fg) \leq \boldsymbol{AUdim}(f)\boldsymbol{AUdim}(g)$.
5. $\boldsymbol{AUdim}(f_n) \leq (n+1)\boldsymbol{AUdim}(f)$.

Proof. Let $(\lambda_A, \{A_i\}, \gamma_A)$ and $(\lambda_B, \{B_i\}, \gamma_B)$ be an AU-basis for f and g, respectively. Items (1) and (2) are trivial. For (3) take the basis

$$((\lambda_A, \lambda_B), diag(A_i, B_i), (\gamma_A', \gamma_B')').$$

For (4) take the basis

$$(\lambda_A \otimes \lambda_B, A_i \otimes B_i, \gamma_A \otimes \gamma_B)$$

where $\otimes$ is the Kroneker product. For (5) take $L \otimes A_i$ where L is an $(n+1) \times (n+1)$ matrix that contains zero in all entries except the entries $i, i+1$ for $i = 1, \ldots, m-1$.$\Box$

When $r \leq |\mathcal{K}|$ we can replace Z_r by $\mathcal{J} \subset \mathcal{K}$ where $|\mathcal{J}| = r$. In that case we write $\mathcal{J} = \{\alpha_1, \ldots, \alpha_r\}$ and $\mathcal{A} = \{A_{\alpha_1}, \ldots, A_{\alpha_r}\}$. Let $f^\star : \mathcal{J}^\star \to \mathcal{K}$ be a function such that

$$f^\star(\alpha_1, \ldots, \alpha_s) = \lambda A_{\alpha_1} \cdots A_{\alpha_s} \gamma.$$

In this case we will write the class $AU(Z_r, \mathcal{K})$ as $AU(\mathcal{J}, \mathcal{K})$.

By Lemma 1 and by multiplying $\mathbf{AUdim}(f^\star)$ by n, we can change the function to a function on n variables $f = f_n^\star$. Since A_x can be regarded as a function from $\mathcal{J}$ to $K^{m\times m}$ we can find a matrix $A(x)$ with entries that are of polynomial degree $r-1$ in x that satisfies $A(\alpha_i) = A_{\alpha_i}$. Such a matrix is called a *polynomial matrix* of degree $r-1$. Therefore

$$f(x_1, \ldots, x_n) = \lambda A(x_1) \cdots A(x_n)\gamma.$$

To make our functions independent of λ and γ we prove the following.

Lemma 2. *Let d be an integer. The set $AU(\mathcal{J}, \mathcal{K})$ with n variables and $\mathbf{AUdim} = poly(n, m)$ is the set of all functions of the form*

$$(A_1(x_1) \cdots A_n(x_n))_{1,1}$$

where $A_i(x_i)$ is a polynomial matrix of degree $|\mathcal{J}| - 1$ and for a matrix B, $(B)_{1,1}$ is the $1,1$ entry of B.

Proof: First notice that $f(x_1, \ldots, x_n) = \lambda A(x_1) \cdots A(x_n)\gamma$ can be written as

$$\begin{aligned} f(x_1, \ldots, x_n) &= e_1(BA(x_1))A(x_2) \cdots A(x_{n-1})(A(x_n)C)e_1' \\ &= ((BA(x_1))A(x_2) \cdots A(x_{n-1})(A(x_n)C))_{1,1} \end{aligned}$$

where $e_1 = (1, 0, 0, \cdots, 0)$ and B and C are matrices that satisfies $\lambda = Be_1$ and $\gamma = Ce_1'$.

Now given a function of the form $f(x_1, \ldots, x_n) = (A_1(x_1) \cdots A_n(x_n))_{1,1}$ we can write

$$f(x_1, \ldots, x_n) = \lambda A(x_1) \cdots A(x_n)\gamma$$

where

$$A(x) = \begin{pmatrix} & A_1(x) & & \\ 0_{nm\times m} & & \ddots & \\ & & & A_n(x) \\ 0_{m\times m} & & 0_{m\times nm} & \end{pmatrix}$$

$\gamma = (0_m, \ldots, 0_m, e_1)$ and $\lambda = \gamma'$. Here $0_{p\times q}$ is the $p \times q$ zero matrix and 0_t is the column 0 vector of length t. This multiplies the $AUdim$ by $n+1$ so it is still polynomial in n and m.□

We will write $\mathbf{AUdim}^\star(f)$ for the minimal m where there are $m \times m$ polynomial matrices $A_i(x)$ such that $f(x_1, \ldots, x_n) = (A_1(x_1) \cdots A_n(x_n))_{1,1}$. Notice that from the above lemma we have following.

Lemma 3.

$$\mathbf{AUdim}(f) \leq (n+1)\mathbf{AUdim}^\star(f).$$

Similar to Lemma 1 we can also prove the following.

Lemma 4. *Let $f, g \in AU(\mathcal{J}, \mathcal{K})$ and $c \in \mathcal{K}$. then*

1. $\boldsymbol{AUdim}^\star(c) = 1$.
2. $\boldsymbol{AUdim}^\star(cg) = \boldsymbol{AUdim}^\star(g)$.
3. $\boldsymbol{AUdim}^\star(f+g) \leq \boldsymbol{AUdim}^\star(f) + \boldsymbol{AUdim}^\star(g)$.
4. $\boldsymbol{AUdim}^\star(fg) \leq \boldsymbol{AUdim}^\star(f)\boldsymbol{AUdim}^\star(g)$.

In the following lemma we show that if functions f and g depend on distinct variables then we can get a better bound.

Lemma 5. *Let $f(x)$ and $g(y)$ be two functions on two distinct set of variables. Then*

1. $\boldsymbol{AUdim}^\star(fg) \leq \max(\boldsymbol{AUdim}^\star(f), \boldsymbol{AUdim}^\star(g))$.
2. $$\boldsymbol{AUdim}^\star(f+g)$$
$$\leq \begin{cases} \max(\boldsymbol{AUdim}^\star(f), \boldsymbol{AUdim}^\star(g)) & \boldsymbol{AUdim}^\star(f) \neq \boldsymbol{AUdim}^\star(g) \\ \boldsymbol{AUdim}^\star(f) + 1 & \boldsymbol{AUdim}^\star(f) = \boldsymbol{AUdim}^\star(g) \end{cases}$$

Proof.
If $f(x) = (A_1(x_1) \cdots A_n(x_n))_{1,1}$ then we may assume that $A_1(x_1) \cdots A_n(x_n)$ contains f in the $1,1$ entry and zero elsewhere. This is because we can multiply both sides of $A_1(x_1) \cdots A_n(x_n)$ by the matrix that contains 1 in the 1,1 entry and 0 in the other entries. We do the above for $g(y) = (B_1(y_1) \cdots B_l(y_l))_{1,1}$. Now changing the sizes of A_i and B_i to the maximum size (just add 0 to the new entries) and then putting them side by side will give the function $f(x)g(y)$.

Again, as above, we assume that after the multiplication only entry 1,1 is not zero. Suppose the size of A_i is $m_1 \times m_1$ and the size of B_i is $m_2 \times m_2$.

If $m_1 < m_2$ then we extend A_i to size $m_2 \times m_2$ by adding zeros and adding 1 in the entry m_2, m_2. Then we can add a permutation matrix P to make $g(y) = (PB_1(y_1) \cdots B_l(y_l)P)_{m_2,m_2}$. Now multiplying both matrices will give a matrix that contains f in entry 1,1 and g in entry m_2, m_2 and zero in the other entries. Now it is easy to see that we can multiply both sides by some matrix to obtain a matrix that contain $f + g$ in the 1,1 entry. If $m_1 = m_2$ then we add another entry $m_1 + 1$ to do the above trick.□

Using the above operations we can build nontrivial functions with small **AUdim**.

3 Results

In [2], Bergadano and Varricchio proved the following.

Theorem 1. For any field $\mathcal{K}$ the set $AU(\mathcal{J}, \mathcal{K})$ is learnable from substitution and equivalence queries in polynomial time in the number of variables n, $|\mathcal{J}|$ and the **AUdim** of the target.

Every $\mathcal{K}$-Automaton over an alphabet A corresponds to some function in $AU(\mathcal{J}, \mathcal{K})$ (see [2] for details).
A *$\mathcal{K}$-monomial* is $p_1(x_1) \cdots p_n(x_n)$ where each $p_i : \mathcal{K} \to \mathcal{K}$ is any function. A *$\mathcal{K}$-linear function* is $p_1(x_1) + \cdots + p_n(x_n)$.

Claim 1: Let T_i be $\mathcal{K}$-monomials and L_i be linear functions. Then

$$\mathbf{AUdim}^\star \left(\sum_{i=1}^{s} T \right) \leq s.$$

$$\mathbf{AUdim}^\star \left(\prod_{i=1}^{s} L \right) \leq 2^s.$$

Proof. Use Lemma 5 and Lemma 4.□

Result 1: *Let $\mathcal{K}$ be a finite field. The class of functions of the form*

$$\sum_{\alpha \in \mathcal{I}} \alpha p_{\alpha_1}(x_1) \cdots p_{\alpha_n}(x_n),$$

where p_i are any functions, is learnable from substitution and equivalence queries in time polynomial in the number of variables, $|\mathcal{I}|$ and the size of the field $|\mathcal{K}|$.

Proof. Follows immediately from Claim 1, Lemma 1 and Theorem 1. Also from Lemma 2 and Theorem 1.□

In particular we have.

Result 1.1: *The class of the XOR of terms in the boolean domain is learnable from membership and equivalence queries in polynomial time in the number of variables and the number of terms.*

This solves an open problem in [6].

Result 1.2: *The class of Sat j-DNF is learnable from membership and equivalence queries in polynomial time in n^j, the number of variables and the number of terms.*

Proof. You can change any Sat j-DNF with m terms to the XOR of m^j terms. Just write $T_1 \vee \cdots \vee T_n$ as $(T_1+1)\cdots(T_n+1)+1$ and multiply. Each multiplication of j terms is 0. □

This improves the result in [3].

Result 1.3: *The class of multivariate polynomials over a finite field is learnable from substitution and equivalence queries in polynomial time in the number of variables and the number of terms and the size of the field*

This also solves an open problem in [6].

In the case of infinite field our learning algorithm is not polynomial. We will change our algorithm to a polynomial time learning in the number of variables, $|I|$ and the *maximal degree of the variables.*

Result 2: *The class of functions of the form*

$$\sum_{\alpha \in \mathcal{I}} \alpha p_{\alpha_1}(x_1) \cdots p_{\alpha_n}(x_n),$$

where p_{α_i} are any polynomials, is learnable from substitution and equivalence queries in time polynomial in the number of variables, $|I|$ and the maximal degree of p_{α_i}.

Proof. Learn the polynomial over the domain $\mathcal{J} \subset \mathcal{K}$ for $|\mathcal{J}| = d+1$ where d is the maximal degree. Now notice that the equivalence query may not return a counterexample from this domain but by using substitution queries we can find a counterexample in this domain as follows. The hypothesis can be written as a multivariate polynomial with maximal degree d (for each variable). This is because the hypothesis in the algorithm is $h = \lambda A(x_1)\cdots A(x_n)\gamma$ where $A(x)$ is degree d matrix. Therefore the maximal degree of each variable is d. Now let h be a hypothesis and f be the

target. Let a be a counterexample for h with entries not necessarily from $\mathcal{J}$. Since $f(a) \neq h(a)$ we must have

$$g = h(x_1, a_2, \ldots, a_n) - f(x_1, a_2, \ldots, a_n) \neq 0.$$

Since g is of degree at most d there must be $b_1 \in \mathcal{J}$ such that $g(b_1, a_2, \ldots, a_n) \neq 0$ and therefore $(b_1, a_2, \ldots, a_n)$ is again a counterexample. We can use substitution queries to find this b_1. We now do the above for all the other entries. This will give us a counterexample from $\mathcal{J}^n$.□

For large or infinite fields we can learn from substitution queries only. Functions of the form

$$\sum_{\alpha \in \mathcal{I}} \alpha p_{\alpha_1}(x_1) \cdots p_{\alpha_n}(x_n),$$

where p_i are polynomials of bounded degree d are learnable from substitution queries only.

The idea of the algorithm is simple. Since the degree of the hypothesis and the target is bounded by d we can simulate equivalence queries using random membership queries.

Theorem 3. *Let $\mathcal{K}$ be a field and d be an integer where $|\mathcal{K}| > nd$. The class of functions*

$$\sum_{\alpha \in \mathcal{I}} \alpha p_{\alpha_1}(x_1) \cdots p_{\alpha_n}(x_n),$$

where p_{α_i} are any polynomials of degree at most d is learnable from substitution queries in time polynomial in the number of variables, $|I|$ and d.

Proof. We choose $\mathcal{L} \subseteq \mathcal{K}$ such that $|\mathcal{L}| = nd + 1$. To answer equivalence queries for h we randomly and uniformly choose $x \in \mathcal{L}^n$ and use membership queries to check if $f(x) \neq h(x)$.

Since $f - h$ has degree n we have, by Schwartz's lemma in [7], that the probability of $f(x) - h(x) \neq 0$ is greater than $1/(nd+1)$ which is enough for polynomial learning.□

Result 4. *All the above results are also true for the product of $k = O(\log n)$ linear functions*

$$f = \prod_{i=1}^{k} (p_1(x_1) + \cdots + p_n(x_n)).$$

Proof. This is because, by Claim 1, $\mathbf{AUdim}^{\star}(f) = 2^k = poly(n)$.□

4 Learning Decision Trees

The problem of learning decision trees from membership and equivalence queries has been solved in [4] using the monotone theory. In this section we show some learning results on more general classes of decision trees. In particular we consider the following classes of decision trees:

1. Depth $O(\log n)$ decision trees over constant $AUdim^\star$ functions, i.e., decision trees of depth $O(\log n)$ where the nodes may contain any function that have a constant $AUdim^\star$. This in particular gives a learning algorithm for $O(\log n)$-depth decision trees over XOR defined by Kushilevitz and Mansour in [5] and $O(\log n)$-depth decision trees over terms defined by Bshouty in [4].

2. Decision trees with integer variables and "less than" conditions in the nodes. This solves the open problem in [4].

Decision trees over XOR were shown to be PAC learnable with membership queries under the uniform distribution in [5]. Constant depth decision trees over terms were shown to be learnable in [4]. Notice also that result 1 is a generalization of the learnability of $O(\log n)$-term DNF.

Result 4.1 *The set of depth $O(\log n)$ decision trees over constant $AUdim^\star$ functions are learnable with membership and equivalence queries in polynomial time in the number of variables n.*

Proof. Let f be a depth $O(\log n)$ decision tree over constant $AUdim^\star$ functions. Every leaf v in the decsion tree for f labeled with 1 defines a function that is a product of $(p_1^v + \alpha_1^v) \cdots (p_{k_v}^v + \alpha_k^v)$ where $p_1^v, \ldots, p_k^v$ are the functions that we will encounter in the path from the root to this leaf and $\alpha_1^v, \ldots, \alpha_k^v$ are the labels of the edges we follow in this path. Then

$$f = \sum_{v \text{ labeled 1 in } f} (p_1^v + \alpha_1^v) \cdots (p_{k_v}^v + \alpha_k^v).$$

By Claim 1 and Lemma 3 the $AUdim^\star$ is polynomial.□

Finally, we study decision trees over the integers $[n] = \{1, ..., n\}$ and any condition $x_i \in P$ in the nodes for any $P \subseteq [n]$.

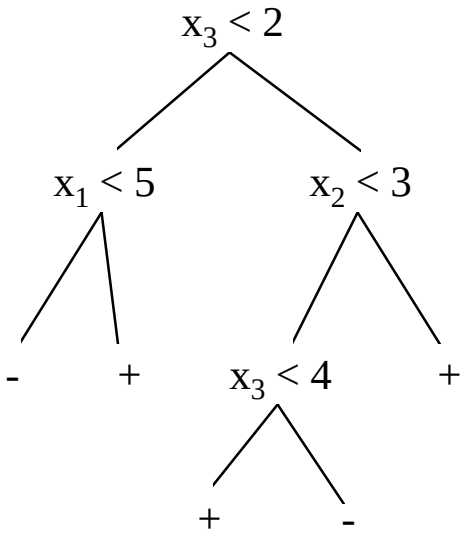

Fig. 1(a) - Decision Tree

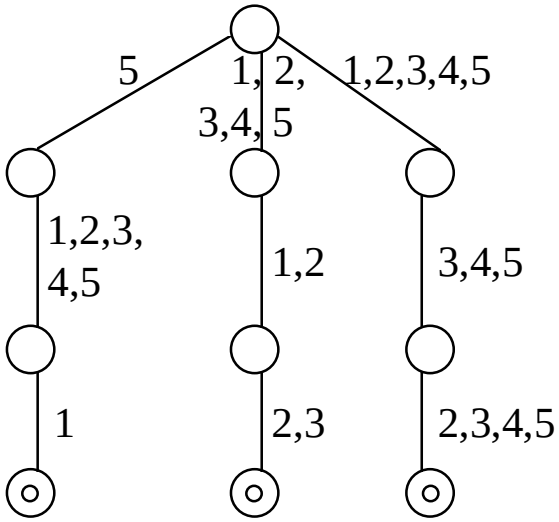

Fig. 1(a) - Non-ambiguous automaton

Result 4.2 *Decision trees over the integers {1,...,n} and $x_i \in P$ conditions for any $P \subseteq [n]$ are learnable with membership and equivalence queries.*

Proof. We show how to change the tree to a multivariate polynomial of the form $\sum_{\alpha \in \mathcal{I}} p_{\alpha_1}(x_1) \cdots p_{\alpha_n}(x_n)$. Define $I_{[x \in P]}$ to be the polynomial that takes values 0 for $x \notin P$ and 1 for $x \in P$. This polynomial has degree at most n. Now it is easy to see that each positive leaf in the tree is a monomial of the form $\prod_{i=1}^{n} I_{[x_i \in P_i]}$ and the tree is the sum of those monomials.□

One example is the tree in figure 1. This tree can be written as

$$I_{[x_1<5]} I_{[x_2 \in [n]]} I_{[x_3 \geq 2]} + I_{[x_1 \in [n]]} I_{[4 \leq x_3 < 2]} I_{[x_2 \geq 3]} + I_{[x_1 \in [n]]} I_{[x_2<3]} I_{[x_3<2]}.$$

Also the tree can be regarded as non-ambiguous automata.

References

[1] D. Angluin. Learning Regular Sets from Queries and Counterexamples. In *Information and Computation*, 75:87–106, 1987.

[2] F. Bergadano and S. Varricchio. Learning Behaviors of Automata from Multiplicity and Equivalence Queries. In *SIAM Journal on Computing*, To Appear.

[3] F. Bergadano, D. Catalano, and S. Varricchio. Learning Sat-k-DNF Formulas from Membership Queries. STOC, 1996.

[4] N. Bshouty. Exact Learning via the Monotone Theory. In *FOCS*, 1991.

[5] E. Kushilevitz and Y. Mansour. Learning Decision Trees using the Fourier Spectrum. In *Proceedings of the ACM Symposium on Theory of Computing*, 1991.

[6] R. E. Shapire and L. M. Sellie. Learning Spparse Multivariate Polynomials Over a Field with Queries and Counterexamples. In *Proceedings of the ACM Workshop on Computational Learning Theory*, 1993.

[7] J. T. Schwartz Fast polynomial algorithms for verification of polynomial identities *J. Assoc. Compt. Mach.* **27** (1980) pp.701-707.