On the Learnability of Z_N-DNF Formulas

(Extended Abstract)

Nader H. Bshouty* **Zhixiang Chen†** **Scott E. Decatur‡** **Steven Homer§**

Abstract

Although many learning problems can be reduced to learning Boolean functions, in many cases a more efficient learning algorithm can be derived when the problem is considered over a larger domain. In this paper we give a natural generalization of DNF formulas, Z_N-DNF formulas over the ring of integers modulo N. We first show using elementary number theory that for almost all larger rings the learnability of Z_N-DNF formulas is easy. This shows that the difficulty of learning Boolean DNF formulas lies in the fact that the domain is small. We then establish upper and lower bounds on the number of equivalence queries required for the exact learning of Z_N-terms. We show that $\alpha(N)n + 1 \leq (\log N)n + 1$ equivalence queries are sufficient and $\gamma(N)n$ equivalence queries are necessary, where $\alpha(N)$ is the sum of the exponents in the prime decomposition of N, and $\gamma(N)$ is the sum of logarithms of the exponents in the prime decomposition of N. We also demonstrate how the additional power of membership queries allows improved learning in two different ways: (1) more efficient learning for some classes learnable with equivalence queries only, and (2) learnability of other classes not known to be learnable with equivalence queries only. Classes which we show learnable with substantially fewer equivalence queries by using membership queries include diagonal Z_N-terms and binary weighted read-once Z_N-terms. Classes which we show learnable with the additional power of membership queries include (1) monotone Z_N-DNF formulas and (2) conjunctions of a bounded number of negated counting functions with a prime modulus.

*Department of Computer Science, the University of Calgary, 2500 University Drive N.W., Calgary, Alberta T2N 1N4, Canada. bshouty@cpsc.ucalgary.ca.

†Department of Computer Science, Boston University,Boston, MA 02215. zchen@cs.bu.edu. The author was supported by NSF grants CCR-9103055 and CCR-9400229.

‡Aiken Computation Laboratory, Harvard University, Cambridge, MA 02138. sed@das.harvard.edu. Supported by an NDSEG Fellowship and by NSF Grant CCR-92-00884.

§Department of Computer Science, Boston University, Boston, MA 02215. homer@cs.bu.edu. The author was supported by NSF grants CCR-9103055 and CCR-9400229.

1 Introduction

Symmetric Boolean functions, especially parity functions and modulo functions, have received much attention in computational learning theory. It is known that the class of single parity functions (see Helmbold *et al.*[HRS92]) and the class of single modulo functions with modulus p for any given prime number p (see Blum *et al.* [BCJ93]) are pac-learnable. In Fisher and Simon [FS92] it was proved that parity functions of monomials with at most k literals are pac-learnable, while given the assumption that $RP \neq NP$ parity functions of k monomials are not pac-learnable with the same type of functions as hypotheses, for any fixed $k \geq 2$. In Blum and Singh [BS90] it was proved that for any constant k, Boolean functions of k monomials are pac-learnable by the more expressive hypothesis class of general DNF formulas. They also showed that, for any $k \geq 2$, for any fixed symmetric function f on k inputs, f consisting of k monomials is not pac-learnable with the same type of functions as hypothesis under the assumption that $RP \neq NP$.

In the on-line learning model with queries, It is known (see Angluin *et al.* [AHK93]) that read-once Boolean functions over the basis (AND, OR, NOT) are polynomial time learnable with equivalence and membership queries. This result was extended in Hancock and Hellerstein [HH91] to Boolean functions over a larger basis including arbitrary threshold functions and parity functions. Further, it was shown in Bshouty *et al.* [BHH92a, b] that read-once functions over the basis of arbitrary symmetric functions are polynomial time learnable with equivalence and membership queries. However, it was also proved in [BHH92b] that read-twice functions over the same basis are not learnable under standard cryptographic assumptions.

In this paper, by introducing counting functions which include parity and modulo functions, we investigate the learnability of a much larger class of functions defined over the domain Z_N^n, the class of Z_N-DNF formulas. This is a broad class of functions and in essence includes all Boolean functions, and especially Boolean DNF formulas, as special cases. Our goal in this approach is to develop new techniques in this setting by means of number theory and algebra and thus derive general results that extend learnability over Boolean domains.

The study of learnability of extensions of Boolean functions to arbitrary domains has also been proposed and investigated by many other researchers (see, for example, [SS93] and [B93]). Schapire and Sellie [SS93] designed efficient algorithms for learning multilinear polynomials over the domain F^n for any finite field F, and for learning polynomials over any semilattice of finite height. Bshouty [B93] extended DNF and CNF formulas to an arbitrary domain Σ^n for any finite abelian group Σ and, he proved that any polynomial size decision tree over Σ^n is learnable. The learnability of counting functions over the domain Z_N^n was initially proposed by Chen and Homer in [CH93] and many preliminary results were obtained there.

This paper is organized as follows. In section 2, we define Z_N-terms, Z_N-DNF formulas, and the learning models as well as parameters $\alpha(N)$ and $\gamma(N)$. In section 3, we show using elementary number theory that the learnability of Z_N-DNF formulas is easy for almost all larger rings. This shows that the difficulty of learning Boolean DNF formulas lies in the fact that the domain is small. In section 4, we determine the number of equivalence queries sufficient and necessary for learning any Z_N-terms over the domain Z_N^n. In section 5, we demonstrate how the additional power of membership queries allows more efficient learning for some classes learnable with equivalence queries only. Those classes are diagonal Z_N-terms and binary weighted read-once Z_N-terms. In section 6, we investigate the problem of learning conjunctions of negated counting functions with equivalence and membership queries. We show that this is in general harder than learning DNF formulas. On the other hand, we show that this problem is learnable when the modulus is prime and the number of negated counting functions in the conjunction is $O(\frac{n}{\log(N-1)})$. In section 7, We show that monotone Z_N-DNF formulas, which are generalizations of monotone DNF formulas, are learnable using equivalence and membership queries. We conclude the paper by listing several open problems in section 8.

2 Preliminaries

2.1 Z_N-Terms and Z_N-DNF formulas

We assume that Z is the set of all integers. Let $Z_N = \{0, \ldots, N-1\}$ for any integer $N \geq 2$. For any example $\vec{a} \in Z_N^n$, we use a_i to denote the i-th component of $\vec{a}$ for $i \in \{1, \ldots, n\}$. A counting function with a modulus $N \geq 2$ is defined as follows[1]:

$$C^N_{\vec{a},b}(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_{j=1}^n a_i x_j \equiv b \pmod N, \\ 0 & \text{otherwise,} \end{cases}$$

where $\vec{a} = (a_1, \ldots, a_n) \in Z_N^n$ and $b \in Z_N$. For convenience, we may use $C^N_{\vec{a},b}$ to stand for $C^N_{\vec{a},b}(x_1, \ldots, x_n)$. A Z_N-term T is a conjunction of counting functions as follows:

$$T = C^N_{\vec{a_1},b_1} \wedge \cdots \wedge C^N_{\vec{a_m},b_m}.$$

We say that a counting function $C^N_{\vec{a},b}$ is *diagonal* if there is at most one $i \in \{1, \ldots, n\}$ such that $a_i \neq 0$. We say that a Z_N-term is *diagonal* if all counting functions in it are diagonal. A Z_N-DNF formula F is a disjunction of Z_N-terms.

For any $a, b \in Z_N$, let ab denote $(ab \bmod N)$. Given $b \in Z_N$ and $\vec{r} \in Z^n$, define

$$Z_N b = \{ab \mid a \in Z_N\} \quad \textit{and}$$

$$Z_N \vec{r} = \{a\vec{r} \mid a \in Z_N\},$$

where $a\vec{r} = (ar_1, ar_2, \ldots, ar_n)$. For subsets $A, B \subseteq Z_N^n$, for any $\vec{b} \in Z_N^n$, define

$$A + \vec{b} = \{\vec{a} + \vec{b} \mid \vec{a} \in A\} \quad \textit{and}$$

$$A + B = \{\vec{u} + \vec{v} \mid \vec{u} \in A \textit{ and } \vec{v} \in B\}.$$

2.2 Parameters $\alpha(N)$ and $\gamma(N)$

Given any integer $N \geq 2$, let $N = p_1^{r_1} p_2^{r_2} \cdots p_t^{r_t}$, where p_i are distinct primes and $r_i \geq 1$ for $i = 1, \ldots, t$. We define

$$\alpha(N) = \sum_{i=1}^t r_i \quad \textit{and}$$

$$\gamma(N) = \sum_{i=1}^t \lceil \log(r_i + 1) \rceil .$$

[1] For convenience, we define counting function by switching 1 and 0 in the definition in [CH94]. However, both definitions in essence have the same representation capacity.

It is easy to see that, for any integer $N \geq 2$,

$$1 \leq \gamma(N) \leq \alpha(N) \leq \log N.$$

2.3 Learning Models

Our first model is the on-line learning model with equivalence queries. The goal of a learning algorithm (or learner) for a class of Boolean-valued function $\mathbf{C}$ over a domain X^n is to learn any unknown target function $f \in C$ that has been fixed by a teacher. In order to obtain information about f, the learner can ask equivalence queries by proposing hypotheses h from a fixed hypothesis space $\mathbf{H}$ of functions over X^n with $C \subseteq H$ to an equivalence oracle $EQ()$. If $h = f$, then $EQ(h) =$ "*yes*", so the learner succeeds. If $h \neq f$, then $EQ(h) = \vec{x}$ for some $\vec{x} \in X^n$ such that $h(\vec{x}) \neq f(\vec{x})$, called a counterexample. x is called a positive example if $f(\vec{x}) = 1$ and a negative example otherwise. A learning algorithm exactly learns C, if for any target function $f \in C$, it can find a $h \in H$ that is logically equivalent to f. A learning algorithm exactly learns C with high probability, if for any $f \in C$, it can infer a hypothesis $h \in H$ that is logically equivalent to f on all inputs with probability at least $1-\delta$, where $0 < \delta < 1$, and the probability is taken over all examples in the domains. We say that a class $\mathbf{C}$ is polynomial time learnable (with high probability) if there is a learning algorithm that exactly learns any target function in $\mathbf{C}$ (with probability at least $1 - \delta$) and runs in time polynomial in the logarithm of the size of the domain and the size of the target function (and in $\frac{1}{\delta}$ as well).

Our second model is the on-line learning model with equivalence and membership queries. This model is the same as the first, but in addition to equivalence queries, the learner can also ask membership queries by presenting examples in the domain to a membership oracle $MQ()$. For any example $\vec{x}$, $MQ(\vec{x}) =$ "*yes*" if $f(\vec{x}) = 1$, otherwise $MQ(\vec{x}) =$ "*no*".

3 Learning Z_N-DNF formulas

Our main result in this section is

Theorem 3.1. *The class of k-term Z_N-DNF formulas is learnable for all k and N such that $k < p(N)$ where $p(N)$ is the minimal prime that divides N in*

$$O\left(\frac{p(N)}{p(N)-k} kn \log N\right)$$

expected time and queries.

Notice that our algorithm is efficient for Ns that have large prime factors and $k < cp(N)$ for some constant c. In particular our result gives an efficient learning algorithm for k-term Z_N-DNF for prime N and $k \leq cN$ for some constant c. The algorithm we present here is a Las Vegas algorithm that guarantees learning after expected polynomial time.

We can regard the set of ones of the target as union of cosets. I.e., for a k-term Z_N-DNF f there are linear spaces $L_1, \ldots, L_k$ and vectors $\vec{a}_1, \ldots, \vec{a}_k$ such that

$$f^{-1}(1) = L_1 + \vec{a}_1 \cup \cdots \cup L_k + \vec{a}_k.$$

Each $L_i + \vec{a}_i$ is a coset and we will call $\vec{a}_i$ the coset leader.

Our algorithm guarantees with high probability that the examples received from the equivalence queries are positive.

We first prove the following

Lemma 3.2. *Let $L_1, L_2 \subseteq Z_N^n$ be linear spaces over Z_N. If $L_2 + \vec{a}_2 \not\subseteq L_1 + \vec{a}_1$ then under the uniform distribution*

$$\Pr[L_1 + \vec{a}_1 | L_2 + \vec{a}_2] \leq \frac{1}{p(N)}.$$

Proof. Notice first that for any coset $L + \vec{a}$ we have $|L + \vec{a}| = |L|$. Since L is a subgroup of Z_N^n we must have $|L|$ divides $|Z_N^n| = N^n$. Therefore we have $|L_2 + \vec{a}_2|$ divides N^n. Now since intersection of cosets is a coset and since $L_2 + \vec{a}_2 \not\subseteq L_1 + \vec{a}_1$ we also have $|L_2 + \vec{a}_2 \cap L_1 + \vec{a}_1|$ divides N^n and is (strictly) smaller than $|L_2 + \vec{a}_2|$. Therefore we can have two cases. Either $|L_2 + \vec{a}_2| = 1$ and $|L_2 + \vec{a}_2 \cap L_1 + \vec{a}_1| = 0$ or $|L_2 + \vec{a}_2 \cap L_1 + \vec{a}_1| / |L_2 + \vec{a}_2|$ is at least $p(N)$, the smallest prime that divide N. In both cases we have

$$\begin{aligned} \Pr[L_1 + \vec{a}_1 | L_2 + \vec{a}_2] &= \frac{|L_2 + \vec{a}_2 \cap L_1 + \vec{a}_1|}{|L_2 + \vec{a}_2|} \\ &\leq \frac{1}{p(N)}. \square \end{aligned}$$

The key idea in the learning algorithm is the following. Suppose we get two positive examples $\vec{x}_1$ and $\vec{x}_2$. If $\vec{x}_1$ and $\vec{x}_2$ are in the same coset $L + \vec{a}$ then

$$\vec{x}_1 + \lambda(\vec{x}_2 - \vec{x}_1) \in L + \vec{a}$$

for any $\lambda \in Z_N$ and therefore is positive in the target for every λ. Now if $\vec{x}_1$ and $\vec{x}_2$ are not in the same coset for all $L_i + \vec{a}_i$, $i = 1, \ldots, k$, then for a random uniform λ we have

$$\begin{aligned} &\Pr_\lambda[\vec{x}_1 + \lambda(\vec{x}_2 - \vec{x}_1) \text{ is positive}] \\ &\quad = \Pr_\lambda[(\exists i)\vec{x}_1 + \lambda(\vec{x}_2 - \vec{x}_1) \in L_i + \vec{a}_i] \\ &\quad \leq k \Pr_\lambda[\vec{x}_1 + \lambda(\vec{x}_2 - \vec{x}_1) \in L_i + \vec{a}_i] \\ &\quad = k \Pr[L_i + \vec{a}_i | L + \vec{x}_1], \end{aligned}$$

where $L + \vec{x} = \{\vec{x}_1 + \lambda(\vec{x}_2 - \vec{x}_1) | \lambda \in Z_N\}$. Since

$L + \vec{x}_1 \not\subseteq L_i + \vec{a}_i$ by the lemma we have

$$\Pr_{\lambda}[\vec{x}_1 + \lambda(\vec{x}_2 - \vec{x}_1) \text{ is positive}] \le \frac{k}{p(N)}.$$

Therefore to decide whether or not $\vec{x}_1$ and $\vec{x}_2$ are in the same coset we randomly uniformly choose $\lambda \in Z_N$ and ask membership queries with $\vec{x}_1 + \lambda(\vec{x}_2 - \vec{x}_1)$. If the answer is negative then $\vec{x}_1$ and $\vec{x}_2$ are not in the same coset. If they are not in the same coset then with probability at least $1 - k/p(N)$ we get a negative counterexample. Therefore this algorithm run in expected time

$$O\left(\frac{p(N)}{p(N)-k}\log\frac{1}{\delta}\right)$$

to get confidence at least $1-\delta$.

Now we may have three positive examples $\vec{x}_1, \vec{x}_2$ and $\vec{x}_3$ where each pair is in the same coset but not all of them are. Therefore we need the following lemma. The proof is similar to the above

Lemma 3.3. *Suppose $\vec{x}_1, \ldots, \vec{x}_j$ are in the same coset $L + \vec{a}$. If $\vec{x}_{j+1}$ is not in the same coset $L + \vec{a}$ then with probability at least $1 - \frac{k}{p(N)}$ we have for random uniform $\lambda_2, \ldots, \lambda_{j+1}$ from Z_N*

$$f(\vec{x}_1 + \lambda_2(\vec{x}_2 - \vec{x}_1) + \cdots + \lambda_j(\vec{x}_j - \vec{x}_1) + \lambda_{j+1}(\vec{x}_{j+1} - \vec{x}_1))$$
$$= 0.$$

Now we can write the algorithm. The algorithm starts by asking equivalence queries with 0 to get a positive example. At the ith stage of the algorithm we have positive examples in $S_1, S_2, \ldots, S_r$ where each set of examples contains bases from the same coset. For each set we also have a leader $l(S_i)$ for this set. The leader is the leader of the coset and is the first positive example that we obtained in this set. Let

$$f_S^{-1}(1) = l(S) + \sum_{s \in S \backslash \{l(S)\}} Z_N(s - l(S)).$$

Notice that since S_i is in the same coset we have (with high probability) $f_{S_i}^{-1}(1)$ is a subset of the target. At stage $i+1$ we ask equivalence queries $h = f_{S_1} \vee \cdots \vee f_{S_r}$. Since $h^{-1}(1)$ is a subset of the target we will get a positive counterexample x_{i+1}. We now use the membership queries to test whether $\vec{x}_{i+1}$ belong to the coset S_j for all j using Lemma 3.3. It may happen that $\vec{x}_{i+1}$ belongs to many cosets in which case we add it to each one and it may also happen that $\vec{x}_{i+1}$ is not in any of them in which case we create a new set S_{r+1} and put $\vec{x}_{i+1}$ in it and make it the leader of the new set.

4 Learning Z_N-Terms

We give upper and lower bounds on the number of equivalence queries required for learning Z_N-terms by proving following theorems.

Theorem 4.1. *There is an algorithm for learning any Z_N-term over the domain Z_N^n using at most $\alpha(N)n + 1 \le (\log N)n + 1$ equivalence queries.*

Theorem 4.2. *To learn a diagonal Z_N-term (and hence a Z_N-term) we need at least $\gamma(N)n$ equivalence queries.*

Theorem 4.3. *Any term (i.e., a conjunction of counting functions) with distinct moduli $\{N_i\}$ are learnable using at most $\alpha(lcm(N_i))n + 1$ equivalence queries provided that $lcm(N_i)$ is known a prior to the learner, where $lcm(N_i)$ is the least common multiple of N_i.*

Theorem 4.4. *Diagonal Z_N-terms are learnable using at most $\gamma(N)n + 1$ equivalence queries.*

Here, we only prove the first two theorems. For two integers a and b we write $a|b$ if b is divisible by a.

Lemma 4.5. *For an element $a \in Z_N$, the following properties hold.*

1. *$Z_N a = Z_N gcd(a, N)$.*
2. *If $a|N$ then $\forall b \in Z_N a$, $a|b$.*
3. *$Z_N a_1 + \cdots + Z_N a_m = Z_N \gcd(a_1, \ldots, a_m, N)$.*
4. *If $a|N$ and the elements of A and B are divisible by a then the elements of $A + B$ are divisible by a.*
5. *$a \notin Z_N b$ if and only if $\gcd(b, N) \not| \gcd(a, N)$.*

A sequence $(\vec{a}_1, \ldots, \vec{a}_m)$ of elements in Z_N^n is called an *independent sequence* if

$$\vec{a}_1 \ne \vec{0}, \cdots, \vec{a}_i \notin Z_N\vec{a}_1 + \cdots + Z_N\vec{a}_{i-1}, \cdots,$$
$$\vec{a}_m \notin Z_N\vec{a}_1 + \cdots + Z_N\vec{a}_{m-1}.$$

The integer m is called the *length* of the sequence. The length of the longest independent sequence in Z_N^n is denoted by $len(n, N)$.

Lemma 4.6. $len(1, N) = \alpha(N)$.

Proof. Let $N = p_1^{r_1} \cdots p_t^{r_t}$. The sequence

$$\begin{aligned}(a_1, \ldots, a_{\alpha(N)}) &= (p_1^{r_1-1}p_2^{r_2}\cdots p_{t-1}^{r_{t-1}}p_t^{r_t}, \ldots, p_1^0 p_2^{r_2}\cdots p_{t-1}^{r_{t-1}}p_t^{r_t},\\ &\quad p_1^0 p_2^{r_2-1}\cdots p_{t-1}^{r_{t-1}}p_t^{r_t}, \ldots, p_1^0 p_2^0\cdots p_{t-1}^{r_{t-1}}p_t^{r_t},\\ &\quad \vdots\\ &\quad p_1^0 p_2^0\cdots p_{t-1}^0 p_t^{r_t-1}, \ldots, p_1^0 p_2^0 \cdots p_{t-1}^0 p_t^0)\end{aligned}$$

is an independent sequence of length $\alpha(N)$. To show that the sequence is independent, notice that

$$a_s = p_1^0 \cdots p_i^0 p_{i+1}^j p_{i+2}^{r_{i+2}} \cdots p_t^{r_t}$$

is not divisible by p_{i+1}^{j+1} but $a_1, \ldots, a_{s-1}$ are divisible by p_{i+1}^{j+1}. Therefore

$$a_s \notin Z_N a_1 + \cdots + Z_N a_{s-1}$$

because all the integers in $Z_N a_1 + \cdots + Z_N a_{s-1}$ are divisible by p_{i+1}^{j+1}. (Notice here that we would not have this property if N were not divisible by p_{i+1}^{j+1}.) This proves that

$$len(1, N) \geq \alpha(N).$$

To show that $len(1, N) \leq \alpha(N)$ we suppose there is a sequence

$$(a_1, \ldots, a_s), \quad s > \alpha(N)$$

that is independent. By Property 2 of Lemma 4.5 we have

$$\begin{aligned} a_i &\notin Z_N a_1 + \cdots + Z_N a_{i-1} \\ &= Z_N gcd(a_1, \ldots, a_{i-1}, N). \end{aligned}$$

Therefore if $N = p_1^{r_1} \cdots p_t^{r_t}$ and $a_j = b_j p_1^{r_{j,1}} \cdots p_t^{r_{j,t}}$ where $gcd(b_j, N) = 1$ then

$$\begin{aligned} a_i &\notin Z_N gcd(a_1, \ldots, a_{i-1}, N) \\ &= Z_N p_1^{\min_{0\leq j<i} r_{j,1}} \cdots p_t^{\min_{0\leq j<i} r_{j,t}}. \end{aligned}$$

where $r_{0,k} \doteq r_k$. By Property 5 of Lemma 4.5

$$\begin{aligned} &p_1^{\min_{0\leq j<i} r_{j,1}} \cdots p_t^{\min_{0\leq j<i} r_{j,t}} \\ &\not| \; p_1^{\min(r_{0,1}, r_{i,1})} \cdots p_t^{\min(r_{0,t}, r_{i,t})}. \end{aligned}$$

Therefore there must be a k such that $\min(r_{0,k}, r_{i,k}) < \min_{0\leq j<i} r_{j,k}$ which implies that

$$r_{i,k} < \min_{0\leq j<i} r_{j,k}.$$

Thus s can be at most $\alpha(N)$. □

Lemma 4.7. $len(n, N) = \alpha(N)n$.

Lemma 4.7 can be proved by Lemma 4.6 and by induction on n. □

Proof of Theorem 4.1. Let f be a Z_N-DNF formula. Let $\vec{a}_1, \ldots, \vec{a}_m$ be the counterexamples and $s_i = (\vec{a}_1, \ldots, \vec{a}_i)$. Define the Boolean functions $f_0(x) = 0$ and, for $i \geq 1$,

$$f_i(x) = \begin{cases} 1 & x \in a_1 + Z_N(\vec{a}_2 - \vec{a}_1) + \cdots + \\ & Z_N(\vec{a}_m - \vec{a}_1) \\ 0 & \text{otherwise.} \end{cases}$$

The Learning Algorithm

Step 1. $s \leftarrow ()$.

Step 2. While $EQ(f_s) \rightarrow a$ does not answer "YES" do $s \leftarrow (s, \vec{a})$.

Positive examples of f are the solutions to some linear system of equations $AX = B$ where $X = (x_1, \ldots, x_n)^T$ and A is a $t \times n$ matrix and B is a column t-vector both with entries from Z_N. Now, Theorem 4.1 follows from Lemma 4.7 and the following claims which can be verified easily.

C1: $\vec{a}_i$ is a positive counterexample.

C2: $A\vec{a}_i = B$.

C3: $f_i \Rightarrow f$.

C4: $(\vec{a}_2 - \vec{a}_1, \ldots, \vec{a}_i - \vec{a}_1)$ is an independent sequence. □

Proof of Theorem 4.2. Let $N = p_1^{r_1} \cdots p_t^{r_t}$. Consider the class of functions $f_{\lambda_1, \ldots, \lambda_n}$ where $\lambda_1, \ldots, \lambda_n | N$ and

$$f^{-1}_{\lambda_1, \ldots, \lambda_n}(1) = \{\vec{x} | x_i = 0 \text{ mod } \lambda_i\}.$$

Suppose the learner already knows $\lambda_1, \ldots, \lambda_{s-1}$ and knows that

$$p_1^{q_1} \cdots p_j^{q_j} p_{j+1}^{\delta_1} \leq \lambda_s \leq p_1^{q_1} \cdots p_j^{q_j} p_{j+1}^{\delta_2} p_{j+2}^{r_{j+2}} \cdots p_t^{r_t}$$

and knows nothing about $\lambda_{s+1}, \ldots, \lambda_n$. Suppose the learner asks equivalence queries with h. Let $A = h^{-1}(1)$. Let

$$L_1 = \{(x_1, \ldots, x_n) | x_i = 0 \text{ mod } \lambda_i, i = 1, \ldots, s-1\}$$

and let

$$L_2 = \{((x_1, \ldots, x_n) | x_s = p_1^{q_1} \cdots p_j^{q_j} p_{j+1}^{\left\lfloor \frac{\delta_1+\delta_2}{2} \right\rfloor} \text{ mod } N\}.$$

If $A \not\subseteq L_1$ then the adversary can give the learner $\vec{x} \in L_1 - A$ as a counterexample and the learner cannot gain any information from this counterexample. If $A \subseteq L_1$ but $A \not\subseteq L_2$ then the adversary will return an element from $A - (L_1 \cap L_2)$ and the learner gains only the additional information that

$$\lambda_s \geq p_1^{q_1} \cdots p_j^{q_j} p_{j+1}^{\left\lfloor \frac{\delta_1+\delta_2}{2} \right\rfloor}.$$

If on the other hand if $A \subseteq L_1 \cap L_2$ then the adversary will return

$$(0, \overset{s-1}{\ldots}, p_1^{q_1} \cdots p_j^{q_j} p_{j+1}^{\left\lfloor \frac{\delta_1+\delta_2}{2} \right\rfloor}, 0, \ldots, 0)$$

and the only information that the learner gains is that

$$\lambda_s \leq p_1^{q_1} \cdots p_j^{q_j} p_{j+1}^{\left\lfloor \frac{\delta_1+\delta_2}{2} \right\rfloor + 1}.$$

Now by induction the result follows. □

5 Reducing the Number of Equivalence Queries by Using Membership Queries

As suggested in [BGHM93], it is reasonable to believe that an equivalence query is more expensive than a membership query. A pratically ideal learning algorithm will use as few equivalence queries as possible. Here we show how to reduce the number of equivalence queries using membership queries when learning certain restricted syntax classes. A read-once Boolean weighted Z_N-term is a Z_N-term in which all weight vectors are Boolean valued vectors and for all components j, no two distinct vectors have a 1 in component j.

Theorem 5.1. *There is an algorithm for learning diagonal Z_N-terms over the domain Z_N^n using 1 equivalence query and at most $\gamma(N)n$ membership queries.*

One first notes that, given a diagonal Z_N-term T, there are $d_i \in Z_N$, $i = 1, \ldots, n$, such that the set of all examples making T true is $Z_N(d_1, \ldots, 0) + \cdots + Z_N(0, \ldots, d_n) + \vec{y}$ for any $\vec{y}$ making T true. One can ask the first equivalence query for ϕ to get $\vec{y}$. Let $N = p^{r_1} p^{r_2} \cdots p^{r_t}$. Since $Z_N d_i = Z_N gcd(d_i, N)$, to learn d_i we only need to determine u_j such that $d_i = p^{u_1} p^{u_2} \cdots p^{u_t}$ with $0 \leq u_j \leq r_j$. Thus, we can find u_j using the binary search with membership queries among integers $0, 1, \ldots, r_j$. Hence, with at most $\gamma(N)n$ membership queries, we can find all the d_i.

Theorem 5.2. *There is an algorithm for learning read-once Boolean weighted Z_N-terms over the domain Z_M^n using 1 equivalence query and at most n^2 membership queries when $M \geq N$, while at most $2n$ equivalence queries are needed if $2 \leq M < N$.*

It is worth noting that in Theorem 5.2 the bounds on the number of membership queries and equivalence queries are respectively independent of N.

To prove Theorem 5.2, note that it is sufficient to determine the relevant variables in each of the counting functions which compose the Z_N-term. When the domain is Z_M^n for $M \geq N$, one need only obtain a single positive example $\vec{x}$ (using an equivalence query) and for each pair of variables, use $\vec{x}$ to test whether the pair of variables appear in the same counting function. The test for any two variables entails a membership query on an example obtained from $\vec{x}$ by incrementing (mod N) one variable and decrementing (mod N) the other. If the variables appear in the same counting function, then this new example is also positive, otherwise it is negative.

When the domain is Z_M^n for $M < N$, then it is possible that the positive example $\vec{x}$ cannot be used to test all pairs of variables, *e.g.* when both variables are set to 0 in $\vec{x}$. However, after making all possible tests with $\vec{x}$, one can formulate an equivalence query that will result in a new example which allows progress to be made. Such a strategy uses at most $2n$ equivalence queries.

6 Learning Conjunctions of Negated Counting Functions

One open problem proposed by A. Blum [B94] and R. Rivest [R94] regarding learning conjunctions of counting functions is whether conjunctions of negated counting functions with a modulus N over the domain Z_N^n are poly-time learnable. When $N = 2$, it is easy to see that a positive answer to the problem exists. In this section, we will prove two related results.

Theorem 6.1. *If conjunctions of negated counting functions with modulus $N > n$ over the domain Z_N^n are polynomial time learnable, then CNF (and thus DNF) formulas are polynomial time learnable.*

Given a CNF formula $F = \bigwedge_{i=1}^m F_i$, where F_i are clauses. we can construct a conjunction of negated counting functions $C(F)$ such that $F(\vec{x}) = 1$ if and only if $C(F)(\vec{x}) = 1$. In order to do so, for each clause F_i, define a vector $\vec{a}_i = (a_{i1}, \ldots, a_{in})$ such that $a_{ij} = 1$ if x_j appears at F_i, $a_{ij} = (N-1)$ if $\bar{x}_j$ appears and, $a_{ij} = 0$ otherwise, where $i \in \{1, \ldots, m\}$ and $j\{1, \ldots, n\}$. We also define, for any $i \in \{1, \ldots, m\}$, b_i as the difference of N and the number of negated variables appearing at F_i. Then, $C(F) = \neg C^N_{\vec{a}_1, b_1} \wedge \cdots \wedge \neg C^N_{\vec{a}(m), b_m}$ satisfies the requirement.

Theorem 6.2. *We can learn conjunctions of m negated counting functions with a prime modulus $N > 2$ over the domain Z_N^n using at most $n + (N-1)^m$ equivalence queries and at most $N(n + (N-1)^m)(N-1)^m$ membership queries. (Hence, when $m = O(\frac{\log n}{\log(N-1)})$, the algorithm is polynomial.)*

One can verify that a conjunction of m negated counting functions F is equivalent to the "union" of the systems $A_{m,n}X = D_i$, $i = 1, \ldots, (N-1)^m$, where $D_i = (d_{i1}, \ldots, d_{im})^T$ are distinct and $d_{ij} \in (Z_N - \{b_i\})$. $A_{m,n}$ and b_i are determined by F. Thus, to learn F, we only need to learn all the systems $A_{m,n}X = D_i$ over the vector space Z_N^n. The only difficulty involved in this task is how to decide whether two positive examples for F are solutions to the same system. However, this difficulty is overcome by the following lemma.

Lemma 6.3. *Given any two examples $\vec{\psi} = (\psi_1, \ldots, \psi_n)$ and $\vec{\omega} = (\omega_1, \ldots, \omega_2)$ making F true, then both $\vec{\psi}$ and $\vec{\omega}$ are solutions to the same system if and only if, for all $u \in Z_N$, $u(\vec{\omega} - \vec{\psi}) + \vec{\psi}$ makes F true.*

A recent result obtained by Bertoni *et al.* in [BCF95] implies that any conjunction of negated counting functions $C = \neg C^N_{\vec{a}_1, b_1} \wedge \cdots \wedge \neg C^N_{\vec{a}(m), b_m}$ over the domain Z_N^n is poly-time learnable with at most $n^{N-1}+1$ equivalence queries, provided that when N is a constant prime and $b_i = 0$ for all $i = 1, \ldots, m$. Enlightened by this result, Chen [C95] further proved that for any constant prime N, conjunctions of counting functions and negated counting function with modulus N over the domain Z_N^n are poly-time learnable with at most $(n+1)^{N-1}+1$ equivalence queries.

7 Learning Monotone Z_N-DNF formulas

A monotone concept (see the formal definition in [B93]) is uniquely determined by one element based on a partial order. This relatively simple structure limits the representation capacity of monotone concepts. There are natural concept classes that can not be represented as disjunctions (or conjunctions) of monotone concepts, for example, unions of axis parallel discretized rectangles (as noted in [B93]), unions of discretized polytopes, and unions of modules over a finite ring. Thus, in order to learn those concepts, new techniques are required. As a first step along this line, we prove Theorem 7.1 that is an extension of Angluin's algorithm for learning monotone DNF formulas [A88]. For any Z_N-DNF formula

$$F = \bigvee_{i=1}^{m} L_i, \text{ where } L_i = C^N_{\vec{a}_{i1}, b_{i1}} \wedge \cdots \wedge C^N_{\vec{a}_{im_i}, b_{im_i}}.$$

We say that F is monotone if, (1) for any $i \in \{1, ..., m\}$ and $j \in \{1, ..., m_i\}$, L_i is diagonal and $b_{ij} \neq 0$, (2) for any two distinct $i, j \in \{1, ..., m\}$, L_i and L_j have different sets of relevant variables. One should note that a monotone Z_N-DNF formula is equivalent to a union of cosets of Z_N^n and the method developed in [B93] can not be applied to learn it.

Theorem 7.1. *There is an algorithm for learning any monotone Z_N-DNF formula F over the domain Z_N^n using at most $m(n\alpha(N)+1)$ equivalence queries and at most $mn(n\alpha(N)+1)$ membership queries.*

Given a positive example $\vec{x}$, let $R(\vec{x})$ be the example obtained by flipping all the components in $\vec{x}$ to 0 such that each flipping still makes F true. Let $V(\vec{x})$ denote the set of all the variables x_i such that the i-th component in $R(\vec{x})$ is not zero. The algorithm works in stages. At any stage s, let $W(s)$ be the class of the sets of relevant variables, $E(s)$ be the set of all the examples received. For each $\psi \in W(s)$, let (ψ) be the set of all $\vec{x} \in E(s)$ such that $V(\vec{x}) = \psi$. For $(\psi) = \{\vec{x}_{i_1}, ..., \vec{x}_{i_k}\}$ with $i_1 < \cdots < i_k$, define $H(s, \psi) = \vec{y}_1 + Z(\vec{y}_2 - \vec{y}_1) + \cdots + Z(\vec{y}_k - \vec{y}_1)$, where $\vec{y}_j = R(\vec{x}_{i_j})$. The hypothesis issued by the learner at stage s is $H(s) = \bigcup\{H(s, \psi) | \psi \in W(s)\}$.

The Learning Algorithm:

Stage 0. Set $H(0) = W(0) = E(0) = \phi$.
Stage $s+1 \geq 1$. Ask an equivalence query for the hypothesis $H(s)$. If yes then stop. Otherwise the learner receives a counterexample $\vec{x}_s$. Set $E(s+1) = E(s) \cup \{\vec{x}_s\}$. Set $W(s+1) = W(s) \cup \{V(\vec{x}_s)\}$, if $V(\vec{x}_s) \notin W(s)$. Otherwise, set $W(s+1) = W(s)$.

8 Open Problems

We list some open problems.

1. We have shown that conjunctions of counting functions with modulus $N \geq 2$ over the domain Z_N^n can be learned using at most $n\alpha(N)+1$ equivalence queries. On the other hand, we have also proved that any algorithm for learning conjunctions of counting functions with modulus $N \geq 2$ over the domain Z_N^n requires at least $n\gamma(N)$ equivalence queries. Can one close the gap between the upper and lower bounds of equivalence queries?
2. Can one substantially decrease the number of equivalence queries required for learning conjunctions of counting functions with modulus $N \geq 2$ over the domain Z_N^n, provided that one is allowed to use $poly(n, \log N)$ many membership queries?
3. We know that the problem of learning disjunctions of negated counting functions with modulus N over the domain Z_N^n is harder than learning DNF formulas for arbitrary $N > n$. On the other hand, a polytime algorithm exists for this problem when N is a constant prime [C95]. However, we do not know whether this problem is poly-time learnable when for arbitrary $N \leq n$ or for a constant composite $N > 2$.
4. We can extend a decision tree over the domain Z_N^n in such a way that each node of the tree is a counting function $ax \equiv b(mod\ N)$. Can one learn the class of the extended decision trees over the domain Z_N^n using equivalence and membership queries?
5. We have proved that monotone Z_N-DNF formulas over the domain Z_N^n are poly-time learnable using equivalence and membership queries. Can one learn this class using equivalence and incomplete membership queries?
6. Can one learn the class of disjunctions of conjunctions of diagonal counting functions with modulus $N \geq 2$ over the domain Z_N^n? Or in general, can one learn Z_N-DNF formulas over the domain Z_N^n? One should note that those two problems are harder than learning DNF formulas. A systematic approach to the monotone theory has been established in [B93], and successfully used to learn any Boolean functions by decision trees and to learn any

Boolean functions by DNF formulas or CNF formulas (or both). However, the theory developed in [B93] can not be used to learn Z_N-DNF formulas, because the algebraic structures of Z_N-DNF formulas are more complicated than that of Boolean functions.

References

[A88] D. Angluin, "Queries and concept learning", *Machine Learning*, 2, 1988, pages 319-342.

[AHK93] D. Angluin, L. Hellerstein, M. Karpinsky, "Learning learning read-once formulas with queries", *J. ACM*, 1, 1993, pages 185-210.

[BCF95] A. Bertoni, N. Cesa-Bianchi, G. Fiorino, "Efficient learning with equivalence queries of conjunctions of modulo functions, submitted to *Information Processing Letters*, 1995.

[B94] A. Blum, Personal Communications, 1994.

[BR92] A. Blum, S. Rudich , "Fast learning of k-term DNF formulas with queries", STOC, 1992, pages 382-389.

[BCJ93] A. Blum, P. Chalasani, J. Jackson, "On learning embedded symmetric concepts", COLT, pages 337-346, 1993.

[BS90] A. Blum, M. Singh, "Learning functions of k terms", COLT, pages 144-153, 1990.

[B93] N. Bshouty, "Exact Learning via the monotone theory", FOCS, 1993.

[BGHM93] N. Bshouty, S. Goldman, T. Hancock, S. Matar, "Asking queries to minimize errors", COLT, pages 41-50, 1993.

[BHH92a] N. Bshouty, T. Hancock, L. Hellerstein, "Learning arithmetic read-once formulas" STOC, 1992.

[BHH92b] N. Bshouty, T. Hancock, L. Hellerstein, "Learning Boolean read-once formulas with arbitrary symmetric and constant fan-in gates", COLT, pages 1-15, 1992.

[C95] Z. Chen, "Disjunctions of negated counting functions are efficiently learnable with equivalence queries", submitted to COCOON'95, 1995.

[CH94] Z. Chen, S. Homer, "On learning counting functions with queries", COLT, pages 218-227, 1994.

[HH91] T. Hancock, L. Hellerstein, "Learning read-once formulas over fields and extended bases", COLT, pages 326-336, 1991.

[HSW92] D. Helmbold, R. Sloan, M. Warmuth, "Learning integer lattices", *SIAM J. Comput.*, 1992, pages 240-266.

[R94] R. Rivest, Personal Communications, 1994.

[SS93] R. Schapire, L. Sellie, "Learning sparse multivariate polynomials over a field with queries and counterexamples", COLT, 1993.

[V84] L. Valiant, "A theory of the learnable", *Communications of the ACM*, 27, pages 1134-1142, 1984.