Title:

On the Learnability of Counting Functions

Authors:

Nader Bshouty, Zhixiang Chen, Scott E. Decatur, and Steven Homer

Affiliation:

Nader Bshouty
Department of Computer Science
University of Calgary
Calgary, Alberta
Canada T2N 1N4
Email: bshouty@cpsc.ucalgary.ca

Zhixiang Chen
Department of Computer Science
University of Texas – Pan American
Edinburg, TX 78539, USA
Email: chen@cs.panam.edu

Scott E. Decatur
Grantham, Mayo, Van Otterloo & Co.
1936 University Avenue, Suite 350
Berkeley, CA 94704
Email: Scott_Decatur@GMO.com

Steven Homer
Department of Computer Science
Boston University
Boston, MA 02215, USA
Email: homer@cs.bu.edu

Abstract

We examine the learnability of concepts based on *counting functions*. A counting function is a generalization of a parity function in which the weighted sum of n inputs is tested for equivalence to some value k modulo N. The concepts we study therefore generalize many commonly studied boolean functions.

We first show that disjunctions of counting functions (DOCFs) with modulus N are learnable by equivalence queries and determine the number of queries sufficient and necessary. We show that $\alpha(N)n + 1$ equivalence queries are sufficient where $\alpha(N) = O(\log N)$ is the sum of the exponents in the prime decomposition of N. When counting functions in the disjunction have distinct counting moduli $\{N_i\}$, we show that $\alpha(\mathtt{lcm}(N_i))n + 1$ equivalence queries are sufficient. We also give lower bounds on the number of equivalence queries required to learn *diagonal* DOCFs (and therefore general DOCFs) and provide a matching upper bound in the case of diagonal DOCFs.

We then 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.

1 Introduction

One of the central problems in machine learning is to study the learnability of boolean functions, in particular, DNF formulas, using equivalence and membership queries. Many results regarding this problem have been obtained and can be divided into three categories: (1) Many results show that subclasses of boolean functions that are learnable, such as monotone DNF formulas [1], read-twice DNF formulas [7], Horn DNF formulas [3], k-DNF formulas [29], $O(\log n)$-term DNF formulas [12], and disjoint DNF [16]. (2) Some results provide witnesses that the problem of learning boolean functions as DNF formulas is hard (see, for example, [4] and [6]). (3) In contrast to the hardness witness given by the results in the second category, the problem of learning boolean formulas using representations other than DNF formulas were studied in [28], [15] and [14]. It was shown in [15] that boolean functions are learnable as multivariate polynomials (XOR of terms). In [14], Bshouty proved that boolean functions are learnable in polynomial time in the DNF size and the CNF size of the target formula. One should note that here learnable in polynomial time as a class of certain type of functions means learnable in polynomial time in the number of variables and in the minimal size of the target function to be learned using functions in the class as hypotheses.

On the other hand, symmetric boolean functions, especially parity functions and modulo functions, have also received much attention in machine learning. It is known that the class of single parity functions (see [26]) and the class of single modulo functions with modulus p for any given prime number p (see [11]) are pac-learnable. It was also proved in [24] 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 $k \geq 2$. Meanwhile, it was shown in [13] that, for any constant k, boolean functions of k monomials are pac-learnable by the more expressive hypothesis class of general DNF formulas and, 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 model with queries, it was shown in [2] 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 [25] to a larger basis including arbitrary threshold functions and parity functions. Further, it was shown in [20] that read-once functions over the basis of arbitrary symmetric functions are polynomial time learnable with equivalence and membership queries. However, they also proved that read-twice functions over the same basis are not, under standard cryptographic assumptions.

In this paper, by introducing counting functions that includes 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 conjunctions of disjunctions of counting functions. This class in essence includes all boolean functions, and especially DNF formulas, as special cases. Our goal along this approach is to develop new techniques (say, from algebra) and thus to derive general results that imply learnability over boolean domains. A

careful reader may have noted that a disjunction of counting functions is equivalent to a system of linear equations over the domain Z_N^n.

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, [28] and [14]). Schapire and Sellie [28] 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 [14] extended DNF and CNF formulas to an arbitrary domain Σ^n for any finite field 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 [22] and [23], and many preliminary results have been obtained there.

This paper is organized as follows. In Section 2 we define counting functions and two parameters $\alpha(N)$ and $\gamma(N)$ for positive integer N. In Section 3 we introduce learning models. In Section 4 we give upper and lower bounds on the number of equivalence queries required for learning disjunctions of counting functions. We show that $\alpha(N)n+1$ equivalence queries are sufficient for learning disjunctions of counting functions, where $\alpha(N) = O(\log N)$ is the sum of the exponents in the prime decomposition of N. When counting functions in the disjunction have distinct counting moduli $\{N_i\}$, we show that $\alpha(\mathtt{lcm}(N_i))n + 1$ equivalence queries are sufficient for learning the disjunction. We also give lower bounds on the number of equivalence queries required to learn *diagonal* disjunctions of counting functions (and therefore general disjunctions of counting functions) and provide a matching upper bound in the case of diagonal disjunctions of counting functions. In Section 5, 6 and 7, we 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. In Section 5 we show that the class of diagonal disjunctions of counting functions and the class of boolean weighted read-once disjunctions of counting functions are learnable with fewer equivalence queries by using membership queries. For both classes the number of equivalence queries is linear in n and has no dependence on the counting modulus N. For the latter class the number of membership queries is also independent of N, while for the former class the number of membership queries has only a sub-logarithmic dependence on N. In Section 6, we show that disjunctions of no more than $O(\frac{\log n}{\log(P-1)})$ negated counting functions with prime modulus $P > 2$ over the domain Z_P^n are learnable with equivalence and membership queries. However, we also show that if the number of counting functions which may be negated is unbounded, then learning such disjunctions of counting functions is as hard as learning boolean DNF formulas. In Section 7, we study the learnability of conjunctions of disjunctions of counting functions (a generalization of boolean DNF/CNF). We first show that a conjunction of k disjunctions of counting functions with $k < \delta(N)$, where $\delta(N)$ is the minimal prime that divides N, is learnable with the number of queries (membership and equivalence queries) that is polynomial in k, the number of variables n, and $\frac{\delta(N)}{(\delta(N)-k)}$. In Section 8, we show that monotone conjunctions of diagonal disjunctions of counting functions are learnable. In Section 9, we list a number of open problems for further study.

2 Preliminaries

2.1 Counting Functions

We assume that Z is the set of all integers. Let $Z_q = \{0, \ldots, q-1\}$ for any integer $q \geq 2$, and $Z_q^n = \{0, \ldots, q-1\}^n$. Examples in Z_N^n are viewed as $1 \times n$ vectors. For any example $\vec{a} \in Z_N^n$, we use a_i to denote its i-th component for $i \in \{1, \ldots, n\}$. A counting function with a modulus $N \geq 2$ is defined (see [22, 23]) as follows:

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

where $\vec{a} = (a_1, \ldots, a_n) \in Z_N^n$ and $b \in Z_N$. For convenience, we may use $C_{\vec{a},b}^N$ to stand for $C_{\vec{a},b}^N(x_1, \ldots, x_n)$. A disjunction of counting functions (denoted by DOCF) $C_{\vec{a}_1,b_1}^N, \ldots, C_{\vec{a}_m,b_m}^N$ is

$$OR(C_{\vec{a}_1,b_1}^N, \ldots, C_{\vec{a}_m,b_m}^N) = C_{\vec{a}_1,b_1}^N \vee \cdots \vee C_{\vec{a}_m,b_m}^N.$$

We say that a counting function $C_{\vec{a},b}^N$ is *diagonal* if there is exactly one $i \in \{1, \ldots, n\}$ such that $a_i \neq 0$. We say that a DOCF is *diagonal* if all counting functions in it are diagonal. A conjunction F of DOCFs is defined as

$$F = \bigwedge_{i=1}^m L_i, \ \textit{where } L_i = OR(C_{\vec{a}_{i1},b_{i1}}^N, \ldots C_{\vec{a}_{im_i},b_{im_i}}^N).$$

An example $\vec{x} \in Z_N^n$ makes F zero (denoted by $F(\vec{x}) = 0$) if and only if it makes at least one L_i zero, i.e.,

$$\sum_{l=1}^n a_{ijl} x_l \equiv b_{ij} (mod\ N), \ \textit{for } j = 1, \ldots, m_i.$$

We call $\vec{x}$ a negative example for F when $F(\vec{x}) = 0$, and a positive example (denoted by $F(\vec{x}) = 1$) otherwise. Let $S(L_i)$ and $S(F)$ denote the sets of all examples in Z_N^n that make L_i and F zero, respectively. It is obvious that

$$S(F) = \bigcup_{i=1}^m S(L_i).$$

One may also consider redefining counting functions by switching 0 and 1 in the above definition. However, by duality, the representation capacity of disjunctions of conjunctions of such functions will be the same as that of conjunctions of disjunctions of counting functions in our definition.

2.2 Counting Functions via DNF Formulas

Given any DNF formula $F = T_1 \vee \cdots \vee T_m$ with terms

$$T_i = x_{i_1} \cdots x_{i_j} \overline{x}_{i_{j+1}} \cdots \overline{x}_{i_{m_i}},$$

For $l = 1, \ldots, m_i$, define $\vec{a}_{il}$ to be the vector with its i_l-th component 1 and all the other components 0. We also define b_{il} to be 1 for $l = 1, \ldots, j$, and b_{il} to be 0 for $l = j+1, \ldots, m_i$. Let L_i be the DOCF of $C^N_{\vec{a}_{i1},b_{i1}}, \ldots, C^N_{\vec{a}_{im_i},b_{im_i}}$, and L be the conjunction of L_i for $i = 1, \ldots, m$. It is easy to see that for any example $\vec{x} \in Z_2^n$,

$$F(\vec{x}) = 1 \Longleftrightarrow \vec{x} \in Z_2^n \cap S(L).$$

Thus, any DNF formula can be represented by a conjunction of DOCFs when restricted on the boolean domain Z_2^n.

2.3 Counting Functions via Modules

Let R be a ring with an identity element 1_R. A *R-module* is an additive abelian group A together with a function $R \times A \to A$ (the image of (r, a) being denoted by ra) such that for all $r, s \in R$ and $a, b \in A$:

1. $r(a + b) = ra + rb$.
2. $(r + s)a = ra + sa$.
3. $r(sa) = (rs)a$.
4. $1_R a = a$.

When R is a field, then A is called a *vector space*. Given a R-module A, a nonempty subset $B \subseteq A$ is called a *submodule* of A, if B is an additive subgroup of A and for all $r \in R$, $b \in B$, $rb \in B$. Recall that Z_N is a ring with the identity 1. When N is prime, Z_N is a field. Recall also that Z_N^n is an additive abelian group. We will only consider *Z_N-modules* (or *modules* for short) throughout this paper. The reader may refer [27] for more details about rings and modules.

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

$$Z_N b = \{ab \mid a \in Z_N\} \text{ and } Z_N \vec{r} = \{a\vec{r} \mid a \in Z_N\},$$

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

$$A + \vec{w} = \{\vec{a} + \vec{w} \mid \vec{a} \in A\}, \text{ and } A + B = \{\vec{a} + \vec{b} \mid \vec{a} \in A \text{ and } \vec{b} \in B\}.$$

Consider a DOCF $L = OR(C^N_{\vec{a}_1,b_1}, \ldots C^N_{\vec{a}_m,b_m})$. It is easy to see that for any $\vec{x} \in Z_N^n$, $\vec{x} \in S(L)$ if and only if $\vec{x}$ is a solution to the following system of linear equations over the domain Z_N^n

$$(*) \quad \begin{cases} \sum_{i=1}^n a_{1i}x_i & \equiv & b_1 \pmod N \\ \sum_{i=1}^n a_{2i}x_i & \equiv & b_2 \pmod N \\ & \ldots\ldots & \\ \sum_{i=1}^n a_{mi}x_i & \equiv & b_m \pmod N \end{cases}$$

Let $(**)$ be the following homogeneous system of $(*)$

$$(**) \quad \begin{cases} \sum_{i=1}^{n} a_{1i}x_i & \equiv & 0 \pmod N \\ \sum_{i=1}^{n} a_{2i}x_i & \equiv & 0 \pmod N \\ & \cdots\cdots & \\ \sum_{i=1}^{n} a_{mi}x_i & \equiv & 0 \pmod N \end{cases}$$

Then, one can verify that the set of all solutions over the domain Z_N^n to $(**)$ formed a module, that is denoted by $M(L)$. One can further verify that, given any $\vec{y} \in S(L)$, i.e., a solution to $(*)$, $S(L) = M(L) + \vec{y}$. In general, for any conjunction of DOCFs $F = L_1 \wedge \cdots \wedge L_m$, we have

$$S(F) = \bigcup_{i=1}^{m} (M(L_i) + \vec{y}_i), \ \textit{where } \vec{y}_i \in S(L_i).$$

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

We will use parameters $\alpha(N)$ and $\gamma(N)$ to describe query complexity of learning algorithms in later sections.

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

$$\alpha(N) = \sum_{i=1}^{t} r_i,$$

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

Lemma 2.4.1. *For any integer $N \geq 2$,*

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

Proof. The only nontrivial relationship is $\alpha(N) \leq \log N$ which follows by noting

$$N = p_1^{r_1} p_2^{r_2} \cdots p_t^{r_t} \geq 2^{r_1 + r_2 + \cdots + r_t} = 2^{\alpha(N)}.$$

□

3 Learning Models

In 1984, Valiant [29] proposed the pac-learning model and thus founded the modern computational learning theory. Later, Angluin [1] proposed the on-line learning with queries and then initiated the study of exact learning. According to Angluin [1] and

Blum [10], on-line learning with equivalence queries (and with membership queries) implies pac-learning (with membership queries), and there are cases such that on-line learning with equivalence queries is strictly harder than pac-learning.

In this paper, we will focus ourselves on the on-line learning with queries. Our first model is the on-line learning model with equivalence queries. The goal of a learning algorithm (or learner) for a class $\mathbf{C}$ of boolean-valued functions over a domain X^n is to learn any unknown target function $f \in \mathbf{C}$ 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 $\mathbf{C} \subseteq \mathbf{H}$ 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 $\vec{x} \in X^n$, called a counterexample, such that $h(\vec{x}) \neq f(\vec{x})$. x is called a positive counterexample if $f(\vec{x}) = 1$ and a negative counterexample otherwise. Each new hypothesis issued by the leaner may depend on the earlier hypotheses and the received counterexamples. A learning algorithm exactly learns $\mathbf{C}$, if for any target function $f \in \mathbf{C}$, it can find a $h \in \mathbf{H}$ that is logically equivalent to f. A learning algorithm exactly learns $\mathbf{C}$ with high probability, if for any target function $f \in \mathbf{C}$, it can infer a hypothesis $h \in \mathbf{H}$ 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 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 δ) and runs in time polynomially in the size of the domain, the size of the target function, and the size of the largest counterexample received during its learning process (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*".

4 Learning DOCFs

In this section we give upper and lower bounds on the number of equivalence queries required for learning DOCFs.

4.1 An Upper Bound for Learning DOCFs with Modulus N

For two integers a and b we write $a|b$ if b is divisible by a. For an element $a \in Z_N$, the set $Z_N a$ has the following properties:

Property 4.1.1. *$Z_N a = Z_N gcd(a, N)$.*

Property 4.1.2. *If $a|N$ then for every $b \in Z_N a$ we have $a|b$.*

Property 4.1.3. $Z_N a_1 + \cdots + Z_N a_m = Z_N \gcd(a_1, \ldots, a_m, N)$.

Property 4.1.4. *If $a|N$ and the elements of A and B are divisible by a then the elements of $A + B$ are divisible by a.*

Property 4.1.5. *We have $a \notin Z_N b$ 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

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

The integer m is called the *length* of the sequence. The length of the longest independent sequence in Z_N^n is denoted by $len(n, N)$. Notice that when N is prime then $len(n, N) = n$. The following lemmas can be easily verified:

Lemma 4.1.6. *If $(\vec{a}_1, \ldots, \vec{a}_m)$ is an independent sequence in Z_N^n then, for any $\lambda_1, \ldots, \lambda_{i-1} \in Z_N$,*

$$(\vec{a}_1, \ldots, \vec{a}_{i-1}, \vec{a}_i + \lambda_1 \vec{a}_1 + \cdots + \lambda_{i-1} \vec{a}_{i-1}, \vec{a}_{i+1}, \ldots, \vec{a}_m)$$

is an independent sequence in Z_N^n.

Lemma 4.1.7. Any subsequence (with the same order) of an independent sequence is an independent sequence.

We first prove some properties of the function $len(n, N)$, then give the learning algorithm, and finally prove the correctness of the algorithm.

Lemma 4.1.8. $len(1, N) = \alpha(N)$.

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

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

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

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

that is independent. By Property 4.1.3 we have

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

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

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

where $r_{0,k} = r_k$. By Property 4.1.5

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

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

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

Thus s can be at most $\alpha(N)$. □

Lemma 4.1.9. *$len(n, N) = \alpha(N)n$.*

proof. The proof is by induction on n. For $n = 1$ the previous lemma shows that $len(1, N) = \alpha(N)$. For any $n > 1$ we first prove that $len(n, N) \ge \alpha(N)n$. Since by the induction hypothesis $len(n-1, N) = \alpha(N)(n-1)$ there is an independent sequence $(\vec{a}_1, \ldots, \vec{a}_{len(n-1,N)})$ in the $(n-1)$-dimensional space. It can be easily shown that the following sequence is independent in the n-dimensional space and is of length $len(n, N) = \alpha(N)n$

$$((\vec{a}_1, 0), \ldots, (\vec{a}_{len(n-1,N)}, 0), (\vec{\mathbf{0}}_{n-1}, b_1), \ldots, (\vec{\mathbf{0}}_{n-1}, b_{\alpha(N)}))$$

where $\vec{\mathbf{0}}_{n-1}$ is the $(n-1)$-dimensional zero vector and $(b_1, \ldots, b_{\alpha(N)})$ is an independent sequence in Z_N.

We now show that $len(n, N) \le \alpha(N)n$. Suppose

$$(\vec{c}_1, \ldots, \vec{c}_t), \quad t > \alpha(N)n$$

is an independent sequence in Z_N^n. Let $\hat{\vec{c}}_j$ be the $(n-1)$-dimensional vector that is the first $n-1$ entries of $\vec{c}_j$ and $\vec{c}_j^n$ is the last entry (the n-th entry) of $\vec{c}_j$. By the induction hypothesis $(\hat{\vec{c}}_1, \ldots, \hat{\vec{c}}_t)$ is dependent and therefore let w be the minimal integer such that

$$\hat{\vec{c}}_w \in Z_N \hat{\vec{c}}_1 + \cdots + Z_N \hat{\vec{c}}_{w-1}.$$

Therefore $\hat{\vec{c}}_w = \lambda_1 \hat{\vec{c}}_1 + \cdots + \lambda_{w-1} \hat{\vec{c}}_{w-1}$. Now we change $\vec{c}_w$ in the sequence to $\vec{d}_w = \vec{c}_w - \lambda_1 \vec{c}_1 - \cdots - \lambda_{w-1} \vec{c}_{w-1}$ and by Lemma 4.1.6 we again get an independent sequence

$$(\vec{c}_1, \cdots, \vec{c}_{w-1}, \vec{d}_w, \vec{c}_{w+1}, \ldots, \vec{c}_t)$$

and notice that $\hat{\vec{d}}_w = \vec{\mathbf{0}}_{n-1}$. This can be done again and again as long as the number of $\hat{\vec{c}}_i$ that are not $\vec{\mathbf{0}}_{n-1}$ is greater than $len(n-1, N)$. At the end we have $s = t - \alpha(N)(n-1) > \alpha(N)$ vectors

$$\vec{d}_{w_1}, \ldots, \vec{d}_{w_s}, \quad w_1 < \cdots < w_s$$

in the sequence that are independent and that satisfy $\hat{\vec{d}}_{w_j} = \vec{\mathbf{0}}_{n-1}$. Therefore $\vec{d}^n_{w_1}, \ldots, \vec{d}^n_{w_s}$ must be independent. Since $s > \alpha(N)$ we have a contradiction to Lemma 4.1.8. □

We now describe the algorithm used to learn DOCFs. For a sequence $s = (\vec{a}_1, \ldots, \vec{a}_m)$ define the boolean function

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

For the empty sequence $s = ()$ define $f_{()}(\vec{x}) = 1$.

The Learning Algorithm

Step 1. $s \leftarrow ()$.

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

Theorem 4.1.10. *The above algorithm learns a DOCF with modulus N using*

$$\alpha(N)n + 1 \leq (\log N)n + 1$$

equivalence queries.

Proof. Let f be the target formula. Let $\vec{a}_1, \ldots, \vec{a}_m$ be the counterexamples in the algorithm and $s_i = (\vec{a}_1, \ldots, \vec{a}_i)$ and $f_i = f_{s_i}$. The zeros of a DOCF are the solutions of 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. We prove that for all $i \in \{1, \ldots, m\}$:

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

C2: $A(a_i)^T = B$.

C3: $f \Rightarrow f_i$.

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

For $i = 1$, $s_1 = ()$ and $f_{()} = 1$. Therefore $EQ(f_{()})$ gives negative counterexample $\vec{a}_1$ and $f(\vec{a}_1) = 0$. This proves condition (C1) for $i = 1$. Since $f(\vec{a}_1) = 0$ we have $A(\vec{a}_1)^T = B$ which proves (C2) for $i = 1$. Now since

$$f_{(\vec{a}_1)}(\vec{x}) = \begin{cases} 0 & if\ \vec{x} = \vec{a}_1 \\ 1 & \text{otherwise} \end{cases}$$

we get (C3) for $i = 1$. Condition (C4) follows because, by definition, () is an independent sequence.

Suppose (C1-C4) are true for $i-1$. Since $f \Rightarrow f_{i-1}$ the counterexample $\vec{a}_i$ will be negative. This implies (C1) for i. Since $\vec{a}_i$ is a negative counterexample $f(\vec{a}_i) = 0$ and $A(\vec{a}_i)^T = B$. This proves (C2) for i. Now

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

If $f_i(\vec{x}) = 0$ then $\vec{x} = \vec{a}_1 + \lambda_1(\vec{a}_2 - \vec{a}_1) + \cdots + \lambda_{i-1}(\vec{a}_i - \vec{a}_1)$ for some $\lambda_1, \ldots, \lambda_{i-1} \in Z_N$ and then by (C2)

$$\begin{aligned} A(\vec{x})^T &= A(\vec{a}_1)^T + \lambda_1 A(\vec{a}_2 - \vec{a}_1)^T + \cdots + \lambda_{i-1} A(\vec{a}_i - \vec{a}_1)^T \\ &= B. \end{aligned}$$

Therefore, $f(\vec{x}) = 0$. This implies $f \Rightarrow f_i$ and proves (C3) for i.

Since $\vec{a}_i$ is a negative counterexample for f_{i-1} we have

$$\vec{a}_i \notin \vec{a}_1 + Z_N(\vec{a}_2 - \vec{a}_1) + \cdots + Z_N(\vec{a}_{i-1} - \vec{a}_1)$$

and therefore

$$\vec{a}_i - \vec{a}_1 \notin Z_N(\vec{a}_2 - \vec{a}_1) + \cdots + Z_N(\vec{a}_{i-1} - \vec{a}_1).$$

This implies (C4) for i. Thus, we have shown that all the four conditions hold for all i.

Now since the longest independent sequence is

$$len(n, N) = \alpha(N)n$$

we have that the number of counterexamples can be at most

$$m \leq \alpha(N)n + 1.$$

This implies the theorem. □

We note that this algorithm works whether the components of the input vectors are either boolean or from Z_N.

4.2 An Upper Bound for Learning DOCFs with Different Moduli

In this section, we show that one can also learn a DOCF even if it has different counting moduli provided that the least common multiple of all moduli is known. Thus, we consider the problem of learning a DOCF of the following form:

$$f = \mathrm{OR}(C^{N_1}_{\vec{a}_1,b_1}, C^{N_2}_{\vec{a}_2,b_2}, \ldots, C^{N_m}_{\vec{a}_m,b_m})$$

when the learner knows the value $N = \mathtt{lcm}(N_1, N_2, \ldots, N_m)$. Note that for all $\vec{x}$, $f(\vec{x}) = 0$ if and only if all of the following equations hold:

$$\begin{aligned} \vec{a}_1 \cdot X &\equiv b_1 \pmod{N_1} \\ \vec{a}_2 \cdot X &\equiv b_2 \pmod{N_2} \\ &\vdots \\ \vec{a}_m \cdot X &\equiv b_m \pmod{N_m} \end{aligned} \tag{1}$$

where $X = (\vec{x})^T$. For each i, let integer $d_i = N/N_i$ and consider the new system of equations:

$$\begin{aligned} (d_1\vec{a}_1) \cdot X &\equiv d_1 b_1 \pmod{N} \\ (d_2\vec{a}_2) \cdot X &\equiv d_2 b_2 \pmod{N} \\ &\vdots \\ (d_m\vec{a}_m) \cdot X &\equiv d_m b_m \pmod{N} \end{aligned} \tag{2}$$

Note that for all $\vec{x}$, $\vec{x}$ is a solution to all equations in system (1) if and only if it is a solution to all equations in system (2). Thus the function which is 0 on the solutions to system (1) is identical to the function which is 0 on the solutions to system (2). It is therefore sufficient to learn the function $f' = f$:

$$f' = \mathrm{OR}(C^N_{d_1\vec{a}_1,d_1b_1}, C^N_{d_2\vec{a}_2,d_2b_2}, \ldots, C^N_{d_m\vec{a}_m,d_mb_m}).$$

But since the learner knows N, this is simply the problem of learning a DOCF with an identical modulus N. By Theorem 4.1.10, such a function is learnable with at most $\alpha(N)n + 1$ equivalence queries.

4.3 A Lower Bound for (Diagonal) DOCFs

We give a lower bound for diagonal DOCF which therefore holds for general DOCF.

Theorem 4.3.1. *To learn a DOCF from equivalence queries we need at least $\gamma(N)n$ equivalence queries.*

Proof. We give a lower bound for diagonal DOCFs which therefore holds for general DOCF. 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

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

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 $A = h^{-1}(0)$. Let

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

and let

$$L_2 = \{(x_1, \ldots, x_n) | x_s = 0 \bmod p_1^{q_1} \cdots p_j^{q_j} p_{j+1}^{\lfloor \frac{\delta_1+\delta_2}{2} \rfloor}\}.$$

If $A \not\subseteq L_1$ then the adversary can give the learner $x \in A \backslash L_1$ 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 $A \backslash (L_1 \cap L_2)$ and the learner gains only the additional information that

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

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

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

and the only information that the learner gains is that

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

Now by induction the result follows. □

4.4 Matching Upper Bounds for Diagonal DOCFs

Using a similar strategy as described in Theorem 4.3.1 for the learner we get Theorem 4.4.1 and Corollary 4.4.2.

Theorem 4.4.1. *Homogeneous diagonal DOCFs can be learned from $\gamma(N)n$ equivalence queries.*

Corollary 4.4.2. *(Non-homogeneous) diagonal DOCFs can be learned from $\gamma(N)n+1$ equivalence queries.*

5 Reducing the Number of Equivalence Queries by Asking Membership Queries

One can use membership queries to reduce the number of equivalence queries. Here we show how to reduce the number of equivalence queries using membership queries when learning certain restricted classes of DOCFs.

5.1 Reduced Queries for Diagonal DOCFs

We have shown in Section 4 that there is an algorithm for learning diagonal DOCFs using at most $\gamma(N)n + 1$ equivalence queries. On the other hand, we have also shown that any algorithm for learning diagonal DOCFs requires at least $\gamma(N)n$ equivalence

queries. We will design a new learning algorithm that substantially decrease the number of equivalence queries by using membership queries as well.

We first give some easy facts about the structures of submodules of Z_N.

Lemma 5.1.1 *For any submodule S of Z_N, there is an element $d \in Z_N$ such that $S = Z_N d$.*

Proof. If $S = \{0\}$, then $S = Z_N 0$. Now assume that $S \neq \{0\}$. Fix $d_1 \in S$ such that $d_1 \neq 0$. If $S = Z_N d_1$ then we are done. Otherwise, fix $t_1 \in S - Z_N d_1$. Set $d_2 = gcd(d_1, t_1, N)$. Then, $Z_N d_1 + Z_N t_1 = Z_N d_2$. $t_1 \notin Z_N d_1$ implies that $d_2 \notin Z_N d_1$. If $S = Z_N d_2$, then we are done, otherwise fix $t_2 \in S - Z_N d_2$. Let $d_3 = gcd(d_2, t_2, N)$. Then, $Z_N d_2 + Z_N t_2 = Z_N d_3$, and $d_3 \notin Z_N d_2$. Repeat the above procedure. Because Z_N contains N elements, the above procedure must terminate at an element d_m with $m \leq N - 1$. Hence, $S = Z_N d_m$. □

Given any $\vec{x} \in Z_N^n$, for $i = 1, \ldots, n$, let $\vec{x}[i]$ denote the example obtained by changing all its components into 0 except the i-th component.

Lemma 5.1.2. *For any diagonal DOCF F, there exist $d_i \in Z_N$, $i = 1, \ldots, n$, such that*

$$M(F) = (Z_N(d_1, \ldots, 0) + \cdots + Z_N(0, \ldots, d_n)),$$

where $M(F)$ is the set of all solutions over Z_N^n to the homogeneous system of linear equations derived from counting functions in F.

Proof. Define

$$M_i(F) = \{\vec{x}[i] | \vec{x} \in M(F)\}, \quad i = 1, \ldots, n.$$

Then, $M(F) = M_1(F) + \cdots + M_n(F)$. It is easy to see that $M_i(F)$ is a submodule of Z_N^n. This means that $\{x_i | \vec{x} \in M_i(F)\}$ is a submodule of Z_N. By Lemma 5.1.1, there exists $d_i \in Z_N$ such that $\{x_i | \vec{x} \in M_i(F)\} = Z_N d_i$. Thus, $M_i(F) = Z_N(0, \ldots, d_i, \ldots, 0)$. □

Theorem 5.1.3. *There is an algorithm for learning diagonal DOCFs with modulus N over the domain Z_N^n using $n + 1$ equivalence queries and $n\gamma(N)$ membership queries.*

Proof. Given a diagonal DOCF F, by Lemma 5.1.2, there are $d_i \in Z_N$, $i = 1, \ldots, n$, such that

$$S(F) = M(F) + \vec{y} = Z_N(d_1, \ldots, 0) + \cdots + Z_N(0, \ldots, d_n) + \vec{y}$$

for any $\vec{y} \in S(F)$. Thus, in order to learn $S(F)$, we only need to find an example $\vec{y} \in S(F)$ and the elements $d_1, \ldots, d_n$ in Z_N. Since $Z_N d_i = Z_N gcd(d_i, N)$, we assume without loss of generality that $d_i | N$. The learning algorithm issues hypotheses for $S(F)$ and works as follows.

Factor N to obtain $N = p_1^{r_1} p_2^{r_2} \cdots p_t^{r_t}$. Ask an equivalence query for $H^{-1}(0) = \phi$. If yes then stop, otherwise we obtain a counterexample denoted by $\vec{y}$.

Ask an equivalence query for $H^{-1}(0) = \{\vec{y}\}$. If yes then stop, otherwise we receive a counterexample $\vec{x}^1$. Then, $\vec{z}^1 = \vec{y} - \vec{x}^1 \in M(F)$, and $\vec{z}^1 \neq \vec{0}$. Thus, there is at least

one i_1 such that the i_1-th component $z^1_{i_1} \neq 0$ and $z^1_{i_1} \in Z_N d_{i_1}$. Factor $z^1_{i_1}$, let $z^1_{i_1} = a_1 p_1^{k_{i_1 1}} p_2^{k_{i_1 2}} \cdots p_t^{k_{i_1 t}}$, where $gcd(a_1, p_j) = 1$, for $j = 1, \ldots, t$. Suppose

$$d_{i_1} = p_1^{q_{i_1 1}} p_2^{q_{i_1 2}} \cdots p_t^{q_{i_1 t}}.$$

Then, $0 \leq q_{i_1 j} \leq k_{i_1 j} \leq r_j$. Now, for each $j \in \{1, \ldots, t\}$, use the binary search with membership queries to find the $q_{i_1 j}$ among integers $0, 1, \ldots, k_{i_1 j}$. Thus, with at most $\gamma(N)$ membership queries, we can find $q_{i_1 j}$, for $j = 1, \ldots, t$, hence we find d_{i_1}.

At next step, ask an equivalence query for $H^{-1}(0) = Z_N(0, \ldots, d_{i_1}, \ldots, 0) + \vec{y}$. If yes then stop, otherwise we receive a counterexample $\vec{x}^2$. Let $\vec{z}^2 = \vec{y} - \vec{x}^2$. Then, there is at least one $i_2 \neq i_1$ such that the i_2-th component $z^2_{i_2} \neq 0$ and it is in $Z_N d_{i_2}$. With the same manner, we can find d_{i_2} using at most $\gamma(N)$ membership queries. Next time, we ask an equivalence query for $H^{-1}(0) = Z_N(0, \ldots, d_{i_1}, \ldots, 0) + Z_N(0, \ldots, d_{i_2}, \ldots, 0) + \vec{y}$. Repeat the above procedure. Hence, we can find $d_1, \ldots, d_n$ (thus, $S(F)$) using at most $n + 1$ equivalence queries and at most $n\gamma(N)$ membership queries. □

5.2 Reduced Queries for Boolean Weighted Read-Once DOCFs

A read-once disjunction of boolean weighted counting functions is a DOCF in which all weight vectors $\vec{a}_i$ are boolean valued vectors and for any component, no two distinct vectors have a 1 in it. Although this class of functions can be learned with $\alpha(N)n + 1$ equivalence queries by the algorithm in Section 4.1, we show here that with the addition of n^2 membership queries, this restricted class can be learned using at most $2n$ equivalence queries. Thus, the number of queries is *independent* of the counting modulus N. This result holds for any input domain Z_M^n where M need not be the same as N. If the input domain is Z_N^n (or even Z_M^n for any $M > N$), then n^2 membership queries and *one* equivalence query and sufficient. Note that results of Bshouty, Hancock and Hellerstein [20] also show this class of functions to be learnable, but their algorithm uses $O(n^3)$ equivalence queries.

The strategy of our algorithm is to determine which variables are relevant, and for those which are, to determine which variables are in the same counting functions. Note that if two variables occur in the same counting function, this can be detected by making a membership query on an example obtained from a negative example by incrementing one of the two variables and decrementing the other. This detection works since if they are in the same counting function, then the sum of variables modulo N remains the same, and the example is still negative. If they are in different counting functions, then on this example both of these counting functions no longer equal their count values, and therefore the example is positive. Thus, by making $n(n-1)/2$ membership queries, each pair can be examined in this manner.

When the input domain is Z_M^n for $M \geq N$, then a single negative counterexample $\vec{y}$ can be used to create all required membership queries, since one can always increment one variable and decrement the other. This is nontrivial only when both variables are either 0 or both are $N-1$, and although one cannot increment and decrement in absolute terms, one can do so with modulo N operation while not affecting the sum of variables

modulo N. Specifically, for an example $\vec{y} \in Z_M^n$ let $\vec{y}^{i+}$ be the example identical to $\vec{y}$ except that its i-th component is incremented by 1 modulo N. Let $\vec{y}^{ij\pm}$ be the example identical to $\vec{y}$ except that its i-th component is incremented by 1 modulo N and its j-th component is decremented by 1 modulo N. One first asks an equivalence query with the always TRUE hypothesis in order to get a negative counterexample $\vec{y}$. Then the set of relevant variables can be determined by asking a membership query on $\vec{y}^{i+}$ for each $i \in \{1, \ldots, n\}$. The membership query response for $\vec{y}^{i+}$ is "yes" if and only if the i-th variable is relevant. For each pair of relevant variables x_i and x_j, a membership query is made with $\vec{y}^{ij\pm}$. These variables are in the same counting function if and only if the response to the membership query is "no"

If the input domain is Z_M^n for some $M < N$, then the algorithm becomes more complicated, since there may be cases in which $\vec{y}^{ij\pm}$ is not in Z_M^n. The learner must instead force the equivalence query oracle to provide counterexamples on which these increments and decrements can be made. Recall that we need only make one membership query for each pair of variables. Once this query is made, we definitively know whether or not the pair of variables is in the same counting function. We will therefore keep track of which pairs of variables have not yet been compared. After the first equivalence query, using counterexample $\vec{y}$, all relevant variables can be compared except for two classes: All variables assigned 0 in $\vec{y}$ cannot be compared against each other (although they can be compared against all others) and similarly all variables assigned $M-1$ in $\vec{y}$ cannot be compared against each other.

Let R be the set of variables whose counting functions have not been found after all possible comparisons of $\vec{y}$. Consider the subset of variables in R assigned $M-1$ in $\vec{y}$ and the subset of variables in R assigned 0 in $\vec{y}$. Let the set S contain these two subsets, unless either of these two sets is empty or contains only one variable, in which case that set is not placed in S. We evolve S according to the following invariants:

- The sets in S are disjoint subsets of R and each set in S has cardinality at least 2.
- For any set $s \in S$, and any variables $x_i \in s$ and $x_j \notin s$, x_i and x_j are known not to be in the same counting function.

This evolution of S is done in stages. At each stage we remove from S at least one set s and replace it by either two disjoint subsets of s, one proper subset of s, or no sets. Since $|R| \leq n$, there can be at most $n-1$ such stages. The variables in a set s removed from S which are not in sets reinserted to S are the variables of counting functions which were found at that stage. The sets that are reinserted to S denote sets of variables which have had the same assignment in every negative counterexample and have therefore not yet been compared.

Each set $s \in S$ has associated with it a nonempty set $t_s \subseteq \{0, M-1\}$. This set denotes the values v for which all elements of s were set to v in some negative counterexample seen so far. Therefore, when a set s splits, each of its children s' inherits the set t_s (*i.e.* $t_{s'} \supseteq t_s$).

We describe the hypothesis submitted to the equivalence query oracle as a general program, and note that such a hypothesis may be stated as a hypothesis from the class of "OR's of AND's of OR's of counting functions." Given S, we construct a hypothesis h which classifies an example as negative only if it is in *all* of the sets described below:

- For each counting function C_i found so far, let $N(C_i)$ be the set of examples which are negative for this counting function.
- For each set $s \in S$, let $E(s)$ be the set of examples for which there exists a $v \in t_s$ such that each of the examples assigns v to *all* variables in s.

Note that any example classified negative by h is a negative example of the target function. This is because such examples are negative examples of all counting functions found so far, and for the variables in the remaining counting functions, the example has the same values as in some previous negative example. Therefore, an equivalence query made with this hypothesis must either return "yes", in which case learning is complete, or else return a new *negative* counterexample $\vec{y}$. If a negative counterexample is returned, then since it is in all $N(C_i)$ sets by definition, it must not be in at least one of the $E(s)$ sets. Therefore, there exists an $s \in S$ such that either:

(1). The set s contains two variables assigned different values in $\vec{y}$, or

(2). The negative counterexample $\vec{y}$ causes t_s equal to $\{M-1\}$ or $\{0\}$ to be set to $\{0, M-1\}$.

Note that case (2) can occur at most n times since each variable in s inherits t_s whenever s is split and each variable can add an element to its t set only once. Whenever case (2) occurs, we create a new hypothesis (based on the new t_s sets), make another equivalence query, and get a new negative counterexample. This does not count as a new stage. Only when in case (1) do we move to the next stage.

Once in case (1), for each set s which contains two variables assigned different values, we remove s from S and perform all comparisons between variables in s with different values. The variables in the newly discovered counting functions will not be in sets reinserted to S. All remaining variables in s set to $M-1$ in $\vec{y}$ form a single new set s'. If there is only one variable in this set, a counting function is constructed using only this variable. Otherwise, if s' is nonempty, it is inserted into S and $t_{s'} = t_s \cup \{M-1\}$. Similar work can be done for all variables set to 0 in $\vec{y}$.

We ask one equivalence query to start the algorithm, n more equivalence queries may result in case (2) and $n-1$ more equivalence queries may result in case (1). Therefore, this algorithm makes at most $2n$ equivalence queries.

Putting all analysis together proves the following result.

Theorem 5.2.1. *When $M \geq N$, a boolean weighted read-once DOCF over Z_M^n is learnable using one equivalence query and at most n^2 membership queries. When $M < N$,*

a boolean weighted read-once DOCF over Z_M^n is learnable using at most $2n$ equivalence queries and at most n^2 membership queries.

6 Learning DOCFs with Negated Counting Functions

One open problem regarding learning DOCFs is whether disjunctions of negated counting functions with modulus N over the domain Z_N^n are poly-time learnable. Because this problem is substantially related to the problem of learning conjunctions of DOCFs with modulus N, it seems very difficult to resolve it. However, when $N = 2$, it is easy to see that a positive answer to the problem exists.

In this section, we will prove two results. First, we will provide a partial solution to the problem by showing that disjunctions of no more than $O(\frac{\log n}{\log(P-1)})$ negated counting functions with prime modulus $P > 2$ over the domain Z_P^n are poly-time learnable with equivalence and membership queries. Secondly, we will show that in general learning disjunctions of negated counting functions is harder than learning CNF formulas, i.e., if one can learn disjunctions of negated counting functions with modulus N over the domain Z_N^n in polynomial time using equivalence and membership queries, then one can learn CNF formulas in polynomial time using equivalence and membership queries.

When P is a constant prime, it was shown in [21] that disjunctions of negated counting functions with modulus P is polynomial time learnable using equivalence queries. The technique used in [21] is an extension of the linear transformation developed in [9].

6.1 A Positive Result for Bounded Negations

Assume that K is a field with Q elements. For any $m \times n$ matrix $A_{m,n} = (a_{ij})_{m,n}$ and $m \times 1$ vector $B = (b_1, \ldots, b_m)^T$ with elements from K, consider the following expression

$$\bigwedge_{i=1}^{m} (\sum_{j=1}^{n} a_{ij} x_j \neq b_i). \tag{6.1}$$

Let $N(A_{m,n}, B)$ denote the set of examples in K^n satisfying (6.1). By simple analysis, we know that (6.1) is equivalent to

$$\bigvee_{i=1}^{(Q-1)^m} A_{m,n} X = D_i, \tag{6.2}$$

where $D_i = (d_{i1}, \ldots, d_{im})^T$ are distinct and $d_{ij} \in (K - \{b_i\})$. In other words,

$$N(A_{m,n}, B) = \bigcup_{i=1}^{(Q-1)^m} S(A_{m,n}, D_i),$$

where $S(A_{m,n}, D_i)$ denotes the set of all examples in K^n satisfying the system of linear equations $A_{m,n} X = D_i$. Now, we prove the following lemma.

Lemma 6.1.1. *Given any two examples $\psi = (\psi_1, \ldots, \psi_n)^T$ and $\omega = (\omega_1, \ldots, \omega_2)^T$ in $N(A_{m,n}, B)$, define*

$$\xi(u) = u(\omega - \psi) + \psi, \quad \forall u \in K.$$

Then, there is an i such that ψ and ω are both in $S(A_{m,n}, D_i)$ if and only if $\xi(u) \in N(A_{m,n}, B)$, $\forall u \in K$.

Proof. Suppose that both ψ and ω are in $S(A_{m,n}, D_i)$ for some i. Then, for any $u \in K$, $\xi(u) \in S(A_{m,n}, D_i)$, because

$$A_{m,n}\xi(u) = u(A_{m,n}\omega - A_{m,n}\psi) + A_{m,n}\psi = u(D_i - D_i) + D_i = D_i.$$

Now assume that $\xi(u) \in N(A_{m,n}, B)$ for any $u \in K$. Suppose by contradiction that there are i and j with $i \neq j$ such that $\psi \in S(A_{m,n}, D_i)$ and $\omega \in S(A_{m,n}, D_j)$. Since $D_i \neq D_j$, there is at least one e such that

$$\sum_{r=1}^{n} a_{er}\psi_r = d_{ie} \neq d_{je} = \sum_{r=1}^{n} a_{er}\omega_r.$$

Since K is a field, $(d_{je} - d_{ie})^{-1}$ exists. By (6.1)

$$\sum_{r=1}^{n} a_{er}\psi_r = d_{ie} \neq b_e.$$

We now choose $u = (b_e - d_{ie})(d_{je} - d_{ie})^{-1}$. Then, $u \in K$ and $u \neq 0$. Note that

$$A_{m,n}\xi(u) = u(A_{m,n}\omega - A_{m,n}\psi) + A_{m,n}\psi = u(D_j - D_i) + D_i.$$

Thus, the e-th element of the above vector is

$$u(d_{je} - d_{ie}) + d_{ie} = (b_e - d_{ie})(d_{je} - d_{ie})^{-1}(d_{je} - d_{ie}) + d_{ie} = b_e.$$

This means that

$$\sum_{r=1}^{n} a_{er}\ \xi(u)_r = b_e,$$

where $\xi(u)_r$ is the r-th element of $\xi(u)$. Hence, $\xi(u)$ doesn't satisfy (6.1), this contradicts to the assumption that $\xi(u) \in N(A_{m,n}, B)$. □

Remark 6.1.2. *By Lemma 6.1.1, for any given two examples satisfying* (6.1)*, we can use at most Q membership queries to decide whether they satisfy the same linear system derived from* (6.2)*. It is also easy to see that for any D_i and D_j,*

$$S(A_{m,n}, D_i) \bigcap S(A_{m,n}, D_j) = \phi.$$

Remark 6.1.3. *Assume that ψ and ω are in $S(A_{m,n}, D_i)$. Then, both $\psi - \omega$ and $\omega - \psi$ are in $S(A_{m,n}, \vec{0})$.*

Theorem 6.1.4. *Given any $m \times n$ matrix $A_{m,n}$ and any $m \times 1$ vector B with their elements in K, one can learn $N(A_{m,n}, B)$ over the domain K^n using at most $n+(Q-1)^m$ equivalence queries and at most $Q(n+(Q-1)^m)(Q-1)^m$ membership queries. Moreover, if addition and multiplication operations of elements in K are of poly-time complexity, then the learning process runs in time polynomial in $\log Q$, n, and $(Q-1)^m$.*

Proof. By (6.2), in order to learn $N(A_{m,n}, B)$, one only needs to learn all the linear systems $A_{m,n}X = D_i$, i.e., $S(A_{m,n}, D_i)$, for $i = 1, \ldots, (Q-1)^m$. We will design a learning algorithm running in stages. Define $W(s)$ to be the set of all linear systems constructed by the end of stage s. Define $Z(s)$, which is constructed by the end of stage s, to be the set of all linearly independent solutions to the homogeneous system $A_{m,n}X = \vec{0}$. Define $R(s)$ to be the set of all examples obtained by the end of stage s that are solutions to different linear systems contained in (6.2). Each linear system L in $W(s)$ is uniquely defined by $Z(s)$ and one element in $R(s)$ in the sense that the set of all solutions to L is exactly the following

$$sol(L) = lspan(Z(s), \mu_L) = \{k_1\psi_1 + \cdots + k_l\psi_l + \mu_L | k_i \in K\},$$

where $Z(s) = \{\psi_1, \ldots, \psi_l\}$, and $\mu_L \in R(s)$ is a solution to the system L. In particular, when $Z(s) = \phi$, $sol(L) = \{\mu_L\}$. The hypothesis $H(s)$ issued at stage s is the disjunction of all linear systems in $W(s)$. In other words, $H(s)$ is the union of all solutions to all the linear systems in $W(s)$. Let $\omega(s)$ denote the counterexample received for the hypothesis $H(s)$.

Learning Algorithm.

Stage 0. Set $W(0) = Z(0) = R(0) = H(0) = \phi$. Ask an equivalence query for $H(0)$. If yes then stop, otherwise one receives a counterexample ω_0.

Stage $s+1 \geq 1$. For each solution $\mu \in R(s)$, decide whether $\omega(s)$ and μ are solutions to the same linear system in (6.2). By Lemma 6.1.1, this can be done with at most Q membership queries. If they are, then it is easy to see that $\omega(s) - \mu$ is linearly independent from all solutions in $Z(s)$ and $\omega(s) - \mu$ is a solution to the homogeneous system $A_{m,n}X = \vec{0}$. Thus, set $Z(s+1) = Z(s) \cup \{\omega(s) - \mu\}$, and $R(s+1) = R(s)$. If for any $\mu \in R(s)$, $\omega(s)$ and μ are not solutions to the same linear system in (6.2), then set $R(s+1) = R(s) \cup \{\omega(s)\}$, and $Z(s+1) = Z(s)$.

Ask an equivalence query for $H(s)$. If yes then stop, otherwise one receives a new counterexample $\omega(s+1)$.

According to (6.2), $|R(s)| \leq (Q-1)^m, \forall s \geq 0$. Since the homogeneous system $A_{m,n}X = \vec{0}$ has at most n linearly independent solutions, $|Z(s)| \leq n, \forall s \geq 0$. Hence, at most $n + (Q-1)^m$ equivalence queries are required. For each counterexample received at stage s, we need to decide whether there is one solution in $R(s)$ such that they are solutions to the same linear system in (6.2). By Lemma 6.1, at most $Q(n + (Q-1)^m)(Q-1)^m$ membership queries are required. In the algorithm, only addition and multiplication operations of elements in K are involved. So, the algorithm runs in

time polynomial in $\log Q$, n, and $(Q-1)^m$, if addition and multiplication operations of elements in K are of poly-time complexity. □

We now consider how to apply Theorem 6.1.4 to learn a disjunction of negated counting functions with a prime modulus $P > 2$. Note that Z_P is a field with P elements, and addition and multiplication operations of elements in Z_P are of poly-time complexity. Given a disjunction F of negated counting functions

$$F = OR(\neg C^P_{\vec{a}_1,b_1}, \ldots, \neg C^P_{\vec{a}_m,b_m})$$

over the domain Z_P^n, It is easy to see that F is equivalent to (6.1) in the sense that, for any example $\psi \in Z_P^n$, $F(\psi) = 0$ if and only if ψ satisfies (6.1). So, in order to learn F, we only need to learn (6.1). Hence, the following corollary follows directly from Theorem 6.1.4.

Corollary 6.1.5. *We can learn a disjunction of m negated counting functions with prime modulus $P > 2$ over the domain Z_P^n using at most $n + (P-1)^m$ equivalence queries and at most $P(n+(P-1)^m)(P-1)^m$ membership queries. The time complexity of the learning algorithm is polynomial in $\log P$, n and $(P-1)^m$. (Hence, when m is at most $O(\frac{\log n}{\log(P-1)})$, the algorithm is of polynomial time complexity.)*

6.2 The Hardness Result for Unbounded Negations

Given a CNF formula

$$F = \bigwedge_{i=1}^{m} T_i, \text{ where } T_i = (x_{i_1} \vee \cdots \vee x_{i_{m_1}} \vee \bar{x}_{i_{m_1+1}} \vee \cdots \vee \bar{x}_{i_{m_2}}). \tag{6.3}$$

We assume without loss of generality that at each clause T_i, any variable can appear at most once, and no variable and its negation can appear together. For T_i, we define a vector $\vec{a}(i) = (a_{i1}, \ldots, a_{in})$ such that $a_{ij} = 1$ if x_j appears at T_i, $a_{ij} = n$ if $\bar{x}_j$ appears and, $a_{ij} = 0$ otherwise. We also define b_i as the difference of $(n+1)$ and the number of negated variables appearing at T_i. Now, we define

$$D(F) = OR(\neg C^{n+1}_{\vec{a}(1),b_1}, \ldots, \neg C^{n+1}_{\vec{a}(m),b_m}).$$

Then, $D(F)$ is a disjunction of negated counting functions.

Lemma 6.2.1. *The size of F is of the same order as the size of $D(F)$. For any example $\alpha = (\alpha_1, \ldots, \alpha_n) \in Z_2^n$, $F(\alpha) = 1$ if and only if $D(F)(\alpha) = 0$.*

Proof. The number of clauses in F is the same as the number of counting functions in $D(F)$, and the number of variables occurring at each clause of F is the same as the number of nonzero weights in the corresponding counting functions in $D(F)$. So, the size of F is of the same order as the size of $D(F)$.

$F(\alpha) = 1$ if and only if $T(i)(\alpha) = 1$, for $i = 1, \ldots, m$. $T(i)(\alpha) = 1$ if and only $\sum_{j=1}^{n} a_{ij}\alpha_j \not\equiv b_i \pmod{n+1}$, if and only if $\neg C^{n+1}_{\vec{a}(i),b_i}(\alpha) = 0$. Hence, $F(\alpha) = 1$ if and only if $D(F)(\alpha) = 0$. □

Theorem 6.2.2. *If one can learn disjunctions of negated counting functions with modulus N over the domain Z_N^n in time $poly(n, m', \log N)$ using equivalence and membership queries, then one can CNF formulas in time $poly(n, m'')$ using equivalence and membership queries, where m' and m'' are the sizes of the input disjunctions of negated counting functions and the input CNF formulas, respectively.*

Proof. Given any CNF formula F with n variables. Suppose the size of F is m''. Convert F to $D(F)$. If there is an algorithm for learning disjunctions of negated counting functions with modulus N over the domain Z_N^n in time $poly(n, m', \log N)$ using equivalence and membership queries, then we can use it to learn $D(F)$ (and hence F according to Lemma 6.2.1). The time complexity for learning $D(F)$ is $poly(n, m'', \log(n+1)) = poly(n, m'')$, because the size of $D(F)$ is of the same order as m'' and the modulus of $D(F)$ is $(n+1)$. □

7 Learning Conjunctions of DOCFs

In this section we show that if the target is a conjunction of k DOCFs and $k < \delta(N)$ where $\delta(N)$ is the minimal prime that divides N then the target can be learned with the number of queries (membership and equivalence queries) that is polynomial in k, the number of variables n and $\delta(N)/(\delta(N) - k)$. We then show that if the class of conjunction of k DOCFs over Z_N is learnable then the class of conjunction of k DOCFs over any $Z_{N'}$ where $N'|N$ is learnable. This justifies the use of $\delta(N)$ in the condition $k < \delta(N)$ and in the query complexity of the algorithm. It also shows that the learnability of conjunctions of DOCFs over Z_N when N is even implies the learnability of boolean DNF.

We have noticed before that learning one DOCF is equivalent to learning the set

$$L = \{x \in Z_N^n \mid Ax = b\}$$

where A is an $m \times n$ matrix over Z_N, $x = (x_1, \ldots, x_n)^T$ and $b = (b_1, \ldots, b_m)^T \in Z_N^m$. This is because the set of negative examples of any DOCF is of the above form. Therefore, learning a conjunction of k DOCFs is equivalent to learning a set

$$W = L_1 \cup L_2 \cup \cdots \cup L_k$$

where

$$L_i = \{x \in Z_N^n \mid A^{(i)}x = b^{(i)}\} \quad i = 1, \ldots, k.$$

To describe our algorithm we give the following definitions. Given examples $s_1, \ldots, s_m \in W$ we write $\sim (s_1, \ldots, s_m)$ if $s_1, \ldots, s_m$ is in the same L_j for some $j = 1, \ldots, k$. For a set S of examples, we write $\sim S$ is the $\sim$ relation holds for all its examples. We have the following lemma.

Lemma 7.1. *Let $s_1, \ldots, s_k \in W$. If $\not\sim (s_1, \ldots, s_k)$ then*

$$\Pr_{\lambda_2, \ldots, \lambda_m}[s_1 + \lambda_2(s_2 - s_1) + \cdots + \lambda_m(s_m - s_1) \in W] \leq \frac{k}{\delta(N)}.$$

If $\sim(s_1,\ldots,s_k)$ *then*

$$\Pr_{\lambda_2,\ldots,\lambda_m}[s_1+\lambda_2(s_2-s_1)+\cdots+\lambda_m(s_m-s_1)\in W]=1.$$

The proof of this lemma is given at the end of this section. We now show how to use this lemma to learn conjunction of k DOCFs when $k<\delta(N)$.

This lemma gives a randomized algorithm that runs with query complexity

$$O\left(\frac{\delta(N)}{\delta(N)-k}\right),$$

for $\delta(N)>k$ to decide whether $\sim(s_1,\ldots,s_m)$ using a randomized membership queries.

The learning algorithm collects counterexamples in sets $S_1,S_2,\ldots$ where each S_i satisfies $\sim S_i$. When the algorithm receives a new counterexample s_i it uses the test in the above lemma to know to which set it belongs. The learning algorithm then adds it to the the appropriate set and asks equivalence query with the new hypothesis.

In the following learning algorithm the first hypothesis is Ø (that corresponds to the constant boolean function 1). For a set $S=\{s_1,\ldots,s_k\}$ we have

$$L(S)=\{s_1+\lambda_2(s_2-s_1)+\ldots+\lambda_k(s_k-s_1)|\lambda_2,\ldots,\lambda_k\in Z_N\}.$$

Algorithm.

1. $j\leftarrow 0$.

2. Ask $EQ(L(S_1)\cup L(S_2)\cup\cdots\cup L(S_j)))\rightarrow s$.

3. For the negative counterexample s and for each S_i, if $\sim(\{s\}\cup S_i)$ then add s to S_j.

4. If for no i we have $\sim(\{s\}\cup S_i)$ do

4.1. $j\leftarrow j+1$.

4.2. $S_j\leftarrow\{s\}$.

5. Goto 2.

The correctness of the algorithm is clear. The complexity of the algorithm is at most kn equivalence queries and

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

membership queries.

We now prove Lemma 7.1. We first give the following proposition.

Proposition 7.2. *Let*

$$L=\{x|Ax=b\},\quad L'=\{x|A'x=b'\}$$

where A and A' are $m \times n$ and $m' \times n'$ matrices over Z_n and $b, b' \in Z_N^m$s. Suppose $\emptyset \neq L \not\subseteq L'$. Then

$$\Pr_x[x \in L' | x \in L] \leq \frac{1}{\delta(N)}.$$

Proof. If $L \cap L' = \emptyset$ then

$$\Pr_x[x \in L' | x \in L] = 0 \leq \frac{1}{\delta(n)}.$$

Let $x^{(1)} \in L \cap L'$. Since $L \not\subseteq L'$ there is $x^{(2)} \in L' \backslash L$. Notice that

$$A'x^{(1)} = b', \quad A'x^{(2)} = b', \quad Ax^{(1)} = b, \quad Ax^{(2)} \neq b.$$

let $y = x^{(1)} - x^{(2)}$ and consider the following relation on L'. For $w_1, w_2 \in L'$

$$w_1 \ R \ w_2 \ \text{ if and only if } \quad w_1 - w_2 = \lambda y \quad \text{for some} \quad \lambda \in Z_N.$$

It is easy to verify that this relation is an equivalence relation. Notice also that for any w the equivalence class of w is

$$[w] = \{w, w + y, w + 2y, \ldots, w + (M-1)y\}$$

where M is the minimal integer that satisfies $My = 0$. This is because for any $\delta \in Z_N$ we have $A'(w + \lambda y) = b'$ and therefore $w + \lambda y \in L'$. Notice also that this M divides N because y is not 0. Therefore, all equivalence classes are of the same size M. We now show that in each equivalence class $[w]$ we either have $T < M$ elements in L' or none of the elements are in L' and $T|M$. If this is true then because all equivalence classes have the same size this implies that the probability that $x \in L$ is also in L' is at most the probability that $x \in [w]$ is also in L' in those $[w]$ that contains T elements in L'. Therefore,

$$\Pr_x[x \in L' | x \in L] \leq \frac{T}{M} \leq \frac{1}{\delta(N)}.$$

It remains to show that for any $w \in L'$ we have that $|[w] \cap L|$ is either 0 or some $T < M$ where $T|M$. Suppose $|[w] \cap L| \neq 0$. Let $u \in [w] \cap L$. Then $[u] = [w]$ and $u \in L$. Consider

$$u, u + y, u + 2y, \ldots, u + (M-1)y.$$

We have

$$A'(u + \lambda y) = A'u = b'$$

and

$$A(u + \lambda y) = b + A(\lambda y) = b + \lambda(b - Ax^{(2)}).$$

Therefore, $u + \lambda y \in L'$ but $u + \lambda y \in L$ if and only if $\lambda z = 0$ where $z = b - Ax^{(2)} \neq 0$. Now

$$|[w] \cap L| = |\{\lambda | \lambda z = 0\}| = T|N$$

and since $z \neq 0$ we have $u + y \notin L$ and therefore $T < M$. Also if Q is the smallest integer that satisfies $Qz = 0$ then $T = M/Q$ of the elements in L' are in L. Therefore $T|M$.□

8 Learning Monotone Conjunctions of Diagonal DOCFs

Angluin [1] showed that any monotone DNF formula, in which each term contained no negated variables, is polynomial time learnable using equivalence and membership queries. In [5] it was further shown that monotone DNF formulas are also polynomial time learnable with high probability using equivalence and incomplete membership queries. Bshouty [14] systematically studied the problem of learning those concept classes that can be represented as conjunctions (or disjunctions) of monotone concepts in general sense, and he obtained many remarkable results by means of monotone theory. For example, it was shown that any boolean function is learnable as decision tree.

A monotone concept is uniquely determined by one element (or example) based on a partially ordered relation. This relatively simple structure limits the representation capacity of monotone concepts. There are natural concepts classes that can not be represented as disjunctions (or conjunctions) of monotone concepts, for example, unions of axis parallel discretized rectangles (as noted in [14]), unions of discretized polytopes, and unions of vector spaces (or modules) over a finite field (or ring). Thus, in order to learn those concepts, new techniques are required. As a first step along this line, we will study the learnability of monotone conjunctions of diagonal DOCFs over the domain Z_N^n, which are direct generalizations of monotone DNF formulas and, in essence, can be represented as unions of modules over the ring Z_N^n.

For any conjunction F of DOCFs over the domain Z_N^n, $N \geq 2$,

$$F = \bigwedge_{i=1}^{m} L_i, \quad where$$

$$L_i = \bigvee_{j=1}^{m_i} a_{ij} x_{i_j} \equiv b_{ij} (mod\ N).$$

We say that F is monotone if, (1) for any $i \in \{1, ..., m\}$ and $j \in \{1, ..., m_i\}$, $a_{ij} \neq 0$, $b_{ij} \neq 0$ and, (2) for any $i, k \in \{1, ..., m\}$, the set $\{x_{i_1}, ..., x_{i_{m_i}}\}$ of relevant variables of L_i is different from the set $\{x_{k_1}, ..., x_{k_{m_k}}\}$ of the relevant variables of L_k. We also say that L_i is a term of F. For simplicity, we may also use L to stand for L_i.

Let $S(L)$ denote the set of all examples in Z_N^n that make the term L zero, and let $S(F)$ denote the set of all examples in Z_N^n that make F zero. In order to learn F, we only need to learn

$$S(F) = \bigcup_{i=1}^{m} S(L_i)$$

Furthermore, we assume without loss of generality that every relevant variable of any term of F appears exactly once in the term.

For any example $\vec{x} \in Z_N^n$, by flipping its i-th component we mean that we change its i-th component $x_i \neq 0$ to 0. If $x_i = 0$, then we will not flip it. Given any example $\vec{x}$ such that $F(\vec{x}) = 0$, let $R(\vec{x})$ be the example obtained by flipping all those components in $\vec{x}$ such that the obtained example still makes F zero. Let $V(\vec{x})$ denote the set of all the variables x_i such that the i-th component of $R(\vec{x})$ is not zero.

Lemma 8.1. *Given any two negative examples $\vec{x}_i$ and $\vec{x}_j$ for F, if $V(\vec{x}_i) = V(\vec{x}_j)$, then $R(\vec{x}_i)$ and $R(\vec{x}_j)$ make the same term of F zero.*

Proof. Suppose that $R(\vec{x}_i)$ and $R(\vec{x}_j)$ make respectively two distinct terms L_r and L_t zero. Because F is monotone, the two sets of relevant variables of L_r and L_t are different. Say, for example, L_r has a relevant variable x_1, but L_t does not. Since L_t does not have x_1, flipping the value of x_1 in any example making L_t zero will result in an example that still makes L_t (and hence F) zero, thus x_1 is not in $V(\vec{x}_j)$.

Since L_r has the relevant variable x_1, let the weight of x_1 in L_r is a_{r1}, then we have

$$a_{r1}x_1 \equiv b_{r1}(mod\ N).$$

Because F is monotone, b_{r1} is not 0, this implies that in order to satisfy the above linear equation the value of x_1 must not be 0. Hence, x_1 must be in $V(\vec{x}_i)$. Since x_1 is not in $V(\vec{x}_j)$ according to the analysis in the above paragraph, we have a contradiction to that fact that $V(\vec{x}_i) = V(\vec{x}_j)$. Therefore, $R(\vec{x}_i)$ and $R(\vec{x}_j)$ make the same term of F zero. □

Lemma 8.2. *Given any negative example $\vec{x}$ for F, if $R(\vec{x})$ makes a term L_i of F zero, then $\vec{x}$ also makes L_i zero.*

Proof. Since $R(\vec{x})$ makes L_i, $R(\vec{x})$ is a solution to the following linear system

$$a_{ij}x_{i_j} \equiv b_{ij}(mod\ N),\ j = 1, \ldots, m_i.$$

Because F is monotone, b_{ij} is not 0. This implies that none of x_{i_j} in $\vec{x}$ have been flipped in the process to get $R(\vec{x})$. Hence, $\vec{x}$ is still a solution to the above linear system, i.e., it still makes L_i zero. □

Theorem 8.3. *There is an algorithm for learning a monotone conjunction F of m diagonal DOCFs over the domain Z_N^n using $m(n\alpha(N) + 1)$ equivalence queries and $mn(n\alpha(N) + 1)$ membership queries.*

Proof. The learner will learn $S(F)$, i.e., the set of all examples making F zero. The learning algorithm LM works in stages. At any stage s, we use $W(s)$ to denote the class of the sets of relevant variables that we have learned by the end of stage s. We also use $E(s)$ to denote the set of all the examples $\vec{x}$ received by the end of stage s. For each set of the relevant variables $\psi \in W(s)$, let $[\psi]$ be the set of all examples $\vec{x} \in E(s)$ such that $V(\vec{x}) = \psi$. For $[\psi] = \{\vec{x}_{i_1}, \ldots, \vec{x}_{i_k}\}$ with $i_1 < \cdots < i_k$, define $H(s, \psi) = \vec{y}_1 + Z_N(\vec{y}_2 - \vec{y}_1) + \cdots + Z_N(\vec{y}_k - \vec{y}_1)$, where $\vec{y}_j = R(\vec{x}_{i_j})$. Define

$$\tilde{H}(s, \psi) = \{\vec{x} \in Z_N^n | \exists \vec{y} \in H(s, \psi)\ such\ that\ \forall x_i \in \psi,\ x_i = y_i\}.$$

The idea for us to use $H(s, \psi)$ is to collect all solutions to a linear system derived from a term L of F. For any example in $H(s, \psi)$, its i-th component is 0 if x_i is not a relevant variable of L. $\tilde{H}(s, \psi)$ is an expansion of $H(s, \psi)$ by filling the i-th component of any example in $H(s, \psi)$ with arbitrary value, where x_i is not a relevant variable of L.

The hypothesis issued by the learner at stage s is

$$H_s = \bigcup\{\tilde{H}(s,\psi)|\psi \in W(s)\}.$$

Algorithm LM:

1. Stage 0. Set $H_0 = W(0) = E(0) = \phi$.
2. Stage $s+1 \geq 1$. Ask an equivalence query for the hypothesis H_s. If yes then stop. Otherwise the learner receives a counterexample $\vec{x}_s$. Find $R(\vec{x}_s)$ by asking at most n membership queries. Set $E(s+1) = E(s) \cup \{\vec{x}_s\}$. If $V(\vec{x}) \notin W(s)$, then set $W(s+1) = W(s) \cup \{V(\vec{x})\}$, otherwise set $W(s+1) = W(s)$.

We now show the following claim about algorithm LM.

Claim 8.4. *For any $s \geq 0$, the following holds.*

(1). *$\vec{x}_s$ is a negative counterexample.*

(2). *Given any $\psi \in W(s)$, for any $\vec{x}_i, \vec{x}_j \in [\psi]$, $R(\vec{x}_i)$ and $R(\vec{x}_j)$ make the same term L zero and $H(s,\psi) \subseteq S(L)$.*

(3). *Assume that $V(\vec{x}_s) = \psi = V(\vec{x}_i) \in W_s$ and $R(\vec{x}_i)$ make the term L zero. Then, $R(\vec{x}_s) \in S(L) - H(s,\psi)$.*

(4). *$H_{s+1} \subseteq S(F)$.*

Proof of Claim 8.4. By induction on s. When $s = 0$, $H_0 = \phi$. Thus, $\vec{x}_0 \notin H_0$ implies $\vec{x}_0 \in S(F)$, i.e., $F(\vec{x}_0) = 0$. So, (1) is true. (2) and (3) are trivially true, because $W(0) = E(0) = \phi$. Since $W(1) = \{V(\vec{x}_0)\}$ and $[V(x_0)] = \{\vec{x}_0\}$, $H_1 = \{R(\vec{x}_0)\}$. By (1) and the definition of $R(\vec{x}_0)$, $H_1 \subseteq S(F)$, hence (4) is true.

Assume that the above claim is true for the case of s. We now consider the case of $s+1$. By (4) in the case of s, $\vec{x}_{s+1} \in S(F) - H_{s+1}$, so (1) is true in the case of $s+1$.

Given any $\psi \in W_{s+1}$, for any $\vec{x}_i, \vec{x}_j \in [\psi]$, we have $V(\vec{x}_i) = V(\vec{x}_j)$. By Lemma 8.1, $R(\vec{x}_i)$ and $R(\vec{x}_j)$ make the same term L zero. This implies that $H(s+1,\psi) \subseteq S(L)$, so (2) is true in the case of $s+1$.

Assume the condition of (3) is true, i.e., $V(\vec{x}_{s+1}) = V(\vec{x}_i)$, and $R(\vec{x}_i)$ makes the term L zero. Then, by Lemma 8.1, $R(\vec{x}_{s+1}) \in S(L)$. Thus, $\vec{x}_{s+1} \in S(L)$ by Lemma 8.2. Note that $\vec{x}_{s+1}$ and $R(\vec{x}_{s+1})$ differs at only those components i such that x_i are not relevant variables of L. If $R(\vec{x}_{s+1}) \in H(s+1,\psi)$, then $\vec{x}_{s+1} \in \tilde{H}(s+1,\psi)$, a contradiction to the fact that $\vec{x}_{s+1}$ is a negative counterexample. Hence, (3) is true in the case of $s+1$.

By (2), for any $\psi \in W(s+1)$, $H(s+1,\psi) \subseteq S(L)$ for a term L of F. According to the definition of $\tilde{H}(s+1,\psi)$, $\tilde{H}(s+1,\psi) \subseteq S(L)$. Hence, $H_{s+1} \subseteq S(F)$. □

For any $\psi \in W_s$, let $[\psi] = \{\vec{x}_{i_1}, \ldots, \vec{x}_{i_l}\}$. Then, by (2) and (3) of Claim 8.4, $(R(\vec{x}_{i_1}), R(\vec{x}_{i_2}) - R(\vec{x}_{i_1}), \ldots, R(\vec{x}_{i_l}) - R(\vec{x}_{i_1}))$ is an independent sequence over Z_N^n.

Again, according to (3) of Claim 8.4, whenever the learner receives a counterexample, we either start to construct a new independent sequence, or add one more element to an existing sequence. We have already known from Lemma 4.1.9 that any independent sequence over Z_N^n contains at most $n\alpha(N)$ elements. Hence, the total number of equivalence queries required to learn F is at most $m(n\alpha(N)+1)$, since F contains at most m terms. For each counterexample $\vec{x}_s$, the learner needs at most n membership queries to find $R(\vec{x}_s)$, hence the total number of membership queries required is at most $mn(n\alpha(N)+1)$. □

9 Open Problems

We list some open problems.

1. We have shown that DOCFs 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 also prove that any algorithm for learning DOCFs 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 DOCFs 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? As suggested in [18], it is reasonable to believe that equivalence queries are practically harder to implement that membership queries.

3. Although we have shown that the problem of learning disjunctions of negated counting functions with modulus $N \geq 2$ over the domain Z_N^n is in general harder than the problem of learning DNF formulas, we do not know whether this problem is learnable when $N < n$ (in particular, when N is a constant greater than 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 the class of monotone conjunctions of diagonal DOCFs with modulus $N \geq 2$ over the domain Z_N^n is learnable using equivalence and membership queries. Can one learn this class using equivalence and incomplete membership queries?

6. Can one learn the class of conjunctions of diagonal DOCFs with modulus $N \geq 2$ over the domain Z_N^n? Or in general, can one learn conjunctions of DOCFs with modulus $N \geq 2$ over the domain Z_N^n? One should note that those two problems are harder than the problem of learning DNF formulas. A systematic approach of monotone theory has been established in [14] 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 [14] can not be used to learn conjunctions of DOCFs, because the algebraic structures of conjunctions of DOCFs are more complicated than those of boolean functions.

References

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

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

[3] D. Angluin, M. Frazier, L. Pitt, "Learning conjunctions of horn clauses", *Machine Learning*, 9, 1992, pages 147-164.

[4] D. Angluin, M. Kharitonov, "When won't membership queries help?" *Proc of the 23th Annual ACM Symposium on Theory of Computing*, 1991, pages 444-454.

[5] D. Angluin, D. Slonim, "Learning monotone DNF with an incomplete membership oracle", *Proc of the Third Annual ACM Workshop on Computational Learning Theory*, pages 139-146, 1991.

[6] H. Aizenstein, L. Hellerstein, L. Pitt, "Read thrice DNF is hard to learn with membership and equivalence queries", *Proc of the 33th Annual ACM Symposium on Foundations of Computer Science*, pages 523-532, 1992.

[7] H. Aizenstein, L. Pitt, "Exact Learning of read-twice DNF formulas", *Proc of the 32th Annual ACM Symposium on Foundations of Computer Science*, 1992.

[8] H. Aizenstein, L. Pitt, "Exact learning of read-k disjoint and not-so-disjoint DNF", *Proc of the Fifth Annual ACM Conference on Computational Learning Theory*, pages 71-76, 1993.

[9] A. Bertoni, N. Cesa-Bianchi, G. Fiorino, "Efficient learning with equivalence queries of conjunctions of modulo functions", Information Processing Letters, 56, pages 15-17, 1995.

[10] A. Blum, "Separating PAC and mistake-bounded learning models over the boolean domain", *Proc of the 31th Annual ACM Symposium on Foundations of Computer Science*, pages 211-218, 1990.

[11] A. Blum, P. Chalasani, J. Jackson, "On learning embedded symmetric concepts", *Proc of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 337-346, 1993.

[12] A. Blum, S. Rudich, "Fast learning of k-term DNF formulas with queries", *Proc of the 24th Annual ACM Symposium on Theory of Computing*, May 1992, pages 382-389.

[13] A. Blum, M. Singh, "Learning functions of k terms", *Proc of the Third Annual Workshop on Computational Learning Theory*, pages 144-153, 1990.

[14] N. Bshouty, "Exact Learning via the monotone theory", *Proc of the 34th Annual ACM Symposium on Foundations of Computer Science*, pages 302-311, 1993.

[15] need reference [BBBKV96]

[16] need reference [BBV96]

[17] N. Bshouty, Z. Chen, S. Decatur, S. Homer, "On the learnability of Z_N-DNF formulas", *Proc of the 8th Annual ACM Conference on Computational Learning Theory*, pages 198-205, 1995.

[18] N. Bshouty, S. Goldman, T. Hancock, S. Matar, "Asking queries to minimize errors", *Proc of the 6th Annual ACM Conference on Computational Learning Theory*, pages 41-50, 1993.

[19] N. Bshouty, T. Hancock, L. Hellerstein, "learning arithmetic read-once formulas" *Proc of the 24th Annual ACM Symposium on Theory of Computing*, 1992.

[20] N. Bshouty, T. Hancock, L. Hellerstein, "Learning boolean read-once formulas with arbitrary symmetric and constant fan-in gates", *Proc of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 1-15, 1992.

[21] Z. Chen, "Disjunctions of negated counting functions are efficiently learnable with equivalence queries", Lecture Notes in Computer Science 959, Springer, 1995, pages 344-349.

[22] Z. Chen, S. Homer, "On learning counting functions with queries" *Proc of the 7th Annual ACM Conference on Computational Learning Theory*, pages 218-227, 1994.

[23] Z. Chen, S. Homer, "On learning counting functions with queries" Theoretical Computer Science, 1997, pages 155-168.

[24] P. Fisher, H. Simon, "On learning ring-sum-expansions", *SIAM J. Comput.*, 1992, pages 181-192.

[25] T. Hancock, L. Hellerstein, "Learning read-once formulas over fields and extended bases", *Proc of the 3th Annual Workshop on Computational Learning Theory*, pages 326-336, 1991.

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

[27] N. Jacobson, Basic Algebra I, W. H. Freeman and Company, New York, 1985.

[28] R. Schapire, L. Sellie, "Learning sparse multivariate polynomials over a field with queries and counterexamples" *Proc of the 6th Annual ACM Conference on Computational Learning Theory*, 1993.

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