Electronic Colloquium on Computational Complexity, Report No. 19 (2002)

On the Proper Learning of Axis Parallel Concepts

Nader H. Bshouty *
Department of Computer Science
Technion
Haifa, Israel

Lynn Burroughs †
Department of Computer Science
Calgary University
Calgary, Alberta, Canada

March 19, 2002

Abstract

We study the proper learnability of axis parallel concept classes in the PAC learning model and in the exact learning model with membership and equivalence queries. These classes include union of boxes, DNF, decision trees and multivariate polynomials.

For the *constant* dimensional axis parallel concepts C we show that the following problems have the same time complexity

1. C is α-properly exactly learnable (with hypotheses of size at most α times the target size) from membership and equivalence queries.
2. C is α-properly PAC learnable (without membership queries) under any product distribution.
3. There is an α-approximation algorithm for the MinEquiC problem. (given a $g \in C$ find a minimal size $f \in C$ that is equivalent to g).

In particular, C is α-properly learnable in polynomial time from membership and equivalence queries **if and only if** C is α-properly PAC learnable in polynomial time under the product distribution **if and only if** MinEquiC has a polynomial time α-approximation algorithm. Using this result we give the first proper learning algorithm of decision trees over the constant dimensional domain and the first negative results in proper learning from membership and equivalence queries for many classes.

For the non-constant dimensional axis parallel concepts we show that with the equivalence oracle (1) $\Rightarrow$ (3). We use this to show that (binary) decision trees are not properly learnable in polynomial time (assuming P$\neq$NP) and DNF is not s^{ϵ}-properly learnable ($\epsilon < 1$) in polynomial time even with an NP-oracle (assuming $\Sigma_2^p \neq P^{NP}$).

*This research was supported by the fund for promotion of research at the Technion. Research no. 120-025. Part of this research was done at the University of Calgary, Calgary, Alberta, Canada and supported by NSERC of Canada.

†This research was supported by an NSERC PGS-B Scholarship, an Izaak Walton Killam Memorial Scholarship, and an Alberta Informatics Circle of Research Excellence (iCORE) Fellowship.

ISSN 1433-8092

1 Introduction

We study the proper learnability of axis parallel concept classes in the PAC-learning model and in the exact learning model with membership and equivalence queries. A class $\mathcal{N}$-$\mathcal{P}\Phi$ of axis parallel concepts is a class of boolean formulas $\phi(T_1, T_2, \ldots, T_t)$ where ϕ is from a class of boolean formulas Φ (such as DNF, decision tree, etc.) and $\{T_i\}$ are boxes in $\mathcal{N}_m^n$, where $\mathcal{N}_m = \{0, \ldots, m-1\}$, that satisfy a certain property $\mathcal{P}$ (such as disjointness, squares, etc.). These classes include union of boxes, union of disjoint boxes, exclusive or of boxes, decision tree partition, and for the boolean case $\mathcal{N}_2^n$, they include DNF, decision trees, disjoint DNF and multivariate polynomials.

The term α-proper learning refers to learning where the intermediate hypotheses used by the learner (in the equivalence queries) and the final hypothesis have size (number of boxes T_i) at most α times the size of the target formula. A class is properly learnable if it is 1-learnable. The following table summarizes the results for the n-dimensional boolean case.

	Upper for	Complexity	Source	Lower	Condition	Source
DNF	Nonproper	$2^{\tilde{O}(n^{1/3})}$	[KS01]	Proper	P$\neq$NP	[PR96]
	Nonproper	NP-oracle	[BCG96]			
	Proper	Σ_p^4-oracle	OPEN	s^ϵ-Proper	$\Sigma_p^2 \neq P^{NP}$	[ours]
CDNF	Nonproper	poly(n)	[B95]			
	Proper	Σ_p^4-oracle	[HPRW96]			
	Proper	poly(n)	OPEN			
Disj-DNF	Nonproper	poly(n)	[BCV96]	Proper	P$\neq$NP	OPEN
	Proper	Σ_p^4-oracle	OPEN			
DT	Nonproper	poly(n)	[B95]	Proper	P$\neq$NP	[ours]
	Proper	Σ_p^4-oracle	OPEN			
MP	Nonproper	poly(n)	[BCV96]	Proper	P$\neq$NP	OPEN
	Proper	Σ_p^4-oracle	OPEN			
MMP	Proper	poly(n)	[SS93]			

Hellerstein et al. [HPRW96] show that proper learnability of a class C from membership and equivalence queries is possible in a machine with unlimited computational power if and only if C has *polynomial certificates*. They also show that if C has a polynomial certificate then C is properly learnable using an oracle for Σ_4^p. They then give a polynomial size certificate for CDNF (a polynomial size DNF that has a polynomial size CNF). This implies that CDNF is properly learnable using an oracle for Σ_4^p. For DNF, decision trees (DT), Disjoint DNF (DNF where the conjunction of every two terms is 0) and multivariate polynomials (with nonmonotone terms (MP)) it is not known whether they have polynomial certificates. Therefore it is not known if they are properly learnable. Pillaipakkamnatt and Raghavan [PR96] show that if DNF is properly learnable then P=NP. On the other hand, Bshouty et al. [BCG96] show that any circuit is (nonproperly) learnable with equivalence queries only and the aid of NP-oracle. The best algorithm today for learning DNF runs in time $2^{\tilde{O}(n^{1/3})}$ [KS01].

CDNF, Decision trees, disjoint DNF and multivariate polynomials are (nonproperly) learnable in polynomial time from membership and equivalence queries [B95, BCV96, BBBKV00]. Multivariate polynomials with monotone terms (MMP) are properly learnable [SS93].

In this paper we use a new technique for finding negative results for learning from membership and equivalence queries (see Theorem 1). We use Theorem 1 and the result of Zantema and Bodlaender [ZB00] to show that if a decision tree is properly learnable from membership and equivalence queries then P=NP. We then use the result of Umans [U99] and show that if DNF is s^ϵ-properly learnable with an NP-oracle, where s is the size of the DNF, then $\Sigma_2^p = P^{NP}$. We show our results are still true even if the learner can use other oracles such as Subset, Superset, Disjointness, etc. Therefore, if P$\neq$NP then decision trees and DNF are not properly learnable from membership and equivalence queries (and all the other oracles defined in subsection 2.3).

The following table summarizes our results for axis parallel classes over $\mathcal{N}_m^n$ for a constant dimension n.

	Positive results poly$(\log m)$	**Negative results iff P=NP**
Union of t Boxes	$\log t$-Proper	Proper
Disjoint Union Boxes	dim=2 Proper	dim>2 Proper
Decision tree	Proper	
Xor of Boxes	α-Proper	OPEN

For axis parallel classes over a constant dimension we show that these classes have polynomial certificates. Therefore by [HPRW96], they are properly learnable from membership and equivalence queries using the Σ_4^p oracle. We further investigate the learnability of these classes and show that an NP-oracle is sufficient for proper learnability. We also show that the following problems have the same time complexity.

1. C is α-properly exactly learnable from membership and equivalence queries.

2. C is α-properly PAC learnable (without membership queries) under any product distribution.

3. There is an α-approximation algorithm for the MinEqui$C$ problem (given a $g \in C$ find a minimal size $f \in C$ that is equivalent to g).

4. C is exactly learnable with a learning algorithm that uses all the queries (membership and nonproper equivalence, subset, superset, etc.) and outputs a hypothesis that has size at most α times the target size.

We start with some surprising results that follow from this. The first is (1)$\Rightarrow$(2). It is known that (proper) learnability from equivalence and membership queries implies (proper) learnability in the PAC model with membership queries [A88]. Here we show that in the case of finite dimensional space and for the product distribution we can change a weak learner (that learns with membership queries) to a strong learner (that learns without membership queries). Another surprising result that we show from this is: a decision tree over any constant dimension is *properly learnable* from membership and equivalence queries. We also show that decision tree is proper PAC-learnable under any distribution.

We also show that union of disjoint DNF in dimension 2 has a polynomial time proper learning algorithm. On the other hand, union of boxes and disjoint union of boxes over dimensions greater than 2 are properly learnable if and only if P=NP. Union of boxes is

$\log t$-properly learnable where t is the number of boxes and Xor of boxes is α-properly learnable for some constant α.

All the results in the literature for the constant dimensional domain give nonproper learning of the above classes in the exact learning model and there were no negative results for proper learning of these classes from membership and equivalence queries.

In [CM94] Chen and Maass give a proper exact learning of one box from equivalence queries. Beimel and Kushilevitz [BK98] show that $\mathcal{N}_m^n$-Disjoint DNF is (nonproperly) learnable from membership and equivalence queries. This result is also true for the nonconstant dimensional domain. The output hypothesis is represented as a $\mathcal{N}_m^n$-Multiplicity Automaton.[1] Since $\mathcal{N}_m^n$-Multiplicity Automaton contains the class of $\mathcal{N}_m^n$-Multivariate Polynomials [BBBKV00], the class of $\mathcal{N}_m^n$-Multivariate Polynomials is (nonproperly) learnable in polynomial time from membership and equivalence queries. In [BGGM99] Bshouty et al. give an $O(d \log t)$-proper learning algorithm that learns a union of t boxes in d-dimensional space. The algorithm in this paper is $\log t$-proper.

There are many algorithms in the literature that learn the union of boxes in the constant dimensional space [CH96, MW98] (and even any combination of thresholds in the constant dimensional space from equivalence queries only [BBK97, B98]). All of these algorithms are nonproper and return hypotheses that have large size.

2 Preliminaries

2.1 Learning Models

The learning criteria we consider are *exact learning* and *PAC-learning*.

In the exact learning model there is a function f called the *target function* $f : \mathcal{N}_m^n \to \{0,1\}$ (where $\mathcal{N}_m = \{0, 1, \ldots, m-1\}$), which has a formula representation in a class C of formulas defined over the variable set $V_n = \{x_1, \ldots, x_n\}$. The goal of the learning algorithm is to halt and output a formula $h \in C$ that is equivalent to f.

The learning algorithm performs a *membership query* by supplying an assignment a to the variables in V_n as input to a *membership oracle* and receives in return the value of $f(a)$. For our algorithms we will regard this oracle as a procedure $MQ_f()$. The procedure's input is an assignment a and its output is $MQ_f(a) = f(a)$.

The learning algorithm performs an *equivalence query* by supplying a formula $h \in C$ as input to an *equivalence oracle* with the oracle returning either "YES", signifying that h is equivalent to f, or a *counterexample*, which is an assignment b such that $h(b) \neq f(b)$. For our algorithms we will regard this oracle as a procedure $EQ_f(h)$.

We say that a class C of boolean functions is *α-properly exactly learnable* from membership and equivalence queries in polynomial time if there is an algorithm that for any $f \in C$ over V_n, the algorithm runs in polynomial time, asks a polynomial number (polynomial in n, $\log m$ and in the size of the target function) of membership and equivalence queries with hypothesis $h \in C$ of size at most α times the size of the target, and outputs a hypothesis

[1] One can also define $\mathcal{N}_m^n$-Multiplicity Automaton. Using the algorithm from that paper, $\mathcal{N}_m^n$-Multiplicity Automaton is properly learnable from membership and equivalence queries.

$h \in C$ that is equivalent to f. The size of h is at most α times the size of the target. We say that C is *properly exactly learnable* if it is 1-properly exactly learnable.

The PAC learning model is as follows. There is a distribution D defined over the domain $\mathcal{N}_m^n$. The goal of the learning algorithm is to halt and output a formula h that is ϵ-close to f with respect to the distribution D, that is,

$$\Pr_D[f(x) = h(x)] \geq 1 - \epsilon.$$

We say that h is an ϵ-approximation of f with respect to the distribution D. In the PAC or *example query* model, the learning algorithm asks for an example from the *example oracle*, and receives an example $(a, f(a))$ where a is chosen from $\mathcal{N}_m^n$ according to the distribution D.

We say that a class of boolean functions C is *α-properly PAC learnable* under the distribution D in polynomial time if there is an algorithm A, such that for any $f \in C$ over V_n and any ϵ and δ, algorithm A runs in polynomial time, asks a polynomial number of queries (polynomial in n, $\log m$, $1/\epsilon$, $1/\delta$ and the size of the target function) and with probability at least $1-\delta$ outputs a hypothesis $h \in C$ that is an ϵ-approximation of f with respect to the distribution D. The size of h is at most α times the size of f. It is known from [A88] that if a class C is α-properly exactly learnable in polynomial time from equivalence queries (and membership queries) then it is α-properly PAC learnable (with membership queries) in polynomial time under any distribution D.

We say that a distribution D is a *product* distribution over $\mathcal{N}_m^n$ if

$$D(x_1, \ldots, x_n) = D_1(x_1) D_2(x_2) \cdots D_n(x_n)$$

where each D_i is a distribution over $\mathcal{N}_m$.

2.2 Axis Parallel Concept Classes

A *boolean function* over $\mathcal{N}_m^n$ is a function $f : \mathcal{N}_m^n \to \{0,1\}$. The elements of $\mathcal{N}_m^n$ are called *assignments*. We will consider the set of *variables* $V_n = \{x_1, \ldots, x_n\}$ where x_i will describe the value of the i-projection of the assignment in the domain $\mathcal{N}_m^n$ of f. For an assignment a, the i-th entry of a will be denoted by a_i.

An *$\mathcal{N}_m^n$-literal* is a function $[x_i \geq a]$ or $[x_i < a]$ where $i \leq n$ and $a \in \mathcal{N}_m \cup \{m\}$. Here $[x_i \geq a] = 1$ if $x_i \geq a$ and 0 otherwise. An *$\mathcal{N}_m^n$-monotone literal* is $[x_i \geq a]$. An *$\mathcal{N}_m^n$-term* is a product (conjunction) of literals. An *$\mathcal{N}_m^n$-monotone term* is a product (conjunction) of monotone literals. An *$\mathcal{N}_m^n$-DNF* is a disjunction of terms. An *$\mathcal{N}_m^n$-Monotone DNF* is a disjunction of monotone terms. An *$\mathcal{N}_m^n$-multivariate polynomial* is a sum of terms (mod 2). An *$\mathcal{N}_m^n$-disjoint DNF* is an $\mathcal{N}_m^n$-DNF where the conjunction of every two terms is 0.

For example, when $n = 3$, $T = [x_1 \geq 2] \wedge [x_1 < 5] \wedge [x_2 \geq 9]$ is an $\mathcal{N}_{11}^3$-term and can be written as

$$T = [2 \leq x_1 < 5] \wedge [9 \leq x_2 < 11] \wedge [0 \leq x_3 < 11].$$

Therefore, every term $\mathcal{N}_m^n$-term can be written as

$$T_i = \bigwedge_{k=1}^{n} [a_{i,k} \leq x_k < b_{i,k}],$$

where $a_{i,k}, b_{i,k} \in \mathcal{N}_m \cup \{m\}$.

Notice that if we take an $\mathcal{N}_m^n$-term T and replace all m with some $m' > m$ then we get an $\mathcal{N}_{m'}^n$-term T'. Therefore we will sometimes ignore m and talk about $\mathcal{N}^n$-terms where each m is replaced by ∞. For an $\mathcal{N}_m^n$-term T we write T^∞ for the corresponding $\mathcal{N}^n$-term. For the above example the corresponding $\mathcal{N}^n$-term is

$$T^\infty = [2 \le x_1 < 5] \wedge [9 \le x_2 < \infty] \wedge [0 \le x_3 < \infty].$$

An $\mathcal{N}_m^n$-*decision tree* ($\mathcal{N}_m^n$-DT) over V_n is a binary tree whose nodes are labeled with $\mathcal{N}_m^n$-literals and whose leaves are labeled with constants from $\{0,1\}$. each decision tree T represents a function $f_T : \mathcal{N}_m^n \to \{0,1\}$. To compute $f_T(a)$ we start from the root of the tree T: if the root is labeled with the literal l and $l(a) = 1$, then $f_T(a) = f_{T_R}(a)$ where T_R is the right subtree of the root (i.e., the subtree of the right child of the root with all its descendents). Otherwise, $f_T(a) = f_{T_L}(a)$ where T_L is the left subtree of the root. If T is a leaf then $f_T(a)$ is the label of this leaf.

Sometimes we write $\mathcal{N}_m$-DNF when we do not want to specify the dimension. A DNF (multivariate polynomial, decision tree) is an $\mathcal{N}_2$-DNF (respectively, $\mathcal{N}_2$-multivariate polynomial, $\mathcal{N}_2$-decision tree).

In general, for every set of boolean functions Φ (e.g., exclusive or, or, etc.) and *property* function $\mathcal{P}_t : (\mathcal{N}^n\text{-}term)^t \to \{0,1\}$ that is computable in polynomial time, (e.g., $\mathcal{P}_t(T_1, \ldots, T_t) = 1$ if $T_1, \ldots, T_t$ are pairwise disjoint) we can build a concept over $\mathcal{N}_m^n$ as follows. We define the concept class $\mathcal{P}\Phi[\mathcal{N}_m^n]$ (or $\mathcal{N}_m^n$-$\mathcal{P}\Phi$) to be the set of all $\phi(T_1, \ldots, T_t)$ where $\phi \in \Phi$ and $\{T_i\}$ are $\mathcal{N}_m^n$-terms with $\mathcal{P}_t(T_1^\infty, \ldots, T_t^\infty) = 1$. The property $\mathcal{P}$ is always computable in polynomial time and independent of m and therefore can be defined for $\mathcal{N}_m^n$-terms for any m. We will sometime ignore the superscript ∞ and just write $\mathcal{P}_t(T_1, \ldots, T_t)$. These classes are called *axis-parallel concept classes*. When $\mathcal{P}_t \equiv 1$ then we write $\Phi[\mathcal{N}_m^n]$ (or $\mathcal{N}_m^n$-Φ).

For example, let $\Phi = \{x_1, x_1 \vee x_2, x_1 \vee x_2 \vee x_3, \cdots\}$. Let $\mathcal{P}_t(T_1, \ldots, T_t) = 1$ if and only if $T_i \wedge T_j = 0$ for every $1 \le i < j \le t$. Then $\mathcal{P}\Phi[\mathcal{N}_2^n]$ is the set of disjoint DNF and $\mathcal{P}\Phi[\mathcal{N}_m^2]$ is the set of union of disjoint rectangles in the two dimensional space.

For an $f \in \mathcal{P}\Phi[\mathcal{N}_m^n]$ we define $size_{\mathcal{P}\Phi}(f)$ to be the minimal number of $\mathcal{N}_m^n$-terms $T_1, \ldots, T_t$ with property $\mathcal{P}_t$ (that is, $\mathcal{P}_t(T_1^\infty, \ldots, T_t^\infty) = 1$) such that there is an $h \in \Phi$ where $f \equiv h(T_1, \ldots, T_t)$. For a decision tree f the size will be the minimal number of non-leaf nodes in an $\mathcal{N}_m^n$-decision tree that is equivalent to f.

A *monotone projection* from $\mathcal{N}_{m'}$ to $\mathcal{N}_m$ is a function $M : \mathcal{N}_{m'} \cup \{m'\} \to \mathcal{N}_m \cup \{m\}$ such that for every $i, j \in \mathcal{N}_{m'}$ where $i < j$ we have $M(i) \le M(j)$ and $M(m') = m$. A monotone projection $\mathcal{M} : (\mathcal{N}_{m'} \cup \{m'\})^n \to (\mathcal{N}_m \cup \{m\})^n$ is $\mathcal{M} = (M_1, \ldots, M_n)$ where each $M_i : \mathcal{N}_{m'} \cup \{m'\} \to \mathcal{N}_m \cup \{m\}$ is a monotone projection. We say that $C = \mathcal{P}\Phi[\mathcal{N}_m^n]$ is *closed under monotone projection* if for any monotone projection $\mathcal{M}$ whenever $\mathcal{P}(T_1, \ldots, T_t) = 1$ we also have $\mathcal{P}(T_1\mathcal{M}, \ldots, T_t\mathcal{M}) = 1$. Notice that if $f = \phi(T_1, \ldots, T_t) \in \Phi[\mathcal{N}_m^n]$ then $f\mathcal{M} = \phi(T_1\mathcal{M}, \ldots, T_t\mathcal{M}) \in \Phi[\mathcal{N}_{m'}^n]$. When the class $C = \mathcal{P}\Phi[\mathcal{N}_m^n]$ is closed under monotone projection then $f\mathcal{M} \in \mathcal{P}\Phi[\mathcal{N}_{m'}^n]$.

For a monotone projection M define the *dual monotone projection* $M^\star$ where $M^\star(y)$ is the minimal x such that $M(x) \ge y$. Since $M(m') = m$, the dual monotone projection is well defined. For a monotone projection $\mathcal{M} = (M_1, \ldots, M_n)$ we define $\mathcal{M}^\star = (M_1^\star, \ldots, M_n^\star)$. We now show

Lemma 1 *If*

$$T = \bigwedge_{k=1}^{n} [a_{i,k} \le x_k < b_{i,k}]$$

then

$$T\mathcal{M} = \bigwedge_{k=1}^{n} [a_{i,k} \le M_k(x_k) < b_{i,k}] = \bigwedge_{k=1}^{n} [M_k^\star(a_{i,k}) \le x_k < M_k^\star(b_{i,k})].$$

and therefore $T\mathcal{M}$ is again an $\mathcal{N}^n$-term.

Proof. We first show the following two properties of monotone projection.

1. If M is monotone then $M^\star$ is monotone.
2. $x \ge M^\star M(x)$.

To prove (1), let $y_1 < y_2$ and $M^\star(y_1) = x_1$ and $M^\star(y_2) = x_2$. Then x_2 is the minimal integer such that $M(x_2) \ge y_2$. Since $M(x_2) \ge y_2 > y_1$, the minimal x_1 such that $M(x_1) \ge y_1$ must be less than or equal to x_2. Therefore,

$$M^\star(y_1) = x_1 \le x_2 = M^\star(y_2).$$

To prove (2) let $M^\star(M(x)) = z$. Then z is the minimal integer such that $M(z) \ge M(x)$. Then $M(z) = M(x)$ and since z is minimal $M^\star(M(x)) = z \le x$.

Now we are ready to prove the result. It is enough to show that

$$M(x) < y \;\text{ if and only if }\; x < M^\star(y).$$

Suppose $M(x) < y$. Let $z = M^\star(y)$. Then z is the minimal integer such that $M(z) \ge y$. Now $M(z) \ge y > M(x)$ and therefore $M^\star(y) = z > x$.

If $M(x) \ge y$ then by properties (1) and (2) we have $x \ge M^\star M(x) \ge M^\star(y)$.□

Now the proof of the following Lemma is straightforward.

Lemma 2 *If $C_m = \mathcal{P}\Phi[\mathcal{N}_m^n]$ is closed under monotone projection and $f \in C_m$, then $f\mathcal{M} \in C_{m'}$ for any monotone projection $\mathcal{M} : \mathcal{N}_{m'}^n \to \mathcal{N}_m^n$. Also,*

$$size_{\mathcal{P}\Phi}(f\mathcal{M}) \le size_{\mathcal{P}\Phi}(f).$$

In a similar way one can define the class $\mathcal{P}\Phi[\mathcal{N}_{m_1} \times \cdots \times \mathcal{N}_{m_n}]$. In that case a monotone projection is $\mathcal{M} = (M_1, \ldots, M_n)$ where $M_i : N_{m_i'} \to N_{m_i}$ and for $f \in \mathcal{P}\Phi[\mathcal{N}_{m_1} \times \cdots \times \mathcal{N}_{m_n}]$. $f\mathcal{M} \in \mathcal{P}\Phi[\mathcal{N}_{m_1'} \times \cdots \times \mathcal{N}_{m_n'}]$. All the results of this paper are also true for that class.

Constructiveness Assumption: We assume that there is an algorithm "Construct" such that for any function $\psi : \mathcal{N}_{m_1} \times \cdots \times \mathcal{N}_{m_n} \to \{0,1\}$ that is computable in polynomial time, Construct(ψ) runs in time $poly(\prod m_i)$ and returns some formula $f \in \mathcal{P}\Phi[\mathcal{N}_{m_1} \times \cdots \times \mathcal{N}_{m_n}]$ that is equivalent to ψ, if there exist such a formula, and returns "error" otherwise. Such algorithms exist (and are in fact very trivial) for all the classes presented in this paper.

2.3 Oracles

The other oracles we will consider in this paper are the following, as defined in [A88].

- **Subset oracle**. $Sub_f(h)$ for $h \in C$. This oracle returns 'YES' if $h \Rightarrow f$ and returns a counterexample a such that $h(a)=1$ and $f(a)=0$ otherwise.
- **Superset oracle**. $Sup_f(h)$ for $h \in C$. This oracle returns 'YES' if $h \Leftarrow f$ and returns a counterexample a such that $h(a)=0$ and $f(a)=1$ otherwise.
- **Disjointness oracle**. $Disj_f(h)$ for $h \in C$. This oracle returns 'YES' if $h \wedge f = 0$ and returns a counterexample a such that $h(a)=1$ and $f(a)=1$ otherwise.
- **Exhaustiveness oracle**. $Exh_f(h)$ for $h \in C$. This oracle returns 'YES' if $h \vee f = 1$ and returns a counterexample a such that $h(a)=0$ and $f(a)=0$ otherwise.

Given a set of oracles $\mathcal{O}$, we say that $\mathcal{O}$ is *easy* (resp. NP-easy) for C if every oracle in $\mathcal{O}$ can be simulated in polynomial time for C (resp. simulated using an NP-oracle).

Lemma 3 *Let Φ be a set of boolean functions. For a constant dimension d, membership and equivalence oracles and all the above oracles (subset, superset, etc.) are easy for $\Phi[\mathcal{N}_m^d]$.*

Proof. Let $R_f(h)$ be an oracle asked for the target f. We take all the literals $[x_i Q a]$ for $Q \in \{\geq, <\}$ in the terms of the target f and of the function h in the oracle. Let A be the set of all the a's in those terms and the two constants 0 and m. Suppose $A = \{0 \leq a_1 < a_2 < \cdots < a_t \leq m\}$. Notice that $t \leq 2d(s_h + s_f)$ where s_h and s_f are the number of terms in h and f, respectively. It is easy to see that the functions h and f are constant functions 0 or 1 in each subdomain $[a_{i_1}, a_{i_1+1}) \times \cdots \times [a_{i_d}, a_{i_d+1})$. We check the oracle R_f for every such subdomain. The number of subdomains is $(t+1)^d$ which is polynomial for any constant d.$\Box$

2.4 Lattice Projection

In this section we will give the definition of the *lattice projection* of functions and prove some claims. This will be one of the main tools used in this paper. Since this technique is used for constant dimension we will use d for the dimension n.

A *lattice* in $\mathcal{N}_m^d$ is $L = L_1 \times \cdots \times L_d$ where $L_i \subseteq \mathcal{N}_m$. Let $f = \phi(T_1, \ldots, T_t) \in \Phi[\mathcal{N}_m^d]$ where $\phi \in \Phi$ and

$$T_i = \bigwedge_{k=1}^{d} [a_{i,k} \leq x_k < b_{i,k}].$$

Let $L = L_1 \times \cdots \times L_d$ be a lattice in $\mathcal{N}_m^d$. For $a, b \in \mathcal{N}_m$ define

$$\lfloor a \rfloor_{L_i} = \max\{x \in L_i \mid x \leq a\} \cup \{0\}, \quad \lceil b \rceil_{L_i} = \min\{x \in L_i \mid x \geq b\} \cup \{m\}.$$

For an assignment $v \in \mathcal{N}_m^d$ we define $\lfloor v \rfloor_L = (\lfloor v_1 \rfloor_{L_1}, \ldots, \lfloor v_d \rfloor_{L_d})$. Define

$$T_i^L = \bigwedge_{k=1}^{d} [\lceil a_{i,k} \rceil_{L_k} \le x_k < \lceil b_{i,k} \rceil_{L_k}]$$

and $f^L = \phi(T_1^L, \cdots, T_t^L)$. We call f^L the *lattice projection* of f on L.

Lemma 4 *For every u we have*

$$f^L(u) = f(\lfloor u \rfloor_L) = f^L(\lfloor u \rfloor_L).$$

Proof. Notice that the lattice projection is a monotone projection with $M_i(x_i) = \lfloor x_i \rfloor_{L_i}$. It is also easy to see that $M_i^\star(x_i) = \lceil x_i \rceil$. Let $\mathcal{M} = (M_1, M_2, \ldots, M_d)$. Then $\mathcal{M}(x) = \lfloor x \rfloor_L$ is a monotone projection and $\mathcal{M}^\star(x) = \lceil x \rceil_L$. We also have $\mathcal{M}\mathcal{M} = \mathcal{M}$. Therefore,

$$f^L(\lfloor u \rfloor_L) = f\mathcal{M}(\mathcal{M}u) = f(\mathcal{M}\mathcal{M}u) = f(\mathcal{M}u) = f^L(u) = f(\mathcal{M}u) = f(\lfloor u \rfloor_L).\Box$$

Lemma 5 *Let $\mathcal{P}\Phi[\mathcal{N}_m^d]$ be closed under monotone projection. For any function $f \in \mathcal{P}\Phi[\mathcal{N}_m^d]$ and lattice L we have $f^L \in \mathcal{P}\Phi[\mathcal{N}_m^d]$.*

Proof. Since $\mathcal{M}(x) = \lfloor x \rfloor_L$ is a monotone projection and $\mathcal{P}\Phi[\mathcal{N}_m^d]$ is closed under monotone projection, the result follows.$\Box$

Lemma 6 *Let $f = \phi(T_1, \ldots, T_t)$ where $\phi \in \Phi$ and $T_i = \bigwedge_{k=1}^d [a_{i,k} \le x_k < b_{i,k}]$. If for every i and k we have $a_{i,k}, b_{i,k} \in L_k$ then $f^L \equiv f$.*

Proof. If $a_{i,k}, b_{i,k} \in L_i$ then $\lceil a_{i,k} \rceil_{L_k} = a_{i,k}$, $\lceil b_{i,k} \rceil_{L_k} = b_{i,k}$, $T_i^L = T_i$ and $f^L = \phi(T_1^L, \ldots, T_t^L) = \phi(T_1, \ldots, T_t) = f$.$\Box$

2.5 Polynomial Certificates

Following the definition of [HPRW96], the class $C = \mathcal{P}\Phi[\mathcal{N}_m^n]$ has polynomial certificates if for every $f \in C$ of size t there are $q = poly(t, n)$ assignments $A = \{a_1^f, \ldots, a_q^f\}$ such that for every $g \in C$ of size less than t, g is not consistent with f on A. That is, $g(a) \ne f(a)$ for some $a \in A$.

Lemma 7 *Let d be constant. If $C = \mathcal{P}\Phi[\mathcal{N}_m^d]$ is closed under monotone projection then it has polynomial certificates.*

Proof. Let $f = \phi(T_1, \ldots, T_t)$ where $\phi \in \Phi$ and $T_i = \wedge_{k=1}^d [a_{i,k} \le x_k < b_{i,k}]$. Let $L_k = \{a_{i,k}, b_{i,k} \mid i = 1, \ldots, t\}$ and $L = L_1 \times \cdots \times L_d$. Notice that $|L| \le (2t)^d = poly(t)$ for constant d. We now show that if $g \in \mathcal{P}\Phi[\mathcal{N}_m^d]$ is consistent with f on L then $size_{\mathcal{P}\Phi}(g) \ge t$.

Since g is consistent with f on L, by Lemmas 4 and 6 we have $g^L(x) = g(\lfloor x \rfloor_L) = f(\lfloor x \rfloor_L) = f^L(x) = f(x)$. By Lemma 5 we have $g^L \in \mathcal{P}\Phi[\mathcal{N}_m^d]$. Therefore, by Lemma 2 $size_{\mathcal{P}\Phi}(g) \ge size_{\mathcal{P}\Phi}(g^L) = size_{\mathcal{P}\Phi}(f)$.$\Box$

It follows from [HPRW96] that if $C = \mathcal{P}\Phi[\mathcal{N}_m^d]$ is closed under monotone projection then C is learnable from membership and equivalence queries using an oracle for Σ_4^p. In this paper we will show that an NP oracle is sufficient.

2.6 Approximation algorithms

We assume the reader is familiar with approximation algorithms, α-approximation and some of the basic concepts in approximation theory and complexity theory. For a problem in which we seek to minimize the size of a formula equivalent to a function f, an α-approximation algorithm (for $\alpha \geq 1$) returns a formula $h \equiv f$ that has size at most α times the size of the smallest formula equivalent to f. Given a class $C = \mathcal{P}\Phi[\mathcal{N}_m^n]$, we define the optimization problem Minimal Equivalent Formula (MinEqui C) to be the following problem:

<u>MinEqui C</u>

Given a formula $f \in C = \mathcal{P}\Phi[\mathcal{N}_m^n]$.

Find a minimal size $h \in C$ that is equivalent to f.

We expect the algorithm to run in time polynomial in n and the size of f. In some cases the function is given as an n-dimensional $m_1 \times m_2 \times \cdots \times m_n$ matrix and the time is expected to be polynomial in $\prod m_i$. We call this the *unary representation* of the input. The problem is defined as follows.

<u>MinEqui C_U</u>

Given an $m_1 \times \cdots \times m_n$ matrix A representing a formula in $C = \mathcal{P}\Phi[\mathcal{N}_{m_1} \times \cdots \times \mathcal{N}_{m_n}]$.

Find a minimal size $h \in C$ that represents A.

A generalization of this problem is the MinEqui* C_U problem. In this problem the input is given in its unary representation. The matrix may contain $\star$ entries which denote unspecified values.

<u>MinEqui* C_U</u>

Given an $m_1 \times \cdots \times m_n$ matrix A with entries $0, 1$ and $\star$, representing a function in $C = \mathcal{P}\Phi[\mathcal{N}_{m_1} \times \cdots \times \mathcal{N}_{m_n}]$.

Find a minimal size $h \in C$ that is equivalent to f on the specified values.

Another problem that is related to the latter problem is the minimal size consistency problem.

<u>MinEqui* C</u>

Given a set $S = \{(a_1, f(a_1)), \cdots, (a_t, f(a_t))\}$ where $f \in C = \mathcal{P}\Phi[\mathcal{N}_m^n]$.

Find a minimal size $h \in C$ that is consistent with S. That is, $h(a_i) = f(a_i)$ for all i.

Some of these problems appear in the literature. For example, MinEqui $\mathcal{N}_m^2$-DNF$_U$ is the task of covering orthogonal polygons by rectangles. MinEqui $\mathcal{N}_m^2$-Disj-DNF$_U$ is the problem of partitioning orthogonal polygons into rectangles.

It turns out that MinEqui C and MinEqui C_U are equivalent for the constant dimensional domain.

Lemma 8 *Let $C = \mathcal{P}\Phi[\mathcal{N}_m^d]$ for a constant d, be closed under monotone projection. Then*

1. *MinEqui C has a polynomial time α-approximation algorithm if and only if MinEqui C_U has a polynomial time α-approximation algorithm.*

2. *MinEqui* C has a polynomial time α-approximation algorithm if and only if MinEqui* C_U has a polynomial time α-approximation algorithm.*

Proof. We prove (1). The proof of (2) is similar. Let $\mathcal{A}$ be a polynomial time α-approximation algorithm for MinEqui C_U. Let $f \in C$ be represented as $\hat{\phi}(\hat{T}_1, \ldots, \hat{T}_{t'})$

where $\hat{\phi} \in \Phi$ and $\hat{T}_i = \bigwedge_{k=1}^{d} [a_{i,k} \leq x_k < b_{i,k}]$. We build an approximation algorithm $\mathcal{B}$ for MINEQUI$^\star C$. Let $L_k = \{a_{i,k}, b_{i,k} \mid i = 1, \ldots, t', k = 1, \ldots, d\}$ and suppose $L_k = \{c_{1,k}, \ldots, c_{m_k,k}\}$ where $c_{1,k} < c_{2,k} < \cdots < c_{m_k,k}$. Algorithm $\mathcal{B}$ defines $L = L_1 \times \cdots \times L_d$. By Lemma 6 and 4 we have $f^L(x) = f(x) = f(\lfloor x \rfloor_L)$. Let A be $m_1 \times \cdots \times m_d$ matrix where $A[i_1, \ldots, i_d] = f(c_{i_1,1}, \cdots, c_{i_d,d})$. This matrix represents the function $f\mathcal{M}$ where $\mathcal{M}(i_1, \ldots, i_d) = (c_{i_1,1}, \cdots, c_{i_d,d})$. Notice that the size of this matrix is at most $\prod m_i \leq (2t')^d$ which is polynomial for a constant d. Define the inverse function $\mathcal{M}^{-1} : L \to \mathcal{N}_{m_1} \times \cdots \times \mathcal{N}_{m_d}$ where $\mathcal{M}^{-1}(c_{i_1,1}, \cdots, c_{i_d,d}) = (i_1, \ldots, i_d)$ and let $\mathcal{M}^{-}(x) = \mathcal{M}^{-1}(\lfloor x \rfloor_L)$. Notice that $\mathcal{M}^{-}$ is a monotone projection. Since C is closed under monotone projection we have $f\mathcal{M} \in \mathcal{P}\Phi[\mathcal{N}_{m_1} \times \cdots \times \mathcal{N}_{m_d}]$. Algorithm $\mathcal{B}$ runs algorithm $\mathcal{A}$ on $f\mathcal{M}$ to find $\phi \in \Phi$ and $T_1, \ldots, T_t$ such that

$$\mathcal{P}_t(T_1, \ldots, T_t) = 1, \qquad f\mathcal{M} = \phi(T_1, \ldots, T_t) \qquad \text{and} \qquad t \leq \alpha \cdot size_{\mathcal{P}\Phi}(f\mathcal{M}).$$

Consider the function

$$g = \phi(T_1\mathcal{M}^{-}, \ldots, T_t\mathcal{M}^{-}).$$

Since C is closed under monotone projection and $\mathcal{M}^{-}$ is a monotone projection we have $\mathcal{P}_t(T_1\mathcal{M}^{-}, \ldots, T_t\mathcal{M}^{-}) = 1$. Therefore, $g \in \mathcal{P}\Phi[\mathcal{N}_m^d]$. We also have by Lemma 2,

$$size_{\mathcal{P}\Phi}(g) \leq t \leq \alpha size_{\mathcal{P}\Phi}(f\mathcal{M}) \leq \alpha \cdot size_{\mathcal{P}\Phi}(f).$$

Finally, we have

$$g(x) = \phi(T_1\mathcal{M}^{-}, \ldots, T_t\mathcal{M}^{-}) = (f\mathcal{M})(\mathcal{M}^{-}(x)) = (f\mathcal{M})(\mathcal{M}^{-1}(\lfloor x \rfloor_L)) = f(\lfloor x \rfloor_L) = f(x).$$

The algorithm returns $g = \phi(T_1\mathcal{M}^{-}, \ldots, T_t\mathcal{M}^{-})$.

For the other direction, suppose that we have an algorithm for MINEQUI C. Given an $m_1 \times \cdots \times m_d$ matrix A, by the constructiveness assumption we can build a formula $f \in \mathcal{P}\Phi[\mathcal{N}_{m_1} \times \cdots \times \mathcal{N}_{m_d}]$ that represents A in time $poly(\prod m_i)$ and then using the algorithm for MINEQUI C we get the desired representation.□

3 Approximation Algorithms and Learning

In this section we show the connection between approximation and learning.

Theorem 9 *If C is α-properly exactly learnable from a set $\mathcal{O}$ of oracles and $\mathcal{O}$ is easy (resp., NP-easy) for C then* MINEQUI C *has an α-approximation algorithm (resp., with the aid of an NP-oracle).*

Proof. Let $\mathcal{A}$ be a learning algorithm that uses the oracles in $\mathcal{O}$ to learn a hypothesis of size less than α times the size of the target. Since $\mathcal{O}$ is easy for C, all the oracles in $\mathcal{O}$ can be simulated in polynomial time for C. So we can run algorithm $\mathcal{A}$ and simulate all the oracles in polynomial time. Since the learning is α-proper, the output hypothesis has size less than α times the size of the target.□

The next Theorem follows from Lemma 8 and the Consistence Theorem [PV88].

Theorem 10 *Let $C = \mathcal{P}\Phi[\mathcal{N}_m^d]$ for a constant d be closed under monotone projection. The following three problems have the same time complexity.*

1. *C is α-properly PAC-learnable.*

2. *There is an α-approximation algorithm for* MINEQUI* *C.*

3. *There is an α-approximation algorithm for* MINEQUI* *C_U.*

Theorem 11 *Let $C = \mathcal{P}\Phi[\mathcal{N}_m^d]$ for a constant d be closed under monotone projection. The following problems have the same time complexity.*

1. *C is α-properly exactly learnable from membership and equivalence queries.*

2. *C is α-properly PAC learnable (without membership queries) under any product distribution.*

3. *There is an α-approximation algorithm for the* MINEQUI *C problem.*

Proof. We first show (1) $\equiv$ (3). By Theorem 9 and since by Lemma 3 the oracles are easy, we have (1) $\Rightarrow$ (3). Now we show (3) $\Rightarrow$ (1). Let $\mathcal{A}$ be an α-approximation algorithm for MINEQUI C. Let $f = \phi(T_1, \ldots, T_t)$ be the target function where $\phi \in \Phi$, $f \in \mathcal{P}\Phi[\mathcal{N}_m^d]$ and

$$T_i = \bigwedge_{k=1}^{d} [a_{i,k} \le x_k < b_{i,k}].$$

The learning algorithm works as follows. At each stage it holds d sets $L_1, \ldots, L_d$ where $L_k \subseteq \{a_{i,k}, b_{i,k} \mid i = 1, \ldots, t\} \cup \{0\}$. L_k is initially $\{0\}$. The elements of L_k are sorted in an increasing order:

$$L_k = \{c_{1,k} < c_{2,k} < \cdots < c_{i_k,k}\}.$$

The learning algorithm then builds the function f^L. It saves a table A of size $\prod_{k=1}^{d}(i_k + 1)$ where $A[j_1, \ldots, j_d] = f(c_{j_1,1}, \ldots, c_{j_d,d})$. This can be done using the membership oracle. Now, f^L can be defined using this table by $f^L(u) = f(\lfloor u \rfloor_L) = A[j_1, \ldots, j_d]$ where $\lfloor u_i \rfloor_{L_i} = c_{j_i}$ for $i = 1, \ldots, d$. Since $f^L \in \mathcal{P}\Phi[\mathcal{N}_m^d]$ (the class is closed under monotone projection), we can use algorithm $\mathcal{A}$ and construct a formula h that is equivalent to f^L and has size at most α times the size of f^L. Since the size of f^L is at most the size of f, the size of h is at most α times the size of f.

After we construct $h \equiv f^L(x)$, we ask the equivalence query EQ$(h(x))$. Let v be the counterexample. That is, $f^L(v) = h(v) \neq f(v)$. Then by Lemma 4 $f(\lfloor v \rfloor_L) = f^L(\lfloor v \rfloor_L) = f^L(v) \neq f(v)$. Intuitively, since $f(\lfloor v \rfloor_L) \neq f(v)$, the straight line that connects the two points v and $\lfloor v \rfloor_L$ hits the "boundary" of f. Now we give an algorithm to find a point close to this boundary and using this point we will add a new point to the lattice L that is equal to one new $a_{i,j}$ or $b_{i,j}$.

The algorithm works as follows. It first finds the smallest value k (or some k using binary search) such that

$$f(\lfloor v_1\rfloor_{L_1},\ldots,\lfloor v_{k-1}\rfloor_{L_{k-1}},v_k,v_{k+1},\ldots,v_d)\neq f(\lfloor v_1\rfloor_{L_1},\ldots,\lfloor v_{k-1}\rfloor_{L_{k-1}},\lfloor v_k\rfloor_{L_k},v_{k+1},\ldots,v_d).$$

Such a k exists since $f(\lfloor v\rfloor_L)\neq f(v)$. Then (again with a binary search) the algorithm finds an integer c such that $\lfloor v_k\rfloor_{L_k}<c\le v_k$ where

$$f(\lfloor v_1\rfloor_{L_1},\ldots,\lfloor v_{k-1}\rfloor_{L_{k-1}},c-1,v_{k+1},\ldots,v_d)\neq f(\lfloor v_1\rfloor_{L_1},\ldots,\lfloor v_{k-1}\rfloor_{L_{k-1}},c,v_{k+1},\ldots,v_d).$$

Now we prove the following.

Claim 1 *We have $c\in\{a_{i,k},b_{i,k}\}$ for some i, and $c\notin L_k$.*

Proof of Claim 1. Let $\theta(x_k)=f(\lfloor v_1\rfloor_{L_1},\ldots,\lfloor v_{k-1}\rfloor_{L_{k-1}},x_k,v_{k+1},\ldots,v_d)$. Here v_i are constant so the function θ is a function on one variable x_k. So $\theta(x_k)=\phi(\hat{T}_1,\ldots,\hat{T}_m)$ where each $\hat{T}_i$ is either 0, 1 or $[a_{i,k}\le x_k<b_{i,k}]$. Since $\theta(c-1)\neq\theta(c)$ we must have some i where either $a_{i,k}\le c$ and $a_{i,k}>c-1$, or $c-1<b_{i,k}$ and $c\ge b_{i,k}$. In the first case $c=a_{i,k}$ and in the second case $c=b_{i,k}$. Thus $c\in\{a_{i,k},b_{i,k}\}$.

Now notice that $\lfloor v_k\rfloor_{L_k}\neq v_k$ and $\lfloor v_k\rfloor_{L_k}<c\le v_k$. Therefore, c was not in L_k. $\Box$

We add c to L_k and update the table by adding all the missing values $f(v)$ where $v\in L_1\times\cdots\times L_d$. We now show that this algorithm runs in polynomial time. Notice that the number of equivalence queries is at most

$$\sum_{k=1}^{d}|\{a_{i,k},b_{i,k}\mid i=1,\ldots,t\}\cup\{0\}|\ \le\ (2t+1)d,$$

and the number of membership queries is at most the size of the table which is $(2t+1)^d$ plus the number of membership queries needed for the binary search. We do one binary search for each equivalence query. Therefore the algorithm uses at most $(2t+1)^d+(2t+1)d^2\log m$ membership queries.

We now give the proof that (2) is equivalent to (3).

To prove (2) $\Rightarrow$ (3), suppose C is α-properly PAC-learnable under any product distribution. We show that (3) is true for MinEqui C_U. Then by Lemma 8 the result follows. Let A be an $m_1\times\cdots\times m_d$ matrix, an instance for MinEqui C_U. Define the product distribution $D(i_1,\ldots,i_d)=1/(\prod_{i=1}^d m_i)$ (uniform over $\mathcal{N}_{m_1}\times\cdots\times\mathcal{N}_{m_d}$). We now run algorithm $\mathcal{A}$ with error $\epsilon=1/(2\prod_{i=1}^d m_i)$. The hypothesis we get is consistent with A with probability at least $1-\delta$ and has size at most α times the target size.

To prove (3) $\Rightarrow$ (2), let $\mathcal{A}$ be an α-approximation algorithm for MinEqui C. Let $r(1/\epsilon,1/\delta)$ be the number of examples needed to learn C, assuming we have unlimited computational power. This r is polynomial and can be upper bounded by the VCdimension Theorem [BEHW89]. Let $\mathcal{B}$ be a polynomial time (nonproper) PAC-learning algorithm for C under any distribution D. Such an algorithm exists. See for example [B98]. The idea of the proof is very simple. Since we cannot use membership queries, we learn a nonproper hypothesis $\hat{h}$ that is close to the target function f and then use $\hat{h}$ for membership queries.

We do that by first running $\mathcal{B}$ to nonproperly learn the target function with a small error. Algorithm $\mathcal{B}$ will output some hypothesis $\hat{h}$. Then we use the hypothesis $\hat{h}$ to simulate membership queries of f. We show that when the distribution is the product distribution then with high probability $\hat{h}$ simulates membership queries of f.

We define the following algorithm to learn f.

Proper_Learning

1. Run $\mathcal{B}$ with $\hat{\epsilon} = 1/(8r^d)$ where $r = r(1/\epsilon, 8)$ and $\hat{\delta} = 1/8$. Let $\hat{h}$ be the output hypothesis.
2. Get $r = r(1/\epsilon, 8)$ examples $(x^{(1)}, f(x^{(1)})), \ldots, (x^{(r)}, f(x^{(r)}))$.
3. Define $L_i = \{x_i^{(j)} \mid j = 1, \ldots, r\}$ and $L = L_1 \times \cdots \times L_d$.
4. Define the $\hat{m}_1 \times \cdots \times \hat{m}_d$ matrix A, where $\hat{m}_i = |L_i|$, and $A[i_1, \ldots, i_d] = \hat{h}(x_1^{(i_1)}, \ldots, x_d^{(i_d)})$.
5. Run $\mathcal{A}$ on A and let $h : \mathcal{N}_{\hat{m}_1} \times \cdots \times \mathcal{N}_{\hat{m}_d} \to \{0, 1\}$ be the output.
6. Define $g = h\mathcal{M}^\star$ where $\mathcal{M}^\star(x) = \mathcal{M}^{-1}(\lfloor x \rfloor_L)$, and $\mathcal{M}^{-1}(x_1^{(i_1)}, \ldots, x_d^{(i_d)}) = (i_1, \ldots, i_d)$.
7. Output (g)

In algorithm **Proper_Learning**, step 1 learns some function $\hat{h}$. Steps 2-3 take examples and build a lattice L. Since we do not have membership queries to find the value of the target on this lattice we use instead $\hat{h}$ to find the values. Steps 4-6 build a consistent hypothesis.

We now show that with probability 7/8 all of the membership queries that are simulated by $\hat{h}$ give a correct answer. Notice that since the distribution is the product distribution, $x_1^{(i_1)}, \ldots, x_d^{(i_d)}$ are chosen independently. Therefore,

$$\Pr[\hat{h}(x_1^{(i_1)}, \ldots, x_d^{(i_d)}) \neq f(x_1^{(i_1)}, \ldots, x_d^{(i_d)})] \leq \hat{\epsilon}$$

and

$$\Pr[\exists x \in L \ : \ \hat{h}(x) \neq f(x)] \leq \hat{\epsilon}|L| \leq \hat{\epsilon} r^d = \frac{1}{8}.$$

Since the learning algorithms $\mathcal{B}$ and $\mathcal{A}$ also have failure probability at most 1/8, algorithm **Proper_Learning** succeeds with probability at least $1 - (3/8) > 1/2$. We can run the above algorithm many times to achieve success $1 - \delta$. This completes the proof of Theorem 11.□

Corollary 12 *Let $C = \mathcal{P}\Phi[\mathcal{N}_m^d]$ for a constant d be closed under monotone projection. Then C is properly learnable from membership and equivalence queries using an oracle for NP.*

Proof. Since MinEqui C_U can be solved in polynomial time using an NP-oracle, by Theorem 11, C is properly learnable from membership and equivalence queries using an oracle for NP.□

Let $C = \mathcal{P}\Phi[\mathcal{N}_m^d]$ for a constant d be closed under monotone projection. Consider these problems:

1. C is exactly learnable with a learning algorithm that uses oracles that are easy for C, and outputs a hypothesis of size at most α times the target size.

2. C is exactly learnable with a learning algorithm that uses equivalence queries only. The hypotheses may not be of small size but the output hypothesis has size at most α times the target size.

Since all the oracles for $C = \mathcal{P}\Phi[\mathcal{N}_m^d]$ are easy, both problems give an α-approximation algorithm for MinEqui C. Therefore they are equivalent to the problems in Theorem 11.

This shows that a negative result for the α-approximation of MinEqui C will give a negative result for the α-proper learnability of C from all of the oracles mentioned in section 2.3.

4 Positive and Negative results for Proper Learning

In this section we will prove positive and negative results for the α-proper learning of different axis parallel classes.

4.1 Decision trees

In this section we give the results for Decision trees.

We first show

Lemma 13 *The class of Decision trees is easy for all the oracles.*

Proof. We show that for any $\phi : \{0,1\}^2 \to \{0,1\}$ there is a polynomial time algorithm $\mathcal{A}$ such that for any two decision trees T_f for f and T_g for g, $\mathcal{A}$ can decide in polynomial time if $\phi(f,g) \equiv 0$ and if $\phi(f,g) \not\equiv 0$ then $\mathcal{A}$ finds an assignment x_0 such that $\phi(f(x_0), g(x_0)) = 1$.

The algorithm takes the decision tree T_f and replaces each leaf v in T_f by a decision tree $T_g^v \equiv T_g$. It then takes each leaf u in T_g^v and label it with $\phi(l_v, l_u)$ where l_v is the label of v in T_f and l_u is the label of u in T_g^v. It is easy to see that this new tree computes $\phi(f,g)$. We will call this tree T'.

Each path in the tree T' from its root to a leaf labeled with 1 defines a term (see for example [B95]). If all such terms are identically 0 then there is no assignment that gives value 1 in T' and therefore $\phi(f,g) \equiv 0$. Otherwise, there is a term that is 1 for some assignment x_0 and then the algorithm returns x_0.□

Theorem 14 *If there is a proper learning algorithm for $\mathcal{N}_2^n$-decision trees from membership and equivalence queries (and other oracles) then $P = NP$.*

Proof. Decision tree is easy for all the oracles. Then, this result follows from Theorem 9 because MinEqui $\mathcal{N}_2^n$-Decision Tree is NP-complete [ZB00].□

Theorem 15 *There is a proper learning algorithm from membership and equivalence queries for $\mathcal{N}_m^d$-decision tree for constant dimension d.*

Proof. We describe an algorithm for MinEqui C_U where C is the class of $\mathcal{N}_m^2$-decision trees. The algorithm easily generalizes to other constant dimensions d. Let A be an $m \times m$ binary matrix. Each node v of the decision tree for A partitions a submatrix of A into two submatrices, which are passed to the left and right subtrees v. The partitioning continues until each submatrix contains only 0s, or only 1s, and the corresponding path in the decision tree terminates with a leaf. Define $S(a,b,x,y)$ to be the size (number of non-leaf nodes) of a minimum decision tree that partitions the submatrix with rows a through x and columns b through y. Then

$$S(a,b,x,y)=\begin{cases} 0 & \text{if the submatrix is monochrome,} \\ 1+\min\left\{ \begin{array}{l} \min_{a<i\le x}(S(a,b,i-1,y)+S(i,b,x,y)), \\ \min_{b<j\le y}(S(a,b,x,j-1)+S(a,j,x,y)) \end{array} \right\} & \text{otherwise.} \end{cases}$$

There are at most $O(m^4)$ submatrices, so the number of subproblems is polynomial, and $S(0,m-1,0,m-1)$ can be computed in time $O(m^5)$. The subproblems also provide information to build the minimal tree. For general d, the algorithm has time complexity $O(dm^{2d+1})$ which is polynomial for constant d. □

Theorem 16 *There is a proper PAC-learning algorithm for $\mathcal{N}_m^d$-decision trees for constant d.*

Proof. The same algorithm above will also solve MinEqui$^\star C_U$. By Theorem 10 the result follows.□

4.2 DNF and Union of Boxes

In this section we give the results for DNF, Union of Boxes and disjoint union of Boxes. We first prove

Theorem 17 *There is an $\epsilon<1$ such that: If the class $\mathcal{N}_m^n$-DNF is s^ϵ-properly learnable with membership and equivalence oracles (and all the other oracles) where s is the size of the DNF, then $\Sigma_2^p = P^{NP}$.*

Proof. The oracles are NP-easy for $\mathcal{N}_2^n$-DNF and approximating MinEqui$\mathcal{N}_2^n$-DNF within s^ϵ is Σ_2^p-hard [U99]. Then the result follows from Theorem 9. □

For union of boxes in a constant dimension we have the following.

Theorem 18 *For union of boxes over dimension 2 ($\mathcal{N}_m^2$-DNF) we have*

1. *There is a $\log t$-proper learning algorithm for union of boxes over dimension 2 from membership and equivalence queries, where t is the optimal number of boxes required.*

2. *There is an α such that there is an α-proper learning algorithm for union of boxes over dimension 2 from any set of oracles if and only if $P=NP$.*

Proof. Part 1 uses the fact that MinEqui$\mathcal{N}_m^2$-DNF$_U$ has a $\log t$-approximation algorithm [Fr89]. Part 2 uses the result that MinEqui$\mathcal{N}_m^2$-DNF$_U$ is NP-complete and does not admit an approximation scheme unless P=NP [BD92]. Then both parts follow from Theorem 11 and Lemma 8. □

For union of disjoint boxes ($\mathcal{N}_m^n$-disjoint DNF) we have the next Theorem.

Theorem 19 *We have*

1. *There is a proper learning algorithm for union of disjoint boxes over dimension 2 from membership and equivalence queries.*

2. *There is a proper learning algorithm for union of disjoint boxes over dimension 3 from membership and equivalence queries if and only if $P = NP$.*

Proof. This follows from Theorem 11 and Lemma 8, and the fact that problem MINEQUI $\mathcal{N}_m$-disjoint DNF$_U$ is the same as the problem of partitioning a (set of) orthogonal polygons into a minimum number of boxes. This problem is in P for dimension 2 [LLLMP79], and NP-complete for dimension 3 [DK91]. □

4.3 Multivariate Polynomials and Xor of Boxes

In this subsection we investigate the learnability of Multivariate Polynomials.

For Multivariate Polynomials with Monotone terms we have

Theorem 20 *There is a proper learning algorithm for $\mathcal{N}_m^d$-Monotone Multivariate Polynomial from membership and equivalence queries.*

Proof. We give an algorithm to optimally solve the MINEQUI C_U problem where C is the class of $\mathcal{N}_m^2$-Monotone Multivariate Polynomials (Xor of monotone rectangles). A monotone term $[x_1 \geq a][x_2 \geq b]$ in the polynomial is a rectangle with lower left corner (a, b), and covering all the points (c, d) with $(a, b) \prec (c, d)$ (i.e., $a < c$ and $b \leq d$ or $a \leq c$ and $b < d$). Any nonzero input matrix $A = (a_{ij})$ over $\{0, 1\}$ will have a point $p = (c, d)$ such that $A[p] = 1$ and $A[q] = 0$ for all $q \prec p$. Then an optimal cover must use the monotone rectangle $R = [x_1 \geq c][x_2 \geq d]$ to cover p. We update the matrix by setting $A[i, j] = 1 - A[i, j]$ for all points (i, j) covered by R. Repeating the process until $A = 0_{ij}$ will yield the minimum number of rectangles. This algorithm for the 2-dimensional case generalizes to d dimensions for any constant d. □

In a moment we will give a result for Multivariate Polynomial with nonmonotone terms. But first we consider "Almost Monotone" Multivariate Polynomials. An Almost-Monotone Multivariate Polynomial is a sum of terms that contain literals of the form $[x_i \geq \alpha]$ for $i = 1, \ldots, d$ and $[x_1 < \beta]$ (only x_1 may be negated).

Lemma 21 *There is a proper learning algorithm for $\mathcal{N}_m^d$-Almost-Monotone Multivariate Polynomial from membership and equivalence queries.*

Proof. We consider the 2-dimensional case and note that it generalizes to d dimensions. An Almost-Monotone Rectangle $[x_1 \geq a][x_1 < a'][x_2 \geq b]$ covers all points (c, d) with $a \leq c < a'$ and $d \geq b$. Suppose that we have two points (a, b), (a', b) such that $A[c, d] = 0$ for all (c, d) with $d < b$, $A[c, b] = 0$ for $c = a'$ and all $c < a$, and $A[c, b] = 1$ for $a \leq c < a'$. Then for $f = T_1 + T_2 + \cdots + T_t$ that represents A, we have $f = [x_2 \geq b]f$ and therefore without loss of generality, no term T_i contains a literal $[x_2 \geq d]$ with $d < b$.

Now consider the row of A that corresponds to $x_2 = b$. Let k be the number of transition points a for which $A[a, b] \neq A[a + 1, b]$. Then f must have at least $\lceil k/2 \rceil$ rectangles T_i covering this row (since a rectangle has two vertical edges, it can cover just two transitions).

So without loss of generality we may assume that consecutive 1s are covered by their own rectangle, and the points $(a,b) \le (c,b) < (a',b)$ are covered by $R = [x_1 \ge a][x_1 < a'][x_2 \ge b]$ in an optimal cover.

So the algorithm works as follows. It finds the two points (a,b) and (a',b) as described above. It adds rectangle $R = [x_1 \ge a][x_1 < a'][x_2 \ge b]$ to the cover. It then toggles the matrix entries that are covered by R, and recurses. $\Box$

Theorem 22 *There is a 2^{d-1}-proper learning algorithm for $\mathcal{N}_m^d$-Multivariate Polynomial from membership and equivalence queries.*

Proof. To prove the Theorem we show that every multivariate polynomial f containing t terms can be written as a sum (Xor) of at most $2^{d-1}t$ Almost-Monotone terms. Thus the algorithm to optimally solve the Almost Monotone Multivariate Polynomial problem, is a 2^{d-1}-approximation algorithm for the Multivariate Polynomial problem. Let $f = T_1 + \cdots + T_t$ be a $\mathcal{N}_m^d$-multivariate polynomial. We change each term to a sum of almost-monotone terms as follows:

$$\begin{aligned} T_i &= \prod_{k=1}^{d} [x_k \ge a_{i,k}][x_k < b_{i,k}] \\ &= [x_1 \ge a_{i,1}][x_1 < b_{i,1}] \prod_{k=2}^{d} ([x_k \ge a_{i,k}] + [x_k \ge b_{i,k}]) \end{aligned}$$

So f is a sum of at most $2^{d-1}t$ almost-monotone terms. $\Box$

References

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

[ABSS97] S. Arora, L. Babai, J. Stern, Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computing System Sciences*, 43, 317-331, 1997.

[BBBKV00] A. Beimel, F. Bergadano, N. H. Bshouty, E. Kushilevitz, S. Varricchio. Learning functions represented as multiplicity automata. *JACM* 47(3): 506-530. (2000).

[BK98] A. Beimel and E. Kushilevitz. Learning boxes in high dimension. *Algorithmica*, 22(1/2):76-90, 1998.

[BBK97] S. Ben-David, N. H. Bshouty, E. Kushilevitz. A composition theorem for learning algorithms with applications to geometric concept classes. *Proceedings of the 29th annual ACM Symposium on Theory of Computing (STOC),* (May 1997).

[BCV96] F. Bergadano, D. Catalano, S. Varricchio. Learning sat-k-DNF formulas from membership queries. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 126–130, 1996.

[BD92] P. Berman, B. DasGupta. Approximating the rectilinear polygon cover problems. In *Proceedings of the 4th Canadian Conference on Computational Geometry*, pages 229–235, 1992.

[B95] N. H. Bshouty, Exact learning of boolean functions via the monotone theory. *Information and Computation*, **123**, pp. 146-153, (1995).

[B95a] N. H. Bshouty. Simple learning algorithms using divide and conquer. In *Proceedings of the Annual ACM Workshop on Computational Learning Theory.* 1995.

[B98] N. H. Bshouty. A new composition theorem for learning algorithms. *Proceedings of the 30th annual ACM Symposium on Theory of Computing (STOC),* (May 1998).

[BCG96] N. H. Bshouty, R. Cleve, R. Gavalda, S. Kannan, C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences* 52(3): pp. 421-433 (1996).

[BGGM99] N. H. Bshouty, P. W. Goldberg, S. A. Goldman, D. H. Mathias. Exact learning of discretized geometric concepts. *SIAM Journal of Comput.* 28(2), pp. 678-699 (1999).

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the A.C.M.*, 36(4), 929-965, 1989.

[CH96] Z. Chen and W. Homer. The bounded injury priority method and the learnability of unions of rectangles. *Annals of Pure and Applied Logic* 77(2):143-168, 1996.

[CM94] Z. Chen, W. Maass. On-line learning of rectangles and unions of rectangles, *Machine Learning*, 17(1/2), pages 201-223, 1994.

[DK91] V. J. Dielissen, A. Kaldewaij. Rectangular partition is polynomial in two dimensions but NP-complete in three. In *Information Processing Letters*, pages 1–6, 1991.

[Fr89] D. Franzblau. Performance guarantees on a sweep-line heuristic for covering rectilinear polygons with rectangles. *SIAM Journal on Discrete Math* 2, pages 307-321, 1989.

[HPRW96] L. Hellerstein, K. Pillaipakkamnatt, V. Raghavan, D. Wilkins. How many queries are needed to learn? *JACM*, 43(5), pages 840-862, 1996.

[KS01] A. Klivans and R. Servedio. Learning DNF in Time $2^{O(n^{1/3})}$. In 33rd Annual Symposium on Theory of Computing (STOC), 2001, pp. 258-265.

[LLLMP79] W. Lipski, Jr., E. Lodi, F. Luccio, C. Mugnai and L. Pagli. On two-dimensional data organization II. In *Fund. Inform.* 2, pages 245-260, 1979.

[MW98] W. Maass and M. K. Warmuth Efficient learning with virtual threshold gates *Information and Computation*, 1998, 141(1): 66-83

[PR96] K. Pillaipakkamnatt, V. Raghavan, On the limits of proper learnability of subclasses of DNF formulas, *Machine Learning*, 25(2/3), pages 237-263, 1996.

[PV88] L. Pitt and L.G. Valiant. Computational Limitations on Learning from Examples. *Journal of the ACM*, 35(4):965–984, October 1988.

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

[U99] C. Umans. Hardness of approximating Σ_2^p minimization problems. In *Proceedings of the 40th Symposium on Foundations of Computer Science*, 1999.

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

[ZB00] H. Zantema, H. Bodlaender. Finding small equivalent decision trees is hard. In *International Journal of Foundations of Computer Science*, 11(2): pages 343–354, 2000.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'