Non-adaptive Learning of a Hidden Hypergraph

Hasan Abasi[1], Nader H. Bshouty[1(✉)], and Hanna Mazzawi[2]

[1] Department of Computer Science Technion, 32000 Haifa, Israel
bshouty@cs.technion.ac.il
[2] Google, 76 Buckingham Palace Rd, London, USA

Abstract. We give a new deterministic algorithm that non-adaptively learns a hidden hypergraph from edge-detecting queries. All previous non-adaptive algorithms either run in exponential time or have non-optimal query complexity. We give the first polynomial time non-adaptive learning algorithm for learning hypergraph that asks an almost optimal number of queries.

1 Introduction

Let $\mathcal{G}_{s,r}$ be a set of all labeled hypergraphs of rank at most r (the maximum size of an edge $e \subseteq V$ in the hypergraph) on the set of vertices $V = \{1, 2, \ldots, n\}$ with at most s edges. Given a hidden Sperner hypergraph[1] $G \in \mathcal{G}_{s,r}$, we need to identify it by asking *edge-detecting queries*. An edge-detecting query $Q_G(S)$, for $S \subseteq V$ is: Does S contain at least one edge of G? Our objective is to *non-adaptively* learn the hypergraph G by asking as few queries as possible.

This problem has many applications in chemical reactions, molecular biology and genome sequencing, where deterministic non-adaptive algorithms are most desirable. In chemical reactions, we are given a set of chemicals, some of which react and some which do not. When multiple chemicals are combined in one test tube, a reaction is detectable if and only if at least one set of the chemicals in the tube reacts. The goal is to identify which sets react using as few experiments as possible. The time needed to compute which experiments to do is a secondary consideration, though it is polynomial for the algorithms we present. See [2,4–7,13,15,17–22,26,28,29,31,34] for more details and many other applications in molecular biology.

In all of the above applications the rank of the hypergraph and the number of edges are much smaller than the number of vertices n. Therefore, throughout the paper, we will assume that $n \geq (\max(r, s))^2$. In the full paper we show that the results in this paper are also true for all values of r, s and[2] n.

The above hypergraph $\mathcal{G}_{s,r}$ learning problem is equivalent to the problem of exact learning a monotone DNF with at most s monomials (monotone terms),

[1] The hypergraph is Sperner hypergraph if no edge is a subset of another. If it is not Sperner hypergraph then learning is not possible.

[2] It is easy to see from Lemma 4 that all the results in this paper are also true for any r, s and $n \geq r + s$.

K. Chaudhuri et al. (Eds.): ALT 2015, LNAI 9355, pp. 89–101, 2015.
DOI: 10.1007/978-3-319-24486-0_6

where each monomial contains at most r variables (s-term r-MDNF) from membership queries [1,7]. In this paper we will use the later terminology rather than the hypergraph one.

The non-adaptive learnability of s-term r-MDNF was studied in [12,20,21,25,28,29,34]. Torney, [34], first introduced the problem and gave some applications in molecular biology. The first explicit non-adaptive learning algorithm for s-term r-MDNF was given by Gao et. al., [25]. They showed that this class can be learned using a $(n,(s,r))$-cover-free family ($(n,(s,r))$-CFF). This family is a set $A \subseteq \{0,1\}^n$ of assignments such that for every distinct $i_1,\ldots,i_s,j_1,\ldots,j_r \in \{1,\ldots,n\}$ there is $a \in A$ such that $a_{i_1} = \cdots = a_{i_s} = 0$ and $a_{j_1} = \cdots = a_{j_r} = 1$. Given such a set, the "folklore algorithm" simply takes all the monomials M of size at most r that satisfy $(\forall a \in A)(M(a) = 1 \Rightarrow f(a) = 1)$. The disjunction of all such monomials is equivalent to the target function. Assuming a set of $(n,(s,r))$-CFF of size N can be constructed in time T, this algorithm learns s-term r-MDNF with N membership queries in time $O(\binom{n}{r} + T)$. Notice that, no matter what is the time complexity of constructing the $(n,(s,r))$-CFF, the folklore algorithm runs in time at least $n^{\Theta(r)}$, which is nonpolynomial for nonconstant r.

In [9,21], it is shown that any set $A \subset \{0,1\}^n$ that non-adaptively learns s-term r-MDNF is an $(n,(s-1,r))$-CFF. We also show that $A \subset \{0,1\}^n$ is an $(n,(s,r-1))$-CFF. Therefore, the minimum size of an $(n,(s-1,r))$-CFF and $(n,(s,r-1))$-CFF is also a lower bound for the number of queries (and therefore, for the time complexity) of any non-adaptively learning algorithm for s-term r-MDNF. It is known, [32], that any $(n,(s,r))$-CFF, $n \geq (\max(r,s))^2$, must have size at least $\Omega(N(s,r)\log n)$ where

$$N(s,r) = \frac{s+r}{\log\binom{s+r}{r}}\binom{s+r}{r}. \tag{1}$$

Therefore, any non-adaptive algorithm for learning s-term r-DNF must ask at least $\max(N(s-1,r),N(s,r-1))\log n = \Omega(N(s,r)\log n)$ membership queries and runs in at least $\Omega(N(s,r)n\log n)$ time.

To improve the query complexity of the folklore algorithm, many tried to construct $(n,(s,r))$-CFF with optimal size in polynomial time. That is, time $poly(n,N(s,r))$. Gao et. al. constructed an $(n,(s,r))$-CFF of size $S = (2s\log n/\log(s\log n))^{r+1}$ in time $\tilde{O}(S)$. It follows from [33] that an $(n,(s,r))$-CFF of size $O\left((sr)^{\log^* n}\log n\right)$ can be constructed in polynomial time. Polynomial time almost optimal constructions of size $N(s,r)^{1+o(1)}\log n$ for $(n,(s,r))$-CFF were given in [10–12,24]. Those constructions give almost optimal query complexity for the folklore algorithm, but still, the running time is nonpolynomial for nonconstant r.

Chin et. al. claim in [20] that they have a polynomial time algorithm that constructs an $(n,(s,r))$-CFF of optimal size. Their analysis is misleading.[3] The size is indeed optimal but the time complexity of the construction is $O(\binom{n}{r+s})$.

[3] Some parts of the construction can indeed be performed in polynomial time, but not the whole construction.

But, as we mentioned above, even if an $(n,(s,r))$-CFF can be constructed in polynomial time, the folklore learning algorithm still takes nonpolynomial time.

The first polynomial time randomized non-adaptive learning algorithm was given by Macula et. al., [28,29]. They gave several randomized non-adaptive algorithms that are not optimal in the number of queries but use a different learning algorithm that runs in polynomial time. They show that for every s-term r-MDNF f and for every monomial M in f there is an assignment a in a $(n,(s-1,r))$-CFF A such that $M(a)=1$ and all the other monomials of f are zero on a. To learn this monomial they compose every assignment in A with a set of assignments that learns one monomial.

We first use the algorithm of Macula et. al., [28,29], combined with the deterministic constructions of $(n,(r,s))$-CFF in [10–12,24] and the fact that the assignments used in any non-adaptive algorithm must be $(n,(s,r-1))$-CFF and $(n,(s-1,r))$-CFF to change their algorithm to a deterministic non-adaptive algorithm and show that it asks $N(s,r)^{1+o(1)}\log^2 n$ queries and runs in polynomial time. The query complexity of this algorithm is almost optimal in s and r but quadratic in $\log n$. We then use a new technique, similar to the one in [16], that changes any non-adaptive learning algorithm that asks $Q(r,s,n)$ queries and runs in polynomial time to a non-adaptive learning algorithm that asks $(rs)^2\cdot Q(r,s,(rs)^2)\log n$ queries and runs in polynomial time. This gives a non-adaptive learning algorithm that asks $N(s,r)^{1+o(1)}\log n$ queries and runs in $n\log n\cdot poly(N(s,r))$ time. Notice that the time complexity of this algorithm is almost linear in n compared to the folklore algorithm that runs in nonpolynomial time $n^{\Theta(r)}+n\log n\cdot N(s,r)^{1+o(1)}$. Our algorithm has almost optimal query complexity and time complexity.

The following table summarizes the results mentioned above for non-adaptive learning algorithms for s-term r-MDNF. In the table we assume that $n\geq(\max(r,s))^2$ and $r=\omega(1)$. For $r=O(1)$ the folklore algorithm is almost optimal and runs in polynomial time.[4]

References	Query Complexity	Time Complexity
[25]	$N(s,r)\cdot(r\log n/\log(s\log n))^{r+1}$	$\binom{n}{r}$
[20]	$N(s,r)\log n$	$\binom{n}{r+s}$
[10–12,24]	$N(s,r)^{1+o(1)}\log n$	$\binom{n}{r}$
Ours+[28,29]+[12]	$N(s,r)^{1+o(1)}\log^2 n$	$poly(n,N(s,r))$
Ours	$N(s,r)^{1+o(1)}\log n$	$(n\log n)\cdot poly(N(s,r))$
Ours, $r=o(s)$	$N(s,r)^{1+o(1)}\log n$	$(n\log n)\cdot N(s,r)^{1+o(1)}$
Lower Bound [21]	$N(s,r)\log n$	$(n\log n)\cdot N(s,r)$

The *adaptive* learnability of s-term r-MDNF was already studied in many papers which gave almost optimal algorithms [5–7,21]. In [5], Abasi et. al. gave a polynomial time adaptive learning algorithm for s-term r-MDNF with almost optimal query complexity.

[4] The factor of n in the lower bound of the time complexity comes from the length n of the queries.

This paper is organized as follows. Section 2 gives some definitions and preliminary results that will be used throughout the paper. Section 3 gives the first algorithm that asks $N(s,r)^{1+o(1)}\log^2 n$ membership queries and runs in time $poly(n, N(s,r))$. Section 4 gives the reduction and shows how to use it to give the second algorithm that asks $N(s,r)^{1+o(1)}\log n$ membership queries and runs in time $(n\log n)\cdot N(s,r)^{1+o(1)}$.

All the algorithms in this paper are deterministic. One can consider a randomized construction of CFF that gives a randomized algorithm that slightly improves (in the $o(1)$ of the exponent) the query and time complexity.

2 Definitions

2.1 Monotone Boolean Functions

For a vector w, we denote by w_i the ith entry of w. Let $\{e^{(i)} \mid i = 1,\ldots,n\} \subset \{0,1\}^n$ be the standard basis. That is, $e^{(i)}_j = 1$ if $i = j$ and $e^{(i)}_j = 0$ otherwise. For a positive integer j, we denote by $[j]$ the set $\{1,2,\ldots,j\}$. For two assignments $a, b \in \{0,1\}^n$ we denote by $(a \wedge b) \in \{0,1\}^n$ the bitwise AND assignment. That is, $(a \wedge b)_i = a_i \wedge b_i$.

Let $f(x_1, x_2, \ldots, x_n)$ be a boolean function from $\{0,1\}^n$ to $\{0,1\}$. For $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ and $\sigma_1, \ldots, \sigma_k \in \{0,1\} \cup \{x_1, \ldots, x_n\}$ we denote by

$$f|_{x_{i_1}\leftarrow\sigma_1, x_{i_2}\leftarrow\sigma_2, \cdots, x_{i_k}\leftarrow\sigma_k}$$

the function $f(y_1, \ldots, y_n)$ where $y_{i_j} = \sigma_j$ for all $j \in [k]$ and $y_i = x_i$ for all $i \in [n]\backslash\{i_1, \ldots, i_k\}$. We say that the variable x_i is *relevant* in f if $f|_{x_i\leftarrow 0} \not\equiv f|_{x_i\leftarrow 1}$. A variable x_i is *irrelevant* in f if it is not relevant in f. We say that the class is *closed under variable projections* if for every $f \in C$ and every two variables x_i and x_j, $i, j \leq n$, we have $f|_{x_i\leftarrow x_j} \in C$.

For two assignments $a, b \in \{0,1\}^n$, we write $a \leq b$ if for every $i \in [n]$, $a_i \leq b_i$. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is *monotone* if for every two assignments $a, b \in \{0,1\}^n$, if $a \leq b$ then $f(a) \leq f(b)$. Recall that every monotone boolean function f has a unique representation as a reduced monotone DNF, [1]. That is, $f = M_1 \vee M_2 \vee \cdots \vee M_s$ where each *monomial* M_i is an AND of input variables, and for every monomial M_i there is a unique assignment $a^{(i)} \in \{0,1\}^n$ such that $f(a^{(i)}) = 1$ and for every $j \in [n]$ where $a^{(i)}_j = 1$ we have $f(a^{(i)}|_{x_j\leftarrow 0}) = 0$. We call such assignment a *minterm* of the function f. Notice that every monotone DNF can be uniquely determined by its minterms [1]. That is, $a \in \{0,1\}^n$ is a minterm of f iff $M := \wedge_{i\in\{j:a_j=1\}} x_i$ is a monomial in f.

An *s-term r-MDNF* is a monotone DNF with at most s monomials, where each monomial contains at most r variables. It is easy to see that the class s-term r-MDNF is closed under variable projections.

2.2 Learning from Membership Queries

Consider a *teacher* that has a *target function* $f : \{0,1\}^n \to \{0,1\}$ that is s-term r-MDNF. The teacher can answer *membership queries*. That is, when receiving $a \in \{0,1\}^n$ it returns $f(a)$. A *learning algorithm* is an algorithm that can ask the teacher membership queries. The goal of the learning algorithm is to *exactly learn* (exactly find) f with a minimum number of membership queries and optimal time complexity.

Let c and $H \supset C$ be classes of boolean formulas. We say that C is *learnable from* H in time $T(n)$ with $Q(n)$ membership queries if there is a learning algorithm that, for a target function $f \in C$, runs in time $T(n)$, asks at most $Q(n)$ membership queries and outputs a function h in H that is equivalent to C. When $H = C$ then we say that C is *properly learnable* in time $T(n)$ with $Q(n)$ membership queries.

In adaptive algorithms the queries can depend on the answers to the previous queries where in non-adaptive algorithms the queries are independent of the answers to the previous queries and therefore all the queries can be asked in parallel, that is, in one step.

2.3 Learning a Hypergraph

Let $\mathcal{G}_{s,r}$ be a set of all labeled hypergraphs on the set of vertices $V = \{1, 2, \ldots, n\}$ with s edges of rank (size) at most r. A hypergraph is called Sperner hypergraph if no edge is a subset of another. Given a hidden Sperner hypergraph $G \in \mathcal{G}_{s,r}$, we need to identify it by asking *edge-detecting queries*. An edge-detecting query $Q_G(S)$, for $S \subseteq V$ is: does S contain at least one edge of G? Our objective is to learn (identify) the hypergraph G by asking as few queries as possible.

This problem is equivalent to learning s-term r-MDNF f from membership queries. Each edge e in the hypergraph corresponds to the monotone term $\wedge_{i \in e} x_i$ in f and the edge-detecting query $Q_G(S)$ corresponds to asking membership queries of the assignment $a^{(S)}$ where $a_i^{(S)} = 1$ if and only if $i \in S$. Therefore, the class $\mathcal{G}_{s,r}$ can be regarded as the set of s-term r-MDNF. The class of s-term r-MDNF is denoted by $\mathcal{G}^*_{s,r}$. Now it obvious that any learning algorithm for $\mathcal{G}^*_{s,r}$ is also a learning algorithm for $\mathcal{G}_{s,r}$.

The following example shows that learning is not possible for hypergraphs that are not Sperner hypergraphs. Let G_1 be a graph where $V_1 = \{1, 2\}$ and $E_1 = \{\{1\}, \{1, 2\}\}$. This graph corresponds to the function $f = x_1 \vee x_1 x_2$ that is equivalent to x_1 which corresponds to the graph G_2 where $V_2 = \{1, 2\}$ and $E_2 = \{\{1\}\}$. Also, no edge-detecting query can distinguish between G_1 and G_2.

We say that $A \subseteq \{0,1\}^n$ is an *identity testing set* for $\mathcal{G}^*_{s,r}$ if for every two distinct s-term r-MDNF f_1 and f_2 there is $a \in A$ such that $f_1(a) \neq f_2(a)$. Obviously, every identity testing set for $\mathcal{G}^*_{s,r}$ can be used as queries to non-adaptively learns $\mathcal{G}^*_{s,r}$. Also, if $\mathcal{A}$ is a nonadaptive algorithm that learns $\mathcal{G}^*_{s,r}$ from membership queries A then A is identity testing set for $\mathcal{G}^*_{s,r}$. We denote by $\mathrm{OPT}(\mathcal{G}^*_{s,r})$ the minimum size of an identity testing set for $\mathcal{G}^*_{s,r}$. We say that

a non-adaptive algorithm $\mathcal{A}$ is *almost optimal* if it runs in $poly(\mathrm{OPT}(\mathcal{G}^*_{s,r}), n)$ time and asks $\mathrm{OPT}(\mathcal{G}^*_{s,r})^{1+o(1)}$ queries.

2.4 Cover Free Families

An $(n,(s,r))$-cover free family ($(n,(s,r))$-CFF), [23], is a set $A \subseteq \{0,1\}^n$ such that for every $1 \leq i_1 < i_2 < \cdots < i_d \leq n$ where $d = s + r$ and every $J \subseteq [d]$ of size $|J| = s$ there is $a \in A$ such that $a_{i_k} = 0$ for all $k \in J$ and $a_{i_j} = 1$ for all $j \in [d] \backslash J$. Denote by $N(n,(s,r))$ the minimum size of such set. The lower bound in [30,32] is, for $n \geq (\max(r,s))^2$,

$$N(n,(s,r)) \geq \Omega\left(N(s,r) \cdot \log n\right) \tag{2}$$

where $N(s,r)$ is as defined in (1). It is known that a set of random

$$\begin{aligned} m &= O\left(r^{1.5}\left(\log\left(\frac{s}{r}+1\right)\right)\left(N(s,r)\cdot\log n + \frac{N(s,r)}{s+r}\log\frac{1}{\delta}\right)\right) \\ &= N(s,r)^{1+o(1)}(\log n + \log(1/\delta)) \end{aligned} \tag{3}$$

assignments $a^{(i)} \in \{0,1\}^n$, where each $a^{(i)}_j$ is 1 with probability $r/(s+r)$, is an $(n,(s,r))$-CFF with probability at least $1-\delta$.

It follows from [10–12,24] that for $n \geq (rs)^c$, for some constant c, there is a polynomial time (in the size of the CFF) deterministic construction algorithm of $(n,(s,r))$-CFF of size

$$N(s,r)^{1+o(1)} \log n \tag{4}$$

where the $o(1)$ is with respect to r. When $r = o(s)$ the construction runs in linear time [10,12].

We now show

Lemma 1. *For any r, s and $n \geq r+s$, there is a polynomial time deterministic construction algorithm of $(n,(s,r))$-CFF of size*

$$N(s,r)^{1+o(1)} \log n.$$

Proof. For $n \geq (rs)^c$, for some constant c, the result follows from [10–12,24]. For $r + s \leq n \leq (rs)^c$ we construct the $(n,(s,r))$-CFF for $N = (rs)^c$ and then truncate the vectors to length n. The size is $N(s,r)^{1+o(1)} \log(rs)^c = N(s,r)^{1+o(1)} \log n$. □

2.5 Perfect Hash Function

Let H be a family of functions $h : [n] \to [q]$. For $d \leq q$ we say that H is an (n,q,d)-*perfect hash family* ((n,q,d)-PHF) [8] if for every subset $S \subseteq [n]$ of size $|S| = d$ there is a *hash function* $h \in H$ such that $h|_S$ is injective (one-to-one) on S, i.e., $|h(S)| = d$.

In [10] Bshouty shows

Lemma 2. *Let $q \geq 2d^2$. There is a (n, q, d)-PHF of size*

$$O\left(\frac{d^2 \log n}{\log(q/d^2)}\right)$$

that can be constructed in time $O(qd^2 n \log n / \log(q/d^2))$.

We now give the following folklore results that will be used for randomized learning algorithms

Lemma 3. *Let $q > d(d-1)/2$ be any integer. Fix any set $S \subset [n]$ of d integers. Consider*

$$N := \frac{\log(1/\delta)}{\log\left(\frac{1}{1-g(q,d)}\right)} \leq \frac{\log(1/\delta)}{\log \frac{2q}{d(d-1)}}$$

uniform random hash functions $h_i : [n] \to [q]$, $i = 1, \ldots, N$ where

$$g(q,d) := \left(1 - \frac{1}{q}\right)\left(1 - \frac{2}{q}\right) \cdots \left(1 - \frac{d-1}{q}\right)$$

With probability at least $1 - \delta$ one of the hash functions is one-to-one on S.

2.6 A Lower Bound For Learning

In this subsection we prove the following lower bound

Lemma 4. *Let $n \geq r + s$. Any identity testing set $A \subseteq \{0,1\}^n$ for s-term r-MDNF is $(n, (s, r-1))$-CFF and $(n, (s-1, r))$-CFF.*

In particular, for $w = \max(r, s)$ and $d = \min(r, s)$,

1. *if $n > w^2$ then $|A| = \Omega(N(s,r) \log n)$,*
2. *if $n = w^{1+\epsilon}$ for some $1/d < \epsilon < 1$, then $|A| = \Omega(N(s,r)/\log^{O(1/\epsilon)} w)$,*
3. *if $r + s \leq n \leq w^{1+1/d}$ then $|A| = \Omega\left(\binom{n}{d}\right)$, and*
4. *for all $n \geq r + s$ we have $|A| = \Omega(\binom{s+r}{r})$.*

Proof. Consider any distinct $1 \leq i_1, \cdots, i_{r+s-1} \leq n$. To be able to distinguish between the two functions $f_1 = (x_{i_1} \cdots x_{i_r}) \vee x_{i_{r+1}} \vee \cdots \vee x_{i_{r+s-1}}$ and $f_2 = (x_{i_1} \cdots x_{i_{r-1}}) \vee x_{i_{r+1}} \vee \cdots \vee x_{i_{r+s-1}}$ we must have an assignment a that satisfies $a_{i_1} = \cdots = a_{i_{r-1}} = 1$ and $a_{i_r} = \cdots = a_{i_{r+s-1}} = 0$. Therefore A is $(n, (s, r-1))$-CFF. To be able to distinguish between the two functions $g_1 = (x_{i_1} \cdots x_{i_r}) \vee x_{i_{r+1}} \vee \cdots \vee x_{i_{r+s-1}}$ and $g_2 = x_{i_{r+1}} \vee \cdots \vee x_{i_{r+s-1}}$ we must have an assignment a that satisfies $a_{i_1} = \cdots = a_{i_r} = 1$ and $a_{i_{r+1}} = \cdots = a_{i_{r+s-1}} = 0$. Therefore A is $(n, (s-1, r))$-CFF.

The bounds *1-4* follows from the lower bounds of $(n, (s-1, r))$-CFF and $(n, (s, r-1))$-CFF in [3,32]. □

2.7 The Folklore Algorithm

The "folklore algorithm" simply construct a $(n,(s,r))$-CFF, A, and takes all the monomials M of size at most r that satisfy $(\forall a \in A)(M(a)=1 \Rightarrow f(a)=1)$. The disjunction of all such monomials is equivalent to the target function. This follows from the following two facts: (1) For any monomial M' where $M' \not\Rightarrow f$, there is an assignment $a \in A$ such that $f(a)=0$ and $M'(a)=1$. (2) $f=\wedge_{M \Rightarrow f} M$. Assuming a set of $(n,(s,r))$-CFF of size N can be constructed in time T, this algorithm learns s-term r-MDNF with N membership queries in time $O(\binom{n}{r}+T)$. In particular we have

Lemma 5. *Let $n \geq r+s$. A $(n,(s,r))$-CFF is an identity testing set for s-term r-MDNF.*

We now show

Theorem 1. *Let $n \geq r+s$. For constant r or constant s there is a non-adaptive proper learning algorithm for s-term r-MDNF that asks*

$$N(s,r)^{1+o(1)} \log n$$

queries and runs in time $poly(n, N(s,r))$.

For $n \geq (\max(r,s))^2$ the algorithm is almost optimal.

Proof. For constant r the folklore algorithm runs in polynomial time and, by Lemma 1, asks $N(s,r)^{1+o(1)} \log n$ queries. By Lemma 4 it is almost optimal for $n \geq (\max(r,s))^2$.

For constant s, every s-term r-MDNF is r^s-clause s-MCNF. We first learn f as r^s-clause s-MCNF. This takes polynomial time and $N(s,r)^{1+o(1)} \log n$ queries.[5] We then change the target to s-term r-MDNF. This can be done in polynomial time $(rs)^s$. By Lemma 4, it is almost optimal for $n \geq (\max(r,s))^2$. □

3 The First Algorithm

In this section we give the first algorithm that asks $N(s,r)^{1+o(1)} \log^2 n$ queries and runs in time $poly(n, N(s,r))$

The first algorithm is similar to the algorithm in [28,29] that were used to give a Monte Carlo randomized algorithm. Here we use the deterministic construction of CFF to change it to a deterministic algorithm and give a full analysis for its complexity.

We first prove

Lemma 6. *Let A be an $(n,(1,r))$-CFF and B be an $(n,(s-1,r))$-CFF. There is a non-adaptive proper learning algorithm for s-term r-MDNF that asks all the queries in $A \wedge B := \{a \wedge b \mid a \in A, b \in B\}$ and finds the target function in time $|A \wedge B| \cdot n$.*

In particular, $A \wedge B$ is an identity testing set for s-term r-MDNF.

[5] To show that a clause $C = x_{i_1} \vee \cdots \vee x_{i_s}$ is not in f and $f \not\Rightarrow C$ we need an assignment a such that $C(a)=0$ and $f(a)=1$. Therefore, f can be learned with an $(n,(s,r))$-CFF.

Proof. Let f be the target function. For every $b \in B$, let $A_b = A \wedge b := \{a \wedge b \mid a \in A\}$. Let I_b be the set of all $i \in [n]$ such that $(a \wedge b)_i \geq f(a \wedge b)$ for all $a \in A$. Let $T_b := \wedge_{i \in I_b} x_i$. We will show that

1. If T is a term in f then there is $b \in B$ such that $T_b \equiv T$.
2. Either $T_b = \wedge_{i \in [n]} x_i$ or T_b is a subterm of one of terms of f.

To prove 1, let T be a term in f and let $b \in B$ be an assignment that satisfies T and does not satisfy the other terms. Such assignment exists because B is $(n, (s-1, r))$-CFF. Notice that $f(x \wedge b) = T(x) = T(x \wedge b)$. If x_i is in T and $f(a \wedge b) = 1$ then $T(a \wedge b) = T(a) = f(a \wedge b) = 1$ and $(a \wedge b)_i = 1$. Therefore $i \in I_b$ and x_i in T_b. If x_i not in T then since A is $(n, (1, r))$-CFF there is $a' \in A$ such that $T(a') = 1$ and $a'_i = 0$. Then $(a' \wedge b)_i = 0$ where $f(a' \wedge b) = 1$. Therefore i is not in I_b and x_i is not in T_b. Thus, $T_b \equiv T$.

We now prove 2. We have shown in 1 that if b satisfies one term T then $T_b \equiv T$. If b does not satisfy any one of the terms in f then $f(a \wedge b) = 0$ for all $a \in A$ and then $T_b = \wedge_{i \in [n]} x_i$. Now suppose b satisfies at least two terms T_1 and T_2. Consider any variable x_i. If x_i is not in T_1 then as before x_i will not be in T_b. This shows that T_b is a subterm of T_1. □

This gives the following algorithm

Learn$(\mathcal{G}^*_{s,r})$
1) Construct an $(n, (1, r))$-CFF A and an $(n, (s-1, r))$-CFF B.
2) Ask membership queries for all $a \wedge b$, $a \in A$ and $b \in B$.
3) For every $b \in B$.
4) $\quad T_b \leftarrow 1$.
5) $\quad$ For every $i \in [n]$.
6) $\quad\quad$ If for all $a \in A$, $(a \wedge b)_i \geq f(a \wedge b)$
7) $\quad\quad\quad$ then $T_b \leftarrow T_b \wedge x_i$.
8) $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{T_b\}$.
9) Remove from $\mathcal{T}$ the term $\wedge_{i \in [n]} x_i$
and all subterms of a larger term.

Fig. 1. An algorithm for learning $\mathcal{G}^*_{s,r}$.

We now have

Theorem 2. *Let $n \geq r + s$. There is a non-adaptive proper learning algorithm for s-term r-MDNF that asks*

$$N(s,r)^{1+o(1)} \log^2 n$$

queries and runs in time poly$(n, N(s,r))$.

For $n \geq (\max(r,s))^2$ the algorithm is almost optimal.

Proof. For constant r or constant s the result follows from Theorem 1. Let $r, s = \omega(1)$. By Lemma 1, constructing a $(n,(1,r))$-CFF of size $|A| = r^2 \log n$ and a $(n,(s-1,r))$-CFF of size $|B| = N(s-1,r)^{1+o(1)} \log n = N(s,r)^{1+o(1)} \log n$ takes $poly(n, N(s,r))$ time. By Lemma 6, the learning takes time $|A \wedge B| \cdot n = poly(n, N(s,r))$ time. The number of queries of the algorithm is $|A \wedge B| \leq |A| \cdot |B| = N(s,r)^{1+o(1)} \log^2 n$ for $r, s = \omega(1)$.

By Lemma 4 the algorithm is almost optimal for $n \geq (\max(r,s))^2$. □

A randomized algorithm with a better query complexity can be obtained using a randomized $(n,(1,r))$-CFF, B, and a randomized $(n,(s-1,r))$-CFF, A.

4 The Second Algorithm

In this section we give the second algorithm

We first prove the following result

Lemma 7. *Let C be a class of boolean functions that is closed under variable projection. Let H be a class of boolean functions and suppose there is an algorithm that, given $f \in H$ as an input, finds the relevant variables of f in time $R(n)$.*

If C is non-adaptively learnable from H in time $T(n)$ with $Q(n)$ membership queries then C is non-adaptively learnable from H in time

$$O\left(qd^2 n \log n + \frac{d^2 \log n}{\log(q/(d+1)^2)}(T(q) + Q(q)n + R(q))\right)$$

with

$$O\left(\frac{d^2 Q(q)}{\log(q/(d+1)^2)} \log n\right)$$

membership queries where d is an upper bound on the number of relevant variables in $f \in C$ and q is any integer such that $q \geq 2(d+1)^2$.

Proof. Consider the algorithm in Figure 2. Let $\mathcal{A}(n)$ be a non-adaptive algorithm that learns C from H in time $T(n)$ with $Q(n)$ membership queries. Let $f \in C_n$ be the target function. Consider the $(n, q, d+1)$-PHF P that is constructed in Lemma 2 (Step 1 in the algorithm). Since C is closed under variable projection, for every $h \in P$ the function $f_h := f(x_{h(1)}, \ldots, x_{h(n)})$ is in C_q. Since the membership queries to f_h can be simulated by membership queries to f there is a set of $|P| \cdot Q(q)$ assignments from $\{0,1\}^n$ that can be generated from $\mathcal{A}(q)$ that non-adaptively learn f_h for all $h \in P$ (Step 2 in the algorithm). The algorithm $\mathcal{A}(q)$ learns $f'_h \in H$ that is equivalent to f_h.

Then the algorithm finds the relevant variables of each $f'_h \in H$ (Step 3 in the algorithm). Let V_h be the set of relevant variables of f'_h and let $d_{max} = \max_h |V_h|$. Suppose $x_{i_1}, \ldots, x_{i_{d'}}$, $d' \leq d$ are the relevant variables in the target function f. There is a map $h' \in P$ such that $h'(i_1), \ldots, h'(i_{d'})$ are distinct and therefore $f'_{h'}$ depends on d' variables. In particular, $d' = d_{max}$ (Step 4 in the algorithm).

After finding $d' = d_{max}$ we have: Every h for which f'_h depends on d' variables necessarily satisfies $h(i_1), \ldots, h(i_{d'})$ are distinct. Consider any other non-relevant variable $x_j \notin \{x_{i_1}, \ldots, x_{i_{d'}}\}$. Since P is $(n, q, d+1)$-PHF, there is $h'' \in P$ such that $h''(j), h''(i_1), \ldots, h''(i_{d'})$ are distinct. Then $f'_{h''}$ depends on $x_{h''(i_1)}, \ldots, x_{h''(i_{d'})}$ and not on $x_{h''(j)}$. This way the non-relevant variables can be eliminated. This is Step 6 in the algorithm. Since the above is true for every non-relevant variable, after Step 6 in the algorithm, the set X contains only the relevant variables of f. Then in Steps 7 and 8, the target function f can be recovered from any f'_{h_0} that satisfies $|V(h_0)| = d'$. □

Algorithm Reduction I
$\mathcal{A}(n)$ is a non-adaptive learning algorithm for C from H.
1) Construct an $(n, q, d+1)$-PHF P.
2) For each $h \in P$
    Run $\mathcal{A}(q)$ to learn $f_h := f(x_{h(1)}, \ldots, x_{h(n)})$.
    Let $f'_h \in H$ be the output of $\mathcal{A}(q)$.
3) For each $h \in P$
    $V_h \leftarrow$ the relevant variables in f'_h
4) $d_{max} \leftarrow \max_h |V_h|$.
5) $X \leftarrow \{x_1, x_2, \ldots, x_n\}$.
6) For each $h \in P$
    If $|V_h| = d_{max}$ then $X \leftarrow X \backslash \{x_i \mid x_{h(i)} \notin V_h\}$
7) Take any h_0 with $|V_{h_0}| = d_{max}$
8) Replace each relevant variable x_i in f'_{h_0} by $x_j \in X$ where $h_0(j) = i$.
9) Output the function constructed in step (8).

Fig. 2. Algorithm Reduction.

We now prove

Theorem 3. *Let $n \geq r + s$. There is a non-adaptive proper learning algorithm for s-term r-MDNF that asks*

$$N(s,r)^{1+o(1)} \log n$$

queries and runs in time $(n \log n) \cdot poly(N(s,r))$ time.

For $n \geq (\max(r,s))^2$ the algorithm is almost optimal.

Proof. For constant r or constant s the result follows from Theorem 1. Let $r, s = \omega(1)$. We use Lemma 7. $C = H$ is the class of s-term r-MDNF. This class is closed under variable projection. Given f that is s-term r-MDNF, one can find all the relevant variables in $R(n) = O(sr)$ time. The algorithm in the previous section runs in time $T(n) = poly(n, N(s,r))$ and asks $Q(n) = N(s,r)^{1+o(1)} \log^2 n$ queries. The number of variables in the target is bounded by

$d = rs$. Let $q = 3r^2s^2 \geq 2d^2$. By Lemma 7, and since $r, s = \omega(1)$, there is a non-adaptive algorithm that runs in time

$$O\left(qd^2n\log n + \frac{d^2\log n}{\log(q/d^2)}(T(q)n + R(q))\right) = (n\log n)poly(N(r,s))$$

and asks

$$O\left(\frac{d^2Q(q)}{\log(q/d^2)}\log n\right) = N(s,r)^{1+o(1)}\log n$$

membership queries.

By Lemma 4 the algorithm is almost optimal for $n \geq (\max(r,s))^2$. □

A randomized algorithm with a better query complexity can be obtained using a randomized $(n, q, d+1)$-PHF, a randomized $(n,(1,r))$-CFF, B, and a randomized $(n,(s-1,r))$-CFF, A.

Acknowledgments. We would like to thank the reviewers for their helpful comments.

References

1. Angluin, D.: Queries and Concept Learning. Machine Learning **2**(4), 319–342 (1987)
2. Alon, N., Asodi, V.: Learning a Hidden Subgraph. SIAM J. Discrete Math. **18**(4), 697–712 (2005)
3. Abdi, A.Z., Bshouty, N.H.: Lower Bounds for Cover-Free Families. CoRR abs/1502.03578 (2015)
4. Alon, N., Beigel, R., Kasif, S., Rudich, S., Sudakov, B.: Learning a Hidden Matching. SIAM J. Comput. **33**(2), 487–501 (2004)
5. Abasi, H., Bshouty, N.H., Mazzawi, H.: On exact learning monotone DNF from membership queries. In: Auer, P., Clark, A., Zeugmann, T., Zilles, S. (eds.) ALT 2014. LNCS, vol. 8776, pp. 111–124. Springer, Heidelberg (2014)
6. Angluin, D., Chen, J.: Learning a Hidden Hypergraph. Journal of Machine Learning Research **7**, 2215–2236 (2006)
7. Angluin, D., Chen, J.: Learning a Hidden Graph using $O(\log n)$ Queries per Edge. J. Comput. Syst. Sci. **74**(4), 546–556 (2008)
8. Alon, N., Moshkovitz, D., Safra, S.: Algorithmic construction of sets for k-restrictions. ACM Transactions on Algorithms **2**(2), 153–177 (2006)
9. Bshouty, N.H.: Exact learning from membership queries: some techniques, results and new directions. In: Jain, S., Munos, R., Stephan, F., Zeugmann, T. (eds.) ALT 2013. LNCS, vol. 8139, pp. 33–52. Springer, Heidelberg (2013)
10. Bshouty, N.H.: Linear time constructions of some d-Restriction problems. In: Paschos, V.T., Widmayer, P. (eds.) CIAC 2015. LNCS, vol. 9079, pp. 74–88. Springer, Heidelberg (2015)
11. Bshouty, N.H.: Testers and their Applications. Electronic Collouium on Computational Complexity (ECCC) 19:11 (2012). ITCS 2014, pp. 327–352 (2014)
12. Bshouty, N.H., Gabizon, A.: Almost Optimal Cover-Free Family (In preperation)

13. Beigel, R., Alon, N., Kasif, S., Serkan Apaydin, M., Fortnow, L.: An optimal procedure for gap closing in whole genome shotgun sequencing. In: RECOMB 2001, pp. 22–30 (2001)
14. Bshouty, N.H., Goldman, S.A., Hancock, T.R., Matar, S.: Asking Questions to Minimize Errors. J. Comput. Syst. Sci. **52**(2), 268–286 (1996)
15. Bouvel, M., Grebinski, V., Kucherov, G.: Combinatorial search on graphs motivated by bioinformatics applications: a brief survey. In: Kratsch, D. (ed.) WG 2005. LNCS, vol. 3787, pp. 16–27. Springer, Heidelberg (2005)
16. Bshouty, N.H., Hellerstein, L.: Attribute-Efficient Learning in Query and Mistake-bound Models. COLT **1996**, 235–243 (1996)
17. Chang, H., Chen, H.-B., Fu, H.-L., Shi, C.-H.: Reconstruction of hidden graphs and threshold group testing. J. Comb. Optim. **22**(2), 270–281 (2011)
18. Chang, H., Fu, H.-L., Shih, C.-H.: Learning a hidden graph. Optim, Lett. (2014)
19. Chen, H.-B., Hwang, F.K.: A survey on nonadaptive group testing algorithms through the angle of decoding. J. Comb. Optim. **15**(1), 49–59 (2008)
20. Chin, F.Y.L., Leung, H.C.M., Yiu, S.-M.: Non-adaptive complex group testing with multiple positive sets. Theor. Comput. Sci. **505**, 11–18 (2013)
21. Du, D.Z., Hwang, F.: Pooling Design and Nonadaptive Group Testing: Important Tools for DNA Sequencing. World Scientific, Singapore (2006)
22. D'yachkov, A., Vilenkin, P., Macula, A., Torney, D.: Families of finite sets in which no intersection of ℓ sets is covered by the union of s others. J. Comb Theory Ser A. **99**, 195–218 (2002)
23. Kautz, W.H., Singleton, R.C.: Nonrandom binary superimposed codes. IEEE Trans. Inform. Theory **10**(4), 363–377 (1964)
24. Fomin, F.V., Lokshtanov, D., Saurabh, S.: Efficient Computation of Representative Sets with Applications in Parameterized and Exact Algorithms. SODA **2014**, 142–151 (2014)
25. Gao, H., Hwang, F.K., Thai, M.T., Wu, W., Znati, T.: Construction of d(H)-disjunct matrix for group testing in hypergraphs. J. Comb. Optim. **12**(3), 297–301 (2006)
26. Grebinski, V., Kucherov, G.: Reconstructing a Hamiltonian Cycle by Querying the Graph: Application to DNA Physical Mapping. Discrete Applied Mathematics **88**(1–3), 147–165 (1998)
27. Kleitman, D.J., Spencer, J.: Families of k-independent sets. Discrete Mathematics **6**(3), 255–262 (1972)
28. Macula, A.J., Popyack, L.J.: A group testing method for finding patterns in data. Discret Appl Math. **144**, 149–157 (2004)
29. Macula, A.J., Rykov, V.V., Yekhanin, S.: Trivial two-stage group testing for complexes using almost disjunct matrices. Discrete Applied Mathematics **137**(1), 97–107 (2004)
30. Ma, X., Wei, R.: On Bounds of Cover-Free Families. Designs, Codes and Cryptography **32**, 303–321 (2004)
31. Reyzin, L., Srivastava, N.: Learning and verifying graphs using queries with a focus on edge counting. In: Hutter, M., Servedio, R.A., Takimoto, E. (eds.) ALT 2007. LNCS (LNAI), vol. 4754, pp. 285–297. Springer, Heidelberg (2007)
32. Stinson, D.R., Wei, R., Zhu, L.: Some New Bounds for Cover free Families. Journal of Combinatorial Theory, Series A **90**(1), 224–234 (2000)
33. Stinson, D.R., Wei, R., Zhu, L.: New constructions for perfect hash families and related structures using combintorial designs and codes. J. Combin. Designs **8**(3), 189–200 (2000)
34. Torney, D.C.: Sets pooling designs. Ann. Comb. **3**, 95–101 (1999)