Error-Tolerant Non-Adaptive Learning of a Hidden Hypergraph

Hasan Abasi
Department of Computer Science, Technion, Haifa, 32000, Israel
hassan@cs.technion.ac.il

Abstract

We consider the problem of learning the hypergraph using edge-detecting queries. In this model, the learner is allowed to query whether a set of vertices includes an edge from a hidden hypergraph. Except a few, all previous algorithms assume that a query's result is always correct. In this paper we study the problem of learning a hypergraph where α-fraction of the queries are incorrect. The main contribution of this paper is generalizing the well-known structure CFF (Cover Free Family) to be Dense (we will call it DCFF - Dense Cover Free Family) while presenting three different constructions for DCFF. Later, we use these constructions wisely to give a polynomial time non-adaptive learning algorithm for a hypergraph problem with at most α-fracion incorrect queries. The hypergraph problem is also known as both monotone DNF learning problem, and complexes group testing problem.

2012 ACM Subject Classification Theory of computation → Boolean function learning

Keywords and phrases Error Tolerant Algorithm, Hidden Hypergraph, Montone DNF, Group Testing, Non-Adaptive Learning

Digital Object Identifier 10.4230/LIPIcs.MFCS.2018.3

1 Introduction

The classical group testing model consists of a set of n items, where s of these items are defective (positive) while others are good (negative) items. The problem is to identify all defective (positive) items with a small number of group tests. A group test is a test with a negative/positive result, where you can choose an arbitrary group $S \subseteq [n]$ and ask whether S contains at least one defective item. The outcome is negative if all the items within the group are good (negative), and positive otherwise (See [13, 12] for more details about group testing).

Torney [20] was the first who generalized the group testing model to the complex group testing model, where we have the same set of n items, as in the first model, but instead of s defective elements, we have s defective groups $M_1, \cdots, M_s$, that are known as positive complexes, where $M_i \subseteq [n]$ and for any $i_1, i_2 \in [s]$ where $i_1 \neq i_2$ $M_{i_1} \not\subseteq M_{i_2}$ and $M_{i_2} \not\subseteq M_{i_1}$. In complex group testing model, a complex group test is a test with a negative/positive result, where you can choose an arbitrary group $S \subseteq [n]$ and ask whether S contains at least one positive complex. The outcome is negative if all the complexes within the group are negative, and positive otherwise. A usual assumption is that each positive complex has at most r items ($|M_i| \leq r$).

This problem is also known as graph testing, where a hypergraph $H = (V, E)$ with s edges and n vertices is given, our objective is to learn the hypergraph by asking edge-detecting queries (denote query function by Q). An edge detecting-query is a test with a negative/positive result, where you can choose an arbitrary group of vertices $S \subseteq [n]$ and ask whether this group of vertices has at least one edge or not. A usual assumption is that each edge has at most r vertices. It is easy to see that this model is equivalent to the complex

group testing model (see [1]). In addition to complex group testing and graph testing, this problem is also known as the learning monotone DNF problem with s monomials, where each monomial contains at most r variables. In this paper, we will use the latter terminology rather than complexes or graphs.

Tests can be *adaptive* (sequential), *non-adaptive* or *multi-stage*. Adaptive tests are tests that may be affected by the outcome of earlier tests. Non-adaptive tests, all tests are executed in parallel without knowledge of any other tests' outcome. Multi-stage tests are divided into stages, and all tests within each stage are executed in parallel, but different stages may be affected by the outcome of earlier stages.

This problem has many applications in chemical reactions, molecular biology and genome sequencing (see [15, 14, 7, 17, 6]). In these types of applications, an experiment corresponding to a group test could take several hours or even several days. Thus, non-adaptive algorithms are most desirable. In addition to the prolonged process, errors in applications are unavoidable. And as the number of tests increases, the number of errors occurring is increased as well. Therefore, it is very important to construct error-tolerant algorithms, where the number of the errors is a fraction of the number of queries. In all of the above applications the size of each term (rank of the hypergraph) is much smaller than the number of terms (edges), and both are much smaller than the number of variables (vertices) n. Therefore, throughout the paper, we will assume that $r < s < n$ (this assumption will be used in the constructions in Section 3) and will denote $d \triangleq r + s$.

Both adaptive and non-adaptive learning of monotone DNF problem are well-known and well-studied. In the Adaptive model, Angluin et al. [5] was the first to present a deterministic optimal adaptive algorithm for learning s-term 2-MDNF (s terms each of size 2). Later, Abasi et al., [1], gave an almost optimal adaptive algorithm for the general s-term r-MDNF case and they proved that any algorithm that learns s-term r-MDNF needs to ask at least

$$\begin{cases} r > s & \Omega\left((r/s)^{s-1} + rs\log n\right) \text{ Queries} \\ r \le s & \Omega\left((2s/r)^{r/2} + rs\log n\right) \text{ Queries} \end{cases}$$

On the non-adaptive side many works studied the s-term r-MDNF. Abasi et al. [2], gave the first deterministic algorithm which runs in polynomial time and non-adaptively learns a hypergraph that asks an almost optimal number of queries. On the other hand, only few algorithms discussed the error-tolerant problems. Stinson and Wei, [18], proved a lower bound

$$0.7c\frac{d\binom{d}{r}}{\log\binom{d}{r}}\log n + \frac{c(z-1)}{2}\binom{d}{r} = s\left(\frac{es}{r}\right)^{r+o(1)}\log n + z\left(\frac{es}{r}\right)^{r+o(1)},$$

where z is constant and the algorithm is tolerant to $z/2$ errors (constant number of errors). In the same paper they proved an upper bound of

$$O\left(z\binom{d}{r}(rs)^{\log^* n}\log n\right) = z\left(\frac{es}{r}\right)^{r+o(1)}(rs)^{\log^* n}\log n.$$

Chen et al., [11], improved the last upper bound to

$$z\left(\frac{d}{r}\right)^{r}\left(\frac{d}{s}\right)^{s}\left(1+\ln\left(\frac{n}{d}+1\right)\right) = z\left(\frac{es}{r}\right)^{r+o(1)}\log n.$$

Lang et al., [16], considered the problem where z errors occur in every m tests. They gave a random algorithm which identifies all the target function's terms with high probability. All

previous algorithms either run in exponential time, are non-deterministic or are tolerant to a constant number of errors.

In this paper we introduce the first algorithm, to the best of our knowledge, that handles α-fraction of incorrect queries which is deterministic and runs in polynomial time. In [2], the authors use the $(n,(s,r))$-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 an assignment $a \in A$ such that

$$a_{i_1} = \cdots = a_{i_s} = 0 \text{ and } a_{j_1} = \cdots = a_{j_r} = 1.$$

In this paper we extend the definition of $(n,(s,r))$-CFF to be

$$(n,(s,r),\alpha)\text{- } \textbf{Dense CFF (DCFF)},$$

where the DCFF is a CFF with an additional requirement, which is that for every distinct $i_1,\ldots,i_s,j_1,\ldots,j_r \in \{1,\ldots,n\}$, an ***α*-fraction** of the assignments $a \in A$ satisfy the following:

$$a_{i_1} = \cdots = a_{i_s} = 0 \text{ and } a_{j_1} = \cdots = a_{j_r} = 1.$$

Improving techniques that were used in [2], by customizing them to comply with DCFF, results in the needed algorithm. In the improvement process we used what we defined above as DCFF. In order to do so, we present three constructions of DCFF, each of which has an advantage over the other. One of these constructions is superior in terms of density, while the other two have better time complexities. One of them is better when $r < T(s)$, while the other is better when $r \geq T(s)$, where T is a function of s. In addition to the extension described above, we also prove that α in $(n,(s,r),\alpha)$-DCFF cannot exceed

$$DNS(s,r) \triangleq \frac{1}{\binom{r+s}{r}} = \frac{1}{\binom{d}{r}}.$$

It is also known, [19], that any non-adaptive learning algorithm must ask at least

$$\Omega\left(\left(\frac{es}{r}\right)^r \log n\right) = \Omega\left(s^{r(1+o(1))} \log n\right)$$

queries, and therefore any algorithm must run in time $poly\,(n,s^r)$. An algorithm that runs in time $poly\,(n,s^r)$ is called efficient algorithm.

This paper is organized as follows. Section 2 gives some definitions and preliminary results that will be used throughout the paper. Section 3 gives three different algebraic constructions for Dense Cover Free family (see Section 3 for full definition). Section 4 gives an inefficient algorithm by using DCFF directly. Section 5 gives a reduction to reduce any non-linear α-error tolerant algorithm that depends on d variables at most to be linear in $\log n$. And finally in Section 6 we use all the constructions and the reduction to generate an efficient algorithm that asks

$$O\left(r^{11}\,(4s)^{r+7} \log n\right) = O\left(s^{r(1+o(1))} \log n\right)$$

noisy membership queries and runs in $poly\,(n,s^r)$, and handles

$$\Omega\left(DNS(s,r)^{1+o(1)}\right) \text{ fraction of incorrect queries.}$$

1.1 Optimality for $r = 2$ (or any constant r)

The learning problem of a graph (when $r = 2$) has been considered and well-studied. Angluin et al., [4, 5], studied the graph problem for the adaptive, nonadaptive and multistage models. Our construction and algorithm are almost optimal when $r = 2$ (or any constant r) in terms of density and query complexity, where the density in case r is constant is $\Omega\left(DNS(s,r)\right)$.

1.2 Algorithm idea overview

One can easily see that using $(n,(s,r),\alpha)$-DCFF solves the problem of s-term r-MDNF when less than $\alpha/2$-fraction of the queries can be incorrect. That is true because for each subset of s terms there are at least $\alpha/2$-fraction of the DCFF's assignments that satisfy one term (r 1's) and assigns 0 to all other terms ($s-1$ 0's) (see Section 4 for more details).
The problem with using DCFF directly is the time complexity, and that is because we need to find out whether each possible subset of s terms each of size at most r satisfies a given set of conditions or not (again, see Section 4 for more details).
In order to achieve a polynomial time complexity, we use bit-wise conjunction of assignments in each DCFF. The general idea is to create two DCFFs. One of them spreads the set of terms over different sets, and the other is used afterward with other structures and techniques to find each term explicitly (see Section 6 and Section 5 for more details).

2 Definitions and Preliminary Results

2.1 Monotone Boolean Functions

For a vector w, we denote by w_i the ith entry of w. 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, $(a\wedge b)_i = a_i\wedge b_i$ for all $i\in[n]$.

Let $f(x_1,x_2,\ldots,x_n)$ be a Boolean function from $\{0,1\}^n$ to $\{0,1\}$. 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 of functions C is *closed under variable projections* if for every function $f\in C$ 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 said to be *monotone* if for every two assignments $a,b\in\{0,1\}^n$, if $a\leq b$ then $f(a)\leq f(b)$. In other words, there is no negated variables in f. Every monotone Boolean function f has a unique representation as a *reduced monotone* DNF, [3].

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.

For a function $f=M_1\vee\cdots\vee M_s$ that belongs to an s-term r-MDNF class, we say that $b\in\{0,1\}^n$ **separates** a term M_i in f from other terms if $M_i(b)=1$ and $M_j(b)=0$ for any $i\neq j$. In the same way, we say that $b\in\{0,1\}^n$ **separates** the terms $M_{i_1},\ldots,M_{i_k}$ in f from other terms if $M_j(b)=1$ for any $j\in\{i_1,\ldots,i_k\}$, and otherwise $M_j(b)=0$.

2.2 α-Error Tolerant Learning from Noisy Membership Queries

Consider a *teacher* that has a *target function* $f:\{0,1\}^n\to\{0,1\}$ that is an s-term r-MDNF. The teacher can answer *noisy membership queries*. That is, when receiving $a\in\{0,1\}^n$, it returns $\overline{f(a)}=1-f(a)$ for α-fraction of the queries, and $f(a)$ for the rest. This teacher is called *α-Liar Teacher*. An *α-error tolerant learning algorithm* is an algorithm that can

ask the α-*Liar Teacher* noisy membership queries. The goal of the learning algorithm is to *exactly learn* (exactly find) f with a minimum number of noisy membership queries and optimal time complexity.

2.3 Dense Perfect Hash Function, Cover Free Family and Dense Cover Free Family

▶ **Definition 1** (Perfect Hash Family)**.** Let H be a family of functions $h : [n] \to [q]$. For $d \leq q$ we say that H is an $(n, q, d, 1-\epsilon)$- *dense perfect hash family* ($(n, q, d, 1-\epsilon)$-DPHF) if for every subset $S \subseteq [n]$ of size $|S| = d$ there are at least $(1-\epsilon)|H|$ *hash functions* $h \in H$ such that $h|_S$ is injective (one-to-one) on S, i.e., $|h(S)| = d$.

▶ **Lemma 2** ([8])**.** *Let q be a power of prime. If $\epsilon > 4(d(d-1)/2+1)/q$, then there is a $(n, q, d, 1-\epsilon)$-DPHF of size*

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

that can be constructed in linear time.

▶ **Definition 3** (Cover Free Family)**.** An $(n, (s, r))$- *cover free family* ($(n, (s, r))$-CFF), 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.

▶ **Definition 4** (Dense Cover Free Family)**.** An $(n, (s, r), \alpha)$- *dense cover free family* ($(n, (s, r), \alpha)$-DCFF), 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 $|J| = s$, the following holds for α-fraction vectors $a \in A$: $a_{i_k} = 0$ for all $k \in J$ and $a_{i_j} = 1$ for all $j \in [d]\backslash J$. Denote by $N(n, (s, r), \alpha)$ the minimum size of such set.

▶ **Lemma 5.** *Let A be an $(n, (s, r), \alpha)$ -DCFF, then $\alpha \leq \frac{1}{\binom{d}{r}} \triangleq DNS(s, r)$.*

Proof. For any $d = r + s$ variables, we must have exactly $\binom{d}{r}$ different assignments to cover all the possible assignments over d variables. Otherwise, one assignments at least will be missed. Denote these assignments by R. If there is at least one assignment $r_0 \in R$ that appears among $(n, (s, r), \alpha)$-DCFF more than $1/\binom{d}{r}$ times, then by pigeonhole principle, there exists at least one assignment $r_1 \in R$, that appears less than $1/\binom{d}{r}$ times. Therefore, α cannot exceed $1/\binom{d}{r}$. ◀

For completeness we show:

▶ **Lemma 6.** *Let H_1 be an $(n, q, d, 1-\epsilon)$-DPHF, and H_0 is a $(q, (r, s), \alpha)$-DCFF, then $DC = \{h_0(h_1)|h_0 \in H_0, h_1 \in H_1\}$ is an $(n, (r, s), \alpha(1-\epsilon))$-DCFF of size $|H_0||H_1|$.*

Proof. We will show that for every $1 \leq i_1 < i_2 < \cdots < i_d \leq n$ where $d = s + r$ and for every $J \subseteq [d]$ of size $|J| = s$ there are $\alpha(1-\epsilon)|H_0||H_1|$ vectors $h \in DC$ such that $h(i_k) = 0$ for all $k \in J$, and $h(i_j) = 1$ for all $j \in [d]\backslash J$.
Since H_1 is $(n, q, d, 1-\epsilon)$-DPHF, then there are $(1-\epsilon)|H_1|$ functions $h_1 \in H$ where $h_1(i_1), \ldots, h_1(i_d)$ are distinct. And since H_0 is a $(q, (r, s), \alpha)$-DCFF, then for each $h_1(i_1), \ldots, h_1(i_d)$ distinct values there are $\alpha|H_0|$ vectors $h_0 \in H_0$, where for every $J \subseteq [d]$ of size $|J| = s$, such that $h_0(h_1(i_k)) = 0$ for all $k \in J$ and $h_0(h_1(i_j)) = 1$ for all $j \in [d]\backslash J$. Accordingly, there are $(1-\epsilon)|H_1| \cdot \alpha|H_0|$ vectors $h \in DC$ where for every $1 \leq i_1 < i_2 < \cdots < i_d \leq n$ where $d = s + r$ and for every $J \subseteq [d]$ of size $|J| = s$, $h(i_k) = 0$ for all $k \in J$ and $h(i_j) = 1$ for all $j \in [d]\backslash J$. ◀

Table 1 Dense CFF, assuming $r \leq O\left((\log^2 d)/(\log\log d)\right)$.

Constr.	Construction Time	Size	Density
I	$O\left(d^{\sqrt{\frac{r}{\log r}}}\left(\frac{d}{r}\right)^r n\log n\right)$	$O\left(d^{\sqrt{\frac{r}{\log r}}}\left(\frac{d}{r}\right)^r \log n\right)$	$\Omega\left(DNS(s,r)d^{-\sqrt{\frac{r}{\log r}}}\right)$
II	$O\left(c^d \cdot d^{2d+6}s^2\left(\frac{es}{r}\right)^{2r} n\log n\right)$	$O\left(d^3 s^2\left(\frac{es}{r}\right)^r \log n\right)$	$\Omega\left(DNS(s,r)\right)$
III	$O\left(d^3\left(\frac{ed^3}{r}\right)^{r+2}(4s)^{2r+4} n\log n\right)$	$O\left(d^3(4s)^{r+2}\log n\right)$	$\Omega\left(\frac{DNS(s,r)}{(c\cdot r)^r}\right)$

3 Explicit Dense Cover Free Family Constructions

In this section we will show three different constructions of $(n,(r,s),\alpha)$-DCFF - each of which has an advantage on the other. Before we introduce the constructions, we will state the results for each construction in **Table** 1.

Note that construction II is the best construction in terms of density. When r is non-constant then construction I and III are much better than construction II in terms of time complexity. Accordingly, when

$$\omega(1) < r < O\left(\frac{\log^2 d}{\log\log d}\right),$$

then we prefer construction III over I. Otherwise we prefer construction I over III.

In our algorithms, we will distinguish between the cases r is constant and r is non-constant. When r is non-constant, we will only deal with the case where $\omega(1) < r < O\left((\log^2 d)/(\log\log d)\right)$, and it can be done similarly for the other case.

3.1 Explicit Construction

Before we start our explicit constructions, we will start with lemmas from [10, 9], that are necessary for our constructions.

▶ **Lemma 7** ([9] - Derandomization lemma). *Let S be a finite sample space with a probability distribution D.*
Let $X_1,\ldots,X_N$ be random variables over S that take values from $\{0,1\}$. Suppose that for every $s \in S = S_1 \times S_2 \times \cdots \times S_n$, the values $X_1(s),\ldots,X_N(s)$ can be computed in $\tilde{O}(N)$[1]. Let

- $N' \leq N, 0 < \epsilon \leq \frac{1}{2}$.
- $\lambda_i = (1-\epsilon)p_i$, *where* $0 < p_i \leq E_{s\sim D}[X_i(s)]$ *for all* $i \in [N'] := \{1,\ldots,N'\}$.
- $\lambda_j = (1+\epsilon)p_j$, *where* $1 > p_j \geq E_{s\sim D}[X_j(s)]$ *for all* $j \in (N',N] := \{N'+1,\ldots,N\}$.
- $\alpha = \min_i \min\left(1/(1-p_i), 1/(1-\lambda_i)\right)$ *and* $m \geq \frac{4\ln N}{\min_i p_i\epsilon^2}$.

If any expectation of the form $E[X_i(x_1,\ldots,x_n)|x_1=\xi_1,\ldots,x_j=\xi_j]$ can be computed in time T, then there is an algorithm that runs in time

$$\tilde{O}\left(T\left(|S_1|+\cdots+|S_n|\right)\cdot Nm^2\log\alpha\right), \text{ and outputs } S' = \{s_1,\ldots,s_{m+1}\} \subseteq S$$

such that for all $i \in [N']$ and $j \in (N',N]$: $E_{s\sim U_{S'}}[X_i(s)] \geq \lambda_i, E_{s\sim U_{S'}}[X_j(s)] \leq \lambda_j$, where $U_{S'}$ is the uniform distribution over S'.

[1] $\tilde{O}(N)$ is $O(N\cdot poly(\log T))$ where T is the time complexity of the construction.

In other words, the lemma says if you have a set of random variables over a big range S, in our case $|S| = 2^n$, then there is a smaller set S' that can replace S while keeping the expectation of each variable the same to the factor of $(1 \pm \epsilon)$. This lemma will help us convert a random algorithm to a deterministic one, since the set S' is "small" and we can go over all possible values in a polynomial time.

▶ **Lemma 8** ([10] - CFF explicit construction). *Fix any integers $r < s < d$ with $d = r+s$, there is an almost optimal $(n, (r, s))$-CFF, i.e., of size $N(r, s)^{1+o(1)} \log n$, where $N(r, s) = \frac{d\binom{d}{r}}{\log\binom{d}{r}}$, that can be constructed in linear time.*

▶ **Corollary 9.** *There is an $\left(n, (r, s), \frac{1}{N(r,s)^{1+o(1)} \log n}\right)$-DCFF of size $N(r, s)^{1+o(1)} \log n$, that can be constructed in linear time.*

3.2 First Construction

From the proof of Lemma 8, we infer that the accurate value of the size is

$$N(n) = t(d) \left(\frac{d}{r}\right)^r \log n.$$

where

$$t(d) = \begin{cases} 2^{O(r \log\log d)} & r > O\left(\frac{\log^2 d}{\log\log d}\right) \\ 2^{O\left(\frac{\sqrt{r}\log d}{\sqrt{\log r}}\right)} & \text{Otherwise} \end{cases}$$

Now we can build the first DCFF by Lemma 2, Lemma 6 and Lemma 8.
Simply choose $\epsilon = 0.01$ (or any small constant) and a prime $q = O(d^3)$. By Lemma 2, there is a $(n, q, d, \Omega(1))$-DPHF of size $O\left(d^2 \log n\right)$. By Corollary 9 and the claim above, for $q = O\left(d^3\right)$, there is

$$\text{an } \left(O\left(d^3\right), r, s, \Omega\left(\frac{DNS(s,r)}{t(d)}\right)\right)\text{-DCFF, that can be constructed in linear time.}$$

By using Lemma 6, we can get

$$\left(n, r, s, \Omega\left(\frac{DNS(s,r)}{t(d)}\right)\right)\text{-DCFF of size } O\left(t(d)\left(\frac{d}{r}\right)^r \log n\right).$$

▶ **Theorem 10.** *There is an $\left(n, (r, s), \Omega\left(\frac{DNS(s,r)}{t(d)}\right)\right)$-DCFF of size $O\left(d^2 \log d N(r, s)^{1+o(1)} \log n\right)$ that can be constructed in linear time.*

3.3 Second Construction

▶ **Theorem 11.** *There is an $(n, (r, s), \Omega\left(DNS(s, r)\right))$-DCFF of size $O\left(s\left(\frac{es}{r}\right)^r \log n\right)$ that can be constructed in poly $\left(n^d\right)$.*

Proof. We define a probability space over the vectors $x = (x_1, \ldots, x_n) \in \{0, 1\}^n$, where

$$x_i = \begin{cases} 0, & \text{with probability } \frac{s}{d} \\ 1, & \text{with probability } \frac{r}{d} \end{cases}$$

For every couple of sets $\{i_1, \ldots, i_r\}$, $\{j_1, \ldots, j_s\}$, we define the random variable $X_{i_1,\cdots,i_r,j_1,\cdots,j_s}$ where

$$X_{i_1,\cdots,i_r,j_1,\cdots,j_s}(x) = \begin{cases} 1, & \text{If } x_{i_1} = \cdots = x_{i_r} = 1, x_{j_1} = \cdots = x_{j_s} = 0. \\ 0, & \text{Otherwise} \end{cases}$$

Note that the number of random variables is $N = \binom{n}{d}\binom{d}{r}$, and that the expectation of each random variable is $E[X_{i_1,\cdots,i_r,j_1,\cdots,j_s}] = \left(\frac{r}{d}\right)^r \left(\frac{s}{d}\right)^s$.
We now use Lemma 7. Note that

$$N' = N, \epsilon = \tfrac{1}{2}, p_{i_1,\cdots,i_r,j_1,\cdots,j_s} = p := \left(\tfrac{r^r s^s}{d^d}\right).$$

$$\lambda_i = \lambda := \tfrac{1}{2}p, \quad m \geq \frac{4\ln\binom{n}{d}\binom{d}{r}}{\frac{p}{4}} = O\left(s\left(\tfrac{es}{r}\right)^r \log n\right).$$

Therefore, according to Lemma 7, there is an algorithm that runs in

$$\tilde{O}(nNm^2\log\alpha) = O\left(n\left(\tfrac{en}{d}\right)^d s^2 \left(\tfrac{es}{r}\right)^{2r}\log^2 n\right) \text{ time, and outputs}$$

$S' = \{s_1,\ldots,s_{m+1}\} \subseteq S$, such that, for all $i = (i_1,\ldots,i_r,j_1\ldots,j_s) \in [n]^d$, holds:
$E_{s\sim U_{S'}}[X_i(s)] \geq \frac{1}{2}\left(\frac{r}{d}\right)^r \left(\frac{s}{d}\right)^s = \Omega\left(DNS(s,r)\right)$ ◀

The size of the construction is almost optimal, while the running time is exponential. In the third construction we will reduce the exponent from d to r.

3.4 Third Construction

▶ **Theorem 12.** *There is an* $\left(n,(r,s),\Omega\left(\frac{DNS(s,r)}{(cr)^r}\right)\right)$*-DCFF of size* $O\left((4s)^{r+2}\log n\right)$ *that can be constructed in poly* (n^r).

Through applying the same technique from the **Second construction** wisely, we reduce the number of random variables this time, leading to a reduced construction time.

Proof. We define a probability space over the vectors $x = (x_1,\ldots,x_n) \in \{0,1\}^n$, where

$$x_i = \begin{cases} 1, & \text{with probability} \frac{1}{4s} \\ 0, & \text{with probability} \frac{4s-1}{4s}. \end{cases}$$

For every couple of sets $\{i_1,\ldots,i_r\}$, $\{j_1,\ldots,j_s\}$, we define the following random variables:

$$X_{i_1,\cdots,i_r,j_1,\cdots,j_s}(x) = \begin{cases} 1, & \text{If } x_{i_1} = \cdots = x_{i_r} = 1, x_{j_1} = \cdots = x_{j_s} = 0. \\ 0, & \text{Otherwise.} \end{cases}$$

$$Y_{i_1,\cdots,i_r,j}(x) = \begin{cases} 1, & \text{If } x_{i_1} = \cdots = x_{i_r} = x_j = 1. \\ 0, & \text{Otherwise.} \end{cases}, j \in \{j_1,\ldots,j_s\}$$

$$Z_{i_1,\cdots,i_r}(x) = \begin{cases} 1, & \text{If } x_{i_1} = \cdots = x_{i_r} = 1. \\ 0, & \text{Otherwise.} \end{cases}$$

Note that we can bound the random variable X, by an expression of Y and Z in the following way:

$$X_{i_1,\cdots,i_r,j_1,\cdots,j_s} \geq Z_{i_1,\cdots,i_r} - \textstyle\sum_{k=1}^{s} Y_{i_1,\cdots,i_r,j_k}$$

The correctness of the above equation follows immediately from the definition.
Note that instead of $\binom{n}{d}$ that we used in the previous construction, now we can do a similar construction by using the following number of random variables

$$N = \tbinom{n}{r+1}(r+1) + \tbinom{n}{r}$$

and the expectation of each random variable is

$$E[X_{i_1,\cdots,i_r,j_1,\cdots,j_s}] \geq E[Z_{i_1,\cdots,i_r}] - \sum_{k=1}^{s} E[Y_{i_1,\cdots,i_r,j_k}] = \frac{3}{4}\frac{1}{(4s)^r}$$

Again we use Lemma 7 for the random variables Z and Y. Note that

1. $N' = \binom{n}{r}$, $\epsilon = \frac{1}{2}$.
2. $p_{i_1,\cdots,i_r} = p_0 := \left(\frac{1}{4s}\right)^r$.
3. $p_{i_1,\cdots,i_r,j} = p_1 := \left(\frac{1}{4s}\right)^{r+1}$.
4. $\lambda_{i_1,\cdots,i_r} = \lambda_0 := \frac{1}{2}p_0$.
5. $\lambda_{i_1,\cdots,i_r,j} = \lambda_1 := \frac{3}{2}p_1$.
6. $m \geq \frac{4\ln\left(\binom{n}{r+1}(r+1)+\binom{n}{r}\right)}{\frac{p_1}{4}} = O\left((4s)^{r+2}\log n\right)$.

Therefore, according to Lemma 7, there is an algorithm that runs

$$\tilde{O}\left(nNm^2\log\alpha\right) = O\left(n\left(\frac{en}{r}\right)^{r+2}(4s)^{2r+4}\log^2 n\right), \text{ and outputs}$$

$$S' = \{s_1,\ldots,s_{m+1}\} \subseteq S,$$

such that for all $i = (i_1,\ldots,i_r) \in [n]^r, j = (j_1,\ldots,j_r,j_{r+1}) \in [n]^{r+1}$:

$$E_{s\sim U_{S'}}[Z_i(s)] \geq \tfrac{1}{2}(\tfrac{1}{4s})^r \text{ and } E_{s\sim U_{S'}}[Y_j(s)] \leq \tfrac{3}{2}(\tfrac{1}{4s})^{r+1}, \text{so}$$

$$E[X_{i_1,\cdots,i_r,j_1,\cdots,j_s}] \geq E[Z_{i_1,\cdots,i_r}] - \sum_{k=1}^{s} E[Y_{i_1,\cdots,i_r,j_k}] \geq \tfrac{1}{2}(\tfrac{1}{4s})^r - s\cdot\tfrac{3}{2}(\tfrac{1}{4s})^{r+1}$$

$$E[X_{i_1,\cdots,i_r,j_1,\cdots,j_s}] \geq \tfrac{1}{8}(\tfrac{1}{4s})^r$$ ◀

Note that we can apply Lemma 2 and Lemma 6 on the second and third constructions, the same way we did in the first construction to get the needed results.

4 Direct Usage of DCFF: α-Error Tolerant Algorithm

In this section we discuss and explain the intuition behind using DCFF, and show how we can use it directly to present an α-error tolerant algorithm of a hidden hypergraph. Unfortunately though, the time complexity of the algorithm will not be polynomial. The issue of the time complexity will be solved in the following sections by wise use of the DCFF combined with other algebraic structures.

We assume our function is $f = M_1 \vee \cdots \vee M_{s'}$, $s' \leq s$, and each M_i is of size at most r. For simplicity we assume that $s' = s$ and each term is of size exactly r. Denote $S_{terms} = \{x_{i_1}\cdots x_{i_r} | 1 \leq i_1 < \ldots < i_r \leq n \text{ and } x_{i_1}\cdots x_{i_r} \notin \{M_1,\ldots,M_s\}\}$.
In order to find M_1 we need a set of assignments A_{M_1} such that:

1. $\exists a \in A_{M_1}$ such that $M_1(a) = 1, M_i(a) = 0$ $(1 < i \leq s)$.
2. $\forall t \in S_{terms}$ $\exists a \in A_{M_1}$ where $t(a) = 1$ and $f(a) = 0$.

If all the above is satisfied, then we can find M_1 simply by trying all the possible terms of size r. One can easily see that any $(n,(s,r),\alpha)$-DCFF(CFF) has a subset that satisfies all the above requirements. To satisfy the first requirement we need r $1's$ (to satisfy M_1) and $s-1$ $0's$ (to falsify $M_i, 1 < i \leq s$), and to satisfy the second requirement we need s $0's$ (to falsify $M_i, 1 \leq i \leq s$) and r $1's$ (to satisfy a given term $t \in S_{terms}$).
Now we present the details of the algorithm:

We construct an $(n,(s,r),\beta)$-DCFF A, where $\beta = 2\alpha + \frac{1}{|A|}$. Denote by $A'(M)$ the set $\{a \in A | M(a) = 1 \text{ and } Q'(a) = 1\}$. Now, take every monomial M of size at most r where $|A'(M)| \leq \alpha|A|$. The disjunction of all such monomials is equivalent to the target function.

This follows from the following two facts: (1) for any monomial M' such that $M' \not\Rightarrow f$, there are at least $\left(\alpha + \frac{1}{|A|}\right)|A|$ assignments $a \in A$ such that $Q'(a) = 0$ and $M'(a) = 1$ (2) for any monomial $M' \Rightarrow f$, there are at least $\left(\alpha + \frac{1}{|A|}\right)|A|$ assignments $a \in A$ such that $Q'(a) = 1$ and $M'(a) = 1$. Since only $\alpha|A|$ queries can be incorrect, and there are $\beta|A|$ assignments where $M'(a) = 1$ and $Q(a) = 0$ for each $M' \not\Rightarrow f$, then there are more than $\alpha|A|$ assignments $a \in A$ where $Q(a) = 0$ and $M'(a) = 1$. Similarly, for each $M' \Rightarrow f$ there are at most $\alpha|A|$ assignments $a \in A$ where $Q(a) = 0$ and $M'(a) = 1$.

▶ **Lemma 13.** *If an* $\left(n, (s, r), 2\alpha + \frac{1}{|A|}\right)$*-DCFF* A *of size* N *can be constructed in time* T*, then there is an algorithm that learns the class* s*-term* r*-MDNF with* N *noisy membership queries in time*

$$O\left(T + Nr\sum_{r'=0}^{r}\binom{n}{r'}\right).$$

Correctness and time complexity of the algorithm follows from the above explanation and from the correctness of the error-free folklore algorithm [2].

5 α-Error Tolerant Reduction

In this section, we give a reduction to reduce the query complexity of any α-error tolerant algorithm that depends on d variables to become linear in $\log n$. The idea behind this is to map the n-dimensional variables space to $g(d)$-dimensional space, where g is a polynomial function of d. Then we run the original algorithm over the new space. The main challenge here is to recover the original function efficiently (in terms of time complexity), while keeping the algorithm $\Omega(\alpha)$-error tolerant. This can be done by using PHF in the following way: by the definition of PHF, it contains a function h such that h maps each variable of MDNF to a distinct variable. After finding this h, we use it to reduce the variables space from n-dimensional to $g(d)$-dimensional. Here is the main theorem of this section:

▶ **Theorem 14.** *Let* H *be a class of boolean functions that is closed under variable projection. 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 H *is non-adaptively learnable in time* $T(n)$ *with* $Q'(n)$ *noisy membership queries, and* $\alpha Q'(n)$*-errors might occur, then* H *is non-adaptively learnable in time*

$$O\left(\frac{d^2 n\log n}{\epsilon\log(\epsilon q/d^3)} + \frac{d^2\log n}{\epsilon\log(\epsilon q/d^3)}(T(q) + Q'(q)n + R(q))\right), \text{ with}$$

$$Q_{new}(n) = O\left(\frac{d^2 Q'(q)}{\epsilon\log(\epsilon q/d^3)}\log n\right)$$

noisy membership queries where less than $\frac{1-\epsilon}{2}\alpha Q_{new}(n)$ *errors might occur, and* d *is an upper bound on the number of relevant variables in* $f \in C$ *and* q *is any integer such that* $q \geq 2(d+1)^2$*.*

Consider the algorithm in Figure 1.

Let $\mathcal{A}(n, \alpha)$ be a non-adaptive algorithm that learns H in $T(n)$ time with $Q'(n)$ noisy membership queries, where $\alpha Q'(n)$ errors might occur. Let $f \in H_n$ be the target function. Consider the $(n, q, d+1, 1-\epsilon)$-DPHF P that was constructed in Lemma 2 (Step 1 in the algorithm).

The following lemma follows from the fact that H is closed under variable projection:

Reduction
$\mathcal{A}(n,\alpha)$ is a non-adaptive α error tolerant learning algorithm for H.
1) Construct an $(n, q, d+1, (1-\epsilon))$-DPHF P.
2) For each $h \in P$
    Run $\mathcal{A}(q,\alpha)$ to learn $f_h := f(x_{h(1)}, \ldots, x_{h(n)})$.
    Let $f'_h \in H$ be the output of $\mathcal{A}(q,\alpha)$.
3) For each $h \in P$
    $V_h \leftarrow$ the relevant variables in f'_h
4) $G(h) = |\{h' || V_{h'}| = |V_h|, h' \in P\}|$, $H' = \{h \in H | G(h) \geq \frac{1-\epsilon}{2}|P|\}$
    $d_{max} \leftarrow \max_{h \in H'} |V_h|$.
5) $X \leftarrow \{x_1, x_2, \ldots, x_n\}$.
6) For each $i \in [n]$, $W(i) = |\{h \in H | x_{h(i)} \notin V_h, |V_h| = d_{max}\}|$
    If $W(i) \geq \frac{1-\epsilon}{2}|P|$, then $X \leftarrow X \backslash \{x_i\}$
7) Take all $h \in H$ with $|V_h| = d_{max}$
8) Replace each relevant variable x_i in f'_h by $x_j \in X$ where $h(j) = i$.
9) Output the function that appears at least $\frac{1-\epsilon}{2}|P|$ times from step (8).

Figure 1 Reduction.

▶ **Lemma 15.** *For every $h \in P$, the function $f_h := f(x_{h(1)}, \ldots, x_{h(n)})$ is in H_q.*

▶ **Lemma 16.** *The total number of queries after the reduction is $Q_{new} = |P|Q'(q)$.*

Proof. For each $h \in P$, we run $\mathcal{A}(q,\alpha)$ to learn f_h. It is known that $\mathcal{A}(q,\alpha)$ generates $Q'(q)$ queries, so overall we have $|P|Q'(q)$ queries. ◀

▶ **Lemma 17.** *(Step 2 in the algorithm) Let f'_h be the output of algorithm $\mathcal{A}(q,\alpha)$ when it runs on f_h, and let $K = \{h | h \in P \text{ and } f'_h \not\equiv f_h\}$, then $K < \frac{1-\epsilon}{2}|P|$.*

Proof. The lemma follows from the fact that the number of times when $\alpha Q'(q)$ errors appear while running $A(q,\alpha)$ is less than $\frac{1-\epsilon}{2}|P|$. Otherwise, the number of errors will be at least $\frac{1-\epsilon}{2}\alpha|P|Q'(q) = \frac{1-\epsilon}{2}\alpha Q_{new}(n)$. ◀

(Step 3 in the algorithm) Note that if the number of errors when running $A(q,\alpha)$ is less than $\alpha Q'(q)$-errors, then the algorithm finds the correct relevant variables of f'_h, by the definition of $A(q,\alpha)$.

▶ **Lemma 18.** *(Step 4 in the algorithm) Suppose $x_{i_1}, \ldots, x_{i_{d'}}$, $d' \leq d$ are the relevant variables in the target function f, then $d_{max} = d'$.*

Proof. Let V_h be the set of relevant variables of f'_h and let $d_{max} = \max_{h \in H'} |V_h|$. By the definition of P there are at least $(1-\epsilon)|P|$ maps $h' \in P$ such that $h'(i_1), \ldots, h'(i_{d'})$ are distinct, and since more than $\alpha Q'(q)$ errors might occur in less than $\frac{1-\epsilon}{2}|P|$ functions $h \in H$ (as mentioned above), then there are at least $(1-\epsilon)|P| - \frac{1-\epsilon}{2}|P| = \frac{1-\epsilon}{2}|P|$ functions $h' \in P$ where $h'(i_1), \ldots, h'(i_{d'})$ are distinct and their $V_{h'}$ equals d'. Such h' satisfies $G(h') \geq \frac{1-\epsilon}{2}|P|$, so $h' \in H'$. Stemming from the same fact, there are less than $\frac{1-\epsilon}{2}|P|$ functions $h \in P$ where the learned function f'_h can have number of relevant variables greater than d'. Such h satisfies $G(h) < \frac{1-\epsilon}{2}|P|$, so $h \notin H'$. And therefore, $d_{max} = d'$. ◀

▶ **Lemma 19.** *(Step 6 in the algorithm) If x_i is an irrelevant variable, then for at least $\frac{1-\epsilon}{2}|P|$ maps $h \in P$ $|V_h| = d_{max}$ and $x_{h(i)} \notin V_h$.*

Learn(f_{MDNF})
1) Construct an $(n,(1,r),\alpha_1)$-DCFF A and an $(n,(s-1,r),\alpha_2)$-DCFF B.
2) Ask noisy 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 at most $\beta_1|A|$ assignments $a \in A$, $(a \wedge b)_i < Q'(a \wedge b)$
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 $\mathcal{T}$ all the terms that appear at most $\beta_2|B|$-times.
10) Remove from $\mathcal{T}$ the term $\wedge_{i\in[n]}x_i$
$\quad$ and all subterms of a larger term.

Figure 2 An algorithm for learning s term r MDNF.

Proof. Consider any irrelevant variable $x_j \notin \{x_{i_1},\ldots,x_{i_{d'}}\}$. Since P is $(n,q,d+1,1-\epsilon)$-DPHF, there are $(1-\epsilon)|P|$ functions $h'' \in P$ such that $h''(j)$ and $h''(i_1),\ldots,h''(i_{d'})$ are distinct. Again since αQ_{new} errors might occur, then $f'_{h''}$ depends on $x_{h''(i_1)},\ldots,x_{h''(i_{d'})}$, and not on $x_{h''(j)}$ and $|V_h| = d_{max}$ for at least $\frac{1-\epsilon}{2}|P|$ functions $h'' \in P$. ◀

This way the irrelevant variables can be eliminated. Since the above is true for every irrelevant variable, the set X contains only the relevant variables of f, after Step 6 in the algorithm. Then in Steps 7 and 8, we find all functions f for which the number of relevant variables is d'. The target function appears at least $\frac{1-\epsilon}{2}|P|$ times, and any other function appears less than $\frac{1-\epsilon}{2}|P|$ times. This is Step 9 in the algorithm. The correctness of the algorithm follows immediately from the above lemmas and explanation.

6 Efficient α-Error Tolerant Algorithm

In Section 4 we used DCFF directly to present a non-polynomial algorithm. In order to improve the time complexity we will now use two DCFFs. One of them is used to spread the set of terms over different sets, and the other is used afterwards to find each term explicitly. The complexity of this process is the bit-wise conjunction of assignments in each DCFF is $f(r,s)\log^2 n$, where $f(r,s)$ is a monotone function of s and r. In order to make our algorithm linear in $\log n$, we use the reduction from the previous section.

▶ **Theorem 20.** *Let A be an $(n,(1,r),\alpha_1)$-DCFF and B be an $(n,(s-1,r),\alpha_2)$-DCFF, and let $\beta_1 \triangleq \frac{\alpha_1}{2} - \frac{1}{|A|}, \beta_2 \triangleq \frac{\alpha_2}{2} - \frac{1}{|B|}$. There is a non-adaptive α-error tolerant learning algorithm for s-term r-MDNF that asks all the queries $A \wedge B$ and finds the target function, when $\alpha = \beta_1\beta_2$.*

This gives the algorithm in Figure 2.

▶ **Lemma 21.** *If $b \in B$ **separates** a term T in f from other terms, and at most $\beta_1|A|$ errors can occur, then $T_b = T$.*

Proof. According to the definition of A, for each distinct $i_1,\ldots,i_r$ and j_1 there are $\alpha_1|A|$ assignments $a \in A$, where $a_{i_1} = \cdots = a_{i_r} = 1$ and $a_{j_1} = 0$. Since $\beta_1|A|$ of the queries might have incorrect results, then each $x_i \in T$ can appear as $x_i = 1$ while also $Q = 1$ in at least

$\alpha_1 - \beta_1 > \beta_1$ fraction of the assignments. Each $x_i \notin T$ can appear as $x_i = 0$ while also $Q = 1$ in at least $\alpha_1 - \beta_1 > \beta_1$ fraction of the assignments. So by the definition of T_b, $T_b = T$. ◀

▶ **Lemma 22.** *Let $A_b = \{a \wedge b | a \in A\}$. If $\alpha \triangleq \beta_1\beta_2$ is the fraction of queries that can be incorrect, then for at most $\beta_2|B|$ assignments $b \in B$, A_b contains more than $\beta_1|A|$ errors.*

Proof. If there are more than $\beta_2|B|$ $b \in B$ such that A_b contains more than $\beta_1|A|$ errors, then the fraction of errors is bigger than $\frac{(\beta_2|B|)(\beta_1|A|)}{|A||B|} = \beta_1\beta_2$. ◀

Now we are ready to prove Theorem 20

Proof. Let $f = M_1 \vee M_2 \vee \cdots \vee M_s$ be the target function. For every $b \in B$, let $F_b(i) = \{a | a \in A, (a \wedge b)_i < Q'(a \wedge b)\}$, $A_b = \{a \wedge b | a \in A\}$, $I_b = \{i || F_b(i)| \leq \beta_1|A|\}$, and T_b be the following term: $T_b := \wedge_{i \in I_b} x_i$. We will show:

1. For each T in f, there are more than $\beta_2|B|$ assignments $b \in B$, such that $T_b = T$.
2. Every other term T_b that appears more than $\beta_2|B|$ times, is either equal to $\wedge_{i\in[n]}x_i$ or to a subterm of one of the terms in f.

In case no errors occur, the term that can be learned from each block A_b is either one of the following:

1. In case b separates a term in f then I_b will be the separated term.
2. In case b separates k terms in f then I_b will be a subterm of one of the terms in f (the intersection of the k terms).
3. In case b separates no term in f then I_b will be $\wedge_{i\in[n]}x_i$.
   (For more details see Lemma 13 in [2]).

By Lemma 22, the number of assignments $b \in B$ where A_b has more than $\beta_1|A|$ errors is $\beta_2|B|$, so any term different than a term in f, a subterm of one of the terms in f and $\wedge_{i\in[n]}x_i$ appears at most $\beta_2|B|$ times.

Now we prove that each term T in f appears more than $\beta_2|B|$ times.

Since there are $\alpha_2|B|$ assignments $b \in B$ that separate T from all other terms in f, among which at most $\beta_2|B|$ assignments A_b can have more than $\beta_1|A|$ errors, then by Lemma 21 $(\alpha_2 - \beta_2)|B| > \beta_2|B|$ of A_b will return $T_b = T$. ◀

▶ **Theorem 23.** *Fix any integers $r < s < d$ with $d = r + s$, and let $d \leq n$. There is a non-adaptive proper α error tolerant learning algorithm for s-term r-MDNF that asks $O\left(r^9(4s)^{r+5}\log^2 n\right)$ queries and runs in time $poly(n, s^r)$, with $\alpha = \Omega\left(\frac{1}{r^r}DNS(1,r)DNS(s-1,r)\right)$ -fraction of the queries might be incorrect.*

Proof. By Theorem 11 (**second construction**), we can construct a $(n, (1, r), \alpha_1)$-DCFF, of size $|A| = O\left(r^6 \log n\right)$, where $\alpha_1 = \Omega\left(DNS(1, r)\right)$. And by Theorem 12 (**third construction**), we can construct a $(n, (s-1, r), \alpha_2)$-DCFF of size $|B| = O\left(d^3\,(4s)^{r+2}\log n\right)$, where $\alpha_2 = \Omega\left(\left(\frac{1}{r^r}\right)DNS(s-1,r)\right)$.

The construction of the above DCFFs takes $poly\,(n, |A|, |B|)$ time. By Theorem 20, the learning takes $|A \wedge B| \cdot n = poly\ (n, |A|, |B|)$ time. The number of queries of the algorithm is $|A \wedge B| \leq |A| \cdot |B| = O\left(r^9(4s)^{r+5}\log^2 n\right)$. ◀

We are now ready to prove the main result:

▶ **Theorem 24.** *Fix any integers $r < s < d$ with $d = r + s$, and let $d \leq n$. There is a non-adaptive proper α error tolerant learning algorithm for s-term r-MDNF that asks*

$$O(r^{11}(4s)^{r+7}\log n)$$

queries where $\alpha = \Omega\left(\frac{1}{r^r}DNS(1,r)DNS(s-1,r)\right)$-fraction of the queries might return incorrect result, and runs in time $(n \log n) \cdot poly(s^r)$.

Proof. We use Theorem 14. 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 $T(n) = poly(n, s^r)$ and asks $Q'(n) = O(r^9(4s)^{r+5}\log^2 n)$ queries. The number of variables in the target is bounded by $d = rs$. Let $q = O\left(d^3\right) \geq 2d^2$. By Theorem 14, there is a non-adaptive algorithm that runs in time $O\left(qd^2 n\log n + \frac{d^2\log n}{\log(q/d^2)}(T(q)n + R(q))\right) = (n\log n)poly(s^r)$, and asks $O\left(\frac{d^2Q'(q)}{\log(q/d^2)}\log n\right) = O(r^{11}(4s)^{r+7}\log n)$, noisy membership queries. ◀

References

1 Hasan Abasi, Nader H. Bshouty, and Hanna Mazzawi. On exact learning monotone DNF from membership queries. In Peter Auer, Alexander Clark, Thomas Zeugmann, and Sandra Zilles, editors, *Algorithmic Learning Theory - 25th International Conference, ALT 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, volume 8776 of *Lecture Notes in Computer Science*, pages 111–124. Springer, 2014. doi:10.1007/978-3-319-11662-4_9.

2 Hasan Abasi, Nader H. Bshouty, and Hanna Mazzawi. Non-adaptive learning of a hidden hypergraph. *Theor. Comput. Sci.*, 716:15–27, 2018. doi:10.1016/j.tcs.2017.11.019.

3 Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987. doi:10.1007/BF00116828.

4 Dana Angluin and Jiang Chen. Learning a hidden hypergraph. *Journal of Machine Learning Research*, 7:2215–2236, 2006. URL: http://www.jmlr.org/papers/v7/angluin06a.html.

5 Dana Angluin and Jiang Chen. Learning a hidden graph using o(logn) queries per edge. *J. Comput. Syst. Sci.*, 74(4):546–556, 2008. doi:10.1016/j.jcss.2007.06.006.

6 Richard Beigel, Noga Alon, Simon Kasif, Mehmet Serkan Apaydin, and Lance Fortnow. An optimal procedure for gap closing in whole genome shotgun sequencing. In Thomas Lengauer, editor, *Proceedings of the Fifth Annual International Conference on Computational Biology, RECOMB 2001, Montréal, Québec, Canada, April 22-25, 2001*, pages 22–30. ACM, 2001. doi:10.1145/369133.369152.

7 Mathilde Bouvel, Vladimir Grebinski, and Gregory Kucherov. Combinatorial search on graphs motivated by bioinformatics applications: A brief survey. In Dieter Kratsch, editor, *Graph-Theoretic Concepts in Computer Science, 31st International Workshop, WG 2005, Metz, France, June 23-25, 2005, Revised Selected Papers*, volume 3787 of *Lecture Notes in Computer Science*, pages 16–27. Springer, 2005. doi:10.1007/11604686_2.

8 Nader H. Bshouty. Linear time constructions of some d -restriction problems. In Vangelis Th. Paschos and Peter Widmayer, editors, *Algorithms and Complexity - 9th International Conference, CIAC 2015, Paris, France, May 20-22, 2015. Proceedings*, volume 9079 of *Lecture Notes in Computer Science*, pages 74–88. Springer, 2015. doi:10.1007/978-3-319-18173-8_5.

9 Nader H. Bshouty. Derandomizing chernoff bound with union bound with an application to k-wise independent sets. *CoRR*, abs/1608.01568, 2016. arXiv:1608.01568.

10 Nader H. Bshouty and Ariel Gabizon. Almost optimal cover-free families. In Dimitris Fotakis, Aris Pagourtzis, and Vangelis Th. Paschos, editors, *Algorithms and Complexity - 10th International Conference, CIAC 2017, Athens, Greece, May 24-26, 2017, Proceedings*, volume 10236 of *Lecture Notes in Computer Science*, pages 140–151, 2017. doi:10.1007/978-3-319-57586-5_13.

11 Hong-Bin Chen, Hung-Lin Fu, and Frank K. Hwang. An upper bound of the number of tests in pooling designs for the error-tolerant complex model. *Optimization Letters*, 2(3):425–431, 2008. doi:10.1007/s11590-007-0070-5.

12 Hong-Bin Chen and Frank K. Hwang. A survey on nonadaptive group testing algorithms through the angle of decoding. *J. Comb. Optim.*, 15(1):49–59, 2008. doi:10.1007/s10878-007-9083-3.

13 Dingzhu Du, Frank K Hwang, and Frank Hwang. *Combinatorial group testing and its applications*, volume 12. World Scientific, 2000.

14 Vladimir Grebinski and Gregory Kucherov. Reconstructing a hamiltonian cycle by querying the graph: Application to DNA physical mapping. *Discrete Applied Mathematics*, 88(1-3):147–165, 1998. doi:10.1016/S0166-218X(98)00070-5.

15 Hwang Frank Kwang-ming and Du Ding-zhu. *Pooling designs and nonadaptive group testing: important tools for DNA sequencing*, volume 18. World Scientific, 2006.

16 Weiwei Lang, Yuexuan Wang, James Yu, Suogang Gao, and Weili Wu. Error-tolerant trivial two-stage group testing for complexes using almost separable and almost disjunct matrices. *Discrete Math., Alg. and Appl.*, 1(2):235–252, 2009. doi:10.1142/S1793830909000191.

17 Anthony J. Macula and Leonard J. Popyack. A group testing method for finding patterns in data. *Discrete Applied Mathematics*, 144(1-2):149–157, 2004. doi:10.1016/j.dam.2003.07.009.

18 Douglas R. Stinson and Ruizhong Wei. Generalized cover-free families. *Discrete Mathematics*, 279(1-3):463–477, 2004. doi:10.1016/S0012-365X(03)00287-5.

19 Douglas R. Stinson, Ruizhong Wei, and Lie Zhu. Some new bounds for cover-free families. *J. Comb. Theory, Ser. A*, 90(1):224–234, 2000. doi:10.1006/jcta.1999.3036.

20 David C Torney. Sets pooling designs. *Annals of Combinatorics*, 3(1):95–101, 1999.