Adaptive Learning a Hidden Hypergraph [1]

A. G. D'yachkov   agd-msu@yandex.ru
Lomonosov Moscow State University, Moscow, Russia

I.V. Vorobyev   vorobyev.i.v@yandex.ru
Lomonosov Moscow State University, Moscow, Russia

N.A. Polyanskii   nikitapolyansky@gmail.com
Lomonosov Moscow State University, Moscow, Russia

V.Yu. Shchukin   vpike@mail.ru
Lomonosov Moscow State University, Moscow, Russia

Abstract. Learning a hidden hypergraph is a natural generalization of the classical group testing problem that consists in detecting unknown hypergraph $H_{un} = H(V, E)$ by carrying out edge-detecting tests. In the given paper we focus our attention only on a specific family $\mathcal{F}(t, s, \ell)$ of localized hypergraphs for which the total number of vertices $|V| = t$, the number of edges $|E| \leqslant s$, $s \ll t$, and the cardinality of any edge $|e| \leqslant \ell$, $\ell \ll t$. Our goal is to identify all edges of $H_{un} \in \mathcal{F}(t, s, \ell)$ by using the minimal number of tests. We provide an adaptive algorithm that matches the information theory bound, i.e., the total number of tests of the algorithm in the worst case is at most $s\ell \log_2 t(1 + o(1))$.

1 Introduction

Before we introduce the problem, let us recall some definitions and notations.

Let $|A|$ denote the size of a set A, and $[N] \triangleq \{1, 2, \ldots, N\}$ - the set of integers from 1 to N. A *hypergraph* is a pair $H = H(V, E)$ such that $E \subset 2^V \setminus \varnothing$, where V is the set of vertices and $E = \{\mathbf{e}_1, \ldots \mathbf{e}_s\}$ is a set of edges. A vertex $v \in V$ is called *active*, if there exists at least one edge $\mathbf{e} \in E$ such that $v \in e$. A set $S \subset V$ is called an *independent* set of H if it contains no entire edge of H. We denote by $\dim(H)$ the cardinality of the largest edge, i.e. $\dim(H) = \max\limits_{\mathbf{e} \in E} |\mathbf{e}|$.

1.1 Statement of the problem

The problem of learning a hidden hypergraph is described as follows. Suppose there is an unknown (hidden) hypergraph $H_{un} = H(V, E)$ whose edges are not

---

known to us, but we know that the unknown hypergraph H_{un} belongs to some family $\mathcal{F}$ of hypergraphs that have a specific structure (e.g, $\mathcal{F}$ consists of all Hamiltonian cycles on V). Our goal is to identify all edges of E by carrying out the minimal number N of *edge-detecting queries* $Q(S)$, where $S \subseteq V$: $Q(S) = 0$ if S is independent of H_{un}, and $Q(S) = 1$ otherwise.

In the given paper we focus our attention only on the family of localized hypergraphs. We consider the family $\mathcal{F}(t, s, \ell)$, that consists of all hypergraphs $H(V, E)$ such that $\dim(H) \leqslant \ell$ and $|E| \leqslant s$. Suppose we know that the hypergraph H_{un} belongs to the family $\mathcal{F}(t, s, \ell)$. An algorithm is said to be $\mathcal{F}(t, s, \ell)$-*searching* algorithm if it finds H_{un}, i.e. there exists only one hypergraph from $\mathcal{F}(t, s, \ell)$ that fits all answers to the queries.

One of the most important aspects of any searching strategy is its adaptiveness. An algorithm is *non-adaptive* if all queries are carried out in parallel. An algorithm is *adaptive* if the later queries may depend on the answers to earlier queries.

By $N^{na}(t, s, \ell)$ ($N^a(t, s, \ell)$) denote the minimal number of queries in a $\mathcal{F}(t, s, \ell)$-searching non-adaptive (adaptive) algorithm. Introduce the *asymptotic rate* for optimal $\mathcal{F}(t, s, \ell)$-searching algorithms:

$$R^{na}(s, \ell) \triangleq \varlimsup_{t \to \infty} \frac{\log_2 t}{N^{na}(t, s, \ell)}, \qquad R^a(s, \ell) \triangleq \varlimsup_{t \to \infty} \frac{\log_2 t}{N^a(t, s, \ell)}.$$

The given paper is organized as follows. In Sect. 2, we discuss previously known results and remind the concept of cover-free codes which is close to the subject. In Sect. 3, we present the main result of the paper and provide the deterministic adaptive algorithm that matches the information theory bound.

2   Previous Results

For the particular case $\ell = 1$, the above definitions were already introduced to describe the model called *designing screening experiments*. It is a classical group testing problem. We refer the reader to the monograph [7] for a survey on group testing and its applications. It is quite clear (e.g., see [7]) that a $\mathcal{F}(t, s, 1)$-searching adaptive algorithm can achieve the information theory bound, i.e. $N(t, s, 1) = s \log_2 t(1 + o(1))$ as $t \to \infty$. Therefore, $R^a(s, 1) = 1/s$.

If $\ell = 2$, then we deal with learning a hidden graph. One important application area for such problem is bioinformatics [6], more specifically, chemical reactions and genome sequencing. Alon et al. [5], and Alon and Asodi [4] give lower and upper bounds on the minimal number of tests for non-adaptive searching algorithms for certain families of graphs, such as stars, cliques, matchings. In [6], Boevel et al. study the problem of reconstructing a Hamiltonian

cycle. In [3], Angluin et al. give a suboptimal $\mathcal{F}(t, s, 2)$-searching adaptive algorithm. More precisely, they prove $R^a(s, 2) \geqslant 1/(12s)$.

For the general case of parameters s and ℓ, Abasi et al. have recently provided [8] a suboptimal $\mathcal{F}(t, s, \ell)$-searching adaptive algorithm. In particular, from their proofs it follows $R^a(s, \ell) \geqslant 1/(2s\ell)$. This bound differs up to the constant factor from the information theory upper bound $R^a(s, \ell) \leqslant 1/(s\ell)$.

2.1 Cover-Free Codes

A binary $N \times t$-matrix

$$X = \|x_i(j)\|, \quad x_i(j) = 0, 1, \ i \in [N], \ j \in [t] \tag{1}$$

is called a *code of length N and size t*. By $\boldsymbol{x}_i$ and $\boldsymbol{x}(j)$ we denote the i-th row and the j-th column of the code X, respectively.

Before we give the well-known definition of cover-free codes, note that any $\mathcal{F}(t, s, \ell)$-searching non-adaptive algorithm consisting of N queries can be represented by a binary $N \times t$ matrix X such that each test corresponds to the row, and each vertex stands for the column. We put $x_i(j) = 1$ if the j-th vertex is included to the i-th test; otherwise, $x_i(j) = 0$.

Definition 1. A code X is called a *cover-free (s, ℓ)-code* (briefly, *CF (s, ℓ)-code*) if for any two non-intersecting sets $\mathcal{S}, \mathcal{L} \subset [t]$, $|\mathcal{S}| = s$, $|\mathcal{L}| = \ell$, $\mathcal{S} \cap \mathcal{L} = \varnothing$, there exists a row $\boldsymbol{x}_i$, $i \in [N]$, for which

$$x_i(j) = 0 \text{ for any } j \in \mathcal{S}, \qquad x_i(k) = 1 \text{ for any } k \in \mathcal{L}. \tag{2}$$

Taking into account the evident symmetry over s and ℓ, we introduce $N_{cf}(t, s, \ell) = N_{cf}(t, \ell, s)$ - the minimal length of CF (s, ℓ)-codes of size t and define the *rate* of CF (s, ℓ)-codes:

$$R_{cf}(s, \ell) = R_{cf}(\ell, s) \triangleq \varlimsup_{t \to \infty} \frac{\log_2 t}{N_{cf}(t, s, \ell)}. \tag{3}$$

In [1], Dyachkov et al. show that any CF (s, ℓ)-code represents a $\mathcal{F}(t, s, \ell)$-searching non-adaptive algorithm, while any $\mathcal{F}(t, s, \ell)$-searching non-adaptive algorithm corresponds to both a CF $(s, \ell-1)$-code and CF $(s-1, \ell)$-code. The best presently known upper and lower bounds on the rate $R(s, \ell)$ of CF (s, ℓ)-codes were presented in [2]. If $\ell \geqslant 1$ is fixed and $s \to \infty$, then these bounds lead to the following asymptotic equality:

$$\frac{(\ell+1)^{\ell+1}}{2e^{\ell-1}} \frac{\log_2 s}{s^{\ell+1}}(1 + o(1)) \geqslant R^{na}(s, \ell) \simeq R_{cf}(s, \ell) \geqslant \frac{\ell^\ell}{e^\ell} \frac{\log_2 e}{s^{\ell+1}}(1 + o(1)). \tag{4}$$

3   New Result

By a counting argument, the lower bound is true.

Theorem 1.   *Any $\mathcal{F}(t, s, \ell)$-searching algorithm has at least $s\ell \log_2 t(1 + o(1))$ edge-detecting queries. In other words, the rate $R(s, \ell) \leqslant 1/(s\ell)$.*

The key result of this paper is given as follows.

Theorem 2.   *There exists an adaptive $\mathcal{F}(t, s, \ell)$-searching algorithm which has at most $s\ell \log_2 t(1 + o(1))$ edge-detecting queries. In other words, the rate $R^a(s, \ell) = 1/(s\ell)$.*

Proof of Theorem 2.

We present the full description of $\mathcal{F}(t, s, \ell)$-searching algorithm by Alg. 1, and this algorithm is based on Alg. 3, 2 and 4. Notice that Alg. 2 is a variation of the binary vertex search. Also one can check that at each step of the algorithm, set S' contains at least one new active vertex. Alg. 3 and 4 represent an exhaustive search of edges and an exhaustive query search, respectively.

Now we upper bound the number of tests of Alg. 1 in the worst scenario. Let $|V| = t$. It is easy to check that Alg. 2 makes use of at most $\lceil \log_2 |S| \rceil \leqslant \lceil \log_2 t \rceil$ tests. One can see that the number of active vertices of the hidden hypergraph $H_{un} \in \mathcal{F}(t, s, \ell)$ is at most $s\ell$. Alg. 3 uses at most $F_1(s, \ell)$ tests, while Alg. 4 uses at most $F_2(s, \ell)$ tests, where the functions F_1 and F_2 do not depend on t. We can upper bound the number of cycles in Alg. 1 by the number of active vertices. Therefore, the total number of tests for the given adaptive $\mathcal{F}(t, s, \ell)$-searching algorithm does not exceed $s\ell(\log_2 t + F_1(s, \ell) + F_2(s, \ell) + 1)$. $\square$

Data: set of vertices V of $H(V, E) \in \mathcal{F}(t, s, \ell)$
Result: set of edges of H_{un}
initialization $E' := \varnothing$; $F := \varnothing$; $S := V$;
while $S \neq \varnothing$ **do**
    perform Alg. 2, and find $v \notin F$;
    $F := F \sqcup v$;
    perform Alg. 3, and find subset of edges E';
    perform Alg. 4, and find query S;
end
set of edges $E' = E$;
      Algorithm 1: Searching edges of the hidden hypergraph

Data: query $S \subseteq V$ such that $Q(S) = 1$, and the set of found active
       vertices F
Result: vertex $v \in V$, $v \notin F$, and $\exists\, \mathbf{e} \in E$, $v \in \mathbf{e}$
initialization $S' := S \setminus F$; $S'' := S \setminus S'$;
while $|S'| > 1$ **do**
    split up S' into two subsets S_1 and S_2 of sizes $\lceil |S'|/2 \rceil$ and $\lfloor |S'|/2 \rfloor$:
    $S' = S_1 \sqcup S_2$;
    carry out a query $S_1 \sqcup S''$;
    if $Q(S_1 \sqcup S'') = 1$ **then**
        $S' := S_1$;
    else
        $S' := S_2$;
        $S'' := S'' \sqcup S_1$;
    end
end
vertex $\{v\} = S'$ satisfies the required conditions;
    Algorithm 2: Searching another active vertex on the query

Data: subset of active vertices $F \subset V$
Result: subset of edges $E' \subset E$ consisting of vertices of F
initialization $E' := \varnothing$;
for $\forall S \subset F: 1 \leqslant |S| \leqslant \ell$ **do**
    if $\nexists\, \mathbf{e} \in E' : \mathbf{e} \subset S$ **then**
        carry out query S;
        if $Q(S) = 1$ **then**
            for $\forall \mathbf{e} \in E' : S \subset \mathbf{e}$ **do**
                delete $\mathbf{e}$ from E';
            end
            add edge $\mathbf{e} = S$ to E';
        else
            proceed to the next step of the loop;
        end
    else
        proceed to the next step of the loop;
    end
end
  Algorithm 3: Searching edges composed on found active vertices

Data: subset of edges $E' \subset E$
Result: or $S \subset V$ such that $Q(S) = 1$, $\mathbf{e} \not\subset S$ for $\forall \mathbf{e} \in E'$, either $S = \varnothing$
initialization $A := \{v : v \in \mathbf{e} \in E'\}$; $B := V \setminus A$; $S := \varnothing$;
for $\forall C \subset V$: $B \subset C$ and $\nexists \mathbf{e} \in E'$, $\mathbf{e} \subset C$ **do**
    carry out query C;
    if $Q(C) = 1$ **then**
        $S := C$;
        break "for loop";
    else
        proceed to the next step of the loop;
    end
end

Algorithm 4: Searching a query on found edges

References

[1] *A. G. Dyachkov, P. Vilenkin, A. Macula*, and *D. Torney*, "Families of finite sets in which no intersection of sets is covered by the union of s others", *J. Combin. Theory.* Ser. A, 99 (2002), pp. 195-218.

[2] *D'yachkov A.G., Vorobyev I.V., Polyanskii N.A., Shchukin V.Yu.*, "Bounds on the Rate of Disjunctive Codes", *Problems of Information Transmission*, vol. 50, no. 1, pp. 27-56, 2014.

[3] *Angluin D., Chen J.*, "Learning a hidden graph using O(log n) queries per edge", *J Comput Syst Sci*, v.74, pp. 546-556, 2008.

[4] *Alon, N.*, and *Asodi, V*, "Learning a hidden subgraph". *SIAM J. Discrete Math.* 18, 4 (2005), pp. 697-712.

[5] *Alon, N., Beigel, R., Kasif, S., Rudich, S.*, and *Sudakov, B.*, "Learning a hidden matching". *SIAM J. Comput.* 33, 2 (2004), pp. 487-501.

[6] *Bouvel, M., Grebinski, V.*, and *Kucherov, G.* "Combinatorial search on graphs motivated by bioinformatics applications: A brief survey". In WG (2005), pp. 1627.

[7] *Du, D.-Z.* and *Hwang, F.K.*, "Combinatorial Group Testing and Its Applications", Singapore: *World Sci.*, 2000, 2nd ed.

[8] *Abasi, H., Bshouty, N.H.*, and *Mazzawi, H.*, "On Exact Learning Monotone DBF from Membership Queries", *Lecture Notes in Artificial Intelligence*, 2014, pp. 111-124.