Learning Multiple Secrets in Mastermind

Milind Prabhu [*1] **David Woodruff** [*2]

Abstract

In the Generalized Mastermind problem, there is an unknown subset H of the hypercube $\{0,1\}^d$ containing n points. The goal is to learn H by making a few queries to an oracle which given a point $\mathbf{q}$ in $\{0,1\}^d$, returns the point in H nearest to $\mathbf{q}$. We give a two-round adaptive algorithm for this problem that learns H while making at most $\exp(\widetilde{O}(\sqrt{d \log n}))$ queries. Furthermore, we show that any r-round adaptive randomized algorithm that learns H with constant probability must make $\exp(\Omega(d^{3^{-(r-1)}}))$ queries even when the input has $\text{poly}(d)$ points; thus, any $\text{poly}(d)$ query algorithm must necessarily use $\Omega(\log \log d)$ rounds of adaptivity. We give optimal query complexity bounds for the variant of the problem where queries are allowed to be from $\{0,1,2\}^d$. We also study a continuous variant of the problem in which H is a subset of unit vectors in $\mathbb{R}^d$, and one can query unit vectors in $\mathbb{R}^d$. For this setting, we give an $O(n^{\lfloor d/2 \rfloor})$ query deterministic algorithm to learn the hidden set of points.

1. Introduction

Mastermind is a classic codebreaking board game that originated in 1970 and over the past few decades has inspired several lines of research in theoretical computer science. The game is played by two players one of whom is a codemaker who chooses a secret sequence of four code pegs each of which has one of six colors; the second player is the codebreaker who wishes to determine the sequence by guessing various four peg patterns. For each pattern the codebreaker guesses, they learn the number of pegs in their guess that are of the correct color and appear in the correct position, as well as the number of pegs that are of the correct color but are in the wrong position. The codebreaker wishes to learn the pattern while making as few guesses as possible. The work of (Knuth, 1977) showed that the original Mastermind game has an optimal strategy that uses 5 queries in the worst case. The obvious generalization of the Mastermind game to n pegs and k colors was studied in (Chvátal, 1983) which gave an optimal strategy up to constant factors when $k \leq n^{1-\varepsilon}$ for any fixed constant $\varepsilon > 0$ and n arbitrarily large. Better strategies for the regime when $k \geq n$ were proposed in (Chen et al., 1996) and subsequently improved in (Goodrich, 2009).

In this paper, we study variants of the Mastermind problem in which the codemaker chooses not one hidden point but a set H of n hidden points. The codebreaker operates under a certain query model and has to learn all the points in H while minimizing the number of queries they make. Specifically, we consider three variants of this problem, each of which is defined by the space from which H is chosen and the query model under which the codebreaker operates. In the first variant, we consider the setting of the problem when the hidden points and queries lie in the hypercube.

> **Problem 1.** The hidden set H is a subset of $\{0,1\}^d$ of size n. The codebreaker queries $\mathbf{q} \in \{0,1\}^d$ and learns $\operatorname{argmin}_{\mathbf{x} \in H} \text{dist}(\mathbf{x}, \mathbf{q})$, i.e., the point in H with the least Hamming distance to $\mathbf{q}$.

Next, we consider a continuous variant of the problem.

> **Problem 2.** The hidden set H is an n point subset of S^{d-1}, the unit sphere in d dimensions. In each query, the codebreaker picks $\mathbf{q} \in S^{d-1}$ and learns $\operatorname{argmin}_{\mathbf{x} \in H} \|\mathbf{x} - \mathbf{q}\|_2$, i.e., the point in H that has the least Euclidean distance to $\mathbf{q}$.

We also study the hypercube version of the problem with a stronger query model.

> **Problem 3.** The hidden set H is a subset of $\{0,1\}^d$ of size n. The codebreaker queries $\mathbf{q} \in \{0,1,2\}^d$ and learns $\min_{\mathbf{x} \in H} \text{dist}(\mathbf{x}, \mathbf{q})$, i.e., the Hamming distance of $\mathbf{q}$ to the point in H nearest to it.

One motivation for the study of such a query model with extra characters is the work of (Hu et al., 2022). They study the problem in which a secret binary string in $\{0,1\}^d$ has to be recovered via access to distance oracles for non-

[*]Equal contribution [1]Department of Computer Science and Engineering, University of Michigan [2]Department of Computer Science, Carnegie Mellon University. Correspondence to: Milind Prabhu <milindpr@umich.edu>, David Woodruff <dwoodruf@cs.cmu.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

decomposable distance metrics such as edit distance, (p)-Dynamic Time Warping, and Fréchet distances. They showed that for the exact recovery problem under some distance metrics such as Dynamic Time Warping, recovery is provably impossible unless extra characters are allowed in the queries. For recovery using the edit distance oracle, they gave algorithms with better query complexity when queries were allowed to have extra characters. The question of what advantage allowing extra characters gives us in our setting is also of interest.

While our main goal is to design query-efficient algorithms for Problems 1-3, we are also interested in the *adaptivity* of algorithms. The adaptivity of an algorithm is the number of rounds over which the algorithm makes queries and is a measure of its parallelizability. Formally we have the following definition:

Definition 1.1 (r-Round Adaptive Algorithm)**.** An algorithm is said to be r-round adaptive if it makes queries in r batches $B_1, B_2, \dots B_r$ and for each $j \in [r] \setminus \{1\}$, the queries made by the algorithm in the j-th batch B_j are a function only of the responses to queries in the batches $B_1, \dots B_{j-1}$.

While it would be ideal to design algorithms that are both query efficient and use few rounds of adaptivity (i.e., are very parallelizable) we show that such algorithms do not exist for Problem 1. Given this reality, one of our main objectives is to understand the tradeoff between the query complexity of algorithms and their adaptivity.

1.1. Motivation for Generalized Mastermind

One strong motivation for studying these problems is that they seem to be very fundamental algorithmic questions and have not been studied previously to the best of our knowledge. Furthermore, several other questions similar in spirit have been of interest to the community (see Section 1.3). We now list some areas where the generalized mastermind problem has applications or is relevant.

Data Reconstruction from a Nearest Neighbor Data Structure. An obvious application of the generalized mastermind problem is the efficient recovery of data given access to its nearest neighbor data structure. In particular, a better understanding of these problems may help prevent adversarial attacks from learning private data using a nearest neighbor data structure.

Optimization. Problem 1 can be formulated as the algorithmic task of recovering all global minima of the function $f(x) = \text{dist}(x, H)$ whose input is a point $x \in \{0,1\}^d$ and output is the hamming distance of x to the hidden set H. Our lower bounds show that recovering all global minima even in such a structured and simple setting is hard.

The key difficulty in designing algorithms for Problem 1 stems from a phenomenon akin to "getting trapped in local minima" that arises in optimization. In the lower-bound for Problem 1 (i.e., Theorem 3.8), the hardness arises from a small set of hidden points $H' \subset H$ preventing the remaining points $H \setminus H'$ from being discovered. More precisely, H' has the property that a uniformly random point from $\{0,1\}^d$ is closer to some point in it than to any point in $H \setminus H'$ with high probability. Therefore, the algorithm has to actively evade points in H' to discover points in $H \setminus H'$.

Adverserially Robust Learning. Deep learning methods, in general, are not robust to adversarial examples (Zhang et al., 2020; Miller et al., 2020; Akhtar & Mian, 2018). Recent developments such as (Lecuyer et al., 2019; Cohen et al., 2019) have led to methods achieving provable robustness with respect to ℓ_p-norm perturbations for continuous domains. A standard procedure to adapt these ideas for discrete domains is to pre-process the inputs by first mapping them to ℓ_p space before classification. As noted by (Hu et al., 2022), efficient non-adaptive algorithms for the exact recovery problems can be used to do exactly this. A non-adaptive algorithm for the generalized Mastermind problem, for instance, can be used to losslessly map the input to a t-dimensional vector of responses to the t queries made by the algorithm.

1.2. Our Contributions

For Problem 1, we present a two-round adaptive deterministic algorithm which makes $2^{O(\sqrt{d \cdot \log d \cdot \log n})}$ queries to retrieve H. On the negative side, we show that any r-round adaptive algorithm must make $\exp(\Omega(d^{3^{-(r-1)}}))$ queries even when the input has $O(d^3)$ points. Firstly, this implies that algorithms with $\text{poly}(d)$ query complexity must use $\Omega(\log \log d)$ rounds of adaptivity. Also, for the special case of non-adaptive algorithms, this corresponds to an $\exp(\Omega(d))$ query lower bound, and for the case of two-round adaptive algorithms a lower bound of $\exp(\Omega(d^{1/3}))$ queries. For Problem 2, we propose a deterministic algorithm in Section 4. The algorithm makes $O(n^{\lfloor d/2 \rfloor})$ queries over $O(n+d)$ adaptive rounds. Finally, we give a deterministic algorithm for Problem 3 that makes $O(nd)$ queries in Section 5. This shows that allowing extra characters in the queries of Problem 1 makes the problem much easier. A simple information-theoretic lower bound shows that this is essentially optimal; any randomized algorithm that succeeds with constant probability must make $\Omega(\frac{nd}{\log d})$ queries.

1.3. Related Work

Past work has studied several variants of the mastermind problem, sometimes under different names. The work of (Fernández et al., 2019) studied the Mastermind problem in which the codemaker picks $\mathbf{x} \in \{-k, -k+1, \cdots, k-1, k\}^d$ and the codebreaker tries to infer $\mathbf{x}$ by making ℓ_p distance queries in which they pick $\mathbf{y} \in \{-k, -k+1, \cdots, k-1, k\}^d$ of their choice and are told $\|\mathbf{x} - \mathbf{y}\|_p$. Their tech-

niques give non-adaptive algorithms for separable distance metrics (which are metrics that decompose into coordinate sums) such as ℓ_p-norms, smooth max, Huber loss, etc. The study of the Mastermind problem with non-separable distance metrics such as edit distance, p-Dynamic Time Warping, and Fréchet distances was initiated by (Hu et al., 2022). Inference problems involving graph metrics have been considered in (Jiang & Polyanskii, 2019; Rodríguez-Velázquez et al., 2014). Permutation-based variants of Mastermind are studied in (Afshani et al., 2019; El Ouali & Sauerland, 2020).

The well-known coin-weighing problem is equivalent to the original Mastermind problem with two colors (Alon et al., 1996). In this problem, given n coins each of whose weight is either w_0 or w_1, the goal is to determine the weight of each coin in a small number of weighings using a spring balance. Optimal bounds for this problem were achieved by (Bshouty, 2009). The problem of group testing, introduced by (Dorfman, 1943), is also closely related. In this problem, there is a group of people out of which some have been infected by a disease. There is a testing procedure that, when queried with a subset of people, can determine either that there is someone in the subset who is infected or that no one is. The goal of the problem is to identify the infected population using the testing procedure as few times as possible. The work of (Coja-Oghlan et al., 2020) obtains optimal non-adaptive and two-round adaptive group testing algorithms. Several other results for group testing are surveyed in (Aldridge et al., 2019).

2. Notation

We use $\widetilde{O}(.)$ to hide factors that are logarithmic in the dimension d. The set $\{1, \cdots, l\}$ is denoted by $[l]$. We use $\{0, 1, \cdots, k\}^d$ to denote the set of vectors each of whose coordinates is in $[k] \cup \{0\}$. For $\mathbf{x}, \mathbf{y} \in \{0, 1, \cdots, k\}^d$, the Hamming distance between $\mathbf{x}$ and $\mathbf{y}$, denoted by $\text{dist}(\mathbf{x}, \mathbf{y})$, is the number of coordinates $i \in [d]$ such that $\mathbf{x}[i] \neq \mathbf{y}[i]$. For a subset $S \subset \{0, 1\}^d$ and a point $\mathbf{x} \in \{0, 1\}^d$ we use the notation $\text{dist}(\mathbf{x}, S)$ to denote $\min_{\mathbf{y} \in S} \text{dist}(\mathbf{x}, \mathbf{y})$. The support of $\mathbf{x}$, denoted by $\text{Supp}(\mathbf{x})$, is the set of non-zero coordinates of $\mathbf{x}$ and the Hamming weight of $\mathbf{x}$, denoted by $\text{wt}(\mathbf{x})$, is the cardinality of the support. For a set of coordinates $I \subset [d]$, the *restriction* of $\mathbf{x}$ to I is the vector in $\{0, 1, \cdots, k\}^{|I|}$ obtained by deleting all the coordinates of $\mathbf{x}$ except those in I. We denote the restriction of $\mathbf{x}$ to the set I by $\mathbf{x}_I$. For a set $S \subset \mathbb{R}^d$, the convex hull of S will be denoted by $\textsc{Conv}(S)$ and the span of the points in S will be denoted by $\text{Span}(S)$. For a subspace W of $\mathbb{R}^d$, its orthogonal complement is denoted by $W^\perp$. We use S^{d-1} to denote the unit sphere in $\mathbb{R}^d$.

Algorithm 1 Two-Round Adaptive Algorithm

1: For $t := 2^{O\left(\sqrt{d \cdot \log d \cdot \log(n)}\right)}$, query t points $\mathbf{y}_1, \ldots, \mathbf{y}_t$ which are sampled independently and uniformly at random from $\{0, 1\}^d$. Let $\mathbf{z}_1, \ldots, \mathbf{z}_t$ be the responses to the queries where $\mathbf{z}_i = \text{query}(\mathbf{y}_i)$.
2: For each $\mathbf{z}_i$, query all points at Hamming distance at most r from it where $r = O(\sqrt{(d \log n)/\log d})$.
3: Output the set of all points discovered by the queries made by the algorithm.

3. Results for Problem 1

In this section, we present results for Problem 1. Recall that, in this setting the hidden set H consists of n points from the d-dimensional hypercube $\{0, 1\}^d$. The query model allows the codebreaker to query $\mathbf{q} \in \{0, 1\}^d$ to learn the point in H with the smallest Hamming distance to $\mathbf{q}$. We use $\text{query}(\mathbf{q})$ to denote this point.

In the following sections, we give upper and lower bounds on the query complexity of Problem 1.

3.1. A Simple 2-Round Adaptive Algorithm

We analyze the natural algorithm which makes random queries in the first round and then uses the information revealed to adaptively make queries in the second round.

Overview of Algorithm 1. In the first round, the algorithm queries $t = 2^{\tilde{O}(\sqrt{d \log n})}$ points $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_t$ uniformly at random from $\{0, 1\}^d$. In the second round, the algorithm queries all points within Hamming distance r of points discovered in the first round.

Theorem 3.1. *Let H be a hidden set with $n \leq 2^{o(d/\log d)}$ points. Algorithm 1 makes at most $2^{O(\sqrt{d \cdot \log d \cdot \log n})}$ queries and recovers H with probability at least $2/3$.*

Our main lemma is to show that, for any fixed $\mathbf{x} \in H$, the first round recovers a point $\mathbf{z} \in H$ such that $\text{dist}(\mathbf{x}, \mathbf{z}) \leq r = \tilde{O}(\sqrt{d \log n})$ with probability at least $(1 - 1/3n)$. It then follows by a union bound that, with probability at least $2/3$, each point in the hidden set is at a distance at most r from some point recovered in round 1. Thus, the algorithm recovers all points after the second round of queries with constant probability.

Lemma 3.2. *Fix a hidden point $\mathbf{x} \in H$ and let $\mathbf{z}$ be the nearest point to $\mathbf{x}$ among the points $\mathbf{z}_i$ learned after the first round of queries. With probability at least $(1 - 1/3n)$, we have dist$(\mathbf{x}, \mathbf{z}) \leq r$.*

Proof. Let $\mathbf{y} = \text{argmin}_{\mathbf{y}_i} \text{dist}(\mathbf{x}, \mathbf{y}_i)$ be the nearest query to $\mathbf{x}$ among the queries made in the first round. Let $\mathbf{x}' \in H$ be some arbitrary but fixed point hidden point satisfying $\text{dist}(\mathbf{x}, \mathbf{x}') > r$ (the lemma trivially follows if no such $\mathbf{x}'$ exists). We show that $\text{dist}(\mathbf{x}, \mathbf{y}) < \text{dist}(\mathbf{x}', \mathbf{y})$ holds with

probability at least $(1 - 1/3n^2)$. A union bound then shows that with probability at least $(1 - 1/3n)$, we simultaneously have $\text{dist}(\mathbf{x}, \mathbf{y}) < \text{dist}(\mathbf{x}'', \mathbf{y})$ for all $\mathbf{x}'' \in H$ satisfying $\text{dist}(\mathbf{x}'', \mathbf{x}) > r$. It then follows that $\text{dist}(\mathbf{x}, \text{query}(\mathbf{y})) \leq r$ holds with probability $(1 - 1/3n)$ thus proving the lemma.

We have reduced our task to showing that $\text{dist}(\mathbf{x}, \mathbf{y}) < \text{dist}(\mathbf{x}', \mathbf{y})$ holds with probability at least $(1 - 1/3n^2)$. We now make some assumptions that simplify the analysis. For the rest of the analysis, we assume that $\mathbf{x}$ is the origin (i.e., all the d coordinates of $\mathbf{x}$ are 0). This corresponds to "shifting the origin to $\mathbf{x}$" by XOR-ing all hypercube points with $\mathbf{x}$. This operation preserves distances between any two points, so our analysis will not lose any generality.

We now have $\text{dist}(\mathbf{x}, \mathbf{y}) = \text{wt}(\mathbf{y})$ where $\text{wt}(\mathbf{y}) = |\text{supp}(\mathbf{y})|$ is the number of coordinates in the support of $\mathbf{y}$. It is also easy to see that $\text{dist}(\mathbf{x}', \mathbf{y}) = \text{wt}(\mathbf{x}') + \text{wt}(\mathbf{y}) - 2I$ where $I := |\text{Supp}(\mathbf{x}') \cap \text{Supp}(\mathbf{y})|$. It follows that $\text{dist}(\mathbf{x}, \mathbf{y}) < \text{dist}(\mathbf{x}', \mathbf{y})$ is equivalent to

$$\text{wt}(\mathbf{x}') > 2I. \tag{1}$$

In the rest of the proof, we focus on showing that Equation (1) holds with probability at least $(1 - 1/3n^2)$. We now define some useful events that will help us prove this upper bound on I. Specifically, we let G_1 to be the good event that $\text{wt}(\mathbf{y}) = d/2 - \Omega(\sqrt{d \log t})$. This is a good event because we can use the fact that $\mathbf{y}$ has a small support to argue that the size I of the common support of $\mathbf{x}'$ and $\mathbf{y}$. Also, we let G_2 be the good event that I is not too much more than its expectation $(\text{wt}(y)\text{wt}(\mathbf{x}'))/d$.

- For an integer $w \in [0, d]$, let E_w denote the event that $\text{wt}(\mathbf{y}) = w$.
- Let G_1 denote the event that $\text{wt}(\mathbf{y}) \in [d/4, d/2 - (\sqrt{d \log t})/8]$.
- Let G_2 denote the event that $I < \mu + \sqrt{3\mu \log(12n^2)}$ where $\mu = (\text{wt}(\mathbf{x}')\text{wt}(\mathbf{y}))/d$.

Using some simple algebra, we can now show the following claim. The details are provided in Appendix B.

Claim 3.3. Suppose that the number of queries t satisfies $t = 2^{\Omega((d \log n)/r)}$. If the good events G_1, G_2 both occur, then $\text{wt}(\mathbf{x}') > 2I$.

Note that our choice of parameters t, r in Algorithm 1 satisfy the first condition of the above claim. Therefore, our task reduces to proving that $G_1 \cap G_2$ holds with probability $(1 - 1/3n^2)$ to finish the proof of Lemma 3.2. This is what we do next.

First, we show that G_1 holds with high probability. We observe that $\text{wt}(\mathbf{y})$ is the minimum of t i.i.d binomial random variables distributed as $\text{Bin}(d, 1/2)$. The claim below then follows from standard concentration inequalities. Its proof is deferred to Appendix B.

Claim 3.4. $\Pr[G_1] \geq (1 - 1/6n^2)$.

Next, we want to show that the upper bound on I guaranteed by event G_2 also holds with high probability. The key observation is the following: conditioned on $\text{wt}(\mathbf{y}) = w$, the distribution of $\mathbf{y}$ is uniform over points in $\{0, 1\}^d$ with weight w. Hence, the distribution of I is hypergeometric with mean $\mu = (w \cdot \text{wt}(\mathbf{x}'))/d$. Using tail bounds for the hypergeometric distribution we then show that $I = \mu + O(\sqrt{\mu \log n})$ holds with high probability. The full details are provided in Appendix B.

Claim 3.5. For any integer $w \in [d/4, d/2]$ we have $\Pr[G_2 \mid E_w] \geq (1 - 1/6n^2)$.

The below claim follows from Claim 3.4, Claim 3.5, and the law of total probability. The details are again deferred to Appendix B.

Claim 3.6. $\Pr[G_1 \cap G_2] \geq (1 - 1/3n^2)$.

This completes the proof of Lemma 3.2 ∎

The following lemma bounds the number of queries made by Algorithm 1 concluding the proof of Theorem 3.1.

Lemma 3.7. *Algorithm 1 makes at most* $2^{O(\sqrt{d \cdot \log d \cdot \log n})}$ *queries.*

Proof. The number of queries in the first round is clearly $2^{O(\sqrt{d \cdot \log d \cdot \log n})}$. In round 2, the algorithm queries all points at a Hamming distance at most $r = O(\sqrt{(d \log n)/\log d})$ from the points discovered in round 1. This is at most $t \cdot \sum_{i=0}^{r} \binom{d}{i} \leq td^{r+1} = 2^{O(\sqrt{d \cdot \log d \cdot \log n})}$. ∎

3.2. A Lower Bound Against r-Round Adaptive Algorithms

The main goal of the section is to prove query complexity lower bounds. In particular, we study the trade-off between query complexity and adaptivity.

We say that a randomized algorithm has success probability p, if, for each input $H \subset \{0, 1\}^d$, it correctly learns all the points in H with probability at least p. The main result of this section is a lower bound on the query complexity of r-round adaptive randomized algorithms which have constant success probability.

Theorem 3.8. *Let d be a sufficiently large integer and $r = O(\log \log d)$ be any positive integer. Any r-round adaptive randomized algorithm for Problem 1 with success probability at least $2/3$ must make $\exp(\Omega(d^{3^{-(r-1)}}))$ queries even when the size of the hidden set H is promised to be $O(d^3)$.*

Let $\textsc{Det}(q, r)$ denote the set of all deterministic algorithms that make at most q queries over r adaptive rounds. We shall use the following version of Yao's lemma which reduces proving randomized query complexity lower bounds to that of designing hard distributions for deterministic algorithms.

Lemma 3.9 (Yao's Principle)**.** *Suppose that there exists a distribution $\mathcal{D}$ over instances of Problem 1 such that for every deterministic algorithm $\mathcal{A} \in \textsc{Det}(q, r)$, we have, $\Pr_{H\sim\mathcal{D}}[\mathcal{A}$ succeeds on $H] < \delta$; then any r-round randomized algorithm which makes at most q queries has success probability at most δ.*

This motivates the following definition of input distributions, which are hard for deterministic algorithms.

Definition 3.10 ((d, m, q, r, δ)-Hard Distribution)**.** Let $\mathcal{D}$ be a distribution over subsets of $\{0, 1\}^d$ that contain at most m points. Such a distribution $\mathcal{D}$ is called (d, m, q, r, δ)-hard if any algorithm $\mathcal{A} \in \textsc{Det}(q, r)$ has at most a δ-probability of learning a hidden set drawn from $\mathcal{D}$.

It follows from Yao's lemma that if a (d, m, q, r, δ)-hard distribution exists, then any r-round randomized algorithm with query complexity q has success probability at most δ. Next, we construct a hard distribution for non-adaptive algorithms.

Lemma 3.11 (Hard Distribution for 1-Round Algorithms)**.** *For some constant $c \in (0, 1)$, there exists a distribution $\mathcal{D}_1$ which is $(d, d^2, 2^{cd}, 1, 2^{-\Omega(d)})$-hard.*

Proof. For a point $\mathbf{x} \in \{0, 1\}^d$ let $N(\mathbf{x}, r) \subset \{0, 1\}^d$ be the set of points at Hamming distance exactly r from $\mathbf{x}$.

The hard distribution $\mathcal{D}_1$ generates an input H as follows:

(i) Sample a uniformly random point $\mathbf{u}$ from $\{0, 1\}^d$.

(ii) Add all the points in $N(\mathbf{u}, 2)$ to H.

(iii) Add a uniformly random subset S of $N(\mathbf{u}, 1)$ to H.

We now show that for some constant $c \in (0, 1)$, any algorithm in $\textsc{Det}(2^{cd}, 1)$ learns $H \sim \mathcal{D}_1$ with probability at most $2^{-\Omega(d)}$. The intuition is that if the number of queries is 2^{cd}, all the queries of the algorithm will be at a distance greater than 2 from u with high probability. In this case, the points in $N(u, 2)$ "block" the algorithm from learning which subset S of $N(u, 1)$ was added to H in step (iii). We formalize this argument below.

Let $\mathcal{A} \in \textsc{Det}(s, 1)$ be a non-adaptive deterministic algorithm that queries the points $\mathbf{y}_1, \ldots, \mathbf{y}_s$. Let E be the event that for all $i \in [s]$ we have $\text{dist}(\mathbf{y}_i, \mathbf{u}) \geq 2$. Since $\mathbf{u}$ is sampled uniformly at random, for a fixed $\mathbf{y}_i$, $\text{dist}(\mathbf{y}_i, \mathbf{u}) \geq 2$ holds with probability $(1 - (1 + d)2^{-d}) \geq (1 - 2d \cdot 2^{-d})$. A union bound then gives $\Pr[\overline{E}] \geq (1 - 2sd \cdot 2^{-d})$.

Now, observe that if event E occurs, for each query $\mathbf{y}_i$ we have $\text{dist}(\mathbf{y}_i, N(\mathbf{u}, 2)) < \text{dist}(\mathbf{y}_i, N(\mathbf{u}, 1))$. Moreover,

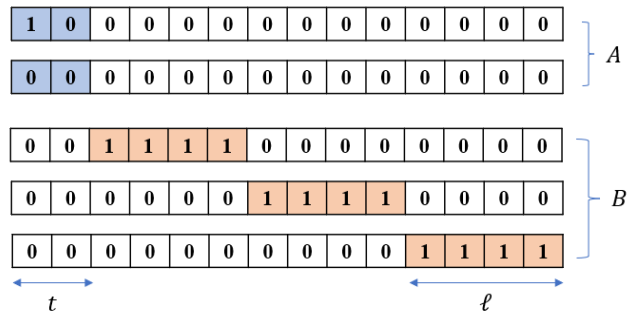

Figure 1. The figure illustrates how a hard distribution for r round algorithms can be used to create a hard distribution for $r + 1$ round algorithms.

since all the points in $N(\mathbf{u}, 2)$ are in H, the responses to the queries are entirely determined by the choice of $\mathbf{u}$ (and, in particular, are independent of S). It follows that if the event E occurs, then the algorithm has at most a 2^{-d} probability of correctly guessing S. The success probability of the algorithm, therefore, is at most $\Pr[E] \cdot 2^{-d} + \Pr[\overline{E}] \leq 2^{-d} + 2sd \cdot 2^{-d} = 2^{-\Omega(d)}$ if $s \leq 2^{cd}$ for a sufficiently small constant $c \in (0, 1)$. ∎

We now show that hard distributions can be constructed recursively: a hard distribution for $(r + 1)$ round algorithms can be constructed using a hard distribution for r-round algorithms. The new distribution will, however, be over points whose dimension is significantly larger (in fact, it will be approximately the cube of the dimension of the former).

Lemma 3.12 (Inductive Step)**.** *Suppose that a (t, m, q, r, δ_1)-hard distribution exists; then a $(t', m', q, r + 1, \delta_1 + \delta_2)$-hard distribution also exists for some $t' \leq 2500t^2(t + \log_2(q/\delta_2)) + t$ and $m' \leq m + 25(t + \log_2(q/\delta_2))$.*

Proof. Let $\mathcal{D}_r$ be the (t, m, q, r, δ_1)-hard distribution. Let $t' = 2500t^2(t + \log_2(q/\delta_2)) + t$. The hard distribution $\mathcal{D}_{r+1}$ generates an input $H_{r+1} \subset \{0, 1\}^{t'}$ as follows (see the illustration above):

(i) Sample $H_r \subset \{0, 1\}^t$ from the distribution $\mathcal{D}_r$.

(ii) Let $A \subset \{0, 1\}^{t'}$ be the set of points obtained by padding $(t' - t)$ zeros to the end of each point in H_r.

(iii) Let $\ell = 100t^2$ and $m_1 = 25(t + \log_2(q/\delta_2))$. Define $B = \{\mathbf{x}_1, \ldots, \mathbf{x}_{m_1}\}$ where $\mathbf{x}_i$ is given by:

$$\mathbf{x}_i[j] = \begin{cases} 1 & \text{If } j \in [t + (i - 1) \cdot \ell + 1,\ t + i \cdot \ell] \\ 0 & \text{Otherwise} \end{cases}$$

(iv) For $\mathbf{u}$ sampled uniformly at random from $\{0,1\}^{t'}$ let $A_u = \{\mathbf{u} \oplus \mathbf{x} \mid \mathbf{x} \in A\}$ and $B_u = \{\mathbf{u} \oplus \mathbf{x} \mid \mathbf{x} \in B\}$ where $\oplus$ denotes the bitwise-XOR operation.

(v) Let $H_{r+1} = A_u \cup B_u$.

It is easy to check that the bounds on the size of H_{r+1} and the dimension t' satisfy the lemma statement.

Next, we prove that $\mathcal{D}_{r+1}$ is indeed a hard distribution for $(r+1)$-round adaptive algorithms. Consider an algorithm in $\textsc{Det}(r+1, q)$ and an input H_{r+1} drawn from $\mathcal{D}_{r+1}$. We show that after the first round of queries, with high probability, the algorithm gains no information about the set H_r. However, the input H_r is itself drawn from a distribution that is hard for r-round algorithms; thus the algorithm has a low probability of learning A in the remaining r rounds.

Let $C \subset \{0,1\}^{t'}$ denote the set of 2^t points whose support is a subset of the first t coordinates. The main claim we show is that if $\mathbf{z}$ is a uniformly random point in $\{0,1\}^{t'}$ then its distance to B is less than its distance to any point in C. To prove this claim, we will require the following simple observation whose proof is provided in Appendix B.

Claim 3.13. Let $\mathbf{x}$ be a point in B and $\mathbf{y}$ be a point in C. If $\mathbf{z}$ is a point in $\{0,1\}^{t'}$ satisfying $|\text{Supp}(\mathbf{x}) \cap \text{Supp}(\mathbf{z})| > \ell/2 + t$ then $\text{dist}(\mathbf{z}, \mathbf{x}) < \text{dist}(\mathbf{z}, \mathbf{y})$.

We now prove the main claim of the argument.

Claim 3.14. If $\mathbf{z}$ is sampled uniformly at random from $\{0,1\}^{t'}$ then $\text{dist}(\mathbf{z}, C) > \text{dist}(\mathbf{z}, B)$ with probability at least $1 - \delta_2/q$.

Proof. Fix a point $\mathbf{y} \in C$. We shall show that for a uniformly random point $\mathbf{z}$, we have $\text{dist}(\mathbf{z}, \mathbf{y}) \leq \text{dist}(\mathbf{z}, B)$ with probability at most $2^{-t} q^{-1} \delta_2$. The claim then follows by a union bound over the $|C| = 2^t$ points in C.

Let $\mathbf{x}$ be a point in B. Since $\mathbf{z}$ is chosen uniformly at random from $\{0,1\}^{t'}$ and $\text{wt}(\mathbf{x}) = \ell$, we have that $|\text{Supp}(\mathbf{x}) \cap \text{Supp}(\mathbf{z})|$ is a $\text{Bin}(\ell, 1/2)$ random variable. Using Fact A.2, the probability that $|\text{Supp}(\mathbf{x}) \cap \text{Supp}(\mathbf{z})| \geq (\ell/2 + 2t)$ is at least $\frac{1}{15}\exp(-64t^2/\ell) = \frac{1}{15}\exp(-16/25) > 1/30$. It follows from Claim 3.13 that $\text{dist}(\mathbf{z}, \mathbf{x}) < \text{dist}(\mathbf{z}, \mathbf{y})$ holds with probability at least $1/30$. Since $B = \{\mathbf{x}_1, \ldots, \mathbf{x}_{m_1}\}$ consists of points with pairwise disjoint support, the random variables $|\text{Supp}(\mathbf{x}_i) \cap \text{Supp}(\mathbf{z})|$ corresponding to different i are mutually independent. Thus, the probability that $\text{dist}(\mathbf{z}, B) > \text{dist}(\mathbf{z}, \mathbf{y})$ is at most $(29/30)^{m_1} < 2^{-t} q^{-1} \delta_2$. ∎ Claim 3.14

We now use the claim above to show that after the first round of queries, with high probability, the algorithm learns no information about the set of points H_r sampled in line (i).

Suppose that the algorithm queries the points $\mathbf{z}_1, \ldots, \mathbf{z}_s$ in the first round. Let E be the event that $\text{dist}(\mathbf{z}_i, C) > \text{dist}(\mathbf{z}_i, B_u)$ holds for all $\mathbf{z}_i$. Observe that $\text{dist}(\mathbf{z}_i, C) > \text{dist}(\mathbf{z}_i, B_u)$ holds iff $\text{dist}(\mathbf{z}_i \oplus \mathbf{u}, C) > \text{dist}(\mathbf{z}_i \oplus \mathbf{u}, B)$. Since $\mathbf{z} \oplus \mathbf{u}$ is distributed uniformly over $\{0,1\}^{t'}$, it follows from Claim 3.14, a union bound that $\Pr[\overline{E}] \leq \delta_2 s/q \leq \delta_2$.

Conditioned on event E, the responses to the queries in round 1 are determined by $\mathbf{u}$ and specifically are independent of the set $A_u \subset C$ and thus are also independent of the points H_r sampled in line (i). Therefore, if the event E occurs, the conditional distribution of H_r after the first round of queries is stochastically identical to the distribution of H_r prior to the queries.

Even if the algorithm is informed what $\mathbf{u}$ is after the first round of queries, it still has to learn the set of points H_r in the remaining r rounds. Since the points in H_r are from a (t, m, q, r, δ_1)-hard distribution, the algorithm learns H_r with probability at most δ_1. Therefore the success probability of the algorithm is bounded by $\Pr[E] \cdot \delta_1 + \Pr[\overline{E}] \leq \delta_1 + \delta_2$. ∎

We now combine Lemma 3.11 and Lemma 3.12 to complete the inductive proof and show the existence of a certain hard distribution against r-round adaptive algorithms. Due to space constraints, the proof of the below lemma has been moved to Appendix B.

Lemma 3.15. *For any integer $t \geq c_1$ and an integer r satisfying $1 \leq r \leq t$ there exists a distribution $\mathcal{D}_r$ which is $((100t)^{3^{r-1}}, t^{3^r}, 2^{ct}, r, 2/3)$-hard where c_1 is a sufficiently large constant and c is a constant in $(0,1)$.*

We now complete the proof of Theorem 3.8.

Proof of Theorem 3.8. Pick d sufficiently large and $r = O(\log\log d)$ so that $r \ll d^{3^{-(r-1)}}$. Setting $t = (d^{3^{-(r-1)}})/100$ in Lemma 3.15 shows the existence of a $(d, d^3, 2^{O(d^{3^{-(r-1)}})}, r, 1/3)$-hard distribution. Theorem 3.8 then follows by Yao's lemma. ∎ Theorem 3.8

4. A Deterministic Algorithm for Problem 2

In Problem 2, there is a hidden set $H = \{\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_n\}$ of n points on the unit sphere $S^{d-1} \subset \mathbb{R}^d$. The goal is to learn the set H while making queries to unit vectors in S^{d-1} where a query to the unit vector $\mathbf{q}$ returns a point in H which has the least Euclidean distance to $\mathbf{q}$. In other words, $\mathbf{x} = \underset{\mathbf{x}_i \in H}{\text{argmin}} \|\mathbf{x}_i - \mathbf{q}\|_2$. Note that since

$$\|\mathbf{x} - \mathbf{q}\|_2 = \|\mathbf{x}\|_2 + \|\mathbf{q}\|_2 - 2 \cdot \langle \mathbf{x}, \mathbf{q} \rangle = 2 - 2 \cdot \langle \mathbf{x}, \mathbf{q} \rangle,$$

the point $\mathbf{x}$ also has the maximum inner product with $\mathbf{q}$ among points in H. We present a deterministic algorithm that learns the set H using $n^{O(d)}$ queries using $O(n+d)$ rounds of adaptivity.

Overview of Algorithm 2: Suppose that the points in H have rank k. The algorithm first finds a basis B of k linearly

Algorithm 2 Deterministic Algorithm for Points on S^{d-1}.

1: Initialize $B := \phi$. {When the loop on line 2 terminates B will be a set of vectors which form a basis of H}.
2: **repeat**
3: Find an orthonormal basis $\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_t\}$ of $\text{Span}(B)^{\perp}$.
4: Query $\mathbf{v}_j$ and $-\mathbf{v}_j$ for $1 \leq j \leq t$. Let $C \subset H$ be the points learned.
5: **if** there exists a point $\mathbf{x}$ in C such that $\langle \mathbf{x}, \mathbf{v}_j \rangle \neq 0$ for some $\mathbf{v}_j$ **then**
6: $B = B \cup \{\mathbf{x}\}$.
7: **end if**
8: **until** No new point is added to B
9: Define $k := |B|$. { k is the rank of points in H.}
10: Initialize $\hat{H} := B$
{When the loop on line 11 terminates $\hat{H}$ will equal H. This loop finds all points in $H - B$.}
11: **repeat**
12: Find all the $(k-1)$ dimensional faces of $\text{Conv}(\hat{H})$.
13: Let D be the set of unit vectors in the $\text{Span}(\hat{H})$ normal to some $(k-1)$-face of the $\text{Conv}(\hat{H})$.
{Note that each $(k-1)$-face of $\text{Conv}(\hat{H})$ has two unit normal vectors.}
14: Let $E \subset H$ denote the set of new points learned on querying points in D.
15: $\hat{H} = \hat{H} \cup E$.
16: **until** $|E| = 0$
17: Output $\hat{H}$.

independent points in H. This is done by repeatedly querying points in the orthogonal complement of points discovered thus far until no new points are discovered. Henceforth, the algorithm only queries points in $\text{Span}(B)$ effectively reducing the problem to k dimensions. In each iteration, the algorithm constructs a convex hull of the points of H found so far and queries the normal vectors of all the $(k-1)$-dimensional faces of the convex hull thus constructed. If no new points are recovered, the algorithm terminates.

Theorem 4.1. *Algorithm 2 recovers the set $H \subset S^{d-1}$ containing n points with rank k while making at most $O(n^{\lfloor \frac{k}{2} \rfloor + 1} + kd)$ queries.*

It is easy to see that at the end of the first loop, Algorithm 2, recovers a basis $B \subseteq H$. We now show that the algorithm discovers all remaining points in the second loop. Towards this end, we state the following simple fact and move its proof to Appendix C.

Fact 4.2. Let T be a finite subset of S^{d-1}; then $\text{Conv}(T) \cap S^{d-1} = T$.

Lemma 4.3. *The output $\hat{H}$ of Algorithm 2 is H.*

Proof. Consider the stage of the algorithm when it is executing an iteration of the loop on line 11. Also, suppose that there is an undiscovered point $\mathbf{y} \in H \setminus \hat{H}$ at this stage. By Fact 4.2, $\text{Conv}(\hat{H}) = \hat{H}$ and therefore y is not in $\text{Conv}(\hat{H})$. Therefore, there is a $(k-1)$-face of $\text{Conv}(B)$ which separates $\mathbf{y}$ from all the points in $\hat{H}$. Therefore, the unit vector $\mathbf{m}$ normal to this face of $\text{Conv}(\hat{H})$ satisfies $\langle \mathbf{m}, \mathbf{y} \rangle > \langle \mathbf{m}, \mathbf{x} \rangle$ for all $\mathbf{x}$ in $\hat{H}$. Hence querying the unit vectors corresponding to all $(k-1)$-faces of $\text{Conv}(\hat{H})$ (line 14) recovers a point in $H \setminus \hat{H}$. This proves that the algorithm terminates only after all the points in H are recovered. ∎

To bound the query complexity of the algorithm, we use the following standard result about the number of facets of the convex hull of n points in d dimensions.

Lemma 4.4 (Upper Bound Theorem, (Theorem 5.5.1 in (Matousek, 2013)))**.** *The number of $(d-1)$-faces of the convex hull of n points in d dimensions is $O(n^{\lfloor \frac{d}{2} \rfloor})$.*

Lemma 4.5. *Algorithm 2 makes at most $O(n^{\lfloor \frac{k}{2} \rfloor + 1} + kd)$ queries where $k = rank(H)$.*

Proof. Algorithm 2 uses $O(kd)$ queries to recover a basis of H. Each iteration of the loop beginning in line 11 queries the two normal vectors to the faces of the convex hull of a set of at most n points in k dimensions. By Lemma 4.4, each iteration makes at most $O(n^{\lfloor \frac{k}{2} \rfloor})$ queries. Since there are at most n iterations of this loop, the total number of queries made by the algorithm overall is $O(n^{\lfloor \frac{k}{2} \rfloor + 1} + kd)$. ∎

5. A Near Optimal Algorithm for Problem 3

In this section, we present results for Problem 3. In this problem, the hidden set H is a subset of $\{0,1\}^d$ of cardinality n and that in each query the codebreaker learns $\min_{\mathbf{x} \in H} \text{dist}(\mathbf{x}, \mathbf{q})$ for $\mathbf{q} \in \{0,1,2\}^d$ of their choice. The codebreaker's goal is to learn H with as few queries as possible. For $\mathbf{x} \in \{0,1\}^d$ and $I \subset [d]$, $\mathbf{x}_I$ denotes the restriction of $\mathbf{x}$ to I (see Section 2 for a definition). Observe that the query model defined above is equivalent to the following query model which is more convenient to work with. In each query, the codebreaker picks $I \subset [d]$ and a vector $\mathbf{q} \in \{0,1\}^{|I|}$ and as a response to this query learns $\min_{\mathbf{x} \in H} \text{dist}(\mathbf{x}_I, \mathbf{q})$ which is the minimum distance of $\mathbf{q}$ to points in H when restricted to the indices in I. We will denote such a query by $\text{query}(I, \mathbf{q})$.

5.1. The Algorithm

In the following section, we present a deterministic algorithm for Problem 3 that finds H using $O(nd)$ queries.

Throughout this section, we define for each $i \in [d]$ the set $C_i = \{1, \ldots, i\}$. In the following lemma, we show that a point in H with a given prefix can be recovered using $O(d)$ queries. This lemma will be used as a subroutine in Algorithm 3.

Algorithm 3 A Deterministic Algorithm for the Stronger Query Model.

1: For $1 \leq i \leq d$, let C_i denote the set of coordinates $\{1, 2, \cdots, i\}$
2: Initialize $H^* := \phi$ {H^* is the subset of H learned by the algorithm.}
3: Initialize $N := \phi$ {N is the "most recently" learned subset of H}
4: Use Lemma 5.1 to find some point $\mathbf{x}_0$ in H (the prefix in the lemma can be set to be the null prefix).
5: Add $\mathbf{x}_0$ to N and H^*.
6: **while** N is non-empty **do**
7: **for** $\mathbf{x} \in H^*$ **do**
8: **for** i from 1 to d **do**
9: Let $\mathbf{x}_{\sim i} \in \{0,1\}^i$ be the i-th prefix of $\mathbf{x}$ but with the i-th coordinate negated.
10: **if** $\mathbf{x}_{\sim i}$ is not the prefix of any point in H^* <u>*and*</u> query$(C_i, \mathbf{x}_{\sim i}) = 0$ **then**
11: Use Lemma 5.1 to find a point in H with prefix $\mathbf{x}_{\sim i}$.
12: **end if**
13: **end for**
14: **end for**
15: Let N be the set of new points learned in this iteration of the while loop. $H^* \leftarrow H^* \cup N$.
16: **end while**
17: Output H^*

Lemma 5.1. *Given $\mathbf{x}' \in \{0,1\}^i$, with i satisfying $0 \leq i \leq d$, we can recover a point $\mathbf{x} \in H$ with prefix $\mathbf{x}'$ (given that such an $\mathbf{x}$ exists in H) using at most d queries.*

Proof. Suppose that $\mathbf{x}$ is a point in H with prefix $\mathbf{x}'$; since $\mathbf{x}'$ is a prefix of $\mathbf{x}$ we have query$(C_i, \mathbf{x}') = 0$. Let $(\mathbf{x}', 0)$ denote the point in $\{0,1\}^{i+1}$ obtained by appending a 0 to $\mathbf{x}'$. To determine the $(i+1)$-th coordinate we perform query$(C_{i+1}, (\mathbf{x}', 0))$. If the response to the query is 0, then $x_{i+1} = 0$; otherwise $x_{i+1} = 1$. We now know the prefix of $\mathbf{x}$ of length $(i+1)$. Repeating this for the subsequent coordinates reveals $\mathbf{x}$ using at most a total of d queries. ∎

Overview of Algorithm 3: Consider the execution of the algorithm when it has discovered a subset H^* of H. It finds the prefix p of some point in $H \setminus H^*$ such that p is not the prefix of any point in H^*. It then uses the subroutine of Lemma 5.1 to learn a point in H with this prefix, thereby learning a previously unknown point in H. To find p, the algorithm iterates over all prefixes of points in H^*. For each prefix q, the algorithm negates the last coordinate of q to obtain $\tilde{q}$. It then checks if $\tilde{q}$ is the prefix of any point in H^* and if this is not true queries $\tilde{q}$. If the query returns 0, then the algorithm has found the required prefix of an undiscovered point.

Theorem 5.2. *Algorithm 3 recovers the set $H \subset \{0,1\}^d$ while making at most $O(nd)$ queries.*

The following lemma proves the correctness of Algorithm 3.

Lemma 5.3. *The set H^* output by Algorithm 3 equals H.*

Proof. First, we prove that the algorithm terminates. The while loop ends as soon as no new points are learned in its most recent iteration. Therefore, there are at most n iterations of the while loop after which the algorithm terminates. Next, we show that the algorithm learns all points in H. Suppose that the execution of the algorithm is at the while loop on line 6. Also, suppose that there exists $\mathbf{y} \in H \setminus H^*$; we show that the algorithm learns a point in $H \setminus H^*$ in this iteration of the while loop. Among points in H^*, let $\mathbf{z}$ be the point that has the largest common prefix with $\mathbf{y}$ and the length of this prefix be l. Observe that $\mathbf{x}_{\sim(l+1)}$ is not the prefix of any point in H^* and that query$(C_{l+1}, \mathbf{x}_{\sim(l+1)}) = 0$ since $\mathbf{x}_{\sim(l+1)}$ is a prefix of $\mathbf{y}$. Therefore, we are done if we argue that z is contained in N. This is true since, if $z \in H^* \setminus N$, then a previous iteration of the while loop would have retrieved a point the length of whose common prefix with $\mathbf{y}$ would be at least $(l+1)$. This leads to a contradiction. ∎

Next, we bound the query complexity of Algorithm 3, completing the proof of Theorem 5.2.

Lemma 5.4. *Algorithm 3 makes $O(nd)$ queries.*

Proof. For each point in H, the algorithm makes at most one query for each of its prefixes. Having learned the prefix of some undiscovered point in line 10, the algorithm uses Lemma 5.1 to learn a new point making at most d queries. Therefore, the total number of queries made is $O(nd)$. ∎

A simple information-theoretic lower bound shows that Algorithm 3 is almost optimal.

Theorem 5.5. *Consider a randomized algorithm for Problem 3 that for any set $H \subset \{0,1\}^d$ containing n points, learns all the points in H with probability at least $2/3$. The worst-case query complexity of such an algorithm is $\Omega\left(\frac{nd}{\log d}\right)$.*

Proof. Let $\mathcal{D}$ denote the uniform distribution over all n-subsets of $\{0,1\}^d$. Let $\mathcal{A}$ be a deterministic algorithm that learns an input $H \sim \mathcal{D}$ with success probability at least $2/3$ while making at most t queries. By Yao's principle, it suffices to show that $t = \Omega(\frac{nd}{\log d})$.

Since $\mathcal{A}$ is deterministic, its output is completely determined by the responses to its queries. For each query, it receives a response of length $O(\log d)$ bits. It, therefore, outputs one of $2^{O(t \log d)}$ subsets of $\{0,1\}^d$ and can be correct on at most $2^{O(t \log d)}$ inputs. Since it succeeds with probability at least $2/3$, we must have $t = \Omega(\frac{1}{\log d} \cdot \log \binom{2^d}{n}) = \Omega(\frac{nd}{\log d})$. ∎

Adaptivity. While Algorithm 3 is almost optimal in terms of query complexity, it requires several rounds of adaptivity. In particular, the algorithm as described in Algorithm 3 can be implemented so as to use $O(n)$ rounds of adaptivity, i.e., $O(1)$ rounds to discover each new point. Alternatively, one can also get a d-round adaptive algorithm with the same query complexity as follows: for $i = 1, \ldots d$, in the i-th round, the algorithm learns the length i prefix of each point in H. Given all length i prefixes, the algorithm can figure out all length $(i + 1)$ prefixes using at most $2n$ queries. Therefore, overall, we can achieve a query complexity of $O(nd)$ using $O(\min(n, d))$ adaptive rounds.

It would be interesting if query-efficient algorithms that use fewer rounds of adaptivity exist. In any case, non-adaptive algorithms for Problem 3 can be ruled out via an argument almost identical to the one used to prove a lower bound for Problem 1 against non-adaptive algorithms.

Theorem 5.6. *Consider a non-adaptive randomized algorithm for Problem 3 that for any set $H \subset \{0, 1\}^d$ containing n points, learns all the points in H with probability at least $2/3$. Such an algorithm must make $\Omega(2^d)$ queries.*

6. Conclusions and Open Problems

This work introduces the natural extension of the Mastermind problem to the setting when there are multiple secrets. In the context of Problem 1, our work focused on understanding the query complexity of two-round adaptive algorithms. We proposed a $2^{\tilde{O}(\sqrt{d \log n})}$ query upper bound and showed that any algorithm with $\text{poly}(d)$ query complexity must use $\Omega(\log \log d)$ rounds of adaptivity. Designing such an algorithm with polynomial query complexity is a natural and interesting open question.

For Problem 2 we gave a simple deterministic algorithm with $n^{O(d)}$ query complexity. This query complexity arose from the fact that convex polytopes with n vertices in d dimensions can have as many as $n^{O(d)}$ faces. For this problem, we do not know of any non-trivial lower bounds and it would be nice to make progress in this direction.

In the case of Problem 3, we have an almost optimal algorithm that recovers the hidden set while making at most $O(nd)$ queries. It however suffers from having to use $O(\min(n, d))$ rounds of adaptive queries. While we show that any non-adaptive algorithms must make $\Omega(2^d)$ queries it would be interesting to establish the correct tradeoff between query complexity and adaptivity for Problem 3.

Another interesting direction to explore is the study of the generalized mastermind problem for non-separable distance metrics such as edit distance, Fréchet distance, and dynamic time warping.

Acknowledgements

We thank Rahul Ilango for the helpful discussions during the initial phases of this work. We would also like to thank Rajiv Gandhi for making the collaboration between the authors possible.

David P. Woodruff was supported in part by a Simons Investigator Award and NSF Grant No. CCF-2335412.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Afshani, P., Agrawal, M., Doerr, B., Doerr, C., Larsen, K. G., and Mehlhorn, K. The query complexity of a permutation-based variant of mastermind. *Discrete Applied Mathematics*, 260:28–50, 2019.

Akhtar, N. and Mian, A. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018. doi: 10.1109/ACCESS.2018.2807385.

Aldridge, M., Johnson, O., Scarlett, J., et al. Group testing: an information theory perspective. *Foundations and Trends® in Communications and Information Theory*, 15 (3-4):196–392, 2019.

Alon, N., Kozlov, D., and Vu, V. The geometry of coin-weighing problems. In *Proceedings of 37th Conference on Foundations of Computer Science*, pp. 524–532, 1996. doi: 10.1109/SFCS.1996.548511.

Bshouty, N. H. Optimal algorithms for the coin weighing problem with a spring scale. In *COLT*, volume 2009, pp. 82, 2009.

Chen, Z., Cunha, C., and Homer, S. Finding a hidden code by asking questions. In *International Computing and Combinatorics Conference*, pp. 50–55. Springer, 1996.

Chvátal, V. Mastermind. *Combinatorica*, 3(3):325–329, 1983.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pp. 1310–1320. PMLR, 2019.

Coja-Oghlan, A., Gebhard, O., Hahn-Klimroth, M., and Loick, P. Optimal group testing. In *Conference on Learning Theory*, pp. 1374–1388. PMLR, 2020.

Dorfman, R. The detection of defective members of large populations. *The Annals of mathematical statistics*, 14 (4):436–440, 1943.

El Ouali, M. and Sauerland, V. The exact query complexity of yes-no permutation mastermind. *Games*, 11(2):19, 2020.

Fernández, M., Woodruff, D. P., and Yasuda, T. The query complexity of mastermind with lp distances. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2019.

Goodrich, M. T. On the algorithmic complexity of the mastermind game with black-peg results. *Information Processing Letters*, 109(13):675–678, 2009.

Hu, Z., Li, X., Woodruff, D. P., Zhang, H., and Zhang, S. Recovery from non-decomposable distance oracles. *arXiv preprint arXiv:2209.05676*, 2022.

Janson, S. Large deviation inequalities for sums of indicator variables. *arXiv preprint arXiv:1609.00533*, 2016.

Jiang, Z. and Polyanskii, N. On the metric dimension of cartesian powers of a graph. *Journal of Combinatorial Theory, Series A*, 165:1–14, 2019.

Knuth, D. E. The computer as master mind. *Journal of Recreational Mathematics*, 9:1–6, 1977.

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 656–672. IEEE, 2019.

Matousek, J. *Lectures on discrete geometry*, volume 212. Springer Science & Business Media, 2013.

Miller, D. J., Xiang, Z., and Kesidis, G. Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks. *Proceedings of the IEEE*, 108(3):402–433, 2020. doi: 10.1109/JPROC.2020.2970615.

Rodríguez-Velázquez, J. A., Yero, I. G., Kuziak, D., and Oellermann, O. R. On the strong metric dimension of cartesian and direct products of graphs. *Discrete Mathematics*, 335:8–19, 2014.

Zhang, W. E., Sheng, Q. Z., Alhazmi, A., and Li, C. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41, 2020.

A. Some Useful Concentration Inequalities

The binomial distribution with n trials and success probability p is denoted by $\text{Bin}(n, p)$. The following are some standard concentration inequalities for the binomial distribution.

Fact A.1 (Hoeffding's Inequality)**.** Let Y be random variable sampled from $\text{Bin}(d, 1/2)$. For $t > 0$ we have,

$$\Pr[\,|Y - d/2| \geq t] \;\leq\; 2 \cdot \exp(-2t^2/d).$$

The following is an *anti-concentration* inequality for binomial random variables.

Fact A.2 (Anti-Concentration Inequality for the Binomial Distribution)**.** Let Y be random variable sampled from $\text{Bin}(d, 1/2)$. When $0 \leq t \leq d/8$ we have:

$$\Pr[\,Y \leq\; d/2 - t] \;\geq\; \frac{1}{15} \cdot \exp(-16t^2/d).$$

The above fact implies the following inequality for the minimum of m independent binomial random variables.

Corollary A.3. *Suppose that $Y_1, Y_2, \cdots, Y_m$ are m independent $\text{Bin}(d, 1/2)$ random variables. Define $Y_{\min}$ to be the minimum of the random variables Y_i, i.e., $Y_{\min} = \min_i Y_i$. For $m = 2^{o(d)}$ and $\frac{1}{2^m} < \delta < 1$ we have,*

$$\Pr\left[\, Y_{\min} \leq \frac{d}{2} - \frac{1}{4}\sqrt{d(\log m - \log(15 \cdot \log(1/\delta)))}\,\right] \geq 1 - \delta.$$

We now define the hypergeometric distribution and state a corresponding concentration inequality.

Definition A.4 (Hypergeometric Distribution)**.** Suppose that d balls are drawn randomly without replacement from an urn containing N balls out of which K are white. The distribution of the random variable X which equals the number of white balls drawn is denoted by $\text{Hypergeometric}(N, K, d)$.

Fact A.5 (Theorem 4 of (Janson, 2016))**.** Let $Y \sim \text{Hypergeometric}(N, K, d)$ and $\mu = \mathbb{E}[Y] = \frac{NK}{d}$. For $\varepsilon > 0$ less than a sufficiently small constant,

$$\Pr[\,|Y - \mu| \geq \varepsilon\mu] \leq 2 \cdot \exp\left(\frac{-\varepsilon^2\mu}{3}\right).$$

B. Missing Details from Section 3

Proof of Claim 3.3 If the event $G_1 \cap G_2$ occurs then we have,

$$\begin{aligned}
2I &< 2\frac{\text{wt}(\mathbf{y}) \cdot \text{wt}(\mathbf{x}')}{d} + 2\sqrt{\frac{3\text{wt}(\mathbf{y}) \cdot \text{wt}(\mathbf{x}') \cdot \log(12n^2)}{d}} \\
&< \frac{2\left(\frac{d}{2} - \frac{1}{8}\sqrt{d \cdot \log t}\right) \cdot \text{wt}(\mathbf{x}')}{d} + 2\sqrt{3 \cdot \frac{\frac{d}{2} \cdot \text{wt}(\mathbf{x}') \cdot \log(12n^2)}{d}} \\
&= \text{wt}(\mathbf{x}') - \frac{1}{4} \cdot \text{wt}(\mathbf{x}') \cdot \sqrt{\frac{\log t}{d}} + \sqrt{6 \cdot \text{wt}(\mathbf{x}') \cdot \log(12n^2)}.
\end{aligned}$$

Therefore, $\text{wt}(x') > 2I$ holds if

$$\frac{1}{4} \cdot \text{wt}(\mathbf{x}') \cdot \sqrt{\frac{\log t}{d}} > \sqrt{6 \cdot \text{wt}(\mathbf{x}') \cdot \log(12n^2)}.$$

The above inequality on rearranging is equivalent to $t > \exp(96\, d\, \log(12n^2)/\text{wt}(x'))$. Since $\text{wt}(x') > r$, we conclude that $t > \exp(96\, d\, \log(12n^2)/r)$ is a sufficient condition for $\text{wt}(\mathbf{x}') > 2I$. ∎ Claim 3.3

Proof of Claim 3.4 The queries $\mathbf{y}_1, \ldots, \mathbf{y}_t$ are chosen independently uniformly at random, and therefore, the distribution of the random variable $\text{dist}(\mathbf{x}, \mathbf{y}_i) = \text{wt}(\mathbf{y}_i)$ is $\text{Bin}(d, 1/2)$. Since $\mathbf{y}$ is the nearest query to $\mathbf{x}$ it follows that $\text{wt}(\mathbf{y})$ is the minimum of t independent $\text{Bin}(d, 1/2)$ random variables. If $n = 2^{o(d/\log d)}$ we have that $t = 2^{O(\sqrt{d \log d \log n})} = 2^{o(d)}$.

Applying Corollary A.3, we conclude that $\mathrm{wt}(\mathbf{y}) < d/2 - \frac{1}{4}\sqrt{d \cdot (\log t - \log(15(\log 12n^2)))}$ holds with probability at least $(1 - 1/12n^2)$. For sufficiently large d we have, $\log(15(\log 12n^2)) \leq 2\log d \leq \frac{3}{4}\log t$. Therefore, $\mathrm{wt}(y) < d/2 - (\sqrt{d \log t})/8$ holds with probability at least $(1 - 1/12n^2)$.

To show the lower bound on $\mathrm{wt}(y)$ we use Fact A.1. The probability that a binomial random variable $\mathrm{Bin}(d, 1/2)$ is less than $d/4$ is at most $2\exp(-d)$. By a union bound the probability that the minimum of $t = 2^{o(d)}$ such binomial random variables is less than $d/4$ is at most $\exp(-\Omega(d)) \leq 1/12n^2$. Therefore, $\mathrm{wt}(\mathbf{y}) \geq d/4$ with probability at least $(1 - 1/12n^2)$.

A final union bound shows that both the lower and upper bounds on $\mathrm{wt}(\mathbf{y})$ hold with probability at least $(1 - 1/6n^2)$. ∎ Claim 3.4

Proof of Claim 3.5 Conditioning on $\mathrm{wt}(\mathbf{y}) = w$ (the event E_w), by symmetry, the distribution of $\mathbf{y}$ is uniform over the points in the hypercube with weight w. It follows that conditioned on E_w, the random variable I is distributed according to $\mathrm{Hypergeometric}(d, \mathrm{wt}(\mathbf{y}), \mathrm{wt}(\mathbf{x}'))$ (see Definition A.4). The claim then follows by setting $\varepsilon = \sqrt{3\log(12n^2)/\mu}$ and $\mu = (\mathrm{wt}(\mathbf{y}) \cdot \mathrm{wt}(\mathbf{x}'))/d$ in Fact A.5. ∎ Claim 3.5

Proof of Claim 3.6 Let J denote the set of integers in the interval $[d/4, d/2 - (\sqrt{d \log t})/8]$. Observe that $G_1 = \bigcup_{w \in J} E_w$. It follows that:

$$\begin{aligned}\Pr[G_1 \cap G_2] &= \sum_{w \in J} \Pr[G_2 \cap E_w] = \sum_{w \in J} \Pr[G_2 \mid E_w] \Pr[E_w] \geq (1 - 1/6n^2) \cdot \sum_{w \in J} \Pr[E_w] \\ &= (1 - 1/6n^2) \Pr[G_1] \geq (1 - 1/6n^2)^2 \geq 1 - 1/3n^2.\end{aligned}$$

∎ Claim 3.6

Proof of Claim 3.14 Note that we have the following formula for the Hamming distance between $\mathbf{a}, \mathbf{b} \in \{0,1\}^{t'}$: $\mathrm{dist}(\mathbf{a}, \mathbf{b}) = \mathrm{wt}(\mathbf{a}) + \mathrm{wt}(\mathbf{b}) - 2|\mathrm{Supp}(\mathbf{a}) \cap \mathrm{Supp}(\mathbf{b})|$. Using this formula and the fact that $|\mathrm{Supp}(\mathbf{x})| = \ell$ and $0 \leq |\mathrm{Supp}(\mathbf{y})| \leq t$ we obtain,

$$\begin{aligned}&\mathrm{dist}(\mathbf{z}, \mathbf{x}) - \mathrm{dist}(\mathbf{z}, \mathbf{y}) \\ &= \mathrm{wt}(\mathbf{x}) - 2|\mathrm{Supp}(\mathbf{x}) \cap \mathrm{Supp}(\mathbf{z})| - \mathrm{wt}(\mathbf{y}) + 2|\mathrm{Supp}(\mathbf{y}) \cap \mathrm{Supp}(\mathbf{z})| \\ &< \ell - 2 \cdot (\ell/2 + t) - 0 + 2t = 0.\end{aligned}$$

∎ Claim 3.14

Proof of Lemma 3.15 We shall prove this by the following induction. For each $i \in [r]$, we show that there is a $(d(i), m(i), 2^{ct}, i, i/3r)$-hard distribution where

$$d(i) \leq 20^{1.5(3^{i-1}-1)} \cdot t^{3^{i-1}} \text{ and } m(i) \leq t^{3^i}.$$

The lemma then follows by setting $i = r$, since $d(r) \leq 20^{1.5(3^{r-1}-1)} \cdot t^{3^{r-1}} \leq (100t)^{3^{r-1}}$.

The bounds on $m(1)$ and $d(1)$ in the base case $i = 1$ follow from Lemma 3.11 which guarantees the existence of $(t, t^2, 2^{ct}, 1, 2^{-\Omega(t)})$-hard distribution. Suppose that the claim holds until $i = j$ for some $j \geq 1$.

We now use Lemma 3.12 to perform the induction step. Setting $\delta_2 = 1/r \geq 1/t$ in this lemma, we deduce that there is a $(d(j+1), m', 2^{ct}, j+1, (j+1)/r)$-hard distribution with $d(j+1) \leq 2500 d(j)^2 (d(j) + \log_2(q/\delta_2)) + d(j)$. Noting that $q \leq 2^t$ and $1/\delta_2 \leq t \leq 2^t$ we conclude that $\log_2(q/\delta_2) \leq 2t \leq 2d(j)$. This implies that

$$d(j+1) \leq 2500 d(j)^2 (3d(j)) + d(j) \leq (20d(j))^3.$$

Using the induction hypothesis for $i = j$, we further conclude that

$$d(j+1) \leq (20 \cdot 20^{1.5(3^{j-1}-1)} \cdot t^{3^{j-1}})^3 \leq 20^{1.5(3^j-1)} \cdot t^{3^j}.$$

Next, we use Lemma 3.12 to bound $m(j+1)$. We have,

$$m(j+1) \leq m(j) + 25(d(j) + \log_2(q/\delta_2)) \leq m(j) + 75d(j) \leq t^{3^j} + 75(100t)^{3^{j-1}} \leq t^{3^{j+1}}$$

where the last inequality holds for t sufficiently large. ∎ Lemma 3.15

C. Missing Details from Section 4

Proof of Fact 4.2. Let $\mathbf{x}$ be a point in $\textsc{Conv}(T)$. By definition, $\mathbf{x} = \sum_{i=1}^{m} \alpha_i u_i$ where $\alpha_i \in [0,1]$ for $1 \leq i \leq m$ and $\sum_{i=1}^{m} \alpha_i = 1$. We have by the Cauchy-Schwarz inequality that,

$$\|\mathbf{x}\|_2^2 = \sum_{1 \leq i,j \leq m} \alpha_i \alpha_j \langle \mathbf{u}_i, \mathbf{u}_j \rangle \leq \sum_{1 \leq i,j \leq m} \alpha_i \alpha_j \|\mathbf{u}_i\|_2 \|\mathbf{u}_j\|_2 = \sum_{1 \leq i,j \leq m} \alpha_i \alpha_j = \left(\sum_{i=1}^{m} \alpha_i \right)^2 = 1.$$

Since the $\mathbf{u}_i$ are distinct, the equality above holds only if $\alpha_i \alpha_j = 0$ whenever $i \neq j$. This is only possible if there exists k such that $\alpha_k = 1$ and $\alpha_i = 0$ for $i \neq k$. It therefore follows that if $\mathbf{x} \in \textsc{Conv}(T)$ and $\|\mathbf{x}\|_2 = 1$ then $\mathbf{x} \in T$. ∎ Fact 4.2