Bandits, Query Learning, and the Haystack Dimension

Kareem Amin **Michael Kearns** **Umar Syed**
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{akareem,mkearns,usyed}@cis.upenn.edu

Abstract

Motivated by multi-armed bandits (MAB) problems with a very large or even infinite number of arms, we consider the problem of finding a maximum of an unknown target function by querying the function at chosen inputs (or arms). We give an analysis of the query complexity of this problem, under the assumption that the payoff of each arm is given by a function belonging to a known, finite, but otherwise arbitrary function class. Our analysis centers on a new notion of function class complexity that we call the *haystack dimension*, which is used to prove the approximate optimality of a simple greedy algorithm. This algorithm is then used as a subroutine in a functional MAB algorithm, yielding provably near-optimal regret. We provide a generalization to the infinite cardinality setting, and comment on how our analysis is connected to, and improves upon, existing results for query learning and generalized binary search.

1 Introduction

A *multi-armed bandit* (MAB) problem proceeds over several rounds, and in each round a decision-maker chooses an action, or *arm*, and receives a random payoff from an unknown distribution associated with the chosen action. MAB problems have been a focus of intensive study in the statistics and machine learning literature because they are an excellent model of many real-world sequential decision making problems that contain an "exploration vs. exploitation" trade-off, such as problems in clinical trials, sponsored web search, quantitative finance, and many other areas. The performance of a MAB algorithm is measured by its *regret*, which is the difference in expected total payoff received by the algorithm and by a highest-payoff action.

The classical formulation of the MAB problem assumes that the set of arms or actions is finite, and regret guarantees typically depend linearly on the number of actions. But many interesting applications have an extremely large, or even infinite, number of actions. As just one example, in (organic or sponsored) web search, a core goal is to select web pages in response to user queries in order to maximize click-throughs; even for a fixed query, the number of possible response web sites may be sufficiently large as to be effectively infinite, thus requiring some notion of similarity or generalization across actions.

Achieving regret that is sublinear in the number of actions clearly requires assumptions. A natural way to make the problem feasible is to make some specific functional assumptions about the action payoffs, i.e., to assume that the expected payoff of each action $x \in \mathcal{X}$ is given by $f^*(x)$, where $f^* \in \mathcal{F}$ is an unknown function belonging to a function class $\mathcal{F}$. One approach along these lines was pioneered by Kleinberg et al. (2008), who assumed that the set of actions $\mathcal{X}$ is endowed with a metric, and that each $f \in \mathcal{F}$ is Lipschitz continuous with respect to this metric. In this way, the observed payoff of any action provides information about the payoffs of "nearby" actions (with respect to the metric). Kleinberg et al. (2008) described efficient algorithms that exploit the structure of $\mathcal{F}$ to achieve no-regret. Nearly all existing algorithms for functional MAB problems rely on some kind of smoothness assumption (Bubeck et al., 2008, Lu et al., 2010, Slivkins, 2011, Flaxman et al., 2005, Dani and Hayes., 2006, Abernethy et al., 2008, Srinivas et al., 2008).

In this paper we consider the MAB problem for a *general* function class $\mathcal{F}$, and focus on the number of rounds required to achieve low regret (ignoring computational efficiency). We give a characterization in terms of a new measure of the complexity of the function class $\mathcal{F}$ that we call the

haystack dimension, which intuitively captures the extent to which maximizing a function via queries requires a search for a small number of items (needles) amongst a much larger number of otherwise undifferentiated possibilities (a haystack). We then give upper and lower bounds involving the haystack dimension of $\mathcal{F}$ that are within a $\log |\mathcal{F}|$ factor. Note that for the hardest MAB problems — where the haystack dimension can be as large as $|\mathcal{F}|$ — this logarithmic factor is relatively benign. Our main results are graphically summarized in Figure 1.

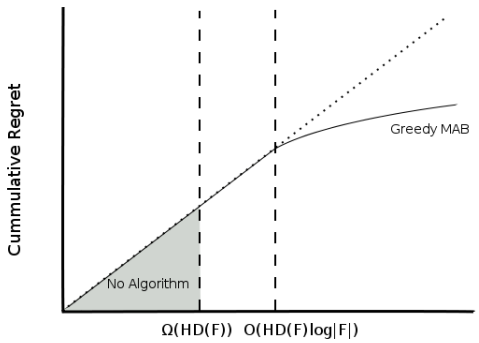

Figure 1: A graphical summary of the main results of this paper. This figure illustrates that *any* MAB algorithm for a function class $\mathcal{F}$ must suffer linear regret for a number of rounds on the order of the haystack dimension (denoted $\mathrm{HD}(\mathcal{F})$), while the MAB algorithm presented in this paper, called Greedy MAB, begins to suffer sublinear regret after roughly $\mathrm{HD}(\mathcal{F}) \log |\mathcal{F}|$ rounds.

An interesting aspect of our methods is the connection drawn between MAB problems and the problem of exact learning of functions from queries. We observe that functional MAB problems implicitly embed the problem of finding a maximum of an unknown function in $\mathcal{F}$ from only input-output queries (generalizations of membership queries), which may or may not be much easier than exact learning. Our analysis shows that any functional MAB algorithm must implicitly be willing to trade off between two distinct types of queries: *max queries* (which attempt to directly guess the maximum of f^*) and *information queries* (which attempt to make progress by reducing the version space, as is traditional in many query learning models). We show that either one of these query types (and essentially no others) can result in progress towards finding the maximum. The haystack dimension can then be viewed as a measure of the extent to which progress can be made at any step via either one of these query types.

Our characterization holds for any finite-cardinality $\mathcal{F}$ (though even the number of *actions* may still be infinite), but we also describe a generalization to the case of infinite $\mathcal{F}$ via covering techniques (which in general does not provide as tight bounds as its finite-cardinality specialization). We also stress that our results only apply to query complexity and regret; we make no claims about computational efficiency (necessarily, due to the generality of our setting). In this sense, the haystack dimension can be seen as playing a role in the study of functional MAB problems analogous to that played by quantities such as VC dimension and teaching dimension in other learning models, which also characterize sample or informational complexity, but not computational complexity. In separate work (Amin et al., 2011), we have developed computationally efficient algorithms for functional *contextual* MAB problems (Langford and Zhang, 2007), in which the payoff function depends on both the chosen action and the current context, both of which may be drawn from very large spaces.

2 Related Work

Many authors (at least since Thompson (1933)) have studied finite MAB problems where the action payoffs are assumed to be correlated; see Mersereau et al. (2009, p. 4) for an excellent survey. As explained in Section 1, more recent work has focused on infinite MAB problems where the action payoffs are related via an unknown function belonging to a known function class, such as the set of all Lipschitz continuous functions (Kleinberg et al., 2008, Bubeck et al., 2008, Slivkins, 2011). Compared to previous work, our results provide a complete analysis for significantly more general function classes.

Obviously, maximizing an unknown function via queries is no harder than exactly learning the function. Hegedüs (1995) characterized the query complexity of exact learning in terms of the *extended teaching dimension* of the function class. For some restricted function classes the haystack dimension and extended teaching dimension coincide[1], and in these cases our analysis approximately

[1]Essentially just those classes for which maximizing an unknown function is as difficult as exactly learning

recovers the bounds due to Hegedüs (1995), but with a significant advantage: our lower bound holds for all *randomized* algorithms, while the earlier bound only applied to deterministic algorithms.

A variant of exact learning (but not maximization) of functions has been considered under the name of *generalized binary search*. Nowak (2009) provided an analysis that only applies under a certain technical condition. In the language of this paper, the condition implies that the haystack dimension is a constant independent of the structure of the function class. In contrast, our analysis applies to any $\mathcal{F}$ and considers the maximization problem and its relationship to MAB directly.

3 Functional Bandits (MAB) and Maximizing From Queries (MAX)

A *functional MAB problem* is defined by a set of *actions* $\mathcal{X}$ and a set of possible *payoff functions* $\mathcal{F}$. A target payoff function $f^* \in \mathcal{F}$ is selected. In each round $t = 1, 2, \ldots$, an algorithm selects an action x_t, and receives an independent payoff from a distribution whose support is contained in $[-b, b]$, and has mean $f^*(x_t)$. The goal of the algorithm is to receive nearly as much cumulative payoff as an algorithm that selects the best action every round. More precisely, the worst-case expected *regret* of algorithm A in round T is $R_A(T) \triangleq \sup_{f^* \in \mathcal{F}} E\left[T \cdot \sup_{x \in \mathcal{X}} f^*(x) - \sum_{t=1}^{T} f^*(x_t)\right]$, where the expectation is with respect to the random payoffs and any internal randomization of the algorithm. We say that algorithm A is *no-regret* if $\lim_{T\to\infty} R_A(T)/T = 0$.

For any functional MAB problem, we can define a corresponding and (as we shall see) closely related functional *maximizing from queries* (or MAX) problem: In each round $t = 1, 2, \ldots$ an algorithm A selects a *query* $x_t \in \mathcal{X}$, and then observes $y_t = f^*(x_t)$. Letting X^f be the set of maxima of the function f (assumed to be non-empty), the goal of the algorithm is to eventually select a $x \in X^{f^*}$. Let $T^{A,f^*} = \min\{t : x_t \in X^{f^*}\}$ be the first round such that x_t is a maximum of f^*. We are interested in bounding the worst-case expected *query complexity* $Q_A \triangleq \sup_{f^* \in \mathcal{F}} E[T^{A,f^*}]$, where the expectation is with respect to any internal randomization of the algorithm.

While the definition of query complexity says that algorithm A will *select* a query in X^{f^*} within Q_A rounds, it does not require that the algorithm be able to *identify* this query. However, if A is deterministic and an upper bound B on Q_A is known, then the latter problem easily reduces to the former: If $Q_A \leq B$ then $x_{t^*} \in X^{f^*}$, where $t^* \in \arg\max_{1 \leq t \leq B}\{x_t\}$. We will describe a deterministic algorithm with near-optimal query complexity in Section 6.

It is important to note that, in a functional MAB problem, in each round t the algorithm only observes a *sample* from a distribution with mean $f^*(x_t)$, while in a functional MAX problem, the algorithm observes $f^*(x_t)$ *directly*. In Sections 4–7, we characterize the query complexity of the MAX problem for $\mathcal{F}$, and then apply these results in Sections 9–10 to characterize the optimal regret for the corresponding MAB problem for $\mathcal{F}$. Consequently, the analysis in Sections 4–7 will not deal with stochasticity, which is addressed afterwards. Also, we refer to elements of $\mathcal{X}$ as *actions* in the MAB context, but as *queries* in the MAX context — this difference exists only to agree with historical usage.

4 The Haystack Dimension

In this section we give the definition of the haystack dimension for function classes $\mathcal{F}$ of *finite cardinality*; generalization to the infinite case is given later.

The formal definition of the haystack definition requires some notation and machinery, but the intuition behind it is rather simple, so we first describe it informally. In words, the haystack dimension identifies the "worst" subset of $\mathcal{F}$, in the sense that on that subset, no matter what query is made and no matter what response is received, only a small fraction θ of the functions in the subset are eliminated due to inconsistency with the query, or are maximized by the query. It turns out mathematically that the right definition of the haystack dimension is the inverse quantity $1/\theta$ for this worst subset. We now proceed with the formal definition.

In the context of a MAX problem, a query $x \in \mathcal{X}$ can be thought of as providing information about the identity of f^*. In particular, f^* cannot be any of the functions in $\mathcal{F}$ inconsistent with with the value $f^*(x)$ observed at x. So one strategy for finding an element of X^{f^*} is to first issue a sequence of *information queries*, that uniquely identify f^*, and then select any $x \in X^{f^*}$.[2]

However, sometimes identifying the true function f^* exactly requires many more queries than necessary for maximization. For example, if many functions $f \in \mathcal{F}$ are maximized by one particular query, it may be useful to play such a *max query*, even if it is not particularly useful for learning f^*.

it; see Example 2.

[2]In the case of boolean functions or concepts, identifying f^* exactly is the problem of *learning from membership queries* (Angluin, 1988).

In the extreme case, there might exist an $x^* \in \mathcal{X}$ such that $x^* \in X^f$ for *all* $f \in \mathcal{F}$. In this case, an element of X^{f^*} can be selected in one query, without ever needing to identify f^*. On the other hand, there are also $\mathcal{F}$ for which exact learning is the fastest route to maximization. [3]

Any general algorithm for maximization from queries thus needs to be implicitly able to consider queries that would eliminate many candidate functions, as well as queries that might be an actual maximum.

Before continuing, we define some convenient notation that we will use throughout the rest of the paper. For any set $F \subseteq \mathcal{F}$, define the *inconsistent set* $F(\langle x, y \rangle) \triangleq \{f \in F : f(x) \neq y\}$ to be the functions in F that are inconsistent with the query-value pair $\langle x, y \rangle$. Also, for any set $F \subseteq \mathcal{F}$, define the *maximum set* $F(x) \triangleq \{f \in F : x \in X^f\}$ to be the functions in F for which x is a maximum.

Intuitively, the *haystack dimension* $\mathrm{HD}(\mathcal{F})$ of a function class $\mathcal{F}$ will characterize a subset of $\mathcal{F}$ on which no query is effective in the two senses previously discussed. For a subset $F \subseteq \mathcal{F}$, let:

$$\rho(F, x) = \inf_{y \in \mathbb{R}} \frac{|F(x) \cup F(\langle x, y \rangle)|}{|F|} \quad \text{and} \quad \rho(F) = \sup_{x \in \mathcal{X}} \rho(F, x).$$

$\rho(F, x)$ is the fraction of functions in F, which are guaranteed to be maximized, or deemed inconsistent, by the query x, for the worst-case possibility for y. If $\rho(F)$ is small, no query is guaranteed to be effective as either a max or information query on the subset F. Now let $F_\theta = \arg\inf_{F \subseteq \mathcal{F}} \rho(F)$ and $\theta = \rho(F_\theta)$.

Definition 1. *Let F_θ and θ be defined as above. Then the haystack dimension $\mathrm{HD}(\mathcal{F})$ of $\mathcal{F}$ is defined as $\frac{1}{\theta}$.*

Note that the haystack dimension can be as small as 1 (all functions share a common maximum-output input) and as large as $|\mathcal{F}|$ (every query eliminates at most one function in the senses discussed, the canonical "needle in a haystack").

5 Examples of the Haystack Dimension

In this section, we provide a few function classes which help illustrate how the haystack dimension characterizes the difficulty of maximizing an unknown $f^* \in \mathcal{F}$. Many of these examples will be useful for subsequent constructions in the paper.

The first construction considered is the "needle in a haystack". Fix a finite $\mathcal{X}$. For each $x \in \mathcal{X}$, let f_x be the function defined to have $f_x(x) = 1$ and $f_x(x') = 0$ for all $x' \neq x$. Now let $\mathcal{H}_\mathcal{X} = \{f_x \mid x \in \mathcal{X}\}$.

Example 1. $HD(\mathcal{H}_\mathcal{X}) = |\mathcal{H}_\mathcal{X}|$

Proof. For any $x \in \mathcal{X}$, f_x is the only function in $\mathcal{H}_\mathcal{X}$ that attains its max at x. Furthermore, all other functions output a 0 on input x. Therefore, letting $F = \mathcal{H}_\mathcal{X}$, $F(x) = \{f_x\}$ and $F(\langle x, 0 \rangle) = \{f_x\}$ for any $x \in \mathcal{X}$. This implies that $\rho(\mathcal{H}_\mathcal{X}) = |\mathcal{H}_\mathcal{X}|^{-1}$. Since $\rho(F)$ cannot be smaller than this quantity for any $F \subseteq \mathcal{F}$, the haystack dimension of $\mathcal{H}_\mathcal{X}$ indeed equals $|\mathcal{H}_\mathcal{X}|$. □

When $f^* \in \mathcal{H}_\mathcal{X}$, maximizing and learning f^* coincide, and both amount to guessing the x for which $f^*(x) = 1$. We now describe a function class $\mathcal{G}$ in which any algorithm is essentially forced to learn the true function f^*.

The input space $\mathcal{X}$ will consist of two components — $\mathcal{X}_1$ and $\mathcal{X}_2$, with $\mathcal{X}$ being the union of these disjoint domains. The high-level idea is to "marry" a small shattered set (in the sense of VC dimension) to a much larger haystack construction. Subdomain $\mathcal{X}_1$ consists of n points $\{a_0, ..., a_{n-1}\}$. We construct $\mathcal{G}$ as follows. On $\mathcal{X}_1$, all possible binary labelings of the n points appear in $\mathcal{G}$, giving a total of 2^n functions. Let us think of each function in $\mathcal{G}$ as being equated with the integer given by its binary labeling of the points in $\mathcal{X}_1$. So a function f is equated with the integer $z(f) = \sum_{i=0}^{n-1} 2^i f(a_i)$. Now let the much larger set $\mathcal{X}_2 = \{0, \ldots, 2^n - 1\}$, and for any $x \in \mathcal{X}_2$ define $f(x) = 2$ if $x = z(f)$ and $f(x) = 0$ otherwise.

Thus, the behavior of a function on $\mathcal{X}_1$ entirely defines the function on $\mathcal{X}_2$ as well, and the labeling on $\mathcal{X}_1$ gives us the index of the function's maximum, which is always equal to 2 and occurs at exactly one point in $\mathcal{X}_2$ (determined by the index).

Note that there is an algorithm that finds the max of f^* in $O(n)$ queries simply by querying every input in $\mathcal{X}_1$ and learning the identity of f^* exactly. Intuitively, no algorithm can do much better. To see why, suppose f^* were drawn uniformly at random from $\mathcal{G}$. Note that an action $r \in \mathcal{X}_2$

[3] See Example 2.

has an exponentially small probability of being the function's max and, in the event that $f^*(r) = 0$, only serves to inform the algorithm that f^* is not the single $f \in \mathcal{G}$ with $z(f) = r$. Also, observe that if the "zooming" algorithm of Kleinberg et al. (2008) is applied to $\mathcal{G}$, it will take exponential time in the worst-case to find a maximum of f^*, essentially because it makes no attempt to exploit the special structure of $\mathcal{G}$.

Example 2. $HD(\mathcal{G}) = \Theta(n) = \Theta(\log |\mathcal{G}|)$

Proof. We first show that there is an $F \subseteq \mathcal{G}$ with $\rho(F) = \frac{1}{n}$. For each $x \in \mathcal{X}_1$, let f_x be the function which outputs $f(x) = 1$, and $f(x') = 0$ for all $x' \in \mathcal{X}_1$, $x \neq x'$. Let $F = \{f_x \in \mathcal{G} \mid x \in \mathcal{X}_1\} = \{f \mid z(f) \in \{2^0, 2^1, ..., 2^{n-1}\}\}$. $|F| = n$.

Consider any query $x \in \mathcal{X}_1$. $F(x) = \emptyset$, since no functions achieve their maximum on a query in $\mathcal{X}_1$. Furthermore, $F(\langle x, 0\rangle) = \{f_x\}$, since f_x is the only function in F which doesn't output a 0 on query x. Thus $\rho(F, x) = \frac{1}{n}$ for any $x \in \mathcal{X}_1$. For a query $r \in \mathcal{X}_2$, $\rho(F, r) \leq \frac{1}{n}$, since at most one function in F (and $\mathcal{G}$) achieves its maximum at r, and all other functions output a zero at r. Thus, $\rho(F) = \frac{1}{n}$.

We now argue that for an arbitrary $F \subseteq \mathcal{G}$, $\rho(F) \geq \frac{1}{2n}$. Let $r_1(x) = |f \in F : f(x) = 1|/|F|$, be the fraction of functions that exhibit a 1 at action $x \in \mathcal{X}_1$. Let $r_0(x) = 1 - r_1(x)$. Suppose there is an action $x \in \mathcal{X}_1$ such that $r_1(x) \geq \frac{1}{2n}$ and $r_0(x) \geq \frac{1}{2n}$. Then, at least $\frac{1}{2n}$ of the functions in F would be inconsistent with any observed output. That is, both $\frac{|F(\langle x,0\rangle)|}{|F|} \geq \frac{1}{2n}$ and $\frac{|F(\langle x,1\rangle)|}{|F|} \geq \frac{1}{2n}$.

Otherwise, for every x in $\mathcal{X}_1$, either $r_1(x) < \frac{1}{2n}$ or $r_0(x) < \frac{1}{2n}$ (i.e. one outcome is quite rare). This implies that more than $1/2$ of the functions in F exhibit the same behavior on all x in $\mathcal{X}_1$. However, unless F is a singleton set (in which case $\rho(F) = 1$), this cannot occur since each $f \in \mathcal{G}$ exhibits unique behavior on $\mathcal{X}_1$. □

The preceeding example illustrates a function class for which any algorithm querying for the max must ultimately learn the true function f^*. However, the opposite extreme is also possible. Consider a function class $\mathcal{G}_{\max}$. Let there be a distinguished $x^* \in \mathcal{X}$ such that every $f \in \mathcal{G}_{\max}$ attains its maximum at x^*. It may be arbitrarily difficult to learn the behavior of f^* on the remainder of $\mathcal{X}$. However, finding the maximum can be done trivially in a single query.

Example 3. $HD(\mathcal{G}_{\max}) = 1$

Proof. For every $F \subseteq \mathcal{G}_{\max}$ $F(x^*) = F$, and the proof is immediate. □

Finally, there are function classes which require a hybrid between learning and maximization. We construct such a class, $\mathcal{G}^+$. Let $\mathcal{X}$ be the disjoint union of three sets $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3$. We let $|\mathcal{X}_1| = n$, $|\mathcal{X}_2| = m$ and create a function in $\mathcal{G}^+$ for each binary labeling of $\mathcal{X}_1$ and $\mathcal{X}_2$, as in the construction of $\mathcal{G}$. We let $z_1(f)$ be the integer corresponding to the labeling f gives X_1 and $z_2(f)$ be the integer corresponding to the labeling f gives X_2.

Now let $\mathcal{X}_3 = M_0 \cup M_1 \cup ... \cup M_{2^n-1}$ where each $|M_i| = c$. Again, like in the construction of $\mathcal{G}$, for each $f \in \mathcal{G}^+$, there will be a single $r \in \mathcal{X}_3$ such that $f(r) = 2$ and $f(r) = 0$ otherwise. However, this time, the maximizing element of f will be found in $M_{z_1(f)}$. And the particular element of $M_{z_1(f)}$ will be given by $z_2(f) \mod c$.

If we think of $c < n < m$, then there is an $n + c$ algorithm for finding the max of $\mathcal{F}$. The algorithm learns the set M_i which contains the max of f^* by querying each element of $\mathcal{X}_1$. It then tries each of the c elements in M_i. Note that the true identity of f^* is never learned, and the "interesting" learning problem is discovering the set M_i containing the maximizing action (i.e., learning the behavior on $\mathcal{X}_1$).

Example 4. *If* $c < n < m$, $HD(\mathcal{G}^+) = \Theta(n)$

Proof. We sketch the proof which proceeds almost identically to that of Example 2. There is an F with that witnesses $\rho(F) = \frac{1}{n}$. F is identical to that used in Example 2 on $\mathcal{X}_1$. The behavior on $\mathcal{X}_2$ is identical across all functions, and the behavior on $\mathcal{X}_3$ is determined by these choices.

To see that for any F, $\rho(F) \geq \frac{1}{2n}$, we use the same reasoning as Example 2. However, if for every x in $\mathcal{X}_1$, either $r_1(x) < \frac{1}{2n}$ or $r_0(x) < \frac{1}{2n}$, rather than implying a contradiction, this implies that there is actually a query $x^* \in \mathcal{X}_3$ such that x^* maximizes more than a $\frac{1}{2c}$ fraction of the functions in F. Therefore for that particular F, $\rho(F) > \frac{1}{2c} > \frac{1}{2n}$, as desired. □

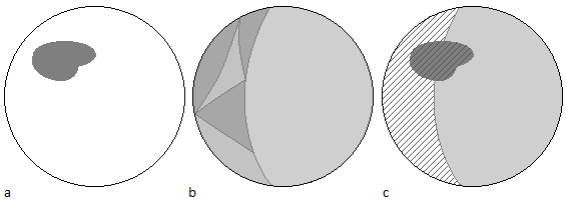

Figure 2: (a) The shaded region represents the subset of functions which attain a maximum at some input x. (b) Querying x also induces a partition of the function space, in which each piece of the partition contains functions that output the same value on x. The least informative query response is thus the largest piece of this partition. (c) Upon querying x, the greedy algorithm eliminates from its attention space a set *at least* the size of the striped region. This region is the union of the maximizing set and the *complement* of the largest partition piece.

6 MAX Query Complexity Upper Bound

Before arguing that the haystack dimension gives lower bounds on the query complexity of any algorithm on $\mathcal{F}$, we observe that it motivates a simple, natural, greedy algorithm. In each round t, the *greedy algorithm* G selects $x_t = \arg\sup_{x \in \mathcal{X}} \rho(F_t, x)$ where the *attention space* F_t is defined inductively as follows: $F_1 = \mathcal{F}$ and $F_{t+1} = F_t \setminus (F_t(x_t) \cup F_t(\langle x_t, y_t \rangle))$. (Note that for fixed F_t, $\rho(F_t, x)$ takes values in $\{1/|F_t|, 2/|F_t|, ..., 1\}$, so the supremum is achieved by an $x \in \mathcal{X}$.)

Essentially, the greedy algorithm always selects the query that results in the largest guaranteed reduction of the attention space [4]; see Figure 2. It is important to note that the attention space differs from the version space of traditional learning in that it *excludes functions that may still be consistent with the observed data*, but for which the algorithm has already played a query which would have maximized such functions. Indeed, in the extreme case the algorithm may choose to query x such that *all functions remaining* have the same output on x — in which case the query conveys no "information" in the traditional learning sense, but nevertheless functions attaining their maximum at x will be discarded.

Theorem 1. *Let G be the greedy algorithm for the MAX problem for $\mathcal{F}$. Then $Q_G \leq \mathrm{HD}(\mathcal{F}) \log |\mathcal{F}|$.*

Proof. We know that the attention space F_t is nonempty for all rounds $t \leq Q_G$ (otherwise the greedy algorithm would have selected a maximum of f^* before round Q_G). And at least a θ fraction of the attention space is removed in each round t, because $\theta = \rho(F_\theta) \leq \rho(F_t) = \rho(F_t, x_t) \leq \frac{|F_t(x_t) \cup F_t(\langle x_t, y_t \rangle)|}{|F_t|}$ by the definition of θ, F_θ, and x_t. Thus we have $(1-\theta)^{Q_G}|\mathcal{F}| \geq 1$. Taking the logarithm of both sides of this inequality, applying $\log(1-x) \leq -x$ for $x < 1$, rearranging, and noting that $\mathrm{HD}(\mathcal{F}) \triangleq \frac{1}{\theta}$ by definition implies the theorem. □

7 MAX Query Complexity Lower Bound

In this section we prove that that haystack dimension in fact provides a lower bound on the query complexity of *any* algorithm finding a maximum of $f^* \in \mathcal{F}$:

Theorem 2. *Let A be any algorithm for the MAX problem for $\mathcal{F}$. Then $Q_A \geq \frac{1}{6} \mathrm{HD}(\mathcal{F})$.*

The proof of Theorem 2 is somewhat involved and developed in a series of lemmas, so we shall first sketch the broad intuition. The lower bound will draw the target f^* uniformly at random from F_θ, where F_θ is as in the definition of haystack dimension, and is the subset of functions on which the greedy algorithm guarantees the least amount of progress (in terms of reducing its attention space, as defined above). By definition no other algorithm A can make more progress than θ on its first query starting from F_θ (since this is the maximum possible, and obtained by greedy). After the first step, the greedy algorithm and algorithm A may have different attention spaces, and thus on subsequent steps A may make greater progress than greedy; but A cannot make "too much" progress on (say) its second step, since otherwise its query there would have made more than θ progress on F_θ. This insight leads to a formal recurrence inequality governing the progress rate of A, whose solution, when combined with a Bayesian argument, leads to a lower bound that is $\Omega(\frac{1}{\theta}) \triangleq \Omega(\mathrm{HD}(\mathcal{F}))$. We now proceed with the formal development.

[4] Note that we assume the specified computations can in fact be implemented in finite computation time.

Suppose we have an algorithm A that, given access to query-value pairs from f^*, generates a sequence of queries $\{x_t\}$ (possibly depending on its random bits). Let $y_t = f^*(x_t)$. Let $H_1 \triangleq F_\theta$ and

$$y_t^-(x) \triangleq \arg\inf_{y\in\mathbb{R}} |H_t(x) \cup H_t(\langle x, y\rangle)| \qquad S_t(x) \triangleq H_t(x) \cup H_t(\langle x, y_t^-(x)\rangle)$$
$$H_{t+1} \triangleq H_t \setminus S_t(x_t) \qquad x_t^* \triangleq \arg\sup_{x\in\mathcal{X}} |S_t(x)| \qquad \delta_t \triangleq \frac{|S_t(x_t^*)|}{|H_t|}$$

These definitions are closely related to those for the greedy algorithm and the haystack dimension. $y_t^-(x)$ is the "least helpful" possible output value on query x, H_t is the attention space of algorithm A when starting from F_θ if only least helpful outputs were returned, $S_t(x)$ is the set of functions in H_t that either attain a maxima at x or would have behavior inconsistent with the observation $y_t^-(x)$ on input x, and δ_t is the progress (fractional reduction of the attention space) made by the greedy algorithm.

Our first lemmas, which codify the aforementioned Bayesian argument, show that when f^* is drawn uniformly from F_θ, the probability a deterministic algorithm (a restriction removed shortly) finds a maximum in fewer than T steps is bounded by the sum of the δ_t.

Lemma 1. *Fix a sequence $\{x_t\}$. Let $B_t = \{f \in F_\theta : x_s \notin X^f \wedge y_s^-(x_s) = f(x_s)\ \forall s < t\}$. Then $H_t = B_t$ for all t.*

Proof. When $t = 1$, $B_1 = \{f \in F_\theta\}$ and the claim is immediate. Otherwise, assume that $H_t = B_t$ for some $t \geq 1$. We have:

$$\begin{aligned} B_{t+1} &= B_t \cap \{f \in F_\theta : x_t \notin X^f \wedge y_t^-(x_t) = f(x_t)\} \\ &= B_t \setminus \{f \in B_t : x_t \in X^f \vee y_t^-(x_t) \neq f(x_t)\} = H_t \setminus S_t(x_t) = H_{t+1} \end{aligned}$$ □

Lemma 2. $\Pr_{f^*\sim U_{F_\theta}}\left[T^{A,f^*} < T\right] \leq \sum_{t=1}^{T} \delta_t$ *for any deterministic algorithm A and constant T.*

Proof. Since A is deterministic, the sequence $\{x_t\}$ is determined by the choice of $f^* \in \mathcal{F}$. By Lemma 1 we have $H_t = \{f \in F_\theta : x_s \notin X^f \wedge y_s^-(x_s) = f(x_s)\ \forall s < t\}$ for all t. Moreover, since $y_s^-(x_s) = f^*(x_s)$ for all $s < t$ and $f^* \in H_t$ and algorithm A is deterministic, the sequence $\{x_s\}_{s\leq t}$ is identical for any choice of $f^* \in H_t$. Let U_F denote the uniform distribution over F. We have:

$$\begin{aligned} \Pr_{f^*\sim U_{F_\theta}}\left[T^{A,f^*} < T\right] &= \Pr_{f^*\sim U_{F_\theta}}\left[\exists t < T : x_t \in X^{f^*}\right] \leq \Pr_{f^*\sim U_{F_\theta}}\left[\exists t < T : x_t \in X^{f^*} \vee y_t^-(x_t) \neq f^*(x_t)\right] \\ &\leq \sum_{t=1}^{T} \Pr_{f^*\sim U_{F_\theta}}\left[x_t \in X^{f^*} \vee y_t^-(x_t) \neq f^*(x_t) \mid x_s \notin X^{f^*} \wedge y_s^-(x_s) = f^*(x_s)\ \forall s < t\right] \\ &= \sum_{t=1}^{T} \Pr_{f^*\sim U_{H_t}}\left[x_t \in X^{f^*} \vee y_t^-(x_t) \neq f^*(x_t)\right] \leq \sum_{t=1}^{T} \frac{|S_t(x_t)|}{|H_t|} \leq \sum_{t=1}^{T} \delta_t \end{aligned}$$ □

We thus see that one approach to lower bounding T^{A,f^*} is to bound the growth rate of the sequence $\{\delta_t\}$. Intuitively, we should expect that $(1-\delta_t)\delta_{t+1} \leq \delta_t$. To see why, recall that δ_t is the most progress that can be guaranteed by a single query if the function is drawn from the space H_t. This is the query that would be selected by the greedy algorithm if it were run on H_t. Suppose that it were instead that case that $(1-\delta_t)\delta_{t+1} > \delta_t$. By playing x_{t+1}^* on H_t at least $(1-\delta_t)\delta_{t+1}$ fraction of the functions in H_t would either attain a maximum point at x_{t+1}^* or be eliminated as inconsistent with the observed value $f^*(x_{t+1}^*)$. Thus the query x_t^*, which only guaranteed that a δ_t fraction of the functions in H_t have this property, was suboptimal, a contradiction . More formally:

Lemma 3. $(1-\delta_t)\delta_{t+1} \leq \delta_t$ *for all t.*

Proof.

$$\begin{aligned} (1-\delta_t)\delta_{t+1} &= \left(1 - \frac{|S_t(x_t^*)|}{|H_t|}\right)\frac{|S_{t+1}(x_{t+1}^*)|}{|H_{t+1}|} = \left(\frac{|H_t| - |S_t(x_t^*)|}{|H_t|}\right)\frac{|S_{t+1}(x_{t+1}^*)|}{|H_t| - |S_t(x_t)|} \\ &\leq \frac{|S_{t+1}(x_{t+1}^*)|}{|H_t|} \leq \frac{|S_t(x_{t+1}^*)|}{|H_t|} \leq \frac{|S_t(x_t^*)|}{|H_t|} = \delta_t \end{aligned}$$

Here the first inequality follows from the the definition of x_t^* as a maximizer of $|S_t(x)|$ (and thus $(|H_t| - |S_t(x_t^*)|)/(|H_t| - |S_t(x_t)|) \leq 1$). The second inequality follows because $|S_{t+1}(x)| \leq |S_t(x)|$ for all $x \in \mathcal{X}$, and the final inequality follows once again from the fact that x_t^* maximizes $|S_t(x)|$. □

We next establish that, roughly speaking, δ_t must remain $O(\delta_1)$ for $\Omega(1/\delta_1)$ steps. More precisely:

Lemma 4. *If $t < \frac{1}{\delta_1}$ then $\delta_t \leq \frac{\delta_1}{1-t\delta_1}$.*

Proof. The base case $t = 1$ clearly holds. Now suppose for induction that $\delta_t \leq \frac{\delta_1}{1-t\delta_1}$. We have

$$\begin{aligned}\delta_{t+1} &\leq \frac{\delta_t}{1-\delta_t} \leq \frac{\delta_1}{1-t\delta_1}\left(\frac{1}{1-\delta_t}\right) = \frac{\delta_1}{1-t\delta_1-\delta_t+t\delta_1\delta_t} \\ &= \frac{\delta_1}{1-(t+1)\delta_1+\delta_1-\delta_t+t\delta_1\delta_t} \leq \frac{\delta_1}{1-(t+1)\delta_1}\end{aligned}$$

The first inequality holds by Lemma 3, and the second inequality holds by the induction hypothesis. For the final inequality, note that $\delta_t \leq \frac{\delta_1}{1-t\delta_1}$ implies that $\delta_1 - \delta_t + t\delta_1\delta_t \geq 0$, as long as $t < \frac{1}{\delta_1}$. □

We are now ready to prove the lower bound of Theorem 2.

Proof. (Theorem 2) The behavior of algorithm A is partly determined by its internal randomization, which we denote as a random string ω drawn from a distribution $\mathcal{P}$. Let us write $A(\omega)$ for the deterministic algorithm corresponding to the string ω.

Fix a positive constant $c < 1/2$ (implying $\frac{c}{1-c} < 1$), with the exact value to be specified later. For any fixed ω

$$\Pr_{f^* \sim U_{F_\theta}}\left[T^{A(\omega),f^*} < \frac{c}{\theta}\right] \leq \sum_{t=1}^{c/\theta}\delta_t \leq \sum_{t=1}^{c/\theta}\frac{\delta_1}{1-t\delta_1} \leq \sum_{t=1}^{c/\theta}\frac{\theta}{1-c} = \frac{c}{1-c} \tag{1}$$

where we used, in order: Lemma 2; Lemma 4 and the fact that $\delta_1 = \theta$ and $c < 1$; the fact that $\delta_1 = \theta$ again; arithmetic. Now we have

$$E_{f^* \sim U_{F_\theta}}\left[T^{A(\omega),f^*}\right] \geq \left(1 - \Pr_{f^* \sim U_{F_\theta}}\left[T^{A(\omega),f^*} < \frac{c}{\theta}\right]\right)\frac{c}{\theta} \geq \left(1 - \frac{c}{1-c}\right)\frac{c}{\theta} \tag{2}$$

where the second inequality follows from (1). Finally, we have

$$\begin{aligned}E_{f^* \sim U_{F_\theta}}\left[T^{A,f^*}\right] &\triangleq E_{\omega \sim \mathcal{P}, f^* \sim U_{F_\theta}}\left[T^{A(\omega),f^*}\right] \\ &= E_{\omega \sim \mathcal{P}}\left[E_{f^* \sim U_{F_\theta}}\left[T^{A(\omega),f^*}|\omega\right]\right] \geq E_{\omega \sim \mathcal{P}}\left[\left(1 - \frac{c}{1-c}\right)\frac{c}{\theta}\right] = \left(1 - \frac{c}{1-c}\right)\frac{c}{\theta}\end{aligned}$$

where the inequality follows from (2). The choice of c implies $\left(1 - \frac{c}{1-c}\right)c > 0$. Letting $c = 2 - \sqrt{3}$ and recalling the definition $\mathrm{HD}(\mathcal{F}) \triangleq \frac{1}{\theta}$ implies the theorem. □

8 Relationship to VC Dimension and Extended Teaching Dimension

As we have demonstrated, the haystack dimension provides a lower bound on the query complexity of any algorithm for the MAX problem on a function class $\mathcal{F}$. This is a role analogous to the VC dimension in the PAC learning model. However, as we will demonstrate, the two are incomparable in general. We will also illustrate the haystack dimension's relationship to the *extended teaching dimension* (Hegedüs, 1995). The extended teaching dimension characterizes the number of queries required to learn $f^* \in \mathcal{F}$ when $\mathcal{F}$ consists of binary functions. Clearly learning f^* is sufficient for maximization and, as we will see, the haystack dimension can be much smaller than extended teaching dimension, but cannot be too much larger. Note that the VC and extended teaching dimensions are defined only for binary functions, whereas the haystack dimension and our results encompass a much more general setting.

For $\mathcal{F}$ consisting of binary functions, we will denote the VC dimension as $VCD(\mathcal{F})$, where the hypothesis class is assumed to equal to concept class. Similarly, we will denote the extended teaching dimension by $XTD(\mathcal{F})$. Both are redefined below.

Definition 2 (Kearns and Vazirani (1994))**.** *Let $\mathcal{F}$ be a function class (concept class) where $f : \mathcal{X} \to \{0,1\}$ for each $f \in \mathcal{F}$. We say a set $S = \{x_1, \ldots, x_m\} \subseteq \mathcal{X}$ is* shattered *by $\mathcal{F}$ if $\{(f(x_1), \ldots, f(x_m)) : f \in \mathcal{F}\} = \{0,1\}^m$. $VCD(\mathcal{F})$ is equal to the cardinality of the largest set shattered by $\mathcal{F}$.*

Definition 3 (Hegedüs (1995))**.** *Let $h : \mathcal{X} \to \{0,1\}$. We say that $S \subseteq \mathcal{X}$ is a* specifying set *for h if there is at most one $f \in \mathcal{F}$ consistent with h on all $x \in S$. Let $XTD(\mathcal{F}, h)$ be equal to the size of the smallest specifying set for h. $XTD(\mathcal{F}) = \sup_h XTD(\mathcal{F}, h)$.*

Example 5. *(a) For any d, there exists a function class and set of inputs $(\mathcal{F}, \mathcal{X})$ such that $VCD(\mathcal{F}) = d$ but $HD(\mathcal{F}) = 1$. (b) For any d, there exists a $(\mathcal{F}, \mathcal{X})$ such that $VCD(\mathcal{F}) = 2$ but $HD(\mathcal{F}) = d$.*

Proof. To prove (a), let $(\mathcal{F}_d, \mathcal{X}_d)$ be any function class, and set of inputs, such that $VCD(\mathcal{F}_d) = d$. Let $\mathcal{X} = \mathcal{X}_d \cup \{x^*\}$. Construct $(\mathcal{F}, \mathcal{X})$ by including a function f in $\mathcal{F}$ for each $f' \in \mathcal{F}_d$, that is identical to f' on all inputs in $\mathcal{X}_d$. Also let $f(x^*) = 1$. For any $F \subseteq \mathcal{F}$, x^* maximizes all functions in F. Therefore, $HD(\mathcal{F}) = 1$. However, no function in $\mathcal{F}$ gives x^* the label 0, and so the size of the largest shattered set does not change, and $VCD(\mathcal{F}) = d$.

To prove (b), consider $\mathcal{F}$ to be the "needle in the haystack" of Example 1, where $|\mathcal{F}| = d$. We have that $HD(\mathcal{F}) = d$. To compute the VC dimension, note that no function in $\mathcal{F}$ labels more than one input with a 1, and so no set of three inputs can be shattered by $\mathcal{F}$. (It's also obvious that any $\{x_1, x_2\} \subseteq \mathcal{X}$ can be shattered). □

Example 6. *(a) For any d, there exists a function class and set of inputs $(\mathcal{F}, \mathcal{X})$ such that $XTD(\mathcal{F}) = d$ but $HD(\mathcal{F}) = 1$. (b) For any $\mathcal{F}$ consisting of binary functions, $HD(\mathcal{F}) = O(XTD(\mathcal{F}) \log |\mathcal{F}|)$.*

Proof. To prove (a), let $(\mathcal{F}_d, \mathcal{X}_d)$ be a function class, and set of inputs, such that $XTD(\mathcal{F}_d) = d$. Construct $(\mathcal{F}, \mathcal{X})$ exactly as in the proof of Example 5(a). For any $h : \mathcal{X} \to \{0,1\}$ with $h(x^*) = 1$, let $h' : \mathcal{X} \to \{0,1\}$ be a function identical to h on $\mathcal{X}_d$. It's clear that $XTD(\mathcal{F}, h) = XTD(\mathcal{F}_d, h')$. For any $h : \mathcal{X} \to \{0,1\}$ with $h(x^*) = 0$, $XTD(\mathcal{F}, h) = 1$, since $\{x^*\}$ is a specifying set for h. Thus, $XTD(\mathcal{F}) = XTD(\mathcal{F}_d) = d$.

For (b), Hegedüs (1995) gives an algorithm which learns $f^* \in \mathcal{F}$ using $O(XTD(\mathcal{F}) \log |\mathcal{F}|)$ membership queries. Since learning f^* is sufficient for maximization, Theorem 2 implies (b). □

9 Functional MAB Regret Upper Bound

In this and the next section, we return to the functional MAB problem, showing a close relationship to the functional MAX problem, and giving upper and lower bounds on regret that are order the haystack dimension HD($\mathcal{F}$). We will describe a no-regret algorithm for functional MAB problems where $\mathcal{F}$ is known, finite, but otherwise arbitrary. Our approach is to use the greedy functional MAX algorithm of Section 6 to find an action/query that maximizes f^*, and then select that action indefinitely. There is a technical complication we must overcome when implementing this approach: Each action returns a sample from a distribution, rather than a single value. We solve this by repeatedly selecting an action x to get an accurate estimate of $f^*(x)$, and also by restarting the algorithm after progressively longer phases (a standard "trick" in MAB algorithms).

Before giving a more detailed description of our algorithm and its analysis, we need some additional definitions. Let $\epsilon_{\min} \triangleq \min_{f,f' \in \mathcal{F}} \inf_{x: f(x) \neq f'(x)} |f(x) - f'(x)|$ be the smallest difference between any two functions in $\mathcal{F}$ on those points where they do differ; $\epsilon_{\min} > 0$ allows us to determine $f^*(x)$ by selecting x enough times. Also, for any set $F \subseteq \mathcal{F}$, let us define the *ϵ-inconsistent set* to be $F(\langle x, y \rangle, \epsilon) \triangleq \{f \in F : |f(x) - y| > \epsilon\}$. Finally, recall that $f^*(x) \in [-b, b]$.

The *greedy bandit algorithm GB* proceeds in phases $i = 1, 2, \ldots$, where phase i lasts $T_i = 2^{2^i}$ rounds, and consists of two consecutive subphases, which we will denote by i_{explore} and i_{exploit}. In subphase i_{explore}, we run a fresh instance of the greedy query algorithm G from Section 6, with some minor modifications. When the query algorithm G selects a query x_t^i, the bandit algorithm GB selects that action for $L = \frac{32b^2}{\epsilon_{\min}^2} (\log \text{HD}(\mathcal{F}) + \log\log |\mathcal{F}|) \log T_i$ consecutive rounds, and lets $\bar{y}_t^i$ be the average of the observations. The attention space F_t^i is then updated according to the rule $F_{t+1}^i = F_t^i \setminus (F_t^i(x_t^i) \cup F_t^i(\langle x_t^i, \bar{y}_t^i \rangle, \epsilon_{\min}))$. Let τ_i be the number of queries required to empty the attention space in subphase i_{explore} (so subphase i_{explore} lasts $\tau_i L$ rounds). In subphase i_{exploit}, in each round the bandit algorithm GB selects the action x_t^i associated with the largest $\bar{y}_t^i$ in subphase i_{explore}.

Theorem 3. *Let GB be the greedy bandit algorithm for the MAB problem for $\mathcal{F}$. Then*

$$R_{GB}(T) = O\left(b^2 \frac{\text{HD}(\mathcal{F}) \log |\mathcal{F}|}{\epsilon_{\min}^2} \Big(\log \text{HD}(\mathcal{F}) + \log\log |\mathcal{F}| \Big) \log T + \log\log T \right)$$

Proof. Let R_i be the regret realized by the greedy bandit algorithm during phase i. We will first bound $E[R_i]$ for any phase i, and then use a 'squaring trick' to tightly bound $E[\sum_i R_i] \triangleq R_{GB}(T)$.

By a similar argument as in Theorem 1, for each phase i we have $\tau_i \leq \mathrm{HD}(\mathcal{F}) \log |\mathcal{F}|$. For each $t \in [\tau_i]$, action x_t^i is selected L times, which is a sufficient number of times to ensure (by the Chernoff and union bounds) that with probability $1 - O(\frac{1}{T_i})$ we have $|\bar{y}_t^i - f^*(x_t^i)| \leq \frac{\epsilon_{\min}}{2}$ for all $t \in [\tau_i]$. Let $\mathrm{good}(i)$ be this event. We have $E[R_i] = E[R_i|\,\mathrm{good}(i)]\Pr[\mathrm{good}(i)] + E[R_i|\neg\,\mathrm{good}(i)]\Pr[\neg\,\mathrm{good}(i)] \leq E[R_i|\,\mathrm{good}(i)] + O(1)$, because $R_i \leq T_i$ and $\Pr[\neg\,\mathrm{good}(i)]$ is $O(\frac{1}{T_i})$.

It therefore remains to bound $E[R_i|\,\mathrm{good}(i)]$. By the definition of $\epsilon_{\min}$, if the event $\mathrm{good}(i)$ occurs then the attention space F_t^i maintained during subphase i_{explore} is updated exactly the same way as in the greedy query algorithm G from Section 6. Therefore, an action $x^* \in X^{f^*}$ is selected in subphase i_{explore} before it ends (by Theorem 1), and thus, again by the definition of $\epsilon_{\min}$, the action x^* is selected in every round of subphase i_{exploit}. Since $\tau_i \leq \mathrm{HD}(\mathcal{F}) \log |\mathcal{F}|$ and subphase i_{explore} lasts $\tau_i L$ rounds, we have $E[R_i|\,\mathrm{good}(i)] = O\left(\frac{b^2\,\mathrm{HD}(\mathcal{F})\log|\mathcal{F}|}{\epsilon_{\min}^2}\left(\log \mathrm{HD}(\mathcal{F}) + \log\log|\mathcal{F}|\right)\log T_i\right)$.

Finally, we apply a 'squaring trick'.[5] Let K be the number of phases. Since $T_i = 2^{2^i}$, we have $K = O(\log\log T)$ and $\sum_{i=1}^{K} \log T_i = \sum_{i=1}^{O(\log\log T)} 2^i = O(\log T)$. Recalling that $R_{GB}(T) \triangleq E[\sum_i R_i]$, proves the theorem. □

Note that the greedy bandit algorithm GB assumes that the value of the haystack dimension $\mathrm{HD}(\mathcal{F})$ is known. It is possible to modify the algorithm in case this information is not available: Instead of selecting each action x_t^i in subphase i_{explore} for $L = \frac{32b^2}{\epsilon_{\min}^2}\left(\log \mathrm{HD}(\mathcal{F}) + \log\log|\mathcal{F}|\right)\log T_i$ consecutive rounds, we select it for $L = \frac{32b^2}{\epsilon_{\min}^2}\left(\log \mathcal{F} + \log\log|\mathcal{F}|\right)\log T_i$ consecutive rounds. Since $\mathrm{HD}(\mathcal{F}) \leq |\mathcal{F}|$ trivially holds, the analysis in the proof of Theorem 3 is essentially unaffected by this modification, and it adds only a $O((\log|\mathcal{F}|)^2)$ term to the regret upper bound in Theorem 3.

10 Functional MAB Regret Lower Bound

In this section, we prove that the greedy bandit algorithm in Section 9 is near-optimal, at least with respect to the haystack dimension (our primary interest) and terms $\log|\mathcal{F}|$ or smaller. With respect to the dependence on the haystack dimension, we can say something quite strong: Every MAB algorithm for every function class must suffer regret that is linear in the haystack dimension of the class. Let $\Delta = \inf_{f\in\mathcal{F}} \inf_{\{x'\in\mathcal{X}: f(x') < \sup_x f(x)\}} \sup_x f(x) - f(x)$, be the difference between the best action and second-best action in $\mathcal{X}$.

Theorem 4. *For any function class $\mathcal{F}$ and MAB algorithm A for $\mathcal{F}$, $R_A(T) = \Omega\left(\Delta \min\{T, \mathrm{HD}(\mathcal{F})\}\right)$.*

The proof of Theorem 4 follows directly from Theorem 2, which implies that no MAB algorithm for function class $\mathcal{F}$ can select the best action in fewer than $\Omega\left(\mathrm{HD}(\mathcal{F})\right)$ steps. The next theorem proves that the terms $\log|\mathcal{F}|$ and $\frac{1}{\epsilon_{\min}}$ cannot be removed from the upper bound in Theorem 3.

Theorem 5. *(a) There exists a function class $\mathcal{F}$ such that $\mathrm{HD}(\mathcal{F}) = \Theta(1)$ and for any MAB algorithm A for $\mathcal{F}$, $R_A(T) = \Omega\left(\log|\mathcal{F}|\right)$. (b) There exists a function class $\mathcal{F}$ such that $\mathrm{HD}(\mathcal{F}) = \Theta(1)$ and for any MAB algorithm A for $\mathcal{F}$, $R_A(T) = \Omega\left(\min\left\{\frac{1}{\epsilon_{\min}}, |\mathcal{F}|\right\}\right)$.*

Proof. To prove (a), we will outline the construction of $\mathcal{F}$ and omit the details of the proof, which are straightforward. The input space $\mathcal{X}$ will have two components — $\mathcal{X}_1$ and $\mathcal{X}_2$, with $\mathcal{X}$ being the union of these disjoint domains. Subdomain $\mathcal{X}_1$ consists of 2^n points, and $\mathcal{X}_2$ of n points, while the function class $\mathcal{F}$ contains n deterministic functions. Each point in $x \in \mathcal{X}_1$ corresponds to a distinct subset $F_x \subseteq \mathcal{F}$, and for each $x \in \mathcal{X}_1$ let $f(x) = \frac{1}{3}$ for half the functions in F_x and $f(x) = \frac{2}{3}$ for the other half, and also let $f(x) = \frac{1}{3}$ for all $f \in \mathcal{F} \setminus F_x$. Note that $\mathcal{X}_1$ contains excellent information queries, because for any subset $F \subseteq \mathcal{F}$ there is a point in $\mathcal{X}_1$ that eliminates at least half the functions in F when that point is issued as a query, and thus $\mathrm{HD}(\mathcal{F}) = 2$. Finally, map each function $f \in \mathcal{F}$ to a distinct point $x_f \in \mathcal{X}_2$, and let $f(x_f) = 1$ and $f(x) = 0$ for all $x \in \mathcal{X}_2 \setminus \{x_f\}$. So each $f \in \mathcal{F}$ has a unique maximum, contained in $\mathcal{X}_2$.

Suppose f^* is chosen uniformly at random from $\mathcal{F}$. It is clear that $Q_A = \Omega(\log|\mathcal{F}|)$. To see why, note that a query $x \in \mathcal{X}_2$ has only probability $\frac{1}{|\mathcal{F}|}$ of being the function's max, and only serves to

[5] If we were to instead apply the more common 'doubling trick', such that $T_i = 2^i$, the upper bound in Theorem 3 would be $O((\log T)^2)$.

inform the algorithm that $f^* \neq f$ whenever $x = x_f$ and $f^*(x) = 0$. Thus any algorithm is forced to learn f^* by playing actions in $\mathcal{X}_1$, requiring that at least $\log_2 |\mathcal{F}|$ actions are played.

To prove (b), we simply modify the construction of $\mathcal{F}$ to add stochasticity, as follows. For an $x \in \mathcal{X}_1$, if we had previously that $f(x) = \frac{1}{3}$, we now let $f(x) = \epsilon$. If we had previously that $f(x) = \frac{2}{3}$, we now let $f(x) = 2\epsilon$. Each function's behavior of $\mathcal{X}_2$ is unchanged. Note that $\epsilon_{\min} = \epsilon$. When playing an $x \in \mathcal{X}$, rather than observe the output of the function, the algorithm now observes the outcome a Bernoulli random variable with mean $f^*(x)$. It can then be shown that any algorithm that wishes to make use of the information in $\mathcal{X}_1$ must sample the actions there $\Omega(1/\epsilon)$ times or else be forced to play actions in $\mathcal{X}_2$. □

Theorem 5(b) establishes that there is at least one function class for which the haystack dimension does not provide an adequate lower bound in the MAB setting. In other words, there is a real gap between the upper and lower bounds in Theorems 3 and 4 that depends on $\frac{1}{\epsilon_{\min}}$. We suspect that a modified version of the haystack dimension which better accounts for the "information value" of a query under stochasticity (i.e. that uses not just the size of the largest inconsistent set, but accounts for the variance of the functions in $\mathcal{F}$) would close this gap, but leave this as an open problem.

11 Infinite Function Classes

In the remainder of the paper, we return to studying the MAX problem. We give extensions of our basic results on query complexity, and also give examples that illustrate some aspects of our analysis. In this section, we describe how to extend our methods from Sections 6 and 7, which were restricted to finite function classes, to infinite function classes that have a *finite covering oracle*.

Definition 4. *We say C is a* finite covering oracle *for $\mathcal{F}$ if for any $\epsilon > 0$ the finite set $C(\epsilon) \subseteq \mathcal{F}$ has the following property: For any $f \in \mathcal{F}$ there exists an $f' \in C(\epsilon)$ such that $\sup_{x \in \mathcal{X}} |f(x) - f'(x)| \leq \epsilon$.*

Fix a possibly infinite function class $\mathcal{F}$ with finite covering oracle C. We consider the ϵ-MAX problem for $\mathcal{F}$, a relaxed version of the MAX problem. In analogy to the MAX problem, for any algorithm A let $T^{A,f^*,\epsilon} \triangleq \min\{t : \sup_x f^*(x) - f^*(x_t) \leq \epsilon\}$ be the first round t such that the query x_t selected by A is an *ϵ-maximum* of $f^* \in \mathcal{F}$. We are interested in bounding the *worst-case ϵ-query complexity* of A, defined $Q_{A,\epsilon} \triangleq \sup_{f^* \in \mathcal{F}} E[T^{A,f^*,\epsilon}]$.

Below we give upper and lower bounds for the ϵ-MAX problem in terms of the *lower and upper ϵ-haystack dimensions*, which are each a different generalization of the haystack dimension. Before introducing these quantities, we need some definitions. For any set $F \subseteq \mathcal{F}$ let $F(x, \epsilon) \triangleq \{f \in F : \sup_{x'} f(x') - f(x) \leq \epsilon\}$ be the functions in F for which x is an ϵ-maximum and, as in Section 9, let $F(\langle x, y \rangle, \epsilon) \triangleq \{f \in F : |f(x) - y| > \epsilon\}$ be the functions in F that are more than ϵ-inconsistent with the labeled example $\langle x, y \rangle$. Also, let

$$\theta^-(\epsilon) \triangleq \inf_{F \subseteq C(\epsilon)} \sup_{x \in \mathcal{X}} \inf_{y \in \mathbb{R}} \frac{|F(x,\epsilon) \cup F(\langle x, y \rangle, 0)|}{|F|} \quad \text{and} \quad \theta^+(\epsilon) \triangleq \inf_{F \subseteq C(\epsilon)} \sup_{x \in \mathcal{X}} \inf_{y \in \mathbb{R}} \frac{|F(x,0) \cup F(\langle x, y \rangle, \epsilon)|}{|F|}$$

Note that the only difference between $\theta^-(\epsilon)$ and $\theta^+(\epsilon)$ is the placement of ϵ and 0. Also note that if $\mathcal{F}$ is finite and $C(\epsilon) = \mathcal{F}$ then $\theta^-(0) = \theta^+(0) = \theta$, where θ was defined in Section 4. Now define the *lower ϵ-haystack dimension* $\mathrm{HD}^-(\epsilon) \triangleq \frac{1}{\theta^-(\epsilon)}$ and the *upper ϵ-haystack dimension* $\mathrm{HD}^+(\epsilon) \triangleq \frac{1}{\theta^+(\epsilon)}$.

A simple approach to solving the ϵ-MAX problem is just to run a slightly modified version of the greedy algorithm from Section 6 using $C(\epsilon)$ as the initial attention space, and removing inconsistent functions only if they are inconsistent by more than ϵ. In other words, in each round t, the *ϵ-greedy algorithm* G_ϵ selects $x_t = \arg\sup_{x \in \mathcal{X}} \inf_{y \in \mathbb{R}} |F_t(x, 0) \cup F_t(\langle x, y \rangle, \epsilon)|$ where the *attention space* F_t is defined inductively as follows: $F_1 = C(\epsilon)$ and $F_{t+1} = F_t \setminus (F_t(x_t, 0) \cup F_t(\langle x_t, y_t \rangle, \epsilon))$.

Theorem 6. *Let G_ϵ be the ϵ-greedy algorithm for the ϵ-MAX problem. Then $Q_{G_\epsilon, 2\epsilon} \leq \mathrm{HD}^+(\epsilon) \log |C(\epsilon)|$ for all $\epsilon > 0$.*

Proof. The proof is nearly identical to the proof of Theorem 1. The algorithm G_ϵ initializes the attention space to $C(\epsilon)$, and after every query at least a $\theta^+(\epsilon)$ fraction of the attention space is eliminated. By the time the attention space is empty, the G_ϵ algorithm has selected a maximum of some function $\hat{f} \in C(\epsilon)$ that ϵ-covers the true function f^*, which implies that a 2ϵ-maximum of f^* has been selected. □

Furthermore, we can also straightforwardly lower bound the query complexity of any algorithm for the ϵ-MAX problem.

Theorem 7. *Let A be any algorithm for the ϵ-MAX problem. Then $Q_{A,\epsilon} \geq \frac{1}{6}\,\mathrm{HD}^-(\epsilon)$ for all $\epsilon > 0$.*

Proof. The proof of Theorem 2 can be repeated, with essentially no changes, to prove this theorem as well. The key is to observe that, when proving Theorem 2, we made no use of the fact that X^{f^*} contained the maxima of the true function f^*. We could have defined X^{f^*} to be any subset of $\mathcal{X}$, including the ϵ-maxima of f^*. □

Notice that the upper and lower bounds in Theorems 6 and 7 are not directly comparable, since we have not related the quantities $\mathrm{HD}^-(\epsilon)$ and $\mathrm{HD}^+(\epsilon)$. If it happens that $\frac{\mathrm{HD}^+(\epsilon)}{\mathrm{HD}^-(\epsilon)} \leq K$ for some constant K, then clearly the upper and lower bounds above are within constant and logarithmic factors of each other, just as we had for finite function classes. Indeed, a simple infinite function class for which this occurs is the class of all bounded norm hyperplanes. Let $\mathcal{X} = \{x \in \mathbb{R}^n \mid \|x\|_\infty \leq 1\}$ and $\mathcal{F}_{\text{hyper}} = \{\langle w, \cdot\rangle \mid w \in \mathbb{R}^n, \|w\|_\infty \leq 1\}$, and let the finite covering oracle C be the appropriate discretization of $\mathcal{F}_{\text{hyper}}$, i.e., $C(\epsilon) = \{w \in \mathbb{R}^n \mid \forall i\ w_i \in \{0, \frac{\epsilon}{n}, \frac{2\epsilon}{n}, \ldots, 1\}\}$. Clearly $|C(\epsilon)| = \Theta((\frac{n}{\epsilon})^n)$, but nonetheless the ratio of the lower and upper ϵ-haystack dimension is not large.

Theorem 8. *For function class $\mathcal{F}_{\text{hyper}}$ we have $\frac{\mathrm{HD}^+(\epsilon)}{\mathrm{HD}^-(\epsilon)} \leq n$ for all $\epsilon > 0$.*

Proof. By examining the definitions of $\mathrm{HD}^+(\epsilon)$ and $\mathrm{HD}^-(\epsilon)$, we see that it suffices to show that $\sup_{x\in\mathcal{X}} \frac{|F(x,0)|}{|F|} \geq \frac{1}{n}$ for any finite $F \subseteq \mathcal{F}$. Let $k_i = |\{\langle w, \cdot\rangle \in F \mid w_j \geq w_i, \forall j\}|$ be the number of functions in F which have their maximal component at i. Clearly there must exist a $k_i \geq \frac{|F|}{n}$. Let e_i be the vector with a one in its ith component and zeros everywhere else. Since $e_i \in \mathcal{X}$ and e_i maximizes k_i functions in F, there exists an $x \in \mathcal{X}$ with $\frac{|F(x,0)|}{|F|} \geq \frac{1}{n}$. □

Unfortunately, the bound in Theorem 8 cannot be generalized to all function classes with finite covering oracles. In the next theorem, we give an example of an infinite function class $\mathcal{F}$ with a finite covering oracle C for which $\frac{\mathrm{HD}^+(\epsilon)}{\mathrm{HD}^-(\epsilon)} = \Omega(|C(\epsilon)|)$, which is essentially the worst possible ratio.

Theorem 9. *There exists a function class $\mathcal{F}$ with a finite covering oracle C such that $\mathrm{HD}^-(\epsilon) = 1$ and $\mathrm{HD}^+(\epsilon) = \Omega\left(|C(\epsilon)|\right)$ for all $\epsilon > 0$.*

Proof. Let $\mathcal{X} = [0, 1]$ and fix any sequence $\{x_n\} \subset \mathcal{X}$, all elements distinct. For visualization, it may be helpful to suppose that $\{x_n\}$ is strictly increasing, but this is not necessary. Let $\{\epsilon_i\}$ be the sequence defined by $\epsilon_i = \frac{1}{2^i}$ for all $i \in \mathbb{N}$.

For each $n \in \mathbb{N}$ let there be a function $f_n \in \mathcal{F}$ whose values are zero everywhere except $x_1, \ldots, x_n$. The nonzero values of f_n are defined as follows: Let $f_n(x_n) = \frac{1}{n}$ and $f_n(x_m) = \epsilon_i - \frac{1}{2^{2^n}}$ for all $m < n$, where $i = \lceil \log_2 n \rceil$. Let $C(\epsilon) = \{f_1, \ldots, f_{N_\epsilon}\}$, where N_ϵ is the smallest integer such that $1/N_\epsilon < \epsilon$. We have that C is finite covering oracle because each $C(\epsilon)$ is finite and because $\sup_{x\in\mathcal{X}} f_n(x) \leq \epsilon$ for all $n \geq N_\epsilon$.

We only need the following properties of this construction, which can be verified: (1) For all $n \in \mathbb{N}$ the elements of $\{f_m(x_n) : m \geq n\}$ are all distinct; (2) Each f_n has a maximum at x_n and nowhere else; (3) For all $n \in \mathbb{N}$, if $i = \lceil \log_2 n \rceil$ then $f_n(x_n) \geq \epsilon_i$ and $f_m(x_n) < \epsilon_i$ for all $m \neq n$.

For the remainder of the proof fix $\epsilon > 0$ and the smallest $i \in \mathbb{N}$ such that $\epsilon_i \leq \epsilon$. We will show that $\theta^-(\epsilon) = 1$ and $\theta^+(\epsilon) \leq \frac{4}{N_{\epsilon_i}}$, which suffices to prove the theorem.

First, we claim that $\theta^-(\epsilon) = 1$. Let $F^+ = \arg\inf_{F\subseteq C(\epsilon)} \sup_{x\in\mathcal{X}} \inf_{y\in\mathbb{R}} \frac{|F(x,\epsilon)\cup F(\langle x,y\rangle,0)|}{|F|}$. Let n be the smallest integer such that $f_n \in F^+$. Then each $f_m \in F^+$ has a distinct value at x_n, by property 1. So $\inf_{y\in\mathbb{R}} |F^+(\langle x_n, y\rangle, 0)| = |F^+|$. Next, we claim that $\theta^+(\epsilon) \leq \frac{4}{N_{\epsilon_i}}$. Let $F_i = \{f_n \in \mathcal{F} : i = \lceil \log_2 n \rceil\}$, and note that $F_i \subseteq C(\epsilon)$ and $|F_i| = \frac{N_{\epsilon_i}}{2}$. For any $x \in \mathcal{X}$ we have $|F_i(x, 0)| \leq 1$, by property 2, and $\inf_{y\in\mathbb{R}} |F_i(\langle x, y\rangle, \epsilon)| \leq 1$, by property 3. □

12 Computational Complexity

Our results have so far ignored computational complexity. In general a function class $\mathcal{F}$, for which finding the optimal query is computationally intractable, might nevertheless have small haystack dimension, admitting algorithms with low query complexity. Consider the following simple example. Let $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3$ where $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ are disjoint, $|\mathcal{X}_1| = |\mathcal{X}_2| = n$ and $\mathcal{X}_3 = 2^n$. Each function in $\mathcal{F}$ will attain its maximum on some action in $\mathcal{X}_3$. The location of that maximum, as in Example 2, will be encoded by $\mathcal{X}_1$ and $\mathcal{X}_2$. However, we will now encode the location cryptographically, with a

function's behavior on $\mathcal{X}_1$ representing a public key, and a function's behavior on $\mathcal{X}_2$ representing the encrypted location of its maximum.

More precisely, let z be an n-bit number. For a pair of $\frac{n}{2}$-bit primes p and q, let $N_{pq} = pq$. Also let $e(z, N_{pq}) = z^2 \mod N_{pq}$ be z encrypted with public key N_{pq}.

Now let $f_{z,p,q}$ be the function that gives the ith bit of N_{pq} as output to the ith input of $\mathcal{X}_1$ and the ith bit of $e(z, N_{pq})$ as output to the ith input of $\mathcal{X}_2$. On the zth input of $\mathcal{X}_3$, $f_{z,p,q}$ outputs a 2. On all other inputs of $\mathcal{X}_3$, $f_{z,p,q}$ outputs 0. Let $\mathcal{F}$ be the function class consisting of all functions $f_{z,p,q}$ for every pair of $\frac{n}{2}$-bit primes p, q and n-bit integer z.

There exists an algorithm with query complexity $O(n)$ for any $f^* \in \mathcal{F}$. That algorithm queries each action in $\mathcal{X}_1 \cup \mathcal{X}_2$, retrieving the public key N_{pq} and the cypher $e(z, N_{pq})$. Information-theoretically, the maximum of f^* can be found in a single additional query. The algorithm may simply test every n-bit z', checking if $e(z, N_{pq}) = e(z', N_{pq})$. However, computing z is as hard as factorization (Kearns and Vazirani, 1994).

Acknowledgments

We thank Alex Slivkins and the anonymous reviewers for their helpful comments.

References

Jacob Abernethy, Elad Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *Proceedings of the 21th Annual Conference on Computational Learning Theory*, 2008.

Kareem Amin, Michael Kearns, and Umar Syed. Graphical models for bandit problems. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, 2011.

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. Online optimization in X-armed bandits. In *Advances in Neural Information Processing Systems 21*, 2008.

Varsha Dani and Thomas P. Hayes. Robbing the bandit: Less regret in online geometric optimization against an adaptive adversary. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2006.

Abraham Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Proceeding of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2005.

Tibor Hegedüs. Generalized teaching dimensions and the query complexity of learning. In *Proceedings of the 8th Annual Conference on Computational Learning Theory*, 1995.

Michael Kearns and Umesh Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, 2008.

John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *Advances in Neural Information Processing Systems 20*, 2007.

Tyler Lu, Dávid Pál, and Martin Pál. Showing relevant ads via Lipschitz context multi-armed bandits. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.

Adam J. Mersereau, Paat Rusmevichientong, and John N. Tsitsiklis. A structured multiarmed bandit problem and the greedy policy. *IEEE Transactions on Automatic Control*, 54(12):2787–2802, 2009.

Rob Nowak. Noisy generalized binary search. In *Advances in Neural Information Processing Systems 22*, 2009.

Aleksandrs Slivkins. Contextual bandits with similarity information. In *Proceedings of the 24th Annual Conference on Computational Learning Theory*, 2011.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning*, 2008.

William Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3–4):285–294, 1933.