Learning the Valuations of a k-demand Agent

Hanrui Zhang [1] **Vincent Conitzer** [1]

Abstract

We study problems where a learner aims to learn the valuations of an agent by observing which goods he buys under varying price vectors. More specifically, we consider the case of a k-demand agent, whose valuation over the goods is additive when receiving up to k goods, but who has no interest in receiving more than k goods. We settle the query complexity for the active-learning (preference elicitation) version, where the learner chooses the prices to post, by giving a *biased binary search* algorithm, generalizing the classical binary search procedure. We complement our query complexity upper bounds by lower bounds that match up to lower-order terms. We also study the passive-learning version in which the learner does not control the prices, and instead they are sampled from some distribution. We show that in the PAC model for passive learning, any *empirical risk minimizer* has a sample complexity that is optimal up to a factor of $\widetilde{O}(k)$.

1. Introduction

The active learning of agents' preferences is also known as *preference elicitation*. Depending on the setting, we may wish to model and represent preferences differently. For example, if there is a set of alternatives to choose from, and agents cannot make payments, then it is natural to represent an agent's preferences by a weak ordering $\succeq$. If agents also express preferences over distributions over alternatives, we may wish to model an agent's preferences by a utility function $u(\cdot)$ and assume the agent is maximizing expected utility. We may learn agents' preferences by asking them *queries*, for example which of two (distributions over) alternatives is preferred.

[1]Department of Computer Science, Duke University, Durham, USA. Correspondence to: Hanrui Zhang <hrzhang@cs.duke.edu>, Vincent Conitzer <conitzer@cs.duke.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

In other contexts, such as the allocation of goods (or bads, e.g., tasks), agents are often able to make payments (or receive them as compensation). In this context, it is natural to model the agent's preferences by a *valuation function* $v(\cdot)$, and assume that utility is valuation minus payment made. Depending on the setting, different types of query may be possible. A *value query* would ask directly for the valuation that the agent has for a specific bundle of goods. But it is not always feasible to ask value queries, for example because the agent finds the query hard to answer, is reluctant to answer it out of fear of exploitation, or because there are simply exogenous restrictions on the type of query we can make. For example, if we are running a grocery store, the only way in which we may learn about an agent's valuation is by setting prices and seeing what he buys. This is what is known as a *demand query*—given these prices, what would you buy? Such queries will be the focus of our paper.

The very simplest setting involves only a single good. In this case, active learning of the agent's valuation is equivalent to the binary search problem: if we we quote a price p that is above the valuation we get a "no" answer, and otherwise a "yes" answer.[1] If there are multiple goods but valuations are *additive*, so that an agent's valuation for a bundle S of items is simply $v(S) = \sum_{j \in S} v(\{j\})$, then the agent's decision on one good is independent of that on the other goods, and we can simply parallelize the binary searches for the individual goods. More interesting is the case of *unit demand*, where there are multiple goods but the agent will buy at most one good, namely the good j that maximizes $v(j) - p(j)$ if this value is nonnegative. Here, the active learning problem can be thought of as the following simple abstract problem. There is a vector of unknown numbers $\vec{v}$; a query consists of subtracting from it an arbitrary other vector $\vec{p}$, and learning the index of the maximum element of $\vec{v} - \vec{p}$, but not its value. (Note that it makes no difference whether we add, subtract, and/or allow negative numbers.) Given the simplicity of this problem, it is likely to have applications outside of economics as well. For example, imagine a physical system, each possible state of which has

[1]Throughout the paper we assume consistent tie-breaking. I.e., whenever $v(j) - p(j) = 0$, the agent either always wants the item, or always does not want the item. Similarly, whenever two items i and j provide the same utility, i.e., $v(i) - p(i) = v(j) - p(j)$, one of the two is always preferred to the other.

a baseline energy that we wish to learn. We can arbitrarily modify the energy of each state, after which the system will go to the lowest-energy state, which we then observe. This is the same problem.[2]

Surprisingly, to our knowledge, how many queries are needed for this very basic problem has not yet been analyzed. In this paper, we settle this up to lower order terms for the generalization of a k-demand agent, who will buy at most k goods, namely the top k goods j as measured by $v(j) - p(j)$ (unless there are fewer than k for which $v(j) - p(j) \geq 0$, in which case only those will be bought). We also study the passive-learning version where we do not control the price vectors, but instead they are generated from some distribution. (This would correspond to the case where the energy modifications are the result of an external random process.)

1.1. Our Results

In Section 2 we study the active elicitation problem, where the learner chooses the price vectors to post, observes the purchased sets, and aims to learn the exact valuations of the agent. We show that when there are n items, and the value of each item is an integer between 0 and W, there is an algorithm that learns the agent's valuations in

$$(1 + o(1)) \left(\frac{n \log W}{k \log(n/k)} + \frac{n}{k} \right)$$

rounds, when k is not too large. We complement this upper bound by showing that both first-order terms of our upper bound are necessary. More specifically, we give adversarial distributions over valuations, where any algorithm needs $(1 - o(1)) \frac{n \log W}{k \log(n/k)}$ and $\lfloor \frac{n-1}{k} \rfloor$ rounds, respectively. Our algorithm is therefore optimal in a strong sense.

In Section 3, we study the passive learning problem. We consider a PAC setting, where price vectors are drawn from a distribution; the learner observes the price vectors as well as the agent's choices, and aims to predict the agent's future choices. We establish sample complexity upper and lower bounds for the passive learning problem by settling the *Natarajan dimension* of the corresponding concept class. We also give efficient algorithms for the *empirical risk minimization (ERM)* problem; by solving this problem, our upper bound is achieved.

Our bounds for the passive learning task are only approximately tight in a worst-case sense, which means that in practice, our learning algorithm is likely to outperform the theoretical upper bound. In Section 4, we experimentally evaluate the performance of ERM algorithms. Our findings show that when prices are i.i.d., the empirical sample complexity of ERM algorithms depends much more mildly on the number of items n and the demand k than the theoretical bound suggests.

1.2. Related Work

In economics, there is a long line of work on *revealed preference theory*, initiated by Samuelson (1938). Here, the idea is to infer consumers' utility functions based on the choices that they make. However, most of this work concerns *divisible goods* and consumers that optimize given a *monetary budget*. Some work concerns the construction of valuations that explain the observed choices of the agent. In particular, Afriat (1967) shows that a sequence of observations can be explained by a utility function if and only if it can be explained by a utility function that is piecewise linear, monotone, and concave. While the proof is constructive, the representation of the constructed utility function is complex in proportion to the number of observed choices, so in general the construction fails to be predictive.

In computer science, researchers have worked on both active and passive learning models. Some of the earliest work focuses on *preference elicitation* (Sandholm & Boutilier, 2006), an active-learning variant in which a party such as an auctioneer asks the agents queries about their valuations, and the agents respond. Typically, the goal is to determine the final outcome (say, allocation of resources) with as few queries as possible, though sometimes this is done by simply learning each agent's valuation function precisely and then computing the allocation.[3] Early work established close connections between preference elicitation and the active learning of valuation functions (Blum et al., 2004; Lahaie & Parkes, 2004).

Multiple types of queries are studied in this line of work. One is a *value query*, where an agent is asked for his valuation for a specific bundle. Another is a *demand query*, where an agent is asked which items he would buy at specific prices for the items. The latter type of query is the one of interest in this paper. Passive variants where in each round, prices are sampled from a distribution (or are otherwise outside our control) and we see what the agent buys under these prices, also fit the demand-query model—every

[2] As a more specific example, suppose there is a set S of nearby natural structures in a lightning-prone area. We are interested in determining the electrical resistance of each structure. To do so, we can place lightning rods of varying heights on the structures, which will reduce the resistance of the electrical path through each structure by a known amount, and see where lightning strikes—which will reveal which of the paths has the lowest resistance for the given lightning rods.

[3] One may worry about incentives, e.g., an agent pretending to have a low valuation in order to be quoted a low price at the end. However, if VCG pricing is used, then it is an ex-post equilibrium for all agents to respond truthfully to every query (Sandholm & Boutilier, 2006; Nisan et al., 2007). This insight also applies to our work here.

round corresponds to a demand query that we do not control. This is what we study towards the end of this paper. (There are also passive-learning variants corresponding to value queries (Balcan & Harvey, 2011; Balcan et al., 2012), but we will not discuss those here.) Various iterative mechanisms (Parkes, 2006), such as ascending combinatorial auctions, require the agent to indicate a preferred bundle while prices adjust; these mechanisms are thus also implemented as sequences of demand queries. In other contexts, it is natural to ask different types of queries yet again: in voting (Conitzer, 2009; Procaccia et al., 2009), one may ask a *comparison query* (which of these two alternatives do you prefer?), and in cake cutting (Brams & Taylor, 1996; Procaccia & Wang, 2017), one may ask how far the knife needs to be moved for the agent to become indifferent between the two parts. However, in this paper we only consider the setting where agents have valuations for items and respond to demand queries.

To study the prediction aspect of revealed preferences theory, Beigman & Vohra (2006) consider a PAC-learning model. They introduce a complexity measure of classes of utility functions, and, based on the complexity measure, characterize the learnability of different classes. Following Beigman and Vohra, Zadimoghaddam and Roth (Zadimoghaddam & Roth, 2012) give efficient learning algorithms for linearly separable concave utility functions in the PAC model. Their bound was later improved by Balcan et al. (2014), who also give generalizations to other classes of utility functions, misspecified models, and non-linear prices. Slightly departing from the PAC setting, Amin et al. (2015) study profit maximization in an online learning model. Bei et al. (2016) extend the results of Balcan et al. to Fisher and exchange markets. All these papers study divisible goods and monetary budgets. In this paper, in contrast, we consider *indivisible* goods and *k-demand* agents without a monetary budget constraint. Our results are therefore of a combinatorial nature.

Basu & Echenique (2018) study the learnability of preference models of choice under uncertainty, and Chase & Prasad (2018) study the learnability of time dependent choices. Their models are intrinsically different from ours, and in particular, they aim to learn *binary* relations, as opposed to predicting combinatorial outcomes. Blum et al. (2018) consider a setting where a seller has unknown priority among the buyers, according to which they are allocated items. They give algorithms that with few mistakes reconstruct both the buyers' valuations and the seller's priority, whenever the buyers have additive, unit-demand, or single-minded valuations. These results are incomparable to ours, since (1) they consider an online model where the goal is to minimize the number of mistakes, whereas we give algorithms that operate either with active querying or in the PAC model, and (2) even in their online model, when there are variable prices, their results apply only to additive or unit-demand buyers, and the mistake bound depends on that of the ellipsoid algorithm. The main complexity of their model comes from the fact that there are multiple agents affecting each other. There is various research on similar, but less closely related, topics (Besbes & Zeevi, 2009; Babaioff et al., 2015; Roth et al., 2016; Brero et al., 2017; Roth et al., 2017; Brero et al., 2018; Balcan et al., 2018; Ji et al., 2018).

2. Active Preference Elicitation

In this section, we study the following active learning model: there is a single k-demand buyer in the market, to whom the learning agent (the seller) may pose demand queries, each consisting of a vector of prices. The buyer values the i-th item v_i, where it is common knowledge that $v_i \in \{0, 1, 2, \ldots, W\}$. The actual values of the buyer, however, are unknown to the seller, and are for the seller to learn. The seller repeatedly posts prices on individual items. The buyer then buys the k (or fewer) items that maximize his quasi-linear utility, and the seller observes the buyer's choice of the k (or fewer) items to buy. The question we are interested in is the following: what is the minimum number of rounds (i.e., demand queries) needed such that the seller can acquire enough information to be sure of $(v_i)_i$, and what algorithm achieves this number?

2.1. The Biased Binary Search Algorithm

We present an algorithm based on *biased binary search*, Algorithm 1. The algorithm, generalizing the classical binary search procedure, works in the following way: first, we fix an item (item 1) as the *reference item*, and learn its valuation using binary search. Then, throughout the execution, the algorithm keeps track of the possible range $[v_i^-, v_i^+]$ of each item i's value. We maintain A as the set of items for which we have not yet learned the exact valuation. If, for a given demand query, the reference item is chosen, then we know that each item i that is not chosen gives utility at most that of the reference item, allowing us to update v_i^+. If the reference item is not chosen, then we know that each item i that is chosen gives utility at least that of the reference item, allowing us to update v_i^-. The algorithm sets prices in such a way that no matter what the chosen set is, the "information gain" (as measured by a potential function) from shrinking the ranges is always about the same. The word "biased" in the name indicates that the ranges do not necessarily shrink by a factor of $\frac{1}{2}$. For example, in the unit demand case, if the reference item is chosen, we get to shrink all the other items' ranges, but only by a little; whereas if another item is chosen, we get to shrink only that item's range, but by a lot. This ensures the information gain is (roughly) invariant. When we learn an item i's valuation and drop it from A, we update the number of items $n' = |A|$ whose valuation we

Algorithm 1 Biased Binary Search

1: **Input:** number of items n, range of value W
2: **Output:** $(v_i)_i$
3: Post price $p_i = \infty$ for $i \geq 2$, and binary search for v_1.
4: Let $v_1^- = v_1^+ = v_1$, $p_1 = v_1 - 0.5$, $A = \{2, \ldots, n\}$, $n' = n$.
5: For each $i \in A$, let $v_i^- = 0$, $v_i^+ = W$.
6: **while** `true` **do**
7: **for** $i \in A$ **do**
8: Set $p_i = v_i^+ - (v_i^+ - v_i^-) \cdot \frac{k \log(n'/k)}{n'} - 0.5$.
9: **end for**
10: Ask a query at these prices; let S be the winning set.
11: **if** $1 \in S$ **then**
12: **for** $i \in A \setminus S$ **do**
13: Let $v_i^+ = p_i + 0.5$.
14: **end for**
15: **else**
16: **for** $i \in S$ **do**
17: Let $v_i^- = p_i + 0.5$.
18: **end for**
19: **end if**
20: **for** $i \in A$ **do**
21: **if** $v_i^+ - v_i^- < 1$ **then**
22: Let $A = A \setminus \{i\}$, $n' = n' - 1$, $p_i = \infty$.
23: **end if**
24: **end for**
25: Break if $2k \geq n'$.
26: **end while**
27: Let B be any subset of A of cardinality $\min(n', k)$.
28: Post price $p_i = \infty$ for all $i \in A \setminus B$, and binary search in parallel for $(v_i)_{i \in B}$.
29: Post price $p_i = \infty$ for all $i \in B$, and binary search in parallel for $(v_i)_{i \in A \setminus B}$.
30: **for** $i \in [n]$ **do**
31: Let $v_i = \lfloor v_i^+ \rfloor$.
32: **end for**
33: Output $(v_i)_i$.

still need to learn. If n' becomes less than twice as large as k, we divide the remaining items in A into two groups of size not exceeding k; for each group, we perform binary search for all items in the group, while posting price ∞ for items in the other group to ensure they are never chosen. Because the size of neither group exceeds k, an item will be chosen if and only if its value exceeds the price, independent of the prices of the other items in the group. Hence, we can learn the values of all items in the group in parallel, via binary search.

We now bound the query complexity of the algorithm.

Theorem 1. *Algorithm 1 computes the values $(v_i)_i$ of the buyer, and has query complexity*

$$(1 + o(1)) \left(\frac{n \log W}{k \log(n/k)} + \frac{n}{k} \right) + O(\log W).$$

Before proceeding to the proof, we note that in the more interesting case where k is not too large compared to n (i.e., $k = o(n)$), the term $O(\log W)$ is dominated by the other terms of the bound.

Proof of Theorem 1. We first prove correctness. We show that throughout the repeat loop, we always have

$$v_i \in [v_i^-, v_i^+].$$

Consider the update procedure from Line 10 to Line 18. When the reference item, item 1, is among the chosen ones, we know that for any unchosen item i,

$$v_i - p_i \leq v_1 - p_1 = 0.5.$$

Therefore,

$$v_i \leq p_i + 0.5$$

and the right-hand side is what v_i^+ is updated to in this case. When item 1 is not chosen, we know that for any chosen item i,

$$v_i - p_i \geq v_1 - p_1 = 0.5.$$

Therefore,

$$v_i \geq p_i + 0.5$$

and the right-hand side is what v_i^- is updated to in this case.

Now we prove the query complexity upper bound. The binary search for v_1 takes $\log W$ demand queries. To analyze the dominant part of the complexity, let us define the following potential function:

$$\begin{aligned} \Phi(((v_i^-, v_i^+))_i) &= \sum_{1 < i \leq n} \phi(v_i^+, v_i^-) \\ &= \sum_{1 < i \leq n} \log(v_i^+ - v_i^-). \end{aligned}$$

The objective is to show that (1) after each query and update, the potential function decreases by a considerable amount, and (2) the total possible decrease of the potential function throughout the execution of the algorithm is bounded. As a result, the total number of queries must also be bounded. We first bound the decrease of the potential function. Observe that when an item is removed from A in Line 22, we fix its price such that it will never be chosen in all future queries. Thus, we maintain the following invariant: every time a query happens, all items chosen are in $\{1\} \cup A$.

Now consider a query where items in S are chosen. When $1 \in S$, for each $i \in A$ that is not chosen (i.e., $i \in A \setminus S$), $\phi(v_i^+, v_i^-)$ decreases by

$$\begin{aligned}&\log(v_i^+ - v_i^-) - \log(p_i + 0.5 - v_i^-)\\ =\ &\log\left(\frac{1}{1 - k\log(n'/k)/n'}\right)\\ \geq\ &\frac{k\log(n'/k)}{n'}.\end{aligned}$$

Since there are at least $n' - k$ such items, the total decrease is

$$-\Delta\Phi \geq k\log(n'/k)(1 - k/n').$$

When $1 \notin S$, for each $i \in A$ that is chosen (i.e., $i \in S$), $\phi(v_i^+, v_i^-)$ decreases by

$$\log\left(\frac{n'}{k\log(n'/k)}\right).$$

Since item 1 gives utility $v_1 - p_1 = 0.5$, and 1 is still not chosen, there must be at least k other items giving utility at least 0.5. As a result, there are exactly k chosen items, so the total decrease is

$$-\Delta\Phi = k\log(n'/k)(1 - \log\log(n'/k)/\log(n'/k)).$$

Putting together the two cases, we see that whenever $n'/k = \omega(1)$,

$$-\Delta\Phi = k\log(n'/k)(1 - o(1)).$$

And whenever $2k \leq n'$,

$$-\Delta\Phi = \Omega(k\log(n'/k)).$$

Consider first the case where $2k \geq n$. In such cases, the algorithm partitions A into two parts of size not exceeding k, and binary searches for each part respectively. Within each part, since the number of items available in the part is no more than the demand of the buyer, and all items in the other part are too expensive to be chosen, whether one item will be chosen depends only on the value and price of the item. As a result, we can binary search for the values of all items in each part in parallel. The query complexity is therefore $O(\log W)$.

Now suppose $2k < n$. The worst-case query complexity happens when sets of items of size k are repeatedly chosen and drop out sequentially. That is, queries keep returning the same set of k items, until the algorithm completely learns the valuations restricted to these k items (i.e., they "drop out" of the learning procedure), and then the queries move on and keep returning another set of k items, until they "drop out" too, etc. There are essentially two stages of the worst case execution pattern: in the first stage of the execution, n' keeps decreasing, and when $2k \geq n'$, the execution enters the second stage, where parallel binary search is performed to determine the values of all items in A. The sequence of "drop-outs" happens in the first stage and we refer to what happens between two "drop-outs" as a substage. By the analysis above, the query complexity of the second stage is simply $O(\log W)$.

The first stage requires more effort. W.l.o.g., assume that k divides n. The first stage can be divided into

$$\ell = n/k - 1 = \omega(1)$$

substages, where in the i-th substage, the number of active items at the beginning of the substage is

$$n' = (\ell - i + 1)k.$$

First observe that in the i-th substage, the minimum possible value of $\phi(v_j^+, v_j^-)$ that can be reached by updating v_j^+ or v_j^- is

$$\log\left(\frac{k\log(n'/k)}{n'}\right) \geq \log\left(\frac{1}{(\ell - i + 1)}\right).$$

This is because once $v_j^+ - v_j^-$ drops below 1, it will never be updated again. Since the maximum possible value of $\phi(v_j^+, v_j^-)$ is $\log W$, the maximum total decrease of Φ in the i-th substage is

$$k\left(\log W + \log(\ell - i + 1)\right).$$

On the other hand, the decrease per query in the i-th substage is

$$(-\Delta\Phi)_i = \begin{cases}\Omega(k), & \text{if } (\ell - i + 1) = O(1)\\ k\log(\ell - i + 1)(1 - o(1)), & \text{otherwise}\end{cases}.$$

This means the number of queries in the i-th substage is at most

$$\frac{k\left(\log W + \log(\ell - i + 1)\right)}{(-\Delta\Phi)_i}.$$

Now for any $0 < t < 1$, the total number of queries in the first stage is upper bounded by

$$\begin{aligned}&\sum_{1 \leq i < \ell} \frac{k\left(\log W + \log(\ell - i + 1)\right)}{(-\Delta\Phi)_i}\\ =\ &\sum_{2 \leq i \leq \ell} \frac{k\left(\log W + \log i\right)}{(-\Delta\Phi)_{\ell - i + 1}}\\ =\ &\sum_{2 \leq i < \ell^t} \frac{k\left(\log W + \log i\right)}{(-\Delta\Phi)_{\ell - i + 1}} + \sum_{\ell^t \leq i \leq \ell} \frac{k\left(\log W + \log i\right)}{(-\Delta\Phi)_{\ell - i + 1}}.\end{aligned}$$

When $\ell^t = \omega(1)$, for any $\ell^t \leq i \leq \ell$, $i = \omega(1)$, and therefore

$$(-\Delta\Phi)_{\ell - i + 1} = (k\log i)(1 - o(1)).$$

So when $t = \omega(1/\log \ell)$, we can further bound the total number of queries by

$$\sum_{2 \le i < \ell^t} (\log W + \log i) + \sum_{\ell^t \le i \le \ell} (1 + o(1)) \frac{\log W + \log i}{\log i} \le \ell^t (\log W + \log \ell) + \ell(1 + o(1)) \left(\frac{\log W}{t \log \ell} + 1 \right)$$

Moreover, when when $t = 1 - \omega(\log \ell / \log\log \ell)$, the second term in the above dominates the first term, so we can further bound the sum by

$$\ell(1 + o(1)) \left(\frac{\log W}{t \log \ell} + 1 \right).$$

Now letting $t = 1 - o(1)$, the number is upper bounded by

$$(1 + o(1)) \left(\frac{n \log W}{k \log(n/k)} + \frac{n}{k} \right).$$

Putting together the two stages, we conclude that the total query complexity is

$$(1 + o(1)) \left(\frac{n \log W}{k \log(n/k)} + \frac{n}{k} \right) + O(\log W). \qquad \square$$

2.2. Matching Lower Bounds

It may appear at first sight that the two-term upper bound in the first part of Theorem 1 is probably suboptimal. However, we show that quite surprisingly, our upper bound is in fact tight *up to lower order terms*. Specifically, we have the following proposition.

Proposition 1. *The following lower bounds hold for actively learning the valuations of a k-demand agent with n items.*

- *When $k = o(n)$, given a uniform prior over the values, any (possibly randomized) algorithm that correctly outputs the values makes at least*

$$(1 - o(1)) \frac{n \log W}{k \log(n/k)}$$

queries in expectation.

- *Even if the values can only be 0 or 1, and there is precisely one item with value 0, any algorithm that correctly outputs the values with probability at least p makes at least $\lceil (np - 1)/k \rceil$ queries.*

Proof. To prove the first part, consider the following mutual information argument. To learn the exact values, the total mutual information gained from observing the query outcomes has to be $n \log W$. On the other hand, since there are only $\binom{n}{k}$ possible outcomes, the conditional mutual information of each query cannot exceed

$$\log \binom{n}{k} = k \log(n/k) + O(k).$$

As a consequence, the number of queries has to be at least

$$\frac{n \log W}{k \log(n/k) + O(k)} = (1 - o(1)) \frac{n \log W}{k \log(n/k)}.$$

For the second part, consider an adversary that obliviously picks the 0-valued item uniformly at random. The algorithm is only required to find the item with value 0. For each query, we consider the values of the k items with the lowest prices (with consistent tie-breaking). Observe that if these items all have value 1, then the only information the algorithm can obtain from this query is that these k items have value 1. Let us hold fixed the values of the algorithm's random bits. Then, as long as the agent keeps choosing the lowest-priced items in each query, the algorithm will follow a fixed sequence of queries. Suppose the algorithm makes T queries, and let i be the item with value 0. Consider the sequence of sets of the k lowest-priced items in these T queries, $S_1, \ldots, S_T$, where $|S_j| = k$. As long as $T < n/k$, with probability $1 - kT/n$, item i is not in any of these sets. In such cases, the best thing the algorithm can do is to output some item in $[n] \setminus (S_1 \cup \cdots \cup S_T)$, which, with probability $\frac{n - kT - 1}{n - kT}$ over the random choice of the adversary, is not the 0-valued item. Hence, regardless of the random bits (and hence also in expectation over them), we see that the failure probability of the algorithm is at least

$$1 - p \ge \frac{(1 - kT/n)(n - kT - 1)}{n - kT},$$

which implies $T \ge \frac{np - 1}{k}$. $\square$

3. PAC Learning from Revealed Preferences

In this section, we consider the following passive learning model: there is a single k-demand buyer in the market. The learner observes a sequence of demand queries and the agent's responses. In each query, a price vector is drawn from a fixed distribution and posted on the items. The buyer then chooses the (at most) k items that maximize his quasi-linear utility. The goal of the learner is to learn an approximately correct hypothesis of the buyer's valuation with probability at least $1 - \delta$, such that when a price vector is drawn from the same distribution, with probability $1 - \varepsilon$, the learner correctly determines the k items that the buyer chooses. The question is: what is the minimum number of queries such that the learner can achieve the above goal?

3.1. Learnability by ERM Algorithms

We show that an algorithm that outputs any value vector that is consistent with the observations (a.k.a. an *empirical*

risk minimization (ERM) algorithm) learns the ground truth efficiently. Moreover, the sample complexity of any ERM learner is optimal up to a factor of $\widetilde{O}(k)$.[4]

The problem here is a *multiclass PAC learning* problem. The data domain $\mathcal{X}_n = \mathbb{R}^n$ contains all possible price vectors, and the set of labels

$$\mathcal{Y}_{n,k} = \{S \subseteq [n] \mid |S| \leq k\}$$

consists of all subsets of $[n]$ of size at most k. Each value vector v acts as a classifier $v : \mathcal{X}_n \to \mathcal{Y}_{n,k}$ that, given a price vector, determines the set of chosen items. Our hypothesis class $\mathcal{H}_{n,k}$ is the set of classifiers induced by all possible value vectors. Given a distribution $\mathcal{D}$ over $\mathcal{X}_n$, we aim to learn, with probability at least $1-\delta$, an approximately correct hypothesis $h \in \mathcal{H}_{n,k}$ by observing sample data points $x_i \sim \mathcal{D}$ and labels $y_i = v(x_i)$, such that

$$\Pr_{x\sim\mathcal{D}}[v(x) \neq h(x)] \leq 1-\varepsilon.$$

To study the above problem, we investigate the *Natarajan dimension* of the hypothesis class $\mathcal{H}_{n,k}$, as defined below.

Definition 1 (Natarajan dimension (Natarajan, 1989))**.** Let $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a hypothesis class and let $S \subseteq \mathcal{X}$. We say that $\mathcal{H}$ N-shatters S if there exists $f_1, f_2 : S \to \mathcal{Y}$ such that $\forall x \in S$, $f_1(x) \neq f_2(x)$, and for every $T \subseteq S$ there is a $g \in \mathcal{H}$ such that

$$\forall x \in T,\ g(x) = f_1(x), \text{ and } \forall x \in S \setminus T,\ g(x) = f_2(x).$$

The *Natarajan dimension* of $\mathcal{H}$, denoted $d_N(\mathcal{H})$, is the maximal cardinality of a set that is N-shattered by $\mathcal{H}$.

The Natarajan dimension of a hypothesis class is closely related to the sample complexity of the corresponding learning task. Let $m^r_{\mathcal{H}}(\varepsilon,\delta)$ be the sample complexity of learning $\mathcal{H}$ with error ε and confidence $1-\delta$ in the realizable case, i.e., when the labels are determined by some $h^* \in \mathcal{H}$. Ben-David et al. (Ben-David et al., 1995) and Daniely et al. (Daniely et al., 2015) together show:

Theorem 2 ((Ben-David et al., 1995; Daniely et al., 2015), rephrased)**.** *There exist constants C_1 and C_2 such that for any $\mathcal{H}$,*

$$C_1\left(\frac{d_N(\mathcal{H}) + \ln\left(\frac{1}{\delta}\right)}{\varepsilon}\right) \leq m^r_{\mathcal{H}}(\varepsilon,\delta) \leq$$
$$C_2\left(\frac{d_N(\mathcal{H})\left(\ln\left(\frac{1}{\varepsilon}\right) + \ln(|\mathcal{Y}|) + \ln(d_N(\mathcal{H}))\right) + \ln\left(\frac{1}{\delta}\right)}{\varepsilon}\right).$$

Moreover, the upper bound is attained by any ERM learner.

In words, Theorem 2 says that up to logarithmic dependence on $1/\varepsilon$, $d_N(\mathcal{H})$, and $|\mathcal{Y}|$, the sample complexity $m^r_{\mathcal{H}}(\varepsilon,\delta)$ of hypothesis class $\mathcal{H}$ is determined solely by the Natarajan dimension $d_N(\mathcal{H})$ of $\mathcal{H}$. It is therefore crucial to determine the Natarajan dimension of the hypothesis class $\mathcal{H}_{n,k}$ corresponding to our problem. We show (see the appendix for a proof):

Lemma 1. *The Natarajan dimension of $\mathcal{H}_{n,k}$ is n.*

The harder part of the lemma is the upper bound on the Natarajan dimension, for which our proof works in the following way. Suppose towards a contradiction that there is a set S of $n+1$ price vectors shattered by $\mathcal{H}_{n,k}$. We create a graph with $n+1$ vertices, where vertices 1 through n correspond to the n items, and vertex $n+1$ corresponds to a dummy item which has value 0. Let f_1 and f_2 be the two classifiers as in Definition 1. For each $x \in S$, we add an undirected edge into the graph with a directed weight determined by x, $f_1(x)$ and $f_2(x)$. Each classifier g induces a way to direct these edges, with the two possible directions corresponding to $g(x) = f_1(x)$ and $g(x) = f_2(x)$, respectively. With $|S| = n+1$ edges, there must be a cycle, and we argue that it is impossible to construct a classifier g such that the cycle becomes a directed cycle in one of the two directions. This leads to a contradiction, since by our assumption, there exists such a classifier $g \in \mathcal{H}_{n,k}$.

Recall that for $\mathcal{H}_{n,k}$, the set of labels $\mathcal{Y}_{n,k}$ containing all subsets of $[n]$ of size at most k has cardinality $O(n^k)$, and therefore $\ln|\mathcal{Y}_{n,k}| = O(k\ln n)$. Given Theorem 2, Lemma 1 directly implies:

Theorem 3. *There exist constants C_1 and C_2, such that*

$$C_1\left(\frac{n + \ln\left(\frac{1}{\delta}\right)}{\varepsilon}\right) \leq m^r_{\mathcal{H}_{n,k}}(\varepsilon,\delta) \leq$$
$$C_2\left(\frac{n\left(k\ln n + \ln\left(\frac{1}{\varepsilon}\right)\right) + \ln\left(\frac{1}{\delta}\right)}{\varepsilon}\right).$$

Moreover, the upper bound is attained by any ERM learner.

That is, any ERM learner achieves the optimal sample complexity up to a factor of $O(k\ln n + \ln(1/\varepsilon))$.

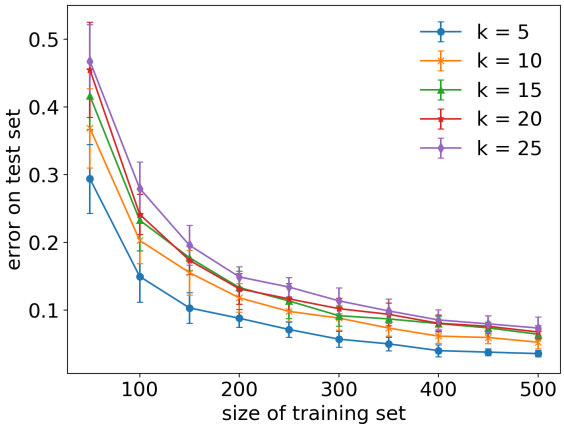
3.2. Computational Efficiency of ERM

While the above theorem establishes *sample complexity* upper and lower bounds for the passive learning problem, it does not address the issue of *computational complexity*. Below we show that there are in fact efficient ERM algorithms for the learning problem.

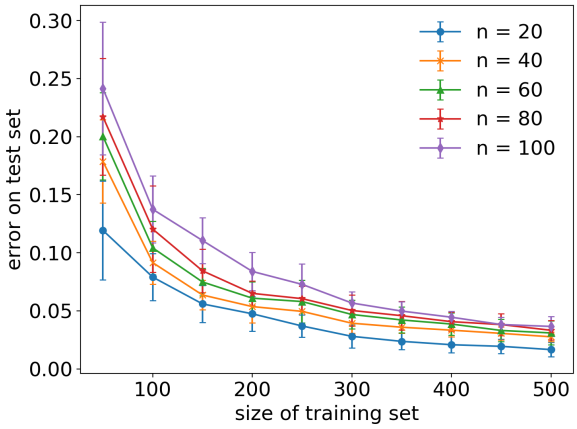
Proposition 2. *Given ℓ consistent samples, the ERM problem can be solved by solving a system of difference constraints with n variables and at most $\ell \cdot k \cdot (n-k+1)$ constraints.*

Proof. Observe that given samples $\{(p^j, S^j)\}_{j\in[\ell]}$, a value

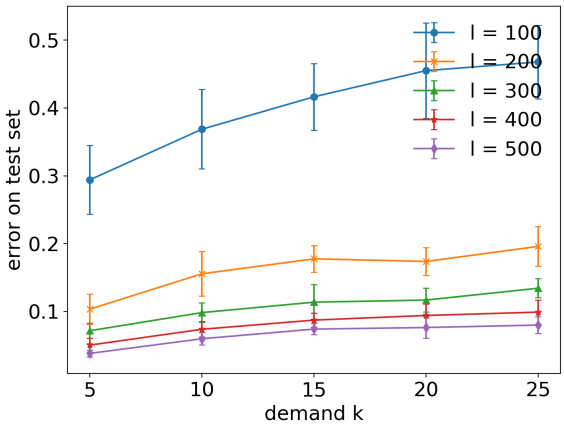
[4] $\widetilde{O}$ hides a polylog factor.



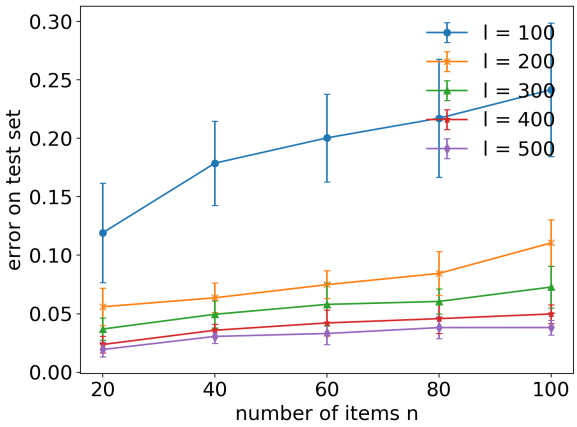
(a) Error rate of ERM vs ℓ ($5 \leq k \leq 25$, $n = 50$).



(b) Error rate of ERM vs ℓ ($20 \leq n \leq 100$, $k = 1$).



(c) Error rate of ERM vs k ($100 \leq \ell \leq 500$, $n = 50$).



(d) Error rate of ERM vs n ($100 \leq \ell \leq 500$, $k = 1$).

Figure 1: Performance of ERM for different number of items n, demand k, and size of training set ℓ.

vector v is an ERM (i.e., v is consistent with all (p^j, S^j)) iff for each $j \in [\ell]$,

- if $|S^j| = k$:
  - for any $i \in S^j$, $v_i - p^j_i \geq 0$,[5] and
  - for any $i_1 \in S^j$ and $i_2 \notin S^j$, $v_{i_1} - p^j_{i_1} \geq v_{i_2} - p^j_{i_2}$;[6]
- if $|S^j| < k$:
  - for any $i \in S^j$, $v_i - p^j \geq 0$, and
  - for any $i \notin S^j$, $v_i - p^j \leq 0$.

For each sample j, if $|S^j| = k$, there are k constraints of the first type, and $k \cdot (n-k)$ constraints of the second type; if $|S^j| < k$, there is one constraint for each $i \in [n]$, and therefore

$$n \leq k \cdot (n - k + 1)$$

constraints in total.

Thus, to compute an ERM, it suffices to solve the system with n variables, $(v_i)_{i \in [n]}$, and the above constraints, whose total number is no more than

$$\ell \cdot k \cdot (n - k + 1). \qquad \square$$

It follows immediately from Proposition 2 that the ERM problem can be solved efficiently by solving the corresponding system of difference constraints using efficient LP solvers or single-source shortest path algorithms.

[5] Tie-breaking issues can be dealt with by requiring a small margin that can be computed efficiently from the samples.

[6] An alternative approach is to introduce an auxiliary variable here for the agent's threshold utility for buying an item, reducing the number of constraints but increasing the number of variables.

4. Experimental Evaluation

In this section, we study empirically the accuracy of ERM learners for PAC learning from revealed preferences.

We implement the ERM learner by solving the system in Proposition 2 using an LP solver, where the objective is to maximize the minimum margin. We draw the ground truth value vector uniformly at random from the unit hypercube $[0, 1]^n$, and for each sample, we draw the price vector uniformly at random from $[-1, 0]^n$. The purchased set is then calculated according to the value and price vectors. Note that since the prices are non-positive, there are always k items purchased. To study the performance of ERM for different values of k, we fix the number of items to be $n = 50$, and examine the accuracy of the ERM learner for

$$k \in \{5, 10, 15, 20, 25\}$$

respectively. To study the performance of ERM for different values of n, we fix the agent to be unit-demand (i.e., $k = 1$), and calculate the accuracy of the ERM learner for

$$n \in \{20, 40, 60, 80, 100\}$$

respectively. In both experiments, we let the size of the training set grow, and plot the empirical error rate for each size of the training set

$$\ell \in \{50, 100, 150, 200, 250, 300, 350, 400, 450, 500\}.$$

When computing the error rate, we train 10 different classifiers using different sample sets, evaluate them on the same test set of size 10000, and take the average.

Figure 1 summarizes the average error rates and standard deviations for different n, k, and ℓ. As can be seen from Figures 1a and 1b, for all values of n and k examined, the empirical error rate decreases sharply as the size of the training set ℓ grows. With a training set of size $\ell = 500$, for all (n, k) pairs examined, the error rate drops below 0.1. This suggests that in practice, the constant factor hidden in our sample complexity upper bound is quite small, especially when the distribution of the price vector is uniform in the unit hypercube. On the other hand, as can be seen from Figures 1c and 1d, although the sample complexity upper bound in Theorem 3 grows roughly linearly in n and k, the empirical error rates of ERM learners for different values of k (in Figure 1c) and different values of n (in Figure 1d) seem quite close, especially when the training set is large enough (e.g., when $\ell = 500$). Based on this, we conjecture that the sample complexity of ERM depends much more mildly on n and k when the distribution of prices is uniform, or is independent over items.

5. Future Directions

The most compelling future direction is to consider more general classes of valuation functions, e.g., matroid-demand agents. Also, real-world agents often do not know exactly their own valuations. To this end, instead of perfectly accurate queries, one may consider noisy queries with various forms of noise.

Acknowledgements

This work is supported by NSF award IIS-1814056. The authors thank anonymous reviewers for helpful feedback.

References

Afriat, S. N. The construction of utility functions from expenditure data. *International Economic Review*, 8(1): 67–77, 1967.

Amin, K., Cummings, R., Dworkin, L., Kearns, M., and Roth, A. Online learning and profit maximization from revealed preferences. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 770–776, 2015.

Babaioff, M., Dughmi, S., Kleinberg, R., and Slivkins, A. Dynamic pricing with limited supply. *ACM Transactions on Economics and Computation (TEAC)*, 3(1):4, 2015.

Balcan, M.-F. and Harvey, N. J. Learning submodular functions. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, pp. 793–802. ACM, 2011.

Balcan, M. F., Constantin, F., Iwata, S., and Wang, L. Learning valuation functions. In *Conference on Learning Theory*, pp. 4–1, 2012.

Balcan, M.-F., Daniely, A., Mehta, R., Urner, R., and Vazirani, V. V. Learning economic parameters from revealed preferences. In *International Conference on Web and Internet Economics*, pp. 338–353. Springer, 2014.

Balcan, M.-F., Sandholm, T., and Vitercik, E. A general theory of sample complexity for multi-item profit maximization. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pp. 173–174. ACM, 2018.

Basu, P. and Echenique, F. Learnability and models of decision making under uncertainty. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pp. 53–53. ACM, 2018.

Bei, X., Chen, W., Garg, J., Hoefer, M., and Sun, X. Learning market parameters using aggregate demand queries. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 404–410, 2016.

Beigman, E. and Vohra, R. Learning from revealed preference. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pp. 36–42. ACM, 2006.

Ben-David, S., Cesabianchi, N., Haussler, D., and Long, P. M. Characterizations of Learnability for Classes of $\{0, ..., n\}$-Valued Functions. *Journal of Computer and System Sciences*, 50(1):74–86, 1995.

Besbes, O. and Zeevi, A. Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Operations Research*, 57(6):1407–1420, 2009.

Blum, A., Jackson, J., Sandholm, T., and Zinkevich, M. Preference elicitation and query learning. *Journal of Machine Learning Research*, 5(Jun):649–667, 2004.

Blum, A., Mansour, Y., and Morgenstern, J. Learning whats going on: Reconstructing preferences and priorities from opaque transactions. *ACM Transactions on Economics and Computation (TEAC)*, 6(3-4):13, 2018.

Brams, S. J. and Taylor, A. D. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.

Brero, G., Lubin, B., and Seuken, S. Probably approximately efficient combinatorial auctions via machine learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 397–405, 2017.

Brero, G., Lubin, B., and Seuken, S. Combinatorial auctions via machine learning-based preference elicitation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 128–136, 2018.

Chase, Z. and Prasad, S. Learning time dependent choice. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

Conitzer, V. Eliciting single-peaked preferences using comparison queries. *Journal of Artificial Intelligence Research*, 35:161–191, 2009.

Daniely, A., Sabato, S., Ben-David, S., and Shalev-Shwartz, S. Multiclass learnability and the ERM principle. *The Journal of Machine Learning Research*, 16(1):2377–2404, 2015.

Ji, Z., Mehta, R., and Telgarsky, M. Social welfare and profit maximization from revealed preferences. In *International Conference on Web and Internet Economics*, pp. 264–281. Springer, 2018.

Lahaie, S. M. and Parkes, D. C. Applying learning algorithms to preference elicitation. In *Proceedings of the Fifth ACM Conference on Electronic Commerce*, pp. 180–188. ACM, 2004.

Natarajan, B. K. On learning sets and functions. *Machine Learning*, 4(1):67–97, 1989.

Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V. *Algorithmic game theory*. Cambridge University Press, 2007.

Parkes, D. Iterative combinatorial auctions. In Cramton, P., Shoham, Y., and Steinberg, R. (eds.), *Combinatorial Auctions*, chapter 2, pp. 41–77. MIT Press, 2006.

Procaccia, A. D. and Wang, J. A lower bound for equitable cake cutting. In *Proceedings of the Eighteenth ACM Conference on Economics and Computation (EC)*, pp. 479–495, Cambridge, MA, USA, 2017.

Procaccia, A. D., Zohar, A., Peleg, Y., and Rosenschein, J. S. The learnability of voting rules. *Artificial Intelligence*, 173(12-13):1133–1149, 2009.

Roth, A., Ullman, J., and Wu, Z. S. Watch and learn: Optimizing from revealed preferences feedback. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, pp. 949–962. ACM, 2016.

Roth, A., Slivkins, A., Ullman, J., and Wu, Z. S. Multidimensional dynamic pricing for welfare maximization. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pp. 519–536. ACM, 2017.

Samuelson, P. A. A note on the pure theory of consumer's behaviour. *Economica*, 5(17):61–71, 1938.

Sandholm, T. and Boutilier, C. Preference elicitation in combinatorial auctions. In Cramton, P., Shoham, Y., and Steinberg, R. (eds.), *Combinatorial Auctions*, chapter 10, pp. 233–263. MIT Press, 2006.

Zadimoghaddam, M. and Roth, A. Efficiently learning from revealed preference. In *International Workshop on Internet and Network Economics*, pp. 114–127. Springer, 2012.

A. Omitted Proofs

Proof of Lemma 1. We first show that

$$d_N(\mathcal{H}_{n,k}) \geq n.$$

We construct a set $S \subseteq \mathcal{X}$, such that $|S| = n$ and $\mathcal{H}_{n,k}$ N-shatters S. For $j \in [n-1]$, let x^j be such that

$$x_i^j = \begin{cases} -1, & i = j \text{ or } i = n \\ -\infty, & 1 \leq i \leq k \text{ and } i \neq \min\{j,k\} \\ \infty, & \text{otherwise} \end{cases}.$$

Let x^n be such that

$$x_i^n = \begin{cases} 0, & i = n \\ -\infty, & 1 \leq i \leq k-1 \\ \infty, & \text{otherwise} \end{cases}.$$

Let $S = \{x^j\}_{j\in[n]}$. Let f_1 and f_2 be such that

$$f_1(x^j) = [k-1] \cup \max\{j,k\},$$

and

$$f_2(x^j) = f_1(x^j) \cup \{n\} \setminus \{j\}$$

for all j. Note that $f_2(x^j) = [k-1] \cup \{n\}$ for $j \in [n-1]$ and $[k-1]$ for $j = n$. For any $T \subseteq S$, we construct $g \in \mathcal{H}_{n,k}$ such that $g(x^j) = f_1(x^j) = j$ for $x^j \in T$, and $g(x^j) = f_2(x^j) = n$ for $x^j \in S \setminus T$. Let g_i denote the value of item i in the value vector that generates g. Let $g_n = 0.1$ if $x^n \in T$ and -0.1 otherwise. For each $j \in [n-1]$ such that $x^j \in T$, let $g_j = 1$. For each j such that $x^j \in S \setminus T$, let $g_j = -1$. We now check that h satisfies the above condition. For any $1 \leq j \leq n-1$, clearly $[k] \setminus \min\{j,k\}$ are among the purchased items, since they have price $-\infty$. Also, any item in $[n-1] \setminus ([k] \cup \{j\})$ cannot be purchased, since they have price ∞. If $x^j \in T$, then $g_j = 1$, and

$$g_j - x_j^j = 1 - (-1) = 2 > 1.1 = 0.1 - (-1) \geq g_n - x_n^j.$$

So the purchased set is $g(x^j) = [k] = f_1(x^j)$. If $x^h \in S \setminus T$, then $g_j = -1$, and

$$g_j - x_j^j = -1 - (-1) = 0 < 0.9 = -0.1 - (-1) \leq g_n - x_n^j.$$

So the purchased set is $g(x^j) = f_1(x^j) \cup \{n\} \setminus \{j\} = f_2(x^j)$. For x^n, if $x^n \in T$, we have

$$g_n - (x_n)^n = 0.1 - 0 = 0.1 > 0,$$

so the purchased set is $g(x^n) = [k-1] \cup \{n\} = f_1(x^n)$. If $x^n \notin T$, we have

$$g_n - (x_n)^n = -0.1 - 0 = -0.1 < 0,$$

so item n will not be purchased, and the purchased set is $g(x^n) = [k-1] = f_2(x^n)$. It follows that S can be N-shattered by $\mathcal{H}_{n,k}$.

Now we show that

$$d_N(\mathcal{H}_{n,k}) < n+1.$$

For any $S \subseteq \mathcal{X}$ where $|S| = n+1$, we show that $\mathcal{H}_{n,k}$ does not N-shatter S. Suppose not, and let f_1 and f_2 be the functions that satisfy the shattering conditions. Given f_1 and f_2, we now construct some $T \subseteq S$, such that there is no $g \in \mathcal{H}_{n,k}$ satisfying $g(x) = f_1(x)$ for $x \in T$ and $g(x) = f_2(x)$ for $x \in S \setminus T$. Suppose $S = \{x^1, \ldots, x^{n+1}\}$. We create a graph with $n+1$ vertices, undirected edges, and directed edge weights, where:

- each item $i \in [n]$ corresponds to a vertex,
- vertex $n+1$ corresponds to a dummy item, which always has value 0,

- each $x^j \in S$ corresponds to an edge (u_j, v_j) where $u_j = \min(f_1(x^j) \cup \{n+1\} \setminus f_2(x^j))$ and $v_j = \min(f_2(x^j) \cup \{n+1\} \setminus f_1(x^j))$, and
- the directed weight on the edge corresponding to x^j from u_j to v_j is $x^j_{u_j} - x^j_{v_j}$.

For notational simplicity, if the directed weight from u to v is w, we say the directed weight on the same edge from v to u is $-w$.

Observe that since there are $n+1$ vertices and $|S| = n+1$ edges in the graph, there must be a cycle. Let $(c_1, \ldots, c_\ell)$ be such a cycle, and $w_1, \ldots, w_\ell$ be the directed weights, where w_i is the weight from c_i to c_{i+1}. We construct the set T using this cycle.

First consider the case where $\sum_i w_i \neq 0$. W.l.o.g. suppose $\sum_i w_i > 0$. Let T be any subset of S satisfying:

- for j where (u_j, v_j) is in the cycle with the same direction (i.e., there exists some $i \in [\ell]$ such that $u_j = c_i$, $v_j = c_{i+1}$), $x^j \in T$, and
- for j where (u_j, v_j) is in the cycle with the opposite direction (i.e., there exists some $i \in [\ell]$ such that $v_j = c_i$, $u_j = c_{i+1}$), $x^j \notin T$.

Suppose g is a price vector such that $g(x^j) = f_1(x^j)$ if $x^j \in T$ and $g(x^j) = f_2(x^j)$ if $x^j \in S \setminus T$. Since for each $i \in [\ell]$, item c_i is preferred to item c_{i+1}, the prices must satisfy:

$$g_{c_i} - x^j_{c_i} \geq v_{c_{i+1}} - x^j_{c_{i+1}},$$

where x^j corresponds to edge (c_i, c_{i+1}) or (c_{i+1}, c_i). Given the definition of the weights on the edges, the above condition of the prices is equivalent to:

$$g_{c_i} - g_{c_{i+1}} \geq x^j_{c_i} - x^j_{c_{i+1}} = w_i.$$

Now summing over $i \in [\ell]$, we have

$$0 = \sum_{i \in [\ell]} (g_{c_i} - g_{c_{i+1}}) \geq \sum_{i \in [\ell]} w_i > 0,$$

a contradiction. In other words, there is no such g for the set T constructed.

Now consider the case where $\sum_i w_i = 0$. We consider two different constructions of T, and show that the only possible values (restricted to the cycle) that may generate the required labels have to be the same up to a constant shift. This leads to a contradiction, since given consistent tie-breaking, the same values cannot lead to different labels. Let T be the set constructed in the previous case. Consider $T_1 = T$ and $T_2 = S \setminus T$. Let g^1 and g^2 be two price vectors such that $g^i(x^j) = f_i(x^j)$ if $x^j \in T_i$, and $g^i(x^j) = f_{3-i}(x^j)$ if $x^j \in S \setminus T_i = T_{3-i}$. By the argument in the previous case, we have

$$g^1_{c_i} - g^1_{c_{i+1}} \geq w_i.$$

Since $\sum_i w_i = 0$, it must be the case that for any $i \in [\ell]$,

$$g^1_{c_i} - g^1_{c_{i+1}} = w_i.$$

On the other hand, for g^2, we have

$$g^2_{c_{i+1}} - g^2_{c_i} \geq -w_i.$$

Again, since $\sum_i w_i = 0$, it must be the case that for any $i \in [\ell]$,

$$g^2_{c_{i+1}} - g^2_{c_i} = -w_i.$$

That is,

$$g^2_{c_i} - g^2_{c_{i+1}} = w_i.$$

So, restricted to the cycle $(c_1, \ldots, c_\ell)$, up to a constant shift, g^1 and g^2 are exactly the same. Yet, they generate different labels restricted to the cycle, leading to a contradiction under consistent tie-breaking. This concludes the upper bound on $d_N(\mathcal{H}_{n,k})$. □