Online Learning and Profit Maximization from Revealed Preferences

Kareem Amin, Rachel Cummings*, Lili Dworkin, Michael Kearns, Aaron Roth†
Computer and Information Science
University of Pennsylvania
{akareem,ldworkin,mkearns,aaroth}@cis.upenn.edu; rachelc@caltech.edu

Abstract

We consider the problem of learning from revealed preferences in an online setting. In our framework, each period a consumer buys an optimal bundle of goods from a merchant according to her (linear) utility function and current prices, subject to a budget constraint. The merchant observes only the purchased goods, and seeks to adapt prices to optimize his profits. We give an efficient algorithm for the merchant's problem that consists of a learning phase in which the consumer's utility function is (perhaps partially) inferred, followed by a price optimization step. We also give an alternative online learning algorithm for the setting where prices are set exogenously, but the merchant would still like to predict the bundle that will be bought by the consumer, for purposes of inventory or supply chain management. In contrast with most prior work on the revealed preferences problem, we demonstrate that by making stronger assumptions on the form of utility functions, efficient algorithms for both learning and profit maximization are possible, even in adaptive, online settings.

1 Introduction

We consider algorithmic and learning-theoretic aspects of the classic *revealed preferences problem*. In this setting, a consumer has a fixed but unknown utility function u over n goods, and is price sensitive. At each period t, she observes a price vector p^t and purchases a bundle of (possibly fractional) goods x^t to maximize her utility given her budget (i.e. $x^t \in \arg\max_{x \cdot p^t \leq B} u(x)$). Given a sequence of T observations $(p^1, x^1), \ldots, (p^T, x^T)$, the revealed preferences problem (introduced by Samuelson (1938); see Varian (2006) for a survey) is to determine whether the observations are *consistent* with a consumer optimizing any utility function u, subject to some mild constraints (e.g. monotonicity of u). In this paper, however, we have different and stronger objectives, motivated by what power the merchant has to set prices. We consider two scenarios:

(*Price-Setting*) First, we consider a monopolist merchant who has the power to set prices as he wishes (without fear of losing the consumer to competition). In this setting, we adopt the natural goal of merchant profit maximization. The merchant has a fixed unit *cost* associated with each good, and his profit, when the consumer buys a bundle x, is his revenue minus the cost of the bundle purchased. The merchant wishes to adaptively set prices so as to minimize his costs, which in turn maximizes his profits. Every round, the consumer purchases her utility maximizing bundle subject to the merchant's prices and her budget constraint. If the merchant *knew* the consumer's utility function, he could optimally set prices, but he does not — instead, the merchant faces a *learning* problem. For the case when the consumer has a linear utility function, we give an efficient algorithm for setting prices to quickly learn the consumer's utility function and then exploit this knowledge to set profit-maximizing prices.

(*Exogenous Prices*) Second, we consider a merchant who cannot unilaterally set prices, but instead must react to a stream of exogenously chosen prices. This setting is relevant to a seller of commodity goods, or the owner of a franchise that must set prices given by the parent company. Despite his lack of control over prices, this merchant would nevertheless like to be able to *predict* which bundle the consumer is going to buy in the next period (e.g. to optimize inventory or streamline the supply chain). The problem remains that the consumer's utility function is unknown, and now the price sequence is also unknown and arbitrary (i.e. it could in the worst case be chosen adaptively, by an adversary). In this setting, when the consumer has a linear utility function, we give an efficient algorithm with a small *mistake bound* — in other words, even against an adaptively chosen set of price vectors, in the worst case over consumer utility functions, our algorithm makes only a bounded number of mistaken predictions of bundles purchased.

We note that there are a variety of scenarios that fall under the two frameworks above. These include sponsored search or contextual advertising on the web (where advertisers typically must obey periodic budget constraints, and prices are set exogenously by the bids of competitors or endogenously by an ad exchange or publisher); consumers who regularly receive gift certificates which can only be used for purchases from a single merchant such as Amazon, who in turn has price-setting powers; and crowdsourcing or labor manage-

*Research performed while the author was visiting the University of Pennsylvania. Author's current affiliation: Computing and Mathematical Sciences, California Institute of Technology.

†Research supported in part by NSF grants CCF-1101389, CNS-1065060, CNS-1253345, and a Google research grant.

ment settings where a manager (merchant) can set rewards or payments for a set of daily tasks, and workers (consumers) with a budget of time or effort select tasks according to the incentives and their own abilities.

From a learning-theoretic point of view, the price-setting framework can be viewed as a form of *query* model, since the merchant is free to experiment with prices (both for the purpose of learning about the consumer, and subsequently in order to set optimal prices); while the exogenous price model falls into the large literature on adversarial, worst-case online learning. The learning problem we consider in both settings is unusual, in that the "target function" we are trying to learn is the vector-valued $\arg\max$ (optimal bundle) of the consumer's utility function. Additionally, despite the linearity of the consumer's utility function, the merchant's reward function is a non-convex function of prices, which further complicates learning.

Our major assumptions — that consumers always spend their entire budget, that there is one divisible unit of each good available in each round, and that consumers repeatedly return to the same merchant — are standard in the revealed preferences model. In order to provide efficient learning algorithms in this setting, we necessarily impose additional restrictions on the form of the consumer's utility function. In particular, we assume that the utility function is linear, and that the coefficients are discretized and lower-bounded. The discretization assumption is necessary to learn the utility function *exactly*, which is required for the merchant's optimization. Even if two functions differ by an arbitrarily small amount, they can induce the consumer to buy very different bundles. We also assume an upper bound on prices, and without loss of generality we rescale this upper bound to be 1. Without such an assumption, the merchant could maximize his profits by setting all prices to infinity. Unbounded prices are neither found in reality, nor lead to an interesting optimization problem.

1.1 Our Results

We first consider the case of a monopolist merchant who has the ability to set prices arbitrarily, and is facing a consumer with an unknown linear utility function. In this setting, we give an algorithm with bounded regret with respect to the optimal (profit-maximizing) set of prices in hindsight. Our argument proceeds in two steps. We first show that, *if we knew* the consumer's utility function u, then we could efficiently compute the optimal profit-maximizing prices p^*:

Theorem 1 (Informal)**.** *There is an efficient algorithm* (*running in time $O(n^2 \log n)$*), *which given as input the linear consumer utility function u, outputs the profit-maximizing prices p^*.*

The analysis of this algorithm first assumes we know only the *set* of goods purchased by the consumer under optimal prices (but not the optimal prices themselves), and introduces a family of linear programs with one free parameter. We then show there is a small set of values for this parameter, one of which yields the optimal prices.

Note that although the *consumer*'s optimization problem when selecting a bundle to purchase given prices is simply a fractional knapsack problem, the problem of computing optimal prices is substantially more complex. The optimal price vector p^* is actually a subgame perfect Nash equilibrium strategy for the merchant in a two-stage extensive form game between the merchant and the consumer (the merchant first picks prices, and then the consumer best responds). Viewed in this way, the fractional knapsack problem that the consumer solves at the second stage is simply her best response function; what we give is an algorithm for computing the merchant's subgame perfect equilibrium strategy in the first stage of this game. Note that doing this in polynomial time is non-trivial, because the merchant's strategy space is continuous (and even after discretization, is exponentially large).

We next give an algorithm that learns the consumer's unknown linear utility function by making price queries.

Theorem 2 (Informal)**.** *There is an efficient algorithm that learns, after at most $O(n)$ price queries, the utility coefficients for all goods except those that are so preferred they will be bought regardless of prices.*

This algorithm has two phases. In the first phase, after setting all prices to 1 (the maximum possible price), the algorithm gradually lowers the prices of unpurchased items in succession until they are purchased, thus learning the ratio of their utility coefficient to that of the least preferred good that was purchased under the price vector of all 1s.

The harder coefficients to learn are those corresponding to goods purchased even when all prices are 1 — these are the consumer's most preferred goods. Some of these are learned by gradually lowering the prices of unpurchased goods until a switch of purchased goods occurs; for the ones that cannot be learned via this procedure, we prove that they are so favored that they will be purchased under any prices, and learning their coefficients is not necessary for price optimization.

These two algorithms combine to prove our first main result:

Theorem 3 (Informal)**.** *There is a price-setting algorithm that, when interacting with a consumer with an unknown linear utility function for T rounds, achieves regret $O(n^2/T)$ to the profit obtained by the optimal* (*profit-maximizing*) *price vector.*

In the last part of the paper, we consider the case of a commodity merchant who does not have the power to set prices. The merchant wishes to predict the bundles that a consumer with an unknown linear utility function will buy, in the face of a stream of arbitrary price vectors. Here, the main quantity of interest is how many mistakes we make (by predicting the incorrect bundle) in the worst case over both consumer utility functions and sequences of price vectors. We call this the *mistake bound* of the algorithm (by analogy to the mistake bounded model of learning). Here we prove our second main result:

Theorem 4 (Informal)**.** *There exists a polynomial time algorithm in the online exogenous price model that has a mistake bound of $O(n^2)$ with high probability.*

1.2 Related Work

The work most directly related to our results is the recent paper of Balcan et al. (2014), which was conducted independently and concurrently. They study the problem of learning from revealed preferences in various settings, including in a query model related to the model we study here. Our "price queries" differ slightly from the queries in the Balcan et al. model, in that our learner can only specify prices, whereas the learner from Balcan et al. can specify prices, as well as the consumer's budget with each query. However, the main distinction between our work and theirs is that our goal is profit maximization (even if we do not exactly learn the buyer's utility function), and the goal of Balcan et al (2014) is to exactly learn the buyer's utility function — they do not consider the profit maximization problem.

More broadly, there is a long line of work on the revealed preference problem, which was first introduced by Samuelson (1938).[1] Most previous efforts have focused on the construction of utility functions that explain a finite sequence of price/bundle observations. Afriat's Theorem (1967) is the seminal result in this field, and proves that a sequence of observations is *rationalizable* (i.e. can be explained by a utility function) if and only if the sequence is rationalizable by a piecewise linear, monotone, concave utility function. However, the hypothesis learned has description length proportional to the number of observations, and hence although it can *explain* previous observations, it usually does not generalize to *predict* the bundles purchased given new price vectors.

The problem of finding a utility function that is both consistent and predictive was first considered by Beigman and Vohra (2006), who formalize the statement that "Afriat's Theorem Learners" do not generalize. Their results essentially show that it is only possible to find predictive hypotheses if we restrict the class of allowable utility functions beyond those that are rationalizable. Roth and Zadimoghaddam (2012) extend this line of work by providing efficient learning algorithms for two specific classes of utility functions — linear and linearly separable and concave utility functions. Cummings, Echenique, and Wierman (2014) consider the revealed preferences problem when the consumer can strategically choose bundles to subvert the merchant's learning. In this setting, they show that without assuming the consumer's utility function is linearly separable, the merchant is unable to learn anything.

Like this prior work, we also seek to find predictive hypotheses for the class of linear utility functions, but we consider two new learning models: one in which prices are directly controlled, rather than observed (which corresponds to a query model of learning), and furthermore we wish to learn optimal prices; and one in which prices are chosen adversarially and adaptively, and arrive online (which corresponds to online learning in the mistake bound model).

Our results in the second model are inspired by the classic halving algorithm for the online learning setting, which is credited to Littlestone (1988). To implement the algorithm efficiently, we instead maintain a continuous hypothesis space from which we predict using a randomly sampled hypothesis (rather than predicting using a majority vote). We track the volume of the hypothesis space (rather than the number of consistent hypotheses), and show that after a bounded number of mistakes, we must have learned one coefficient of the consumer valuation function.

We also make note of a large literature on preference learning, which has similar motivation to our work, but is not directly technically relevant. This literature typically does not consider preferences which are generated in response to *prices*, so the main focus of our work (on profit maximization) does not arise in the preference learning literature. Similarly, in our exogenous prices setting, we are in a mistake bounded model, and make no distributional assumptions, which differentiates us from most of the work on preference learning.

2 Preliminaries

We consider a set of n divisible goods that a merchant wishes to sell to a consumer. We represent a *bundle* of goods $x \in [0,1]^n$ by a vector specifying what fraction of each of the n goods is purchased. The consumer has an unknown utility function $u : [0,1]^n \to \mathbb{R}$ specifying her utility for each possible bundle. The prices (one for each good) are also represented by a vector $p \in [0,1]^n$ (we normalize so that the price of every good i is $p_i \leq 1$). Written in this way, the *price of a bundle* x is simply $x \cdot p = \sum_{i=1}^{n} p_i \cdot x_i$. Finally, the consumer behaves as follows: facing a price vector p, the consumer purchases her most preferred bundle subject to a budget constraint $B \geq 0$. That is, she purchases a bundle in the set $X(u,p,B) = \arg\max_{x \cdot p \leq B} u(x)$ of utility-maximizing bundles. If $X(u,p,B)$ is a singleton set, we say that the consumer's choice is *uniquely specified* by p. We assume the budget is fixed and known to the merchant (although if the budget were unknown, the merchant could learn it from a single price query).

We restrict our attention to linear utility functions, which are defined by a valuation vector $v \in \mathbb{R}^n$ such that $u(x) = x \cdot v$. We assume the valuations vectors are discretized to some increment δ; i.e. each $v_i \in \{0, \delta, 2\delta, \ldots, 1\}$. For this family of utility functions, the consumer's optimization problem to compute $X(u,p,B)$ is a fractional knapsack problem. The capacity of the knapsack is B, and the weight and value of a good i are p_i and v_i, respectively. This problem can be solved greedily by ranking the goods in decreasing order of their v_i/p_i (i.e. bang per buck) ratios, and then buying in this order until the budget is exhausted. Note that in the optimal bundle, there will be at most one fractionally purchased good. Since this ratio is important in many of our algorithms, given u and p, we will denote v_i/p_i by $r_i(u,p)$, or by r_i when u and p are clear from context. If $r_i \geq r_j$ we say that the consumer *prefers item i to item j*.

We consider two problem variants. In the first, the merchant has the power to set prices, and has a production cost $c_i \leq 1$ for each good i. Hence, the merchant's *profit* when the consumer buys a bundle x at prices p is $x \cdot (p - c)$. It always improves the consumer's utility to saturate her

[1] For a textbook introduction to the standard model, see (Mas-Colell, Whinston, and Green 1995) and (Rubinstein 2012), and for a survey of recent work, see (Varian 2006).

budget, so $x \cdot p = B$ for any $x \in X(u,p,B)$ and $x \cdot (p-c) = B - x \cdot c$. Hence, maximizing the merchant's profit is equivalent to minimizing his costs $x \cdot c$. The merchant's goal is to obtain profit close to the maximum possible profit $\mathrm{OPT} = \max_{p \in [0,1]^n} \max_{x \in X(u,p,B)} x \cdot (p-c) = \max_{p \in [0,1]^n} \max_{x \in X(u,p,B)} B - x \cdot c$.

Note that solving this problem requires both learning something about the unknown utility function u, as well as the ability to solve the optimization problem given u. At every round t, the merchant chooses some price vector p^t, and the consumer responds by selecting any consistent $x^t \in X(u,p^t,B)$. We measure our success over T time steps with respect to our regret to the optimal profit the merchant could have obtained had he priced optimally at every round, which is defined as $\mathrm{Regret}(p^1,\ldots,p^T) = \mathrm{OPT} - \frac{1}{T}\sum_{t=1}^{T} x^t \cdot (p^t - c)$.

In the second variant, we view price vectors $p^1,\ldots,p^T$ as arriving one at a time, chosen (possibly adversarially) by Nature. In this setting, the merchant has no control over the bundle purchased by the consumer, and wishes only to predict it. At each time step t, after learning p^t, we get to predict a bundle $\hat{x}^t$. Following our prediction, we observe the bundle $x^t \in X(u,p^t,B)$ actually purchased. We say that the algorithm makes a *mistake* if $\hat{x}^t \neq x^t$, and our goal is to design an algorithm that makes a bounded number of mistakes in the worst case over both u and the sequence of prices $p^1,\ldots,p^T$.

Note: Due to space limitations, some proofs are omitted or only sketched, but are provided in full detail in the full version of this paper, which also contains pseudocode for all algorithms. The full version is available on arXiv at: http://arxiv.org/abs/1407.7294

3 Price-Setting Model

We begin by considering the first model, in which the merchant controls prices, and seeks to maximize profit. First we show that, given the coefficients v_i of the consumer's linear utility function, we can efficiently compute the profit-maximizing prices. We will then combine this algorithm with a query algorithm for learning the coefficients, thus yielding an online no-regret pricing algorithm.

3.1 Computing Optimal Prices Offline

In this section we assume that all the coefficients v_i of the consumer's utility function are known to the merchant. Even then, it is not clear *a priori* that there exists an efficient algorithm for computing a profit-maximizing price vector p. As previously mentioned, the optimal prices are a subgame perfect Nash equilibrium strategy for the merchant in a two-stage extensive form game, in which the merchant has exponentially many strategies. Straightforwardly computing this equilibrium strategy via backwards induction would therefore be inefficient. Our algorithm accomplishes the task in time only (nearly) quadratic in the number of goods.

The key to the algorithm's efficiency will stem from the observation that there exists a restricted family of pricing vectors $\mathcal{P} \subset [0,1]^n$ containing a (nearly) profit-maximizing vector p^*. This subset $\mathcal{P}$ will still be exponentially large in n, but will be "derived" (in a manner which will be made more precise) from a small set of vectors $p^{(1)},...,p^{(n)}$. This derivation will allow the algorithm to efficiently search for p^*. We define $p^{(k)}$ by letting $p_i^{(k)} = \min(v_i/v_k, 1)$. In other words, the price of every good whose value is less than the kth good is set to the ratio v_i/v_k. Otherwise, if $v_i > v_k$, the price of good i in $p^{(k)}$ is set to the ceiling of 1.

To understand the operation of the algorithm, consider the consumer's behavior under the prices $p^{(k)}$. Any good priced at v_i/v_k will have a bang per buck ratio $r_i = v_k$. Therefore, the consumer's choice is not uniquely specified by $p^{(k)}$ in general (since the consumer is indifferent between any of the previously mentioned goods). Moreover, the consumer's choice will have great impact on the merchant's profit since the goods between which the consumer is indifferent might have very different production costs c_i. The algorithm therefore proceeds by computing, for each $p^{(k)}$, which bundle $x^{(k)}$ the merchant *would like* the consumer to purchase under $p^{(k)}$. More precisely, for each k, the algorithm computes $x^{(k)} \in \arg\max_{x \in X(u,p^{(k)},B)} x \cdot (p^{(k)} - c)$. Note that if the merchant were to actually play the price vector $p^{(k)}$, the consumer would be under no obligation in our model to respond by selecting $x^{(k)}$. Therefore, the final step of the algorithm is to output a price vector which attains nearly optimal profit, but for which the consumer's behavior is uniquely specified.

The analysis proceeds by proof of three key facts. (1) For some k, the optimal profit is attained by $(p^{(k)}, x^{(k)})$, or rather, $\mathrm{OPT} = x^{(k)} \cdot (p^{(k)} - c)$ for some k. (2) Given any $p^{(k)}$, $x^{(k)}$ can be computed efficiently (in $O(n \log n)$ time). Finally, (3) there is some price $\hat{p}$ for which the consumer's choice x is uniquely specified, and where $x \cdot (\hat{p} - c)$ is close to OPT.

Theorem 1. *Algorithm* `OptPrice` (*which runs in time* $O(n^2 \log n)$)*, takes coefficients* $v_1,\ldots,v_n$ *as input and computes prices,* $\hat{p}$ *for which the consumer's choice* $\hat{x}$ *is* uniquely specified *and that for any* $\epsilon > 0$ *achieves profit* $x(\hat{p} - c) \geq \mathrm{OPT} - \epsilon$.

We prove the above theorem by establishing the three key facts listed above. The first lemma establishes that optimal profit is attained by some $(p^{(k)}, x^{(k)})$ for some k. We give a sketch of the proof.

Lemma 1. *Let* $p^{(k)}$ *be the pricing vector such that* $p_i^{(k)} = \min(v_i/v_k, 1)$*. For any consumer utility parameterized by* (u,B)*, there exists some* k *and an* $x \in X(u,p^{(k)},B)$ *such that* $\mathrm{OPT} = x \cdot (p^{(k)} - c)$.

Proof. (Sketch). Consider a profit-maximizing price p^*, and corresponding bundle $x^* \in X(u,p^*,B)$, so that $\mathrm{OPT} = x^* \cdot (p^* - c)$. Let $O = \{i : x_i^* > 0\}$ be the set of purchased goods in x^*. We note that there must exist a τ such that $r_i(u,p^*) \leq \tau$ whenever $i \notin O$ and $r_i(u,p^*) \geq \tau$ whenever $i \in O$. In other words, in order for the bundle x^* to maximize the consumer's utility, the bang for buck for every purchased good must be at least as large as the bang for buck for every unpurchased good.

Given (x^*, p^*), we write a linear program: $\max_p \sum_{i \in O} p_i$ s.t. (1) $v_i/p_i \geq \tau, \forall i \in O$, (2) $v_i/p_i \leq \tau, \forall i \notin O$, and (3) $p_i \leq 1$. We claim that any solution to this LP is also a profit-maximizing price, and that $p_i = \min(v_i/\tau, 1)$ is a solution. Finally, we argue that τ can always be taken to be v_k for some $k \in [n]$. □(Lemma 1)

Lemma 1 establishes that for some k, $p^{(k)}$ is optimal, in the sense that there exists an $x^{(k)} \in X(u, p^{(k)}, B)$ such that $\mathrm{OPT} = x^{(k)}(p^{(k)} - c)$. The algorithm proceeds by computing for each such $p^{(k)}$, the most profitable bundle (for the merchant) that the consumer (who is indifferent between all bundles in $X(u, p^{(k)}, B)$) could purchase. $X(u, p^{(k)}, B)$ is potentially a very large set. For example, if the consumer has identical values for all goods (i.e. $v_i = c$ for all i), and $p^{(k)}$ is the all-1s vector, then $X(u, p^{(k)}, B)$ contains any budget-saturating allocation. Despite the potential size of this set, Lemma 2 shows that computing $\max_{x \in X(u, p^{(k)}, B)} x \cdot (p - c)$ simply requires solving a fractional knapsack instance, this time from the merchant's perspective.

Lemma 2. *For any p, u, and B, $\max_{x \in X(u,p,B)} x \cdot (p - c) = B - x \cdot c$ can be computed in $O(n \log n)$ time.*

Proof. Let p be an arbitrary price vector. The merchant-optimal bundle that could be purchased under this price vector is $\max_{x \in X(u,p,B)} B - x \cdot c$, and can be computed as follows. Let $r_i(u, p) = v_i/p_i$. First sort the r_i in decreasing order, so that $r_{i_1} \geq ... \geq r_{i_n}$. The consumer will buy items in this order until the budget B is exhausted. Thus, we can simulate the consumer's behavior, iteratively buying items and decrementing the budget. The consumer's behavior is uniquely specified unless there is some run of items with $r_{i_j} = r_{i_{j+1}} = ... = r_{i_{j+d}}$, and B' budget remaining, where $\sum_{l=0}^{d} p_{i_{j+l}} > B'$. In other words, the consumer is indifferent between these items, and can make different selections to exhaust the remaining budget B'.

In that case, we know that for any bundle in $X(u, p, B)$, $x_{i_l} = 1$ if $l < j$, and $x_{i_l} = 0$ if $l > j + d$. For the remaining items, the merchant's profit is maximized when $x \cdot c$ is minimized. This occurs when the consumer saturates the remaining budget B' while minimizing the cost c to the merchant. This is an instance of min-cost knapsack wherein the size of the items are $p_{i_j}, ..., p_{i_{j+d}}$ and the cost of the items are $c_{i_j}, ..., c_{i_{j+d}}$. A solution to this problem can be computed greedily. Thus the most profitable bundle for p can be computed with at most two sorts (first for r_i then for p_i/c_i). □(Lemma 2)

Finally, to induce the consumer to buy the bundle x^*, rather than another member of $X(u, p^*, B)$, we perturb the vector p^* slightly, and show that this has an arbitrarily small effect on profit.

Lemma 3. *For any $\epsilon > 0$, there exists a price vector $\hat{p}$ that uniquely specifies a bundle $\hat{x}$ that satisfies $\hat{x} \cdot (\hat{p} - c) \geq \mathrm{OPT} - \epsilon$.*

Proof. (Sketch). Recall that the merchant would like the consumer to purchase the bundle x^*, which is a member of the set $X(u, p^*, B)$. Even if the merchant sets prices at p^*, there is no guarantee that the consumer will purchase x^* rather than some other bundle in the set. Our goal is to compute a vector $\hat{p}$ that is a slight perturbation of p^* and will induce the consumer will purchase some bundle $\hat{x}$ arbitrarily close to x^*. For any good i such that $x_i^* = 0$ (a good that consumer should not buy at all), we simply set $\hat{p}_i = 1$. For any good i such that $x_i^* = 1$ (a good that that the consumer should buy in its entirety), we set $\hat{p}_i = p_i^* - \epsilon_0$. Finally, for any good i such that $0 < x_i^* < 1$ (a good that the consumer should buy fractionally), we set $\hat{p}_i = p_i^* - \delta\epsilon_0$. It can be shown that these perturbations ensure that the consumer will buy goods in the order desired by the merchant.

We have decreased each price by at most ϵ_0, so the consumer might have up to an additional $n\epsilon_0$ budget to spend. Recall that prices are chosen by the algorithm to be $p_i = \min(v_i/v_k, 1)$ for each i and for some k. Because values are discretized and lower-bounded, the minimum price possible is δ. Consider setting $\epsilon_0 = \delta\epsilon/n$, which yields at most $\delta\epsilon$ additional budget. Then the consumer can afford to purchase at most an additional $\delta\epsilon/\delta = \epsilon$ fraction of a good. In the worst case, if this good is of maximum cost 1, the merchant will incur an additional cost of ϵ. □(Lemma 3)

3.2 Learning Consumer Valuations

We now provide a query algorithm for learning the coefficients v_i of the consumer's utility function. For the analysis only, we assume without loss of generality that the goods are ordered by decreasing value, i.e. $v_1 > \ldots > v_n$. Our algorithm can learn the values in some suffix of this unknown ordering; the values that cannot be learned are irrelevant for setting optimal prices, since those goods will always be purchased by the consumer.

Theorem 2. *Algorithm* `LearnVal`*, given the ability to set prices, after at most $O(n \log((1 - \delta)/\delta^2))$ price queries (where δ is the discretization of values), outputs the ratio v_i/v_n for all goods i, except those that will be bought under all price vectors.*

Proof. Algorithm `LearnVal` proceeds as follows. On each day we choose a particular price vector, observe the bundle purchased at those prices, and then use this information as part of the learning process. First, we set $p_i = 1$ for all i and observe the bundle x bought by the consumer. Let j be the index of the least-preferred good that the consumer purchases under this price vector. If the consumer buys some good i fractionally (which the algorithm can observe), then $j = i$. Otherwise, to learn j, we incrementally lower the price of some good k that the consumer did not purchase, until k is purchased instead of another good i. Then we have learned $j = i$.

In the next phase of this algorithm, we learn the ratio v_k/v_j for all goods $k > j$ that were not originally purchased. To do so, we lower p_k (while keeping all other prices at 1) until item k is purchased. This will occur when $v_k/p_k = v_j/p_j$, or $v_k/v_j = p_k$. Recall that we assume each v_i is discretized to a multiple of δ. Therefore to guarantee that we learn the ratio v_k/v_j *exactly*, we must learn the ratio up to a precision of $\min_{k \neq k'} |(v_k - v'_k)|/v_j$. This quantity is minimized at $v_k = v'_k + \delta$ and $v_j = 1$ (because $v_j \leq 1$), so it is sufficient to learn the ratio to within $\pm\delta$. Thus if we perform a discrete binary search on p_k, it will require $O(n\log(1/\delta))$ steps to exactly identify the desired ratio. Finally, we renormalize the ratios we have learned in terms of v_n. That is, for all $k \geq j$, we define $s_k = v_k/v_n = (v_k/v_j)/(v_n/v_j)$.

We next attempt to learn $s_k = v_k/v_n$ for all $k < j$. These are the most preferred goods that were originally purchased under the all-1s price vector. We learn s_k inductively in decreasing order of k, so as we learn s_k, we already know the value s_i for all $i > k$. The goal is to now set prices so that the consumer will be indifferent to all goods $i > k$ (i.e. will have a tie in the bang per buck for all these goods). The bang per buck on these goods is initially set low, and gradually raised by adjusting prices. At some critical point, a switch in the behavior of the consumer will be observed in which k is no longer purchased, and s_k is learned.

We therefore introduce a parameter α, which controls the bang per buck ratio of goods $k+1,\ldots,n$. Define $p(\alpha,k)$ to be the price vector where $p_i = 1$ for $i \leq k$, and $p_i = \alpha s_i$ for $i > k$. Let $r(\alpha,k)$ denote the corresponding bang per buck vector; i.e. $r(\alpha,k)_i = v_i/p_i$. It is easy to see that $r(\alpha,k) = (v_1,\ldots,v_k,v_n/\alpha,\ldots,v_n/\alpha)$. Thus, by lowering α, we lower the prices and raise the desirability of goods $k+1,\ldots,n$. This process is illustrated in Figure 1, whereby gradually lowering α from 1, we eventually reach a first point where goods $k+1,...,n$ are preferred to good k. This switch occurs when the bang per buck ratio of goods $k+1,\ldots,n$ equals the bang per buck ratio of k, i.e. $v_n/\alpha = v_k$, or $\alpha = v_n/v_k$. Our goal will be to identify this value of α, which we denote $\alpha^*(k)$. Once we know $\alpha^*(k)$, we will have learned s_k, since $s_k = v_k/v_n = 1/\alpha^*(k)$.

We know that we have found $\alpha^*(k)$ when we identify the highest value of α for which some good goes from being purchased to unpurchased. This good must be k, because as we decrease α, we increase the desirability of goods $k+1,\ldots,n$, so none of these goods will go from being purchased to unpurchased. Of goods $1,\ldots,k$, good k is the least preferred, so it will be the first to become unpurchased.

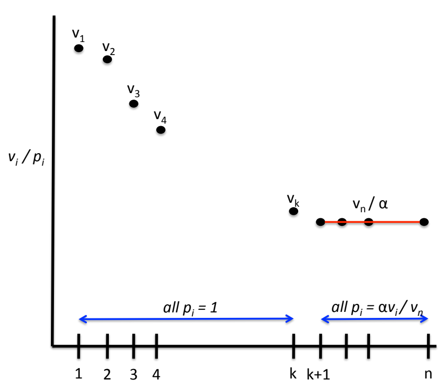

Figure 1: An illustration of the process by which we search over α. The figure depicts the situation just before hitting the critical point $\alpha^*(k)$. By lowering α slightly, we raise the bang per buck v_n/α of all goods $k+1,\ldots,n$ to be slightly larger than v_k, and the consumer now prefers these goods to good k.

To learn the value of $\alpha^*(k) = v_n/v_k$ exactly, we must have a precision of $\min_{k\neq k}|v_n(|1/v_k - 1/v'_k|)$. This quantity is minimized at $v_n = \delta$ (because we assume a lower bound on values, and therefore $v_n \geq \delta$), $v_k = 1$, and $v'_k = 1-\delta$, and the corresponding value is $\delta^2/(1-\delta)$. Thus, we should search for the critical $\alpha^*(k)$ over the interval $[0,1]$ in increments of this size. In the algorithm's implementation, we can perform a binary rather than linear search over the range of α, which therefore requires requires $O(n\log((1-\delta)/\delta^2))$ steps. Once we have identified the value of $\alpha^*(k)$ to within $\delta^2/1-\delta$, we set $s_k = 1/\alpha^*(k)$.

In this manner, we can inductively find the next ratio s_{k-1} by equalizing preferences for the later goods and searching for the critical $\alpha^*(k-1)$. The only problem that might arise is that good k is sufficiently valued, and the budget sufficiently large, that good k is always purchased no matter how low α is set. Lemma 4 shows that if this occurs for some k, then no matter how we price goods $1,\ldots,k$, the consumer will always purchase these goods in full. Thus, it is unnecessary to learn the values of these goods, since the merchant can always set the prices to the maximum of 1.

Lemma 4. *If goods $1,\ldots,k$ are purchased at price vector $p(\alpha',k)$, where $\alpha' = \alpha^*(k)-\delta^2/(1-\delta)$, then goods $1,\ldots,k$ will be purchased at any price vector.*

The number of queries made is $O(n\log((1-\delta)/\delta^2))$, which is linear in the description length of values. □(Theorem 2)

3.3 Putting It All Together

We now combine the previous two sections into a complete online, no-regret algorithm. The informal description of Algorithm `ProfitMax` is as follows. First we use Algorithm `LearnVal` to learn all possible s_i ratios. For any good i for which we did not learn s_i, we set p_i at 1. Finally, we apply Algorithm `OptPrice` to the subset of remaining goods, using the s_i ratios as input. The following main result shows that this approach achieves no-regret.

Theorem 3. *For any $\epsilon > 0$, after T rounds, Algorithm* `ProfitMax` *achieves per-round regret $O((n^2/T)\log((1-\delta)/\delta^2))$ to the profit obtained by the optimal price vector (where ϵ is the additive approximation to the optimal profit and δ is the discretization of values).*

Proof. First we show that this algorithm correctly composes Algorithms `LearnVal` and `OptPrice` and indeed generates an approximately optimal price vector p.

Lemma 5. *An approximately optimal pricing for goods $\{1,\ldots,n\}$ is obtained by setting $p_i = 1$ for all goods for which Algorithm* `LearnVal` *could not learn s_i, and then applying Algorithm* `OptPrice` *to the s_i ratios of the remaining goods.*

As the lemma shows, every time we price according to p, we receive approximately optimal profit. In particular, according to Theorem 1, our profit is at least OPT - ϵ, and so our regret at most ϵ. Furthermore, Algorithm `LearnVal` uses at most $O(n\log((1-\delta)/\delta^2))$ price queries, so there are $O(n \log((1-\delta)/\delta^2))$ days on which we might incur maximum regret. On any given day, the maximum possible profit is B, while the minimum possible is $B-n$, yielding a maximum regret of n. Thus, our overall per-step regret is bounded by $O((n^2/T)\log((1-\delta)/\delta^2)+\epsilon)$. Setting $\epsilon = (n^2/T)\log((1-\delta)/\delta^2)$ yields the bound in the theorem statement. □(Theorem 3)

4 Exogenous Price Model

We now consider our second model, in which an arbitrary and possibly adversarially selected price vector arrives every day, and the goal of our algorithm is to predict the bundle purchased by the consumer. Recall that the motivating scenario for such a setting is a merchant who is forced to set prices according to the market or the choices of a parent company. At each day t, the algorithm observes a price vector p^t and makes a prediction $\hat{x}^t$. The algorithm then learns the bundle purchased, $x^t \in X(u, p^t, B)$, and is said to *make a mistake* if $x^t \neq \hat{x}^t$. Our goal is to prove an upper bound on the total number of mistakes ever made by our algorithm, in the worst case over price vectors and utility functions. We call such an upper bound a *mistake bound*.

The algorithm maintains the set of valuation vectors v consistent with the observations seen thus far: initially this feasible set is simply $C_0 = [0,1]^n$. At every round t, the observed pair (p^t, x^t) provides a set of linear constraints which we add to further constrain our feasible set C_t. In particular, we know that for any pair of goods $i, j \in [n]$ where $x_i^t > x_j^t$, it must be that $v_i/p_i^t \geq v_j/p_j^t$ (Zadimoghaddam and Roth 2012). This follows immediately from the fact that the vector x^t is the solution to a fractional knapsack problem, for which the optimal algorithm is to buy goods in decreasing order of v_i/p_i^t.

The set of all such constraints learned so far at time t forms the feasible set C_t, which is a convex polytope. The idea of the algorithm is to sample a hypothesis valuation function v^t uniformly at random from C_t at each stage, and predict using the bundle that would be purchased if the buyer had valuation function v^t. The event that this hypothesis makes a mistake in its prediction is exactly the event that the hypothesis is eliminated when we update the consistent set to C_{t+1}, and so the probability of making a mistake is exactly equal to the fraction of volume eliminated from our consistent set, which allows us to charge our mistakes to the final volume of our consistent set. However, we need a way to lower bound the final volume of our consistent set.

We define the *width* of a polytope K in dimension i as $\mathrm{width}_i(K) = \max_{x,y\in K} |x_i - y_i|$. Note that the width can be efficiently computed using a linear program. We take advantage of the fact that the true valuation function takes values that are discretized to multiples of δ. Hence, if at any point the width of our consistent set C_t in some dimension i is less than $\delta/2$, then we can exactly identify the i^{th} coefficient of the consumer's valuation function. Note that if $\mathrm{Vol}(C_t) < \delta^n$, then there must be some dimension in which $\mathrm{width}_i(C_t) < \delta/2$. When we detect such a dimension, we fix that coefficient, and restart the algorithm by maintaining a consistent set in one fewer dimension.

Hence, at each *epoch*, we maintain a set of consistent valuation functions restricted to those indices that are not yet fixed, and predict according to the composite of the valuation vectors sampled from this consistent set, together with the fixed indices. Every time we fix an index, a new epoch begins. The volume of the consistent set can never go below δ^n within an epoch, and since we fix an index at the end of every epoch, there can be at most n such epochs.

The only computationally challenging step is sampling a point uniformly at random from the consistent set C_t, which can be done in polynomial time using the technique of (Dyer, Frieze, and Kannan 1991). We thus obtain:

Theorem 4. *Algorithm* `ExogLearnVal` *runs in polynomial time per round, and with probability* $1-\beta$, *makes at most* $O\left(n^2 \log(1/\delta) + n\sqrt{\log(1/\beta)\log(1/\delta)}\right)$ *mistakes over any sequence of adaptively chosen price vectors.*

References

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

Balcan, M.; Daniely, A.; Mehta, R.; Urner, R.; and Vazirani, V. V. 2014. Learning economic parameters from revealed preferences. *arXiv preprint 1407.7937*.

Beigman, E., and Vohra, R. 2006. Learning from revealed preference. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, EC '06, 36–42.

Cummings, R.; Echenique, F.; and Wierman, A. 2014. The empirical implications of privacy-aware choice. In *Proceedings of the Fifteenth ACM Conference on Economics and Computation*, EC '14, 969.

Dyer, M.; Frieze, A.; and Kannan, R. 1991. A random polynomial-time algorithm for approximating the volume of convex bodies. *Journal of the ACM (JACM)* 38(1):1–17.

Littlestone, N. 1988. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning* 2(4):285–318.

Mas-Colell, A.; Whinston, M. D.; and Green, J. R. 1995. *Microeconomic Theory*. Oxford University Press.

Rubinstein, A. 2012. *Lecture notes in microeconomic theory: the economic agent*. Princeton University Press.

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

Varian, H. R. 2006. Revealed preference. In Szenberg, M.; Ramrattan, L.; and Gottesman, A. A., eds., *Samuelsonian economics and the twenty-first century*. Oxford University Press. 99–115.

Zadimoghaddam, M., and Roth, A. 2012. Efficiently learning from revealed preference. In *Proceedings of the 8th International Conference on Internet and Network Economics*, WINE '12, 114–127.

Supplemental Material

1 Pseudocode for Algorithms

Algorithm OptPrice(B, v, c, ϵ)

---

for $k = 1$ to n **do**
    $p_i^{(k)} = \min(v_i/v_k, 1)$ for all i
    $x^{(k)} = \arg\max_{x \in X(u, p^{(k)}, B)} x \cdot (p^{(k)} - c)$ $\triangleright$ $O(n \log n)$ computation
    $\text{Profit}(k) = x^{(k)} \cdot (p^{(k)} - c)$
$k_{\max} = \arg\max_k \text{Profit}(k)$
$p^* = p^{(k_{\max})}$, $x^* = x^{(k_{\max})}$
for $i = 1$ to n **do**
    if $x_i^* = 0$ **then**
        $\hat{p}_i = 1$
    else if $x_i^* = 1$ **then**
        $\hat{p}_i = p_i^* - \epsilon$
    else
        $\hat{p}_i = p_i^* - \epsilon/2$
return $\hat{p}$

Algorithm LearnVal(δ)

$p_i = 1 \quad \forall i$ ▷ First price query
$x \leftarrow \texttt{consumer}^1(p)$
while $\neg\exists i$ such that $0 < x_i < 1$ **do** ▷ Find some fractionally bought good
  Choose k such that $x_k = 0$
  $p_k = p_k - \delta$
  $x \leftarrow \texttt{consumer}(p)$
$j = i$ ▷ j is least-preferred purchased good
for $k = j + 1$ to n **do** ▷ Learn s_k for $j + 1, \ldots, n$
  while $x_k = 0$ **do**
    $p_k = p_k - \delta$
    $x \leftarrow \texttt{consumer}(p)$
  $v_k/v_j = p_k$
$s_k = (v_k/v_j)/(v_n/v_j) \quad \forall k \geq j$ ▷ Renormalize ratios
for $k = j - 1$ to 1 **do** ▷ Learn s_k for $j - 1, \ldots, 1$
  for $\alpha = 1$ to δ (in increments of δ) **do**
    $p_i = 1 \quad i \leq k$
    $p_i = \alpha s_i \quad \forall i > k$
    $x \leftarrow \texttt{consumer}(p)$
    if $x_k > 0$ **then**
      $s_k = 1/\alpha$
      break
  if s_k is undefined **then** ▷ k was always bought
    break
return s

[1]The notation $x \leftarrow \texttt{consumer}(p)$ specifies the bundle bought by the consumer at prices p.

Algorithm ProfitMax(B, c, δ, ϵ)

```
Profit = 0
s ← LearnVal(δ)                              ▷ Learn s_i ratios
for i = 1 to n do
    if s_i = 0 then
        p_i = 1
v = {s_i | s_i ≥ 0}
p ← OptPrice(B, v, c, ε)                     ▷ Compute optimal prices
while t ≤ T do                               ▷ Set prices optimally
    x^t ← consumer(p)
    Profit = Profit + x^t · (p − c)
```

Algorithm ExogLearnVal(B, δ)

```
Fixed = ∅, w_i = 0 for all i.                          ▷ Initialize the fixed coordinates, initially none
C_0 = {z ∈ [0,1]^n | 0 ≤ z_i ≤ 1 ∀i}                  ▷ Initialize the set of consistent hypotheses
for t = 0 to T do
    Observe p^t
    z^t ← sample¹(C_t)                                  ▷ Sample a valuation uniformly from consistent set
    v^t = (z^t_{−Fixed}, w_Fixed)                       ▷ Combine sampled and fixed coefficients
    Predict x̂^t ∈ arg max_{x·p^t ≤ B} x · v^t           ▷ Predict according to sampled valuation
    x_t ← consumer(p^t)
    C_{t+1} = C_t
    for i, j ∉ Fixed | x^t_i > x^t_j do                 ▷ Update Constraints
        C_{t+1} = C_{t+1} ∩ {z_i/p^t_i ≥ z_j/p^t_j}
    for i ∉ Fixed, j ∈ Fixed | x^t_i > x^t_j do         ▷ Update Constraints
        C_{t+1} = C_{t+1} ∩ {z_i/p^t_i ≥ w_j/p^t_j}
    for j ∉ Fixed, i ∈ Fixed | x^t_i > x^t_j do         ▷ Update Constraints
        C_{t+1} = C_{t+1} ∩ {w_i/p^t_i ≥ z_j/p^t_j}
    if There exists i ∉ Fixed such that width_i(C_{t+1}) < δ/2 then        ▷ Start a new Epoch
        for Each i ∉ Fixed such that width_i(C_{t+1}) < δ/2 do              ▷ Fix each determined coordinate
            Fix(i)
        C_{t+1} = {z ∈ [0,1]^{n−|Fixed|} | 0 ≤ z_i ≤ 1 ∀i ∉ Fixed}          ▷ Re-initialize in the unfixed coordinates

procedure Fix(i)
    Fixed = Fixed ∪ {i}
    z ← sample(C_{t+1})                                 ▷ Sample a value for coordinate i
    w_i = round²(z_i/δ) · δ                             ▷ Round sampled value to nearest discrete value and fix it
```

[1]`sample` is any polynomial algorithm for sampling uniformly at random from a convex polytope.
[2]`round`(x) rounds to the nearest integer.

2 Proofs

Lemma 1. *Let $p^{(k)}$ be the pricing vector such that $p_i^{(k)} = \min(v_i/v_k, 1)$. For any consumer utility parameterized by (u, B), there exists a k, and an $x \in X(u, p^{(k)}, B)$, such that* $\mathrm{OPT} = x \cdot (p^{(k)} - c)$.

Proof. Fix u and B. Consider a profit-maximizing price p^*, and corresponding bundle $x^* \in X(u, p^*, B)$, so that $\mathrm{OPT} = x^* \cdot (p^* - c)$. Let $O = \{i : x_i^* > 0\}$ be the set of purchased goods in x^*. If there is a fractionally purchased good in x^* ($0 < x_f^* < 1$), we denote its index by f.

We note that there must exist a τ such that $r_i(u, p^*) \le \tau$ whenever $i \notin O$ and $r_i(u, p^*) \ge \tau$ whenever $i \in O$. In other words, in order for the bundle x^* to maximize the consumer's utility, the bang-for-buck for every purchased good must be at least as large as the bang-for-buck for every unpurchased good. If there is a fractional good in x^*, we take τ to be $r_f(u, p^*)$. Otherwise, we take it to be $\max_{i \notin O} r_i(u, p^*)$, the most desirable, unpurchased, good.

Given (x^*, p^*), we can write the following linear program. We claim that any solution to this LP is also a profit-maximizing price. More precisely, if $p^{(LP)}$ is a solution to the LP, there is an $x^{(LP)} \in X(u, p^{(LP)}, B)$ such that $\mathrm{OPT} = x^{(LP)}(p^{(LP)} - c)$. In what follows we prove this claim.

$$
\begin{array}{lll}
\max & \sum_{i \in O} p_i & \\
\text{s.t.} & v_i/p_i \ge \tau & \forall i \in O \\
 & v_i/p_i \le \tau & \forall i \notin O \\
 & p_i \le 1 & \forall i \in [n]
\end{array}
$$

We can straightforwardly characterize any solution $p^{(LP)}$ to the LP. Note that the constraints on p_i are disjoint, and therefore for each $i \in O$, p_i can increased separately until a constraint is saturated. Thus, the LP is optimized by setting $p_i^{(LP)} = \min\{v_i/\tau, 1\}$ for each $i \in O$, and $p_i^{(LP)} \ge v_i/\tau$ for each $i \notin O$ (which is always possible since p^* is a feasible solution).

The LP constraints imply that, under $p^{(LP)}$, the consumer (weakly) prefers any item in O to any item not in O. Moreover, if there is a fractional good, the definition of τ ensures that any item in $O \setminus \{f\}$ is preferred to f. Finally, we know that $\sum_{i \in O} p_i^* \le \sum_{i \in O} p_i^{(LP)}$ since p^* is a feasible solution to the LP. This, along with the previous preference ordering, tells us that the consumer can saturate her budget at least as quickly under $p^{(LP)}$ as under p^*. In other words, under $p^{(LP)}$, the consumer might saturate her budget before purchasing all items in $O \setminus \{f\}$, or might require a smaller allocation of item f.

More precisely, there exists an $x^{(LP)} \in X(u, p^{(LP)}, B)$ such that point-wise $x_i^{(LP)} \le x_i^*$. Thus, $\mathrm{OPT} = x^*(p^* - c) = B - x^* \cdot c \le B - x^{(LP)} \cdot c = x^{(LP)} \cdot (p^{(LP)} - c)$, and so $p^{(LP)}$ must be a profit-maximizing price. We have established that $p^{(LP)}$, which sets $p_i = \min(v_i/\tau, 1)$, is profit-maximizing. All that's left to show is that we can take $\tau = v_k$ for some k. Consider again a profit-maximizing price and bundle (p^*, x^*). Notice that for any $i \notin O$, we can modify $p_i^* = 1$ and still have (p^*, x^*) be profit-maximizing. That is, we are only making the unpurchased goods more undesirable. Notice that with this modification $\max_{i \notin O} r_i(u, p^*) = \max_{i \notin O} v_i = v_{k^*}$ for some k^*. Next consider modifying the price of the fractional good. The price of f can increased to v_f/v_{k^*} (if $v_f/v_{k^*} < 1$) and 1 otherwise. This increases the price of f while still keeping it as the fractionally purchased item, thus reducing the merchant's cost. This will result in $r_f = v_{k^*}$ or $r_f = v_f$. In either case, there exists a (p^*, x^*) such that $\tau = v_k$ for some k. Thus $p^{(LP)}$ derived from (p^*, x^*) sets $p_i = \min(v_i/v_k, 1)$ for some v_k. □(Lemma 1)

Lemma 2. *For any p, u, B, $\max_{x \in X(u,p,B)} x \cdot (p - c) = B - x \cdot c$ can be computed in $O(n \log n)$ time.*

Proof. Let p be an arbitrary price vector. The merchant-optimal bundle that could be purchased under this price vector is $\max_{x \in X(u,p,B)} B - x \cdot c$, and can be computed as follows. Let $r_i(u, p) = v_i/p_i$. First sort the r_i in decreasing order, so that $r_{i_1} \ge ... \ge r_{i_n}$. The consumer will buy items in this order until the budget B is exhausted. Thus, we can simulate the consumer's behavior, iteratively buying items and decrementing the budget. The consumer's behavior is uniquely specified unless there is some run of items with $r_{i_j} = r_{i_{j+1}} = ... = r_{i_{j+d}}$, and B' budget remaining, where $\sum_{l=0}^{d} p_{i_{j+l}} > B'$. In other words, the consumer is indifferent between these items, and can make different selections to exhaust the remaining budget B'.

In that case, we know that for any bundle in $X(u, p, B)$, $x_{i_l} = 1$ if $l < j$, and $x_{i_l} = 0$ if $l > j + d$. For the remaining items, the merchant's profit is maximized when $x \cdot c$ is minimized. This occurs when the consumer saturates the remaining budget B' while minimizing the cost c to the merchant. This is an instance of min-cost knapsack wherein the size of the items are $p_{i_j}, ..., p_{i_{j+d}}$ and the cost of the items are $c_{i_j}, ..., c_{i_{j+d}}$. A solution to this problem can be computed greedily. Thus the most profitable bundle for p can be computed with at most two sorts (first for r_i then for p_i/c_i). □(Lemma 2)

Lemma 3. *There exists a price vector $\hat{p}$ which uniquely specifies a bundle $\hat{x}$ such that for any $\epsilon > 0$, $\hat{x} \cdot (\hat{p} - c) \geq \text{OPT} - \epsilon$.*

Proof. Recall that the merchant would like the consumer to purchase the bundle x^*, which is a member of the set $X(u, p^*, B)$. Even if the merchant sets prices at p^*, there is no guarantee that the consumer will purchase x^* rather than some other bundle in the set. Our goal is to compute a vector $\hat{p}$ that is a slight perturbation of p^* and will induce the consumer will purchase some bundle $\hat{x}$ arbitrarily close to x^*. For any good i such that $x_i^* = 0$ (a good that consumer should not buy at all), we simply set $\hat{p}_i = 1$. For any good i such that $x_i^* = 1$ (a good that that the consumer should buy in its entirety), we set $\hat{p}_i = p_i^* - \epsilon_0$. Finally, for any good i such that $0 < x_i^* < 1$ (a good that the consumer should buy fractionally), we set $\hat{p}_i = p^* - \epsilon_0/2$. These perturbations ensure that the consumer will buy goods in the order desired by the merchant.

We have decreased each price by at most ϵ_0, so the consumer might have up to an additional $n\epsilon_0$ budget to spend. Recall that prices are chosen by the algorithm to be $p_i = \min(v_i/v_k, 1)$ for each i and for some k. Because values are discretized and lower-bounded, the minimum price possible is δ. Consider setting $\epsilon_0 = \delta\epsilon/n$, which yields at most $\delta\epsilon$ additional budget. Then the consumer can afford to purchase at most an additional $\delta\epsilon/\delta = \epsilon$ fraction of a good. In the worst case, if this good is of maximum cost 1, the merchant will incur an additional cost of ϵ. □(Lemma 3)

Lemma 4. *If goods $1, \ldots, k$ are purchased at price vector $p(\alpha', k)$, where $\alpha' = \alpha^*(k) - \delta^2/(1-\delta)$, then goods $1, \ldots, k$ will be purchased at any price vector.*

Proof. Consider the corresponding bang per buck vector $r(\alpha', k)$. For $i \leq k$, $r(\alpha', k)_i = v_i \geq v_k$, and for $i < k$, $r(\alpha', k)_i > v_k$. Thus, good k has the lowest bang per buck of all goods. Since good k is still purchased, all other goods $i \neq k$ must be purchased as well. Consider raising the price of some good j from its price under $p(\alpha', k)$, so that good j becomes less desirable. It must be that $j > k$, because all other prices are already maximized at 1, so this does not change the fact that goods $1, \ldots, k$ will be purchased. Next consider lowering the price of some good j. This simply frees up more of the consumer's budget and allows the consumer more purchasing power, so all goods will remain purchased. □(Lemma 4)

Lemma 5. *An approximately optimal pricing for goods $\{1, \ldots, n\}$ is obtained by setting $p_i = 1$ for all goods for which Algorithm* `LearnVal` *could not learn s_i, and then applying Algorithm* `OptPrice` *to the s_i ratios of the remaining goods.*

Proof. According to Lemma 4, for any good i for which Algorithm `LearnVal` could not learn s_i, the consumer will always purchase i, regardless of p_i. Thus, our profit is maximized by setting $p_i = 1$. Furthermore, recall that the bundle bought by the consumer is invariant to scaling, and so the consumer will buy the same bundle regardless of whether we price according to the $s_i = v_i/v_n$ ratios or the actual v_i values. Because our profit depends only on the bundle bought, it is therefore sufficient to use Algorithm `OptPrice` to price approximately optimally for the s_i ratios. □(Lemma 5)

Theorem 4. *Algorithm* `ExogLearnVal` *runs in polynomial time per round, and with probability $1-\beta$ makes at most $O\left(n^2 \log(1/\delta) + n\sqrt{\log(1/\beta)\log(1/\delta)}\right)$ mistakes over any sequence of adaptively chosen price vectors.*

Proof. We prove a bound on the number of mistakes made by the algorithm in a single epoch (recall that each epoch ends when a new coordinate is fixed). Since there are only n coordinates, and hence at most n epochs, our final mistake bound is at most n times the mistake bound per epoch, giving the theorem.

Consider an epoch i in which there remain $d \le n$ unfixed coordinates. We will track the d-dimensional volume of the sets C_t during this epoch. Let S_i be the first stage of the epoch, and let F_i be the final stage of the epoch. Note that $\mathrm{Vol}(C_{S_i}) = 1$ (because C_{S_i} is always initialized to be the d-dimensional hypercube). Note also that for $S_i \le t < F_i$, any hypothesis in C_t that leads to an incorrect prediction of x^t is eliminated from C_{t+1}. Hence, if M_t is the indicator random variable specifying whether our algorithm makes a mistake at round t, it is also the indicator random variable specifying whether our algorithm sampled a hypothesis that will be eliminated at the next round. Because we sample our hypothesis at round t at random from C_t, we have:

$$\mathrm{E}[M_t] = 1 - \frac{\mathrm{Vol}(C_{t+1})}{\mathrm{Vol}(C_t)}$$

Note that we can write the volume of C_{F_i} as a telescoping product:

$$\mathrm{Vol}(C_{F_i}) = \mathrm{Vol}(C_{S_i}) \prod_{t=S_i}^{F_i-1} \frac{\mathrm{Vol}(C_{t+1})}{\mathrm{Vol}(C_t)} = \prod_{t=S_i}^{F_i-1} \frac{\mathrm{Vol}(C_{t+1})}{\mathrm{Vol}(C_t)}$$

Note also that before the end of an epoch, the width of C_t in every coordinate is at least $\delta/2$. Hence we know $\mathrm{Vol}(C_{F_i}) \ge \delta^d \ge \delta^n$. Combining these facts, we can write:

$$\delta^n \le \prod_{t=S_i}^{F_i-1} \frac{\mathrm{Vol}(C_{t+1})}{\mathrm{Vol}(C_t)} = \prod_{t=S_i}^{F_i-1} (1 - \mathrm{E}[M_t]) \le \prod_{t=S_i}^{F_i-1} \exp(-\mathrm{E}[M_t]) = \exp\left(-\mathrm{E}\left[\sum_{t=S_i}^{F_i-1} M_t\right]\right)$$

Solving for the expected number of mistakes made in a single epoch, we find that:

$$\mathrm{E}\left[\sum_{t=S_i}^{F_i} M_t\right] \le \mathrm{E}\left[\sum_{t=S_i}^{F_i-1} M_t\right] + 1 \le 1 + n\ln\left(\frac{1}{\delta}\right) = O\left(n\ln\left(\frac{1}{\delta}\right)\right)$$

Now consider the expected number of mistakes made by our algorithm for its entire run, $t \in \{1, \ldots, T\}$. Since we can partition each time step into one of at most n epochs, and have a bound on the expected number of mistakes in each epoch, by linearity of expectation, we have:

$$\mathrm{E}\left[\sum_{t=1}^{T} M_t\right] = \sum_{i=1}^{n} \mathrm{E}\left[\sum_{t=S_i}^{F_i} M_t\right] \le n + n^2\ln\left(\frac{1}{\delta}\right) = O\left(n^2\ln\left(\frac{1}{\delta}\right)\right)$$

Note that each of the random variables M_i is independent and bounded in $[0,1]$. We can therefore apply a multiplicative Chernoff bound. For any $\epsilon < 1$, we have:

$$\Pr\left[\sum_{t=1}^{T} M_t \ge (1+\epsilon)\mathrm{E}\left[\sum_{t=1}^{T} M_t\right]\right] \le \exp\left(\frac{-\epsilon^2}{3}\mathrm{E}\left[\sum_{t=1}^{T} M_t\right]\right)$$

Setting the right hand side to be at most β, plugging in our bound on $\mathrm{E}\left[\sum_{t=1}^{T} M_t\right]$ and solving for ϵ allows us to take

$$\epsilon = O\left(\sqrt{\frac{\ln(1/\beta)}{n^2\ln(1/\delta)}}\right)$$

Plugging this into the Chernoff bound proves the theorem.

□(Theorem 4)