Feature-Based Dynamic Pricing

Maxime C. Cohen
NYU Stern School of Business, New York, NY 10012, maxime.cohen@stern.nyu.edu
Ilan Lobel
NYU Stern School of Business, New York, NY 10012, ilobel@stern.nyu.edu
Renato Paes Leme
Google Research, New York, NY 10011, renatoppl@google.com

We consider the problem faced by a firm that receives highly differentiated products in an online fashion. The firms needs to price these products to sell them to its customer base. Products are described by vectors of features and the market value of each product is linear in the values of the features. The firm does not initially know the values of the different features, but can learn the values of the features based on whether products were sold at the posted prices in the past. This model is motivated by applications such as online marketplaces, online flash sales, and loan pricing. We first consider a multi-dimensional version of binary search over polyhedral sets and show that it has a worst-case regret which is exponential in the dimension of the feature space. We then propose a modification of the prior algorithm where uncertainty sets are replaced by their Löwner-John ellipsoids. We show that this algorithm has a worst-case regret which is quadratic in the dimension of the feature space and logarithmic in the time horizon. We also show how to adapt our algorithm to the case where valuations are noisy. Finally, we present computational experiments to illustrate the performance of our algorithm.

Key words: online learning, contextual bandits, ellipsoid method, revenue management

1. Introduction

Most dynamic pricing models assume that a firm sells identical products to its customer base over time. Even the models that do allow for product differentiation, generally assume that the seller offers a manageable number of distinct products. However, there exist important business settings, such as online marketplaces, where sellers offer an enormous number of different products to its customer base. Our paper addresses the following problem: how should a seller price its products when they arrive in an online fashion and are significantly differentiated from each other?

Specifically, we consider a firm selling products to customers over a finite time horizon. In each period, a new product arrives and the firm must set a price for it. The product features are chosen antagonistically by nature. The firm can base its pricing decision on the features of the product at hand, as well as on the history of past prices and sales. Once a price is chosen, the product is either accepted or rejected by the market, depending on whether the price is below or above

the product's market value. The firm does not know the market value of each product, except for the fact that the market value of each product is linear in the value of the product features (we also consider some commonly used non-linear models in Section 7). The seller can therefore use past prices and sales data to estimate the market values of the different features, and use those estimates to inform future pricing decisions. Our goal is to find a pricing algorithm that performs well in the sense that it generates a low worst-case regret. Our concern is how the regret scales with the time horizon, as well as how it performs with respect to the dimension of the feature space. Assuming the feature vectors are chosen antagonistically by nature ensures that our solution is robust to important considerations such as features appearing in correlated form and the set of relevant features changing over time (some features may have zero value throughout most of the time horizon, but be important in later periods).

Our first attempt is to propose a multi-dimensional version of binary search in order to learn the values of the different features. In each period, the seller represents the possible values of the different features by a polyhedral-shaped uncertainty set. Whenever a new product arrives, the seller solves two linear programs, one to compute the maximum possible market value of the product, and the other to compute the minimum possible market value of the product given the uncertainty set. If these two numbers are close together, the seller uses the minimum possible market value as an "exploit" price, in order to ensure that a sale occurs. If these two numbers are far apart, the seller performs a binary search step (or "explore" step), and chooses a price halfway between the minimum and the maximum possible market values. We call this algorithm PolytopePricing. However, despite seeming to be a suitable algorithm for the problem at hand, we show in Theorem 1 that this algorithm has a worst-case regret that is exponential in the dimension of the feature space. This occurs because nature can choose a sequence of vectors of features that forces the seller to explore for exponentially many periods without exploiting.

Fortunately, we can modify PolytopePricing to make it a low-regret algorithm. The modification invokes an idea from the ellipsoid method for solving a system of linear equations. At every step of the algorithm, we replace the convex uncertainty set (previously a polytope) by its Löwner-John ellipsoid. The Löwner-John ellipsoid of a convex body is the minimal volume ellipsoid that contains that convex body. We call this modified algorithm EllipsoidPricing. The main result of our paper is Theorem 2, which proves that EllipsoidPricing generates a worst-case regret that is quadratic in the dimension of the feature space and logarithmic in the time horizon. The proof is based on two ideas. The first is the classical idea from the ellipsoid method: the volume of the ellipsoidal uncertainty set shrinks exponentially fast with the number of cuts (in our problem, the cuts are explore prices). The second main idea is that, under EllipsoidPricing, the smallest radius of the ellipsoid cannot shrink below a given threshold. To prove this second idea, we build on

linear algebra machinery that characterizes the evolution of the eigenvalues after rank-one updates. This machinery is useful because an ellipsoid can be represented by a matrix whose eigenvalues correspond to the squares of the ellipsoid radii, and using an explore price corresponds to performing a rank-one update. Combining the two ideas, we get a quadratic bound on the number of possible explore steps, which yields our bound on the regret of the algorithm. EllipsoidPricing is also computationally more efficient than PolytopePricing, since it does not require solving linear programs in each iteration. In fact, all computational steps—optimizing a linear function over an ellipsoid and replacing a half-ellipsoid by its own Löwner-John ellipsoid—require nothing more than matrix-vector products.

The basic form of EllipsoidPricing assumes that the market value of each product is a deterministic function of its features. We also propose two variants of the algorithm that add robustness to noisy valuations. We call the first one ShallowPricing. The ShallowPricing algorithm is based on the idea of a shallow cut of an ellipsoid, which is an off-center cut, designed to maintain more than half of the original uncertainty set. By using shallow cuts, we can add a safety margin to each cut, and hence still obtain similar regret guarantees under a low-noise regime. Our second proposal is an algorithm we call EllipsoidEXP4, which is a combination of ShallowPricing with the standard adversarial contextual bandits algorithm EXP4. For EllipsoidEXP4, we show a regret guarantee that (i) matches the bound of EllipsoidPricing in the limit when the noise vanishes, (ii) approximately matches the regret guarantee of EXP4 under high-noise settings, and (iii) leads to an intermediate regret guarantees in moderately noisy environments. We discuss these algorithms and their regret guarantees in detail in Section 6.

Online marketplaces are one area in which the algorithms we develop in this paper can be applied. Consider Airbnb, the popular sharing economy platform for subletting homes and individual rooms. The products in Airbnb are stays, which are highly differentiated products: they involve different locations, amenities, and arrival dates, among many other features. Airbnb offers a service to its hosts called Smart Pricing, which, when turned on, allows Airbnb to choose prices on the hosts' behalf (see Bray 2017, Ye et al. 2018). As in our model, if a given good (in this case, a one night stay in a home at a particular date) is not sold, it generates no revenue and becomes obsolete.[1] To offer a service such as Smart Pricing to its hosts, a platform like Airbnb must use a feature-based dynamic pricing algorithm following the same spirit as our algorithms. Other online marketplaces, such as eBay or Etsy, could also use an algorithm such as ours to help sellers price their products.

An additional application is online flash sales websites such as Gilt, Rue La La, and Belle & Clive. These vendors periodically receive various goods from luxury brands to sell online. Usually,

[1] Though the value of outside options is normalized to zero in our model, they can be incorporated by adding one additional feature associated with the outside option value.

a flash sale lasts for a short time period, and the vendor needs to choose the price of each good. As in our model, the owner sells highly differentiated products and must set prices to balance exploration and exploitation. There also exist classical markets that involve highly differentiated products that arrive over time, such as the high-end art and the premium wine markets. The algorithms we propose in this paper may also be useful in these contexts since it is natural to set prices based on the values of the product features.

One of the key applications that has motivated the dynamic pricing with learning literature is the pricing of financial services (see Phillips 2005, Harrison et al. 2012, Keskin and Zeevi 2014). Consider a bank offering loans or other forms of consumer credits. After a consumer requests a loan, the bank must select a price (an interest rate), which the consumer can accept or reject. This literature studies how such a bank should balance immediate profit maximization with price exploration. A typical assumption in this literature is that consumers are indistinguishable from each other. In reality, consumers have several features (e.g., credit history, annual income, FICO score) which can be used by the bank to price loans. With our framework, the bank would be able to take these features into account when choosing interest rates to offer individual customers.

2. Related Literature

Our work lies at the intersection of two literature streams: dynamic pricing with learning and contextual bandits, and is also connected to learning from revealed preferences, conjoint analysis, and the ellipsoid method from optimization theory.

Dynamic pricing with learning. The literature on dynamic pricing with learning studies pricing algorithms for settings where the demand function is unknown. The problem is typically modeled as a variant of the multi-armed bandit problem, where the arms represent prices and the payoffs from the different arms are correlated since the demand evaluated at different price points are correlated random variables. The first paper that modeled dynamic pricing as a multi-armed bandit problem is Rothschild (1974). Kleinberg and Leighton (2003) deserve credit for formulating the finite-horizon, worst-case regret version of the problem of dynamic pricing with learning, a formulation that we use in our paper. In particular, they solve the one-dimensional version of our problem, as we discuss in Section 4.1. A large body of literature has recently emerged studying this topic. This includes both parametric approaches (Araman and Caldentey 2009, Broder and Rusmevichientong 2012, Harrison et al. 2012, Chen and Farias 2013, den Boer and Zwart 2013, Besbes and Zeevi 2015) as well as non-parametric ones (e.g., Besbes and Zeevi 2009, Keskin and Zeevi 2014). The literature also includes models that, like ours, use a robust optimization approach to model uncertainty (see, e.g., Bertsimas and Vayanos 2015). Another important dimension in pricing problems is that of limited supply (Besbes and Zeevi 2009, Badanidiyuru et al. 2013,

Babaioff et al. 2015). For example, Badanidiyuru et al. (2013) study a problem where the seller has a fixed number of goods, so s/he must trade-off learning and earning not only across time but also across supply levels. The authors provide near optimal results for this setting. In fact, their result is cast in a more general setting of *bandits with knapsacks*, where bandit algorithms have resource constraints. In their follow-up paper, Badanidiyuru et al. (2014) extend this analysis to contextual settings and obtain a non-trivial improvement over the standard reduction to contextual settings. This line of work has been further improved in a series of papers by Agrawal and Devanur (2015a,b) and Agrawal et al. (2016).

Contextual bandits. A crucial aspect of our model is that products arrive over time and are characterized by vectors of features. The literature that studies multi-armed bandit problems in settings where the payoff in each period depends on a particular set of features (that are relevant only for a specific period) is called contextual bandits. This literature started with Auer et al. (2002) and Auer (2003), and has recently grown into a large literature (see, for example, Dudik et al. 2011, Agarwal et al. 2014). Auer et al. (2002) proposed a regret-optimal algorithm for contextual bandits called EXP4 that we use as a building block in one of our algorithms in Section 6. Many models of contextual bandits (but certainly not all) assume that payoffs are linear in the feature vector (Chu et al. 2011, Abbasi-Yadkori et al. 2011, Agrawal and Devanur 2015a). In our model, we make a sightly different assumption: we assume market values are linear in the features. Products having market values which are a function of their features is a typical assumption in marketing, which is referred to as hedonic pricing (see Milon et al. 1984, Malpezzi 2003, Sirmans et al. 2005).

In a related work by Chu et al. (2011), the authors propose an algorithm called LinUCB that also use ellipsoids to design uncertainty regions in contextual learning settings, but both the problem they study and the resulting algorithms are very different from ours. In Chu et al. (2011), the payoffs are assumed to be linear in the context and are observed for the arm played. In our model, in contrast, the payoffs are discontinuous pricing functions and we only observe whether there is a sale or not. Also, the updates in Chu et al. (2011) are not based on cuts (as in our algorithm) but on high-dimensional statistical bounds.

Pricing with features. Closest to our paper is the work by Amin et al. (2014), which also studies a feature-based dynamic pricing problem. In their model, features are stochastically drawn from a distribution, whereas in our model, features are adversarially selected by nature. Amin et al. (2014) propose an algorithm that is based on stochastic gradient descent, and obtain a regret bound of $\widetilde{O}(T^{2/3})$.[2] However, they do not investigate how their algorithm performs with respect to the dimension of the feature set. In their stochastic setting, Amin et al. (2014) also analyze a version of the algorithm in which buyers strategically react to the algorithm.

[2] The $\tilde{O}(\cdot)$ notation is a variant of the $O(\cdot)$ notation that ignores logarithmic terms.

In a paper subsequent to ours, Qiang and Bayati (2016) also consider a dynamic pricing problem in a model where the value of different features (or covariates) need to be learned. Their model is stochastic, as opposed to our adversarial model, and features arrive in an i.i.d. fashion. They show that a greedy least squares approach performs well, which is not the case in a model without covariates. Approaches based on least squares can be used in stochastic models, but not in adversarial models such as the model considered in this paper.

Learning from revealed preferences. There exist additional learning problems for which researchers have developed techniques that are somewhat related to the algorithm we propose in this paper. In the problem called "learning from revealed preferences," a seller sells an identical bundle of goods in each period to a single buyer. The seller does not know the utility function of the buyer, but can learn from the past bundles purchased by the buyer. Amin et al. (2015) and Roth et al. (2016, 2017) study the problem of dynamic pricing in multiple dimensions and propose several algorithms for this problem, some of which are, like our algorithm, based on the ellipsoid method (Khachiyan 1979). There are at least two important differences between this line of work and our paper. First, no features are present in this line of work. Second, the decision variable in our problem at each time period is a single price, while in this literature the seller selects a price for each item at each period. An algorithm that selects multiple prices in each period may seem more general than an algorithm that selects only a single price per period, as in our setting. However, this intuition is misleading. When applying the ellipsoid method to the problem of learning from revealed preferences, one can choose the direction of each cut by selecting an appropriate vector of prices. In our problem, however, we are given a cut direction chosen adversarially by nature (the vector of features) and thus, we are only able to select where to position the hyperplane.

Conjoint analysis. Another related field of study is adaptive choice-based conjoint analysis, where a market researcher wants to design an adaptive survey to elicit the preferences of individuals in a population. Though the problem is different from ours, some of the most commonly used solutions share with our algorithm the property that they heavily rely on the geometry of polyhedra and ellipsoids (see, e.g., Toubia et al. 2003, 2004, 2007, Bertsimas and O'Hair 2013). A key distinction that makes our problem more difficult to solve than conjoint analysis is that we cannot choose directions of cuts (vectors of features), while the market researcher in conjoint analysis is allowed to do so.

The ellipsoid method. One of the key ideas that we use in our paper is to replace an uncertainty set that is polyhedral by its Löwner-John ellipsoid. This idea is not novel, dating back to Khachiyan (1979)'s proof that linear programs are solvable in polynomial time. There are several key advantages of using ellipsoids instead of polyhedra. In particular, it is easy to cut an ellipsoid

through its center. In addition, by cutting an ellipsoid through its center and replacing the remaining half-ellipsoid with its own Löwner-John ellipsoid, a fixed fraction of the volume is removed. The idea of replacing polyhedra with ellipsoids has been used in other papers after Khachiyan (1979), including Toubia et al. (2003, 2004, 2007). Removing a fixed fraction of the volume of the uncertainty set in each iteration is also a well-known idea, and has found applications in preference elicitation (Boutilier et al. 2006) and recommender systems (Viappiani and Boutilier 2009). Several of the challenges that emerge in our problem are related to (i) not being able to control the direction of cuts and thus, not being able to cut orthogonally to the direction in which the uncertainty is the largest, as the aforementioned papers do; and (ii) having to manage not only the volume, but also the radii of the ellipsoids since the regret in our model is a function of the length of an ellipsoid along a direction chosen by nature.

3. Model

Consider a setting with a seller that receives a different product at each time period $t = 1, 2, \ldots, T$. Each product is described by a vector of features $x_t \in \mathcal{X} \subset \mathbb{R}^d$ and has a market value $v_t = v(x_t)$, which is unknown to the seller. Upon receiving each product, the seller observes the vector of features x_t and then, chooses a price p_t. The market either accepts the price, which occurs if the price p_t is less or equal than the market value v_t, or rejects it, in which case the product is lost.[3] The goal of the seller is to design a pricing policy to maximize revenue. The main challenge here is that the market value is unknown to the seller and, at each time, the seller wants to earn revenues but also to refine his/her knowledge about the market value function v.

In order for this problem to be tractable, we need to make assumptions about the market value function v. We assume that the market value of a product is a linear function of its features, i.e., $v_t = \theta' x_t$, an assumption that we partially relax in Section 7. We also assume for the sake of normalization that $\|x_t\| \leq 1$ for all $x_t \in \mathcal{X}$ and that $\|\theta\| \leq R$, where $\|\cdot\|$ refers to the ℓ_2-norm. The exact value of θ is unknown to the seller. We encode the initial uncertainty of the seller as a polytope $K_1 \subseteq \mathbb{R}^d$, which represents all feasible values of θ. The set K_1 could either be a d-dimensional box or encode some initial domain specific knowledge about the problem.

The seller sets a price p_t at each time period, and collects revenues if a sale occurs. If the price selected by the seller is below or equal the market value, i.e., $p_t \leq \theta' x_t$, a sale occurs and the seller earns a revenue of p_t. If the seller sets $p_t > \theta' x_t$, no sale occurs and no revenue is generated. At each time period, the seller may learn some new information about the value of θ that can be used in subsequent time periods. More precisely, the seller naturally updates the uncertainty set with

[3] The assumption of no outside value if the offer is rejected is without loss of generality since the outside value could be encoded as an additional feature with negative value associated with it.

$K_{t+1} = K_t \cap \{\theta \in \mathbb{R}^d : \ \theta' x_t \geq p_t\}$ or $K_{t+1} = K_t \cap \{\theta \in \mathbb{R}^d : \ \theta' x_t \leq p_t\}$ depending on whether a sale occurred or not, where K_t denotes the uncertainty set at time t.[4]

Our goal is to find a simple and computationally efficient dynamic pricing policy that achieves a good performance in terms of regret. Let Π be the seller's policy and $\mathbb{X}$ the strategies available to nature (nature adversarially selects the true value of the parameter θ, and the feature vectors x_t in each round). Both the seller and nature are allowed to use closed-loop policies, where their actions at time t depend on the history of events up to time $t-1$. The worst-case regret induced by policy Π is given by:

$$\textsc{Regret}(\Pi) = \max_{\theta \in K_1, \ X \in \mathbb{X}} \ \sum_{t=1}^{T} \Big[\theta' x_t - p_t I\{\theta' x_t \geq p_t\} \Big], \tag{1}$$

where $I\{\cdot\}$ denotes the indicator function and $X \in \mathbb{X}$ represents the policy used by nature to select the sequence of feature vectors $\{x_t\}$. The first term inside the summation corresponds to the maximal revenue the seller could extract if s/he knew the value of θ, and the second term is the actual revenue generated by policy Π for a given (θ, X) pair. We are concerned not just with how the regret scales with T, as it is typical in multi-armed bandit problems, but also with how the regret scales with the dimension of the feature space d.

Most of the paper focuses on the model described above where the valuation is a fixed linear function of the item's features. This serves as the main building block for tackling richer models. We consider extensions in two directions: noisy and non-linear valuations. The setting with noisy valuations is studied in Section 6. A special case of non-linearity is addressed via a Lipschitz continuity argument in Section 7.

4. A First Attempt: Multi-Dimensional Binary Search

4.1. Binary Search and the One-Dimensional Problem

The simplest special case of our problem is when there is only a single dimension, i.e., $d = 1$. Assume further that $R = 1$, i.e, $\theta \in [0, 1]$ and $x_t = 1$ for every t (note that the exact value of x_t does not affect the problem in the one-dimensional case). Then, the problem consists of picking a price p_t in each time step and collecting revenue $p_t \cdot I\{p_t \leq \theta\}$. A natural strategy is to perform binary search for a few steps, build a good estimate of θ, and then set the price using this estimate. More precisely, start with $K_1 = [0, 1]$, for each step t, keep $K_t = [\ell_t, u_t]$ and then set the price $p_t = \frac{1}{2}(\ell_t + u_t)$. If a sale occurs, set $K_{t+1} = [p_t, u_t]$ and otherwise, $K_{t+1} = [\ell_t, p_t]$. Repeat this as long as $u_t - \ell_t \geq \epsilon$, for some $\epsilon > 0$. From this point onwards, set the price at $p_t = \ell_t$. Note that under this price, the seller

[4] If a sale does not occur at time t, the seller learns that $\theta' x_t < p_t$. However, in order to maintain our uncertainty sets closed, we add the constraint $\theta' x_t \leq p_t$ to update the uncertainty set, rather than using a strict inequality.

is guaranteed to sell the item. The algorithm uses $\log_2(\frac{1}{\epsilon})$ steps to build a good estimate of θ, and from then onwards, uses the lower estimate to price. It is not hard to see that the total regret is:

$$\textsc{Regret} \leq \log_2\left(\frac{1}{\epsilon}\right) + \left(T - \log_2\left(\frac{1}{\epsilon}\right)\right)\cdot\epsilon = O(\log_2 T) \quad \text{for} \quad \epsilon = \frac{1}{T}.$$

This regret is surprisingly not optimal for the one-dimensional problem. Kleinberg and Leighton (2003) show that the optimal regret for the one-dimensional problem is $O(\ln\ln T)$. This result implies a lower bound of $\Omega(d\ln\ln T)$ for any algorithm in our multidimensional problem. Determining if Kleinberg and Leighton (2003) is extendable to higher dimensions is a difficult problem that we do not attempt to address. Instead, we focus on the simpler binary search algorithm, which has sufficiently low regret in T, $O(\ln T)$, and aim to see if we can generalize it to higher dimensions.

4.2. Binary Search in High Dimensions

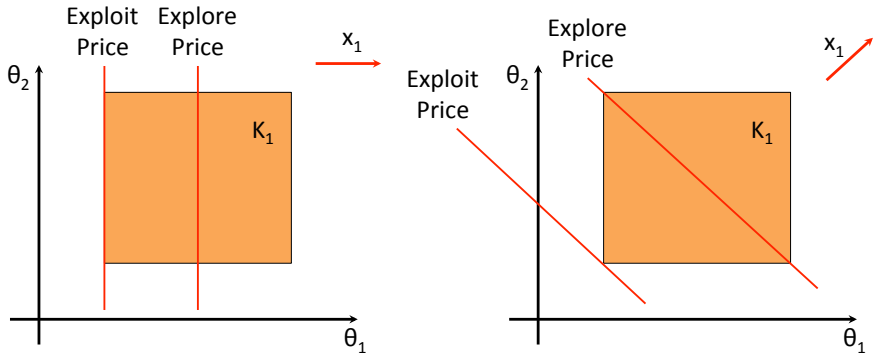
We now return to our original setting with d dimensions and features x_t chosen adversarially by nature. Our first instinct might be to follow the same approach and use the first few iterations to build a good estimate of θ (we call this the *explore* phase), and then use this estimate to set a close-to-optimal price (we call this the *exploit* phase). One problem with this approach is that the features selected by nature may never offer an opportunity for the seller to learn θ precisely. Some features might not appear with a non-zero value often enough to allow for their values to be learned. Features might also be chosen in a correlated fashion by nature, making learning more difficult. Finally, even in the case where all the different features are present and not correlated, it might still not be wise for the seller to wait until s/he reaches a good estimate of θ to start exploiting, as some features may only appear with non-zero values close to the time horizon T. We therefore need an algorithm that dynamically decides whether or not to explore in each period, rather than having fixed exploration and exploitation phases.

4.3. Explore and Exploit Prices

Based on our discussion so far, we know that we cannot hope to learn the value of θ exactly. Also, pre-determined exploration and exploitation phases do not seem to be adequate here. Instead, for each product that arrives, we will decide whether to exploit or not. In particular, we will exploit if we have gathered enough information on the market value for this particular set of features.

To evaluate the amount of information we have for the feature vector x_t, the seller can use the current uncertainty set K_t to compute an interval $[\underline{b}_t, \bar{b}_t]$ that contains the actual market value $v_t = \theta' x_t$, by solving the following pair of linear programs:

$$\underline{b}_t = \min_{\hat{\theta}\in K_t} \hat{\theta}' x_t \qquad \text{and} \qquad \bar{b}_t = \max_{\hat{\theta}\in K_t} \hat{\theta}' x_t. \tag{2}$$



(a) $x_1 = (1,0)$ (b) $x_1 = (1/\sqrt{2}, 1/\sqrt{2})$

Figure 1 **Explore and exploit prices when $x_1 = (1,0)$ and $x_1 = (1/\sqrt{2}, 1/\sqrt{2})$.**

By pricing the item at $p_t = \underline{b}_t$, the seller is guaranteed to sell the item and generate revenue $\underline{b}_t$. However, the seller will learn nothing about the market value from such a price. We call such a price an *exploit price*. Inspired by binary search, we define an *explore price* as the price that will provide us with most information about the buyer's valuation for that particular feature vector, which is: $p_t = \frac{1}{2}(\bar{b}_t + \underline{b}_t)$.

In the simple two-dimensional examples shown in Figure 1, the explore price always divides the feasible region into two parts, whereas the exploit price is always located at the boundary of the set. Note that by definition an exploit price guarantees some revenue, while an explore price may or may not generate a sale.

Now, we describe the algorithm PolytopePricing, which is parametrized by a threshold value $\epsilon > 0$. Starting from an initial uncertainty set K_1, for each t, compute the values $\underline{b}_t$ and $\bar{b}_t$. If $\bar{b}_t - \underline{b}_t \leq \epsilon$, set the exploit price $p_t = \underline{b}_t$, collect revenue p_t, and set $K_{t+1} = K_t$. If $\bar{b}_t - \underline{b}_t > \epsilon$, set the explore price $p_t = \frac{1}{2}(\bar{b}_t + \underline{b}_t)$. If a sale occurs, update the uncertainty set to $K_{t+1} = K_t \cap \{\theta \in \mathbb{R}^d : \ \theta' x_t \geq p_t\}$. Otherwise, update it to $K_{t+1} = K_t \cap \{\theta \in \mathbb{R}^d : \ \theta' x_t \leq p_t\}$.

4.4. The Exponential Regret of PolytopePricing

Although PolytopePricing is a straightforward generalization of the single-dimensional binary search algorithm, it is far from an ideal algorithm. First, it needs to keep track of a complicated polytope. Second, each step is computationally expensive as it requires solving two linear programs.

Furthermore, for any parameter $\epsilon > 0$, the worst-case regret of PolytopePricing is exponential in d. The proof of this result is presented in the Appendix.

Theorem 1. *For any parameter $\epsilon > 0$, the algorithm* PolytopePricing *suffers worst-case regret $\Omega(Ra^d \ln T)$ for some constant $a > 1$.*

We remark that the proof of Theorem 1 shows that the PolytopePricing algorithm has exponential regret in d even for an adversary that draws the feature vectors from a very simple i.i.d. distribution that samples 1/4 of the features.

5. Ellipsoid Pricing

In this section, we modify the PolytopePricing algorithm from the previous section so as to achieve a regret that is polynomial (in fact, quadratic) in the dimension. As a bonus, the algorithm becomes also simpler to implement and computationally cheap. The algorithm now requires only to maintain a $d \times d$ matrix and to perform a few matrix-vector products in each iteration.

Our new algorithm is inspired by Khachiyan's celebrated ellipsoid method (Khachiyan 1979). The central idea is that instead of keeping the uncertainty set K_t in each iteration, we "round" it up to the smallest ellipsoid E_t that contains K_t. This is often referred to as the Löwner-John ellipsoid of the set K_t.

We call our algorithm EllipsoidPricing. The algorithm starts from the smallest ellipsoid E_1 that contains K_1, or in fact any ellipsoid that contains K_1 (see Figure 2). At each time step t, the algorithm computes the values $\underline{b}_t$ and $\bar{b}_t$ using the ellipsoid E_t instead of the uncertainty set K_t:[5]

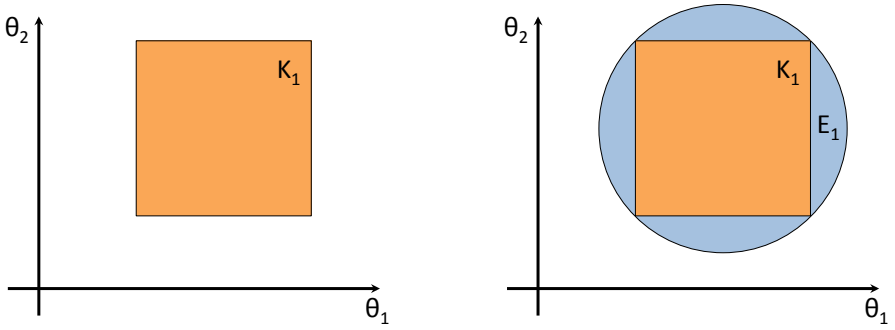
$$\underline{b}_t = \min_{\hat{\theta} \in E_t} \hat{\theta}' x_t \qquad \text{and} \qquad \bar{b}_t = \max_{\hat{\theta} \in E_t} \hat{\theta}' x_t. \tag{3}$$

If $\bar{b}_t - \underline{b}_t \leq \epsilon$, the seller offers the exploit price $p_t = \underline{b}_t$, collects revenue p_t, and sets $E_{t+1} = E_t$ (see Figure 3). If $\bar{b}_t - \underline{b}_t > \epsilon$, the seller offers the explore price $p_t = \frac{1}{2}(\bar{b}_t + \underline{b}_t)$. If a sale occurs, let $H_{t+1} = E_t \cap \{\theta \in \mathbb{R}^d : \ \theta' x_t \geq p_t\}$. Otherwise, let $H_{t+1} = E_t \cap \{\theta \in \mathbb{R}^d : \ \theta' x_t \leq p_t\}$. Now, let E_{t+1} be the smallest ellipsoid that contains the half-ellipsoid H_{t+1} (see Figures 4 and 5). Our main result is reported next.

Theorem 2. *The worst-case regret of the* EllipsoidPricing *algorithm with parameter* $\epsilon = Rd^2/T$ *is* $O(Rd^2 \ln(T/d))$.

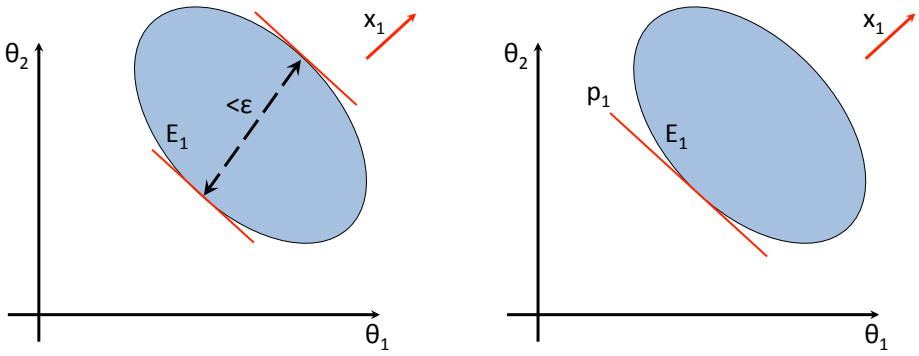
We defer the proof of this theorem until Section 5.3. Interestingly, efficiency is achieved by *enlarging* the uncertainty set. At the expense of adding candidate vectors $\hat{\theta}$ that are known not to be the true θ (when we enlarge H_{t+1} to E_{t+1}) at each iteration t, we are *regularizing* the uncertainty sets. In other words, we are making the uncertainty sets symmetric and easier to analyze. We are not the first to propose this kind of technique. The same principle was at play in Khachiyan (1979)'s proof that the ellipsoid method solved linear programming in polynomial time, as well as in more recent papers that also rely on the underlying mechanics of the ellipsoid method.

[5] Note that we slightly abuse notation by reusing the variable names $\underline{b}_t$ and $\overline{b}_t$ in this section.



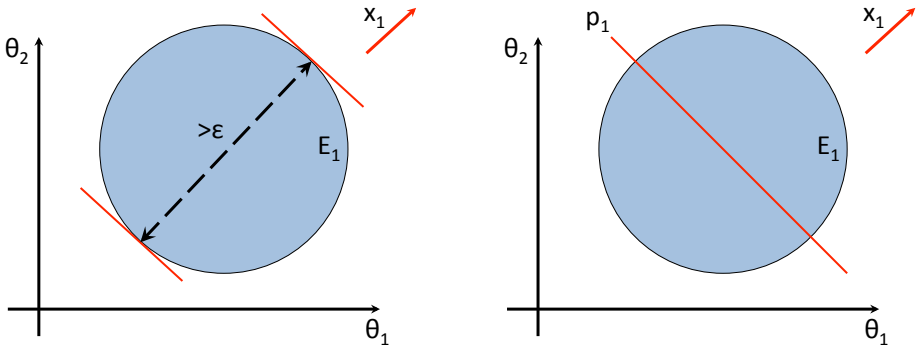
(a) The initial uncertainty set K_1 is a polytope.(b) The smallest ellipsoid E_1 that contains K_1.

Figure 2 The polytope K_1 and its Löwner-John ellipsoid E_1.



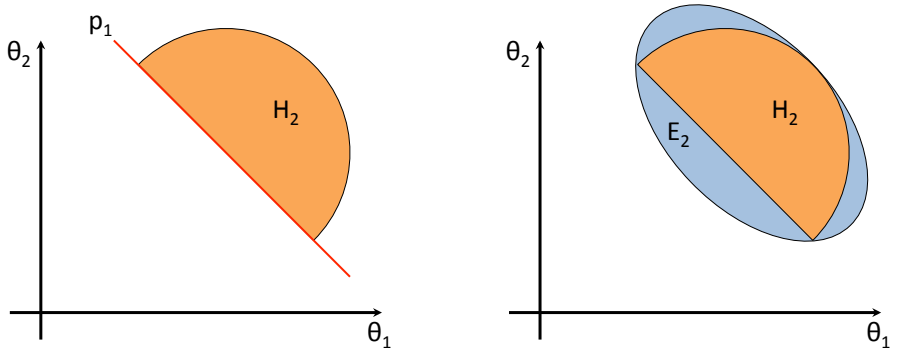
(a) Solve for the max and min over E_1. (b) Compute the exploit price p_1.

Figure 3 The vector $x_1 = (1/\sqrt{2}, 1/\sqrt{2})$ induces an exploit price p_1.



(a) Solve for the max and min over E_1. (b) Compute the explore price p_1.

Figure 4 The vector $x_1 = (1/\sqrt{2}, 1/\sqrt{2})$ induces an explore price p_1.



(a) Update the uncertainty set by observing a sale. (b) Compute the Löwner-John ellipsoid E_2.

Figure 5 Updating the uncertainty set and computing the Löwner-John ellipsoid E_2.

The reader familiar with the mechanics of the ellipsoid method will readily recognize it here: we start with an ellipsoid, and at each time we find a hyperplane passing through its center, cut it in half, and replace the remaining half by its smallest enclosing ellipsoid. The guarantee that the ellipsoid method offers is that the volume of the ellipsoid decreases at each time step. More precisely, after n cuts (which in our case correspond to n exploration rounds), the volume of the ellipsoid is at most $e^{-\frac{n}{2d}}$ of the original volume. However, it provides no guarantee about the shape of the ellipsoid. An ellipsoid of small volume could definitely be very skinny in some dimensions, but quite long in other dimensions.

To prove Theorem 2, we show that if we cut the ellipsoid only along directions in which it is not very skinny yet (i.e., we explore only if the gap $\bar{b}_t - \underline{b}_t$ is large) sufficiently many times, then the ellipsoid will eventually become small in every direction. Consequently, we will not need to explore from that point onwards. To do so, instead of bounding the volume of the ellipsoid, we need to bound the eigenvalues of the matrix defining the ellipsoid.

We will make the statements in the previous paragraph precise in a moment. Before that, we provide the reader with a quick introduction of the theory of ellipsoids. We refer the reader to the book by Grötschel, Lovász, and Schrijver (Grötschel et al. 1993), or the survey by Bland, Goldfarb, and Todd (Bland et al. 1981) for an in-depth discussion.

5.1. A Primer on Ellipsoids

We invite readers who are familiar with the ellipsoid method to skip this subsection and move directly to Section 5.2. A $d \times d$ matrix A is symmetric if $A = A'$, i.e., it is equal to its transposed matrix. It is a basic fact of linear algebra that every symmetric matrix A admits an eigenvalue decomposition, i.e., we can write $A = Q\Lambda Q'$, where Q is a $d \times d$ orthogonal matrix (i.e., $Q'Q = I$)

and Λ is a diagonal matrix with elements $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$ in its main diagonal and zero elsewhere. We refer to $\lambda_i(A)$ as the i-th largest eigenvalue of A. A symmetric matrix is said to be *positive definite* if all of its eigenvalues are strictly positive, i.e., $\lambda_d(A) > 0$.

An ellipsoid E is a subset of $\mathbb{R}^d$ defined by a vector $a \in \mathbb{R}^d$, which we call the center and a positive definite matrix A as follows:

$$E(A,a) := \{\theta \in \mathbb{R}^d : \ (\theta - a)' A^{-1} (\theta - a) \leq 1\}.$$

Each of the d radii of $E(A,a)$ corresponds to the square root of an eigenvalue of A and the volume of the ellipsoid is given by:

$$\text{Vol } E(A,a) = V_d \cdot \sqrt{\textstyle\prod_i \lambda_i(A)},$$

where V_d is a constant that depends only on d and corresponds to the volume of the unit ball in $\mathbb{R}^d$. Since the volume depends on the matrix A and not on a, we will often write Vol $E(A)$ instead of Vol $E(A,a)$, when the center is not important or can be inferred from the context.

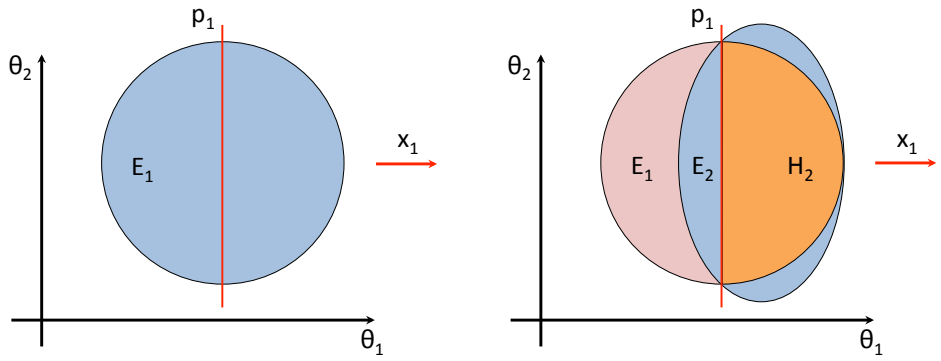
For any vector $x \in \mathbb{R}^d \setminus \{0\}$, $\arg\max_{\theta \in E(A,a)} x'\theta = a + b$ and $\arg\min_{\theta \in E(A,a)} x'\theta = a - b$ for $b = Ax/\sqrt{x'Ax}$ (see, Grötschel et al. 1993). Furthermore, the hyperplane perpendicular to x passing through a is given by $x'(\theta - a) = 0$. This plane cuts the ellipsoid $E(A,a)$ in two symmetric pieces. The smallest ellipsoid containing each of these pieces (called the Löwner-John ellipsoid) can be computed by the following closed form formula. The smallest ellipsoid containing $E(A,a) \cap \{\theta \in \mathbb{R}^d : \ x'(\theta - a) \leq 0\}$ is $E(\tilde{A}, a - \frac{1}{d+1} b)$ and the smallest ellipsoid containing $E(A,a) \cap \{\theta \in \mathbb{R}^d : \ x'(\theta - a) \geq 0\}$ is $E(\tilde{A}, a + \frac{1}{d+1} b)$, where:

$$\tilde{A} = \frac{d^2}{d^2 - 1} \left(A - \frac{2}{d+1} bb' \right). \tag{4}$$

A central fact used in the analysis of the ellipsoid method is the following:

$$\text{Vol } E(\tilde{A}) \leq e^{-1/2d} \cdot \text{Vol } E(A).$$

One can note that while the volume (and hence the product of eigenvalues) decreases after an update, some eigenvalues might increase whereas other eigenvalues decrease. To see this, consider for example the ellipsoid where $A = I$ (here I denotes the identity matrix) and assume $x_1 = e_1 = (1, 0, \ldots, 0)$, i.e., the coordinate vector in the 1-direction. Using Eq. (4), we obtain that $\tilde{A}$ is the diagonal matrix with eigenvalue $\frac{d^2}{(d+1)^2} < 1$ in direction e_1, and $\frac{d^2}{d^2-1}$ in all other directions. In general, the ellipsoid shrinks in the direction of x_1 but expands in directions orthogonal to x_1 (see Figure 6 for an illustration). For example, if one starts with a unit ball, successively cut the ellipsoid along the e_1 direction and replace one of the halves by its Löwner-John ellipsoid, then one direction shrinks exponentially while the other directions grow exponentially.



(a) Assume E_1 is a sphere and $x_1 = (1, 0)$. (b) The half-ellipsoid H_2 and ellipsoid E_2.

Figure 6 **After an explore step where $x_1 = (1, 0)$, the new ellipsoid E_2 shrinks along the θ_1-axis but expands along the θ_2-axis.**

5.2. Revisiting EllipsoidPricing

Before analyzing the regret of the EllipsoidPricing algorithm, we revisit it using the tools introduced in Section 5.1. We can represent the ellipsoid at time t by $E_t = E(A_t, a_t)$. Furthermore, computing $\underline{b}_t$ and $\bar{b}_t$ can be done in closed form:

$$\underline{b}_t = \min_{\hat{\theta} \in E_t} x_t' \hat{\theta} = x_t' \left[a_t - \frac{A_t x_t}{\sqrt{x_t' A_t x_t}} \right] = x_t' a_t - \sqrt{x_t' A_t x_t}.$$

Similarly, $\bar{b}_t = x_t' a_t + \sqrt{x_t' A_t x_t}$, which means that the gap $\bar{b}_t - \underline{b}_t = 2\sqrt{x_t' A_t x_t}$. First, note that deciding between exploration and exploitation as well as setting the appropriate price can be accomplished by computing a matrix-vector product instead of solving two linear programs (as it was the case for PolytopePricing). Second, updating the ellipsoid can be done via Eq. (4). The algorithm needs to keep track only of a $d \times d$ matrix and a d-dimensional vector. Unlike PolytopePricing, the amount of information that the algorithm needs to maintain does not depend on T.

5.3. Regret Analysis for EllipsoidPricing

To show that the regret of EllipsoidPricing is small, we prove that if ϵ is set properly, then the number of exploration rounds is bounded. To be precise:

Lemma 1. EllipsoidPricing *will choose the explore price in at most* $2d^2 \ln(20R(d+1)/\epsilon)$ *time periods.*

We defer the proof of this lemma to Section 5.5. It is simple to see how Lemma 1 can be used to prove our main result by setting the parameter ϵ appropriately.

Proof of Theorem 2. In an exploitation round, since we collect revenue $\underline{b}_t$ and the best possible revenue from that round is $\bar{b}_t$, the regret from that round is at most $\bar{b}_t - \underline{b}_t \leq \epsilon$. For exploration

rounds, we use the trivial bound of regret R per round. So, if we have at most N exploration rounds, $\textsc{Regret} \leq NR + (T-N)\epsilon$. By Lemma 1 we have: $\textsc{Regret} \leq 2Rd^2 \ln(20R(d+1)/\epsilon) + T\epsilon$. Choosing $\epsilon = Rd^2/T$, the total regret becomes $\textsc{Regret} = O(Rd^2 \ln(T/d))$. $\square$

The core of our analysis consists of proving Lemma 1. Recall that the algorithm explores if and only if $\bar{b}_t - \underline{b}_t = 2\sqrt{x_t' A_t x_t} > \epsilon$. If the matrix A_t is such that $\max_{\{x \in \mathbb{R}^d:\ \|x\| \leq 1\}} 2\sqrt{x' A_t x} \leq \epsilon$, then all the feature vectors will lead the algorithm to exploit. We note that the quantity $\max_{\{x \in \mathbb{R}^d:\ \|x\| \leq 1\}} x' A_t x$ corresponds to the largest eigenvalue $\lambda_1(A_t)$ of the matrix A_t. Our goal, then, is to show that after $2d^2 \ln(20R(d+1)/\epsilon)$ exploration steps, all the eigenvalues of A_t are at most $\epsilon^2/4$, so that $\max_{\{x \in \mathbb{R}^d:\ \|x\| \leq 1\}} 2\sqrt{x' A_t x} \leq \epsilon$.

The proof of this claim will crucially rely on the fact that we only perform exploration steps if $\sqrt{x_t' A_t x_t}$ is sufficiently large for the feature vector x_t. If instead we were to explore in every round, then, even though the volume is shrinking by the usual ellipsoid argument, the largest eigenvalue may not shrink, as shown in the example at the end of Section 5.1.

Conceptually, we would like to show that after sufficiently many exploration steps, the largest eigenvalue cannot be too large. We prove this result in a roundabout way. We first construct a lower bound for the smallest eigenvalue. Such a bound automatically implies a lower bound on the volume of the ellipsoid. Since at each exploration step the volume decreases by a constant factor, we also have an upper bound on the volume of the ellipsoid after a given number of exploration steps. Combining these two results, we obtain an upper bound on the number of exploration steps, which allows us to prove that EllipsoidPricing is a low-regret algorithm.

5.4. More Tools from Linear Algebra

To study how the eigenvalues of A_t change when we explore, we introduce some tools from linear algebra to bound the variation in eigenvalues, when a matrix is perturbed by a rank-one matrix.

Given a symmetric $d \times d$ matrix A, its characteristic polynomial is defined as $\varphi_A(z) = \det(A - zI)$, which is a polynomial of degree d with the eigenvalues of A as roots.

Given a vector $b \in \mathbb{R}^d$ and $\beta > 0$, consider the rank-one perturbation $D = A - \beta bb'$. If $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$ are the eigenvalues of A, Wilkinson (1965) showed that the characteristic polynomial of D can be written as:

$$\varphi_D(z) = \det(A - \beta bb^\top - zI) = \prod_j (\lambda_j - z) - \beta \sum_i b_i^2 \prod_{j \neq i} (\lambda_j - z).$$

It is often convenient to write for $z \neq \lambda_i$ for all i, the characteristic polynomial as:

$$\varphi_D(z) = \prod_j (\lambda_j - z) \cdot \hat{\varphi}_D(z) \quad \text{where} \quad \hat{\varphi}_D(z) = 1 - \beta \sum_i \frac{b_i^2}{\lambda_i - z}. \tag{5}$$

We refer to Golub (1973) for an in-depth discussion of this result. An important consequence is the fact that evaluating the characteristic polynomial $\varphi_D(z)$ at λ_i, we obtain: $\varphi_D(\lambda_d) \leq 0$, $\varphi_D(\lambda_{d-1}) \geq 0$, $\varphi_D(\lambda_{d-2}) \leq 0$, and so on. Then, the intermediate value theorem allows us to pin down the exact intervals in which the eigenvalues of D lie. Let $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d$ be the eigenvalues of D, then:

$$\lambda_d - \beta b'b \leq \sigma_d \leq \lambda_d \leq \sigma_{d-1} \leq \lambda_{d-1} \leq \sigma_{d-2} \leq \lambda_{d-2} \leq \ldots \leq \lambda_2 \leq \sigma_1 \leq \lambda_1. \tag{6}$$

Consequently, this provides us with a tool to lower bound the smallest eigenvalue of D, as we show in the next lemma.

LEMMA 2. *Let $\lambda_1 \geq \ldots \geq \lambda_d$ be the eigenvalues of A and $\sigma_1 \geq \ldots \geq \sigma_d$ be the eigenvalues of $D = A - \beta bb'$. Consider any $z < \lambda_d$. If $\varphi_D(z) \geq 0$, then $\sigma_d \geq z$.*

Proof. For any $z < \lambda_d$, the sign of $\varphi_D(z)$ is the same as the sign of $\hat{\varphi}_D(z)$ since in Eq. (5), we have $\prod_j(\lambda_j - z) > 0$. Thus, $\varphi_D(z) \geq 0$ implies $\hat{\varphi}_D(z) \geq 0$. Note that $\hat{\varphi}_D(\cdot)$ is a non-increasing function since $\frac{\partial \hat{\varphi}_D(z)}{\partial z} = -\beta \sum_i \frac{b_i^2}{(\lambda_i - z)^2} \leq 0$. We also have that $\hat{\varphi}_D(\sigma_d) = 0$ by the definition of the characteristic polynomial. Therefore, $\hat{\varphi}_D(z) \geq 0$ implies $\sigma_d \geq z$. □

5.5. Back to the Regret Analysis

As discussed at the end of Section 5.3, our proof strategy is to lower bound the smallest eigenvalue of A_t, and then to use the traditional ellipsoid method argument that upper bounds the volume of the ellipsoid. The two bounds combined can then be used to upper bound the number of exploration steps. First, we use Lemma 2 to show that the smallest eigenvalue does not decrease by much in any given iteration:

LEMMA 3. *For any exploration step t, we have: $\lambda_d(A_{t+1}) \geq \frac{d^2}{(d+1)^2}\lambda_d(A_t)$.*

Proof. From the update rule in Eq. (4), we can write $A_{t+1} = \frac{d^2}{d^2-1}D$ for $D = A_t - \frac{2}{d+1}bb'$, where $b = A_t x_t / \sqrt{x_t' A_t x_t}$. For convenience, we move to the base of eigenvalues of A_t, which we do by writing $A_t = Q\Lambda Q'$. We define $\tilde{A}_{t+1} = Q' A_{t+1} Q$ and $\tilde{D} = Q' D Q$. We thus obtain $\tilde{A}_{t+1} = \frac{d^2}{d^2-1}\tilde{D}$ and $\tilde{D} = \Lambda - \frac{2}{d+1}\tilde{b}\tilde{b}'$, where $\tilde{b} = Q'b = \Lambda c / \sqrt{c'\Lambda c}$ and $c = Q' x_t$.

Since the eigenvalues are invariant by changes of bases, $\lambda_d(A_{t+1}) = \lambda_d(\tilde{A}_{t+1})$. We know that $\lambda_d(\tilde{A}_{t+1}) = \frac{d^2}{d^2-1}\lambda_d(\tilde{D})$, so we only need to prove that $\lambda_d(\tilde{D}) \geq \frac{d^2-1}{d^2} \cdot \frac{d^2}{(d+1)^2}\lambda_d(A_t) = \frac{d-1}{d+1}\lambda_d(A_t)$.

To simplify notation, we refer to $\lambda_d(A_t)$ as simply λ_d from now on. Using Lemma 2, we only need to argue that $\hat{\varphi}_{\tilde{D}}(\frac{d-1}{d+1}\lambda_d) \geq 0$. We have:

$$\hat{\varphi}_{\tilde{D}}\left(\frac{d-1}{d+1}\lambda_d\right) = 1 - \frac{2}{d+1}\sum_i \frac{\tilde{b}_i^2}{\lambda_i - \frac{d-1}{d+1}\lambda_d} \geq 0.$$

Using the fact that $\tilde{b}_i = \lambda_i c_i / \sqrt{\sum_j \lambda_j c_j^2}$, one can rewrite the expression as follows:

$$1 - \frac{2}{d+1} \sum_i \frac{\lambda_i c_i^2}{\sum_j \lambda_j c_j^2} \frac{1}{1 - \frac{d-1}{d+1}\frac{\lambda_d}{\lambda_i}} \geq 1 - \frac{2}{d+1} \max_i \frac{1}{1 - \frac{d-1}{d+1}\frac{\lambda_d}{\lambda_i}} = 1 - \frac{2}{d+1} \frac{1}{1 - \frac{d-1}{d+1}\frac{\lambda_d}{\lambda_d}} = 0.$$

The inequality follows from the fact that the term $\frac{\lambda_i c_i^2}{\sum_j \lambda_j c_j^2}$ depicts a convex combination and can be bounded by its maximal element. The equality follows from λ_d being the smallest eigenvalue. □

Lemma 3 shows that the smallest eigenvalue of A_t decreases in each time step by at most $d^2/(d+1)^2$. The intuition for this result is as follows. At the end of Section 5.1, we argued that when the matrix A_t corresponds to the unit sphere and $x_1 = e_1$, the new matrix A_{t+1} will have $d^2/(d+1)^2$ as its smallest eigenvalue, which will correspond to direction e_1. The same statement is true in general. Assume x_t is the eigenvector that corresponds to the smallest eigenvalue of an arbitrary matrix A_t. Then, the smallest eigenvalue of A_{t+1} is equal to $\frac{d^2}{(d+1)^2}\lambda_d(A_t)$. Lemma 3 proves that this particular x_t is the one that causes the smallest eigenvalue to shrink the most.

In the next lemma, we show that this eigenvalue cannot decrease past a certain point. More precisely, we show that there exists a constant $k(d)$ such that once the smallest eigenvalue is below $k(d)\epsilon^2$, then either (i) $x_t' A_t x_t \leq \frac{1}{4}\epsilon^2$, resulting in an exploit step, or (ii) $\lambda_d(A_{t+1}) \geq \lambda_d(A_t)$, i.e., the smallest eigenvalue does not decrease.

LEMMA 4. *There exists a sufficiently small $k = k(d)$ such that if $\lambda_d(A_t) \leq k\epsilon^2$ and $x_t' A_t x_t > \frac{1}{4}\epsilon^2$, then $\lambda_d(A_{t+1}) \geq \lambda_d(A_t)$, i.e., the smallest eigenvalue does not decrease after the update. In addition, one can take $k = \frac{1}{400d^2}$.*

Proof. In this proof, we assume $d \geq 2$. Note that the lemma trivially holds for $d = 1$. Using the same notation as in the proof of Lemma 3, we need to show that $\lambda_d(A_{t+1}) = \frac{d^2}{d^2-1}\sigma_d \geq \lambda_d(A_t)$, where σ_d is the smallest eigenvalue of $\tilde{D}$. To prove that $\sigma_d \geq \frac{d^2-1}{d^2}\lambda_d(A_t)$, it is sufficient to show that $\varphi_{\tilde{D}}\left(\frac{d^2-1}{d^2}\lambda_d\right) \geq 0$ by using Lemma 2. Note that $\varphi_{\tilde{D}}\left(\frac{d^2-1}{d^2}\lambda_d\right) \geq 0$ holds if and only if $\hat{\varphi}_{\tilde{D}}\left(\frac{d^2-1}{d^2}\lambda_d\right) \geq 0$ since $\frac{d^2-1}{d^2}\lambda_d < \lambda_d$. Therefore, the remainder of the proof focuses on showing that $\hat{\varphi}_{\tilde{D}}\left(\frac{d^2-1}{d^2}\lambda_d\right) \geq 0$.

We next split the sum in the definition of $\hat{\varphi}_D$ into two parts, depending on whether the eigenvalue λ_i is smaller or larger relative to $\sqrt{k}\epsilon^2$. We obtain:

$$\hat{\varphi}_{\tilde{D}}\left(\frac{d^2-1}{d^2}\lambda_d\right) = 1 - \frac{2}{d+1}\left[\sum_{i:\ \lambda_i \leq \sqrt{k}\epsilon^2} \frac{\lambda_i c_i^2}{\sum_j \lambda_j c_j^2} \frac{1}{1 - \frac{d^2-1}{d^2}\frac{\lambda_d}{\lambda_i}} + \sum_{i:\ \lambda_i > \sqrt{k}\epsilon^2} \frac{\lambda_i c_i^2}{\sum_j \lambda_j c_j^2} \frac{1}{1 - \frac{d^2-1}{d^2}\frac{\lambda_d}{\lambda_i}}\right].$$

To bound the previous expression, we use some bounds on the eigenvalues. For the first sum, we know that $\lambda_i \leq \sqrt{k}\epsilon^2$, $\sum_j \lambda_j c_j^2 > \frac{1}{4}\epsilon^2$, and $\lambda_i \geq \lambda_d$. For the second sum, we use $\lambda_i \geq \sqrt{k}\epsilon^2 = \frac{k\epsilon^2}{\sqrt{k}} \geq \frac{\lambda_d}{\sqrt{k}}$. Therefore, we obtain:

$$\hat{\varphi}_{\tilde{D}}\left(\frac{d^2-1}{d^2}\lambda_d\right) \geq 1 - \frac{2}{d+1}\left[\sum_{i:\ \lambda_i \leq \sqrt{k}\epsilon^2} \frac{\sqrt{k}\epsilon^2 c_i^2}{\frac{1}{4}\epsilon^2}\frac{1}{1-\frac{d^2-1}{d^2}} + \sum_{i:\ \lambda_i > \sqrt{k}\epsilon^2} \frac{\lambda_i c_i^2}{\sum_j \lambda_j c_j^2}\frac{1}{1-\frac{d^2-1}{d^2}\sqrt{k}}\right]$$
$$\geq 1 - \frac{2}{d+1}\left[4d^2\sqrt{k} + \frac{1}{1-\frac{d^2-1}{d^2}\sqrt{k}}\right].$$

The last inequality follows from the facts that $\sum_i c_i^2 \leq 1$ and $\sum_i \frac{\lambda_i c_i^2}{\sum_j \lambda_j c_j^2} = 1$. In the limit when $k \to 0$, the above expression approaches $1 - \frac{2}{d+1}$, and hence is positive. Consequently, there exists a sufficiently small $k = k(d)$ such that $\hat{\varphi}_D\left(\frac{d^2-1}{d^2}\lambda_1\right) \geq 0$. This concludes the proof of existence. By substituting $k = 1/400d^2$ in the final bound of $\hat{\varphi}_{\tilde{D}}\left(\frac{d^2-1}{d^2}\lambda_d\right)$, inspecting the first few values of d and the derivative, we conclude that taking $k = 1/400d^2$ is enough. □

The intuition behind Lemma 4 is as follows. Assume λ_d is sufficiently small ($\lambda_d \leq k\epsilon^2$). If x_t is equal to the eigenvector that corresponds to the smallest eigenvalue, the algorithm will choose to exploit (thus preserving the ellipsoid). If x_t is not far from this eigenvector, the algorithm still chooses an exploit price. More generally, any x_t that is approximately a convex combination of eigenvectors associated with small eigenvalues (where small means $\lambda_i \leq \sqrt{k}\epsilon^2$) will induce an exploit step. For the algorithm to choose an explore step, the vector x_t has to be approximately a convex combination of eigenvectors that correspond to large eigenvalues (where large means $\lambda_i > \sqrt{k}\epsilon^2$). However, such an x_t cannot cause the smallest eigenvalue to shrink, as this x_t will be nearly orthogonal to the eigenvectors corresponding to the smallest eigenvalues (see Figure 6 for a 2-dimensional illustration).

Finally, we are in the position of proving Lemma 1, which is the missing piece of our argument:

Proof of Lemma 1. Let $\tilde{E}_1 = E_1$ and $\tilde{E}_n$ be the ellipsoid obtained after the n-th explore step. Let $\tilde{A}_n$ be the matrix defining $\tilde{E}_n$. We will build two bounds on the volume ratio $\text{Vol}\ \tilde{E}_{n+1}/\text{Vol}\ \tilde{E}_1$. The first bound is the usual upper bound from the ellipsoid method (see Section 5.1) given by:

$$\frac{\text{Vol}\ \tilde{E}_{n+1}}{\text{Vol}\ \tilde{E}_1} \leq e^{-\frac{n}{2d}}.$$

Next, we construct a lower bound by using the previous lemmas. Since $\tilde{E}_1$ lies in the ball of radius R, we know that $\text{Vol}\ \tilde{E}_1 \leq V_d \cdot R^d$, for a constant V_d defined in Section 5.1. For $\tilde{E}_{n+1}$, we can use:

$$\text{Vol}\ \tilde{E}_{n+1} = V_d \cdot \sqrt{\prod_i \lambda_i(\tilde{A}_{n+1})} \geq V_d \cdot \lambda_d(\tilde{A}_{n+1})^{d/2}.$$

From Lemma 4, when the smallest eigenvalue is below $\epsilon^2/400d^2$, it cannot shrink further. Also, from Lemma 3, whenever the smallest eigenvalue shrinks, it has to shrink by at most $d^2/(d+1)^2$, and hence, at any given time:

$$\lambda_d(\tilde{A}_{n+1}) \geq \frac{d^2}{(d+1)^2} \cdot \frac{\epsilon^2}{400d^2} = \frac{\epsilon^2}{400(d+1)^2}.$$

Therefore, we have:

$$\textsc{Vol}\ \tilde{E}_{n+1} \geq V_d \cdot \left(\frac{\epsilon}{20(d+1)}\right)^d.$$

The ratio of those two expressions gives us a bound on the volume decrease. Putting the two bounds together, we obtain:

$$\left(\frac{\epsilon}{20R(d+1)}\right)^d \leq \frac{\textsc{Vol}\ \tilde{E}_{n+1}}{\textsc{Vol}\ \tilde{E}_1} \leq e^{-\frac{n}{2d}},$$

which implies that the number of explore steps satisfies $n \leq 2d^2 \ln\left(\frac{20R(d+1)}{\epsilon}\right)$. □

6. Noisy Valuations

Up until now, we have assumed that the market value of product t is determined according to a linear model, that is, $v_t = \theta' x_t$. In this section, we extend the model to allow for idiosyncratic additive noise: $v_t = \theta' x_t + \delta_t$, where δ_t is an i.i.d. zero-mean random variable representing an error in our estimate of v_t.

In this noisy model, the original regret definition, $\textsc{Regret} = \sum_{t=1}^{T} v_t - p_t I\{v_t \geq p_t\}$, becomes overly demanding and no algorithm (even in the one-dimensional context-free case) can achieve sublinear regret. The natural approach is to compare against a benchmark that knows the value of θ but not the realization of δ_t. Therefore, we can redefine the regret as follows:

$$\textsc{Regret} = \mathbb{E}\sum_{t=1}^{T}\left[\max_{p_t^*}\left(p_t^* \cdot \Pr_{\delta_t}(\theta' x_t + \delta_t \geq p_t^*)\right) - p_t \cdot I\{v_t \geq p_t\}\right],$$

where the expectation is taken over both the noise and any randomness used by the algorithm.

Assumption. Throughout this section, we assume that the distribution of the noise δ_t is fixed over time, known, and σ-subgaussian, which are common assumptions for tractability. With this assumption, we can focus on learning the weights for each feature instead of learning the noise distribution itself. We say that a distribution is σ-subgaussian if $\Pr(|\delta_t| > t) \leq e^{-t^2/(2\sigma^2)}$ for all $t > 0$.[6]

Before analyzing the contextual case, it is instructive to start with the one-dimensional version of our problem, where the valuation is simply $v_t = v + \delta_t$ for a fixed $v \in \mathbb{R}_+$. Kleinberg and Leighton

[6] This is a common assumption in the literature. The Gaussian distribution is σ-subgaussian for its standard deviation σ. In addition, uniform, Rademacher, and bounded random variables are all subgaussian for suitable σ parameters.

(2003) proved that no learning algorithm can obtain $o(\sqrt{T})$ regret and also show a matching upper bound of $O(\sqrt{T})$ under certain assumptions on the distribution of δ_t. Given this lower bound, we will generally aim for sublinear, but not logarithmic, regret in the noisy contextual case.

Returning to our setting, we will show that the ellipsoid technique is useful in designing feature-based pricing algorithms with noisy valuations. In particular, we will construct two algorithms, ShallowPricing and EllipsoidEXP4. ShallowPricing is a robust version of EllipsoidPricing that will allow us to be robust to low noise ($\sigma = O(1/T \ln T)$) without performance degradation. EllipsoidEXP4 is a combination of ShallowPricing and EXP4, a standard contextual bandit algorithm from the literature. We will prove a regret bound of $O\left(d^{5/2}\ln(T/d)\cdot\left[1+T^{2/3}d^{2/3}(\sigma\ln(T))^{1/3}\sqrt{\ln(T/\sigma)}\right]\right)$ for EllipsoidEXP4. This regret bound is logarithmic in T when the noise vanishes (i.e., $\sigma \to 0$), and has approximately the same dependence in T as EXP4, i.e., $\tilde{O}(T^{2/3})$ for high-noise settings ($\sigma = O(1)$). In addition, under moderate noise such as $\sigma = O(1/\sqrt{T})$ or $\sigma = O(T^{-2/3})$, we incur $\tilde{O}(\sqrt{T})$ or $\tilde{O}(T^{4/9})$ regret, respectively; that is, lower than EXP4. The EllipsoidEXP4 algorithm uses ShallowPricing to localize the solution to a narrow region, and then applies EXP4 on the localized region. The performance boost comes from the fact that the regret of EXP4 depends heavily on the number of possible actions the learner can choose from. By first localizing the solution to a narrow region using the ellipsoid method, we can run the algorithm with a smaller set of actions, and hence improve the regret.

Though we explicitly combine ShallowPricing with EXP4 to obtain a regret bound, our approach is generic in the sense that we could replace EXP4 with other contextual bandits algorithms. For example, if the features are i.i.d. over time, we could use a stochastic gradient descent algorithm as in Amin et al. (2014) instead of EXP4 and obtain a similar regret bound.

Our approach will be as follows. In Section 6.1, we introduce ShallowPricing, prove that it does not incur too many explore steps, and show a regret bound for low-noise environments. In Section 6.2, we analyze the EXP4 algorithm and show how to obtain $\tilde{O}(d^{1/3}T^{2/3})$ regret under either a noiseless or noisy regime. We then combine the two algorithms to produce EllipsoidEXP4 and prove its regret bound in Section 6.3.

6.1. A Robust Version of EllipsoidPricing

We now propose a version of EllipsoidPricing that is designed to offer some protection against noise. Recall that δ_t is assumed to be σ-subgaussian. We define

$$\delta = \sqrt{2}\sigma \ln T, \tag{7}$$

Then, $\Pr(|\delta_t| > \delta) \le e^{-\ln^2 T}$. Using the union bound, we can write $\Pr(|\delta_t| > \delta \text{ for some } t = 1,..,T) \le Te^{-\ln^2 T}$ and

$$\Pr(|\delta_t| \le \delta \text{ for all } t = 1,..,T) \ge 1 - Te^{-\ln^2 T} \ge 1 - 1/T, \tag{8}$$

where the last inequality holds for $T \geq 8$. That is, with probability at least $1 - 1/T$ all the noise terms δ_t are bounded by δ in absolute value. Therefore, if we use a buffer of size δ when adding cuts to the ellipsoid, we are unlikely to remove the true θ from the uncertainty set.

We will add a buffer in the following way. When we choose a price p_t and observe a sale, we can no longer infer that $\theta' v_t \geq p_t$. Instead, we will infer that $\theta' v_t \geq p_t - \delta$. Similarly, in the event of a no sale, all that we will infer is $\theta' v_t \leq p_t + \delta$. We will call the version of our algorithm that adds these buffers ShallowPricing.

ShallowPricing will keep uncertainty sets K_t in the form of ellipsoids and for each vector x_t, it will compute $\underline{b}_t$ and $\overline{b}_t$ in the same way as EllipsoidPricing. For a given parameter $\epsilon \geq 4d\delta$, if $\overline{b}_t - \underline{b}_t \geq \epsilon$, the algorithm will suggest the price $p_t = \frac{1}{2}(\underline{b}_t + \overline{b}_t)$. If $\overline{b}_t - \underline{b}_t \geq \epsilon$ and a sale occurs, we remove elements from the uncertainty set as if we had used the price $p_t - \delta$, i.e., $K_{t+1} = K_t \cap \{\theta \in \mathbb{R}^d : \ \theta' x_t \geq p_t - \delta\}$. Similarly, when a sale does not occur, we remove elements from the set as if we had used the price $p_t + \delta$, i.e., $K_{t+1} = K_t \cap \{\theta \in \mathbb{R}^d : \ \theta' x_t \leq p_t + \delta\}$. If $\overline{b}_t - \underline{b}_t < \epsilon$, we do not update the uncertainty set. ShallowPricing also reduces the exploit prices from EllipsoidPricing by δ, using $p_t = \underline{b}_t - \delta$. The cuts we used in Section 5 remove half of the volume of the ellipsoid and are called central cuts. The cuts we propose here remove less than half of the volume of the ellipsoid and are called shallow cuts. Figure 7 illustrates the difference between central and shallow cuts.

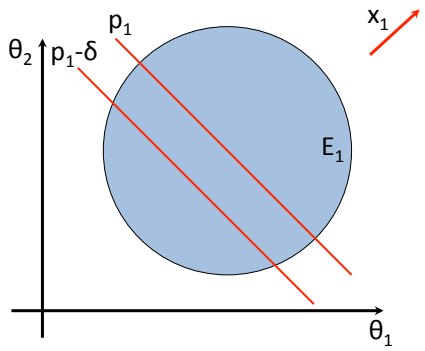


Figure 7 **If we remove the half-ellipsoid below p_1, we are performing a central cut of the ellipsoid E_1. If we remove only the subset below $p_1 - \delta$, we are performing a shallow cut of E_1.**

To analyze ShallowPricing, we need to introduce the concept of the depth of a cut, which is given by:

$$\alpha_t = -\frac{\delta}{\sqrt{x_t' A_t x_t}}.$$

The depth of a cut is a number between -1 and 0, where -1 represents a supporting hyperplane of the ellipsoid and 0 represents a central cut of the ellipsoid. Our analysis does not involve the third

type of standard cuts, the deep cut, which is a cut that removes more than half of the volume of the ellipsoid and thus, has positive depth.

For ShallowPricing to work, the depth of the cuts has to be at least $-1/d$. With $\alpha_t \geq -1/d$, the following machinery allows us to compute the Löwner-John ellipsoids of the sets that remain after shallow cuts (see Eq. (3.1.17) in Grötschel et al. 1993). The Löwner-John ellipsoid of the set $K_{t+1} = E(A_t, a_t) \cap \{\theta \in \mathbb{R}^d : \ \theta' x_t \geq (\underline{b}_t + \overline{b}_t)/2 - \delta\}$ is given by $E(A_{t+1}, a_t + \frac{1+d\alpha_t}{d+1} b_t)$, where $b_t = A_t x_t / \sqrt{x_t' A_t x_t}$ and

$$A_{t+1} = \frac{d^2}{d^2-1}(1-\alpha_t^2)\left(A_t - \frac{2(1+d\alpha_t)}{(d+1)(1+\alpha_t)} b_t b_t'\right). \tag{9}$$

Similarly, the Löwner-John ellipsoid of the set $K_{t+1} = E(A_t, a_t) \cap \{\theta \in \mathbb{R}^d : \ \theta' x_t \leq (\underline{b}_t + \overline{b}_t)/2 + \delta\}$ is given by $E(A_{t+1}, a_t - \frac{1+d\alpha_t}{d+1} b_t)$.

Note that Eq. (9) is not all that different from Eq. (4). We can therefore adapt our analysis for central cuts to allow for shallow cuts. We are now ready to present a bound of the number of explore steps used by ShallowPricing. Recall from Eq. (8) that with probability at least $1 - 1/T$, all the noise terms δ_t are bounded by δ.

Theorem 3. *If $\delta_t \leq \delta$ for all t, then the* ShallowPricing *algorithm with parameter $\epsilon = \max\{Rd^2/T, 4d\delta\}$ will observe $\overline{b}_t - \underline{b}_t > \epsilon$ in at most $O(d^2 \ln(\min\{T/d, R/\delta\}))$ steps.*

Proof. The proof of this result closely mimics the proof of Theorem 2. Therefore, instead of repeating all the steps in the proof of Theorem 2 and its intermediary lemmas, we restrict ourselves to pointing out the necessary changes.

Let N be the number of steps with $\overline{b}_t - \underline{b}_t > \epsilon$. We call such a step an exploration step. To bound this quantity, we first need to show that Lemmas 3 and 4 still apply in the noisy setting. We next show that for any exploration step t,

$$\lambda_d(A_{t+1}) \geq \frac{d^2(1-\alpha_t)^2}{(d+1)^2} \lambda_d(A_t), \tag{10}$$

which is a shallow-cut equivalent of Lemma 3. We define the matrix $D = A_t - \frac{2(1+d\alpha_t)}{(d+1)(1+\alpha_t)} b_t b_t'$ so that $A_{t+1} = \frac{d^2}{d^2-1}(1-\alpha_t^2) D$ according to Eq. (9). We then perform the same change of basis as in the proof of Lemma 3 to define $\tilde{D}$. Eq. (10) is equivalent to $\frac{d^2}{d^2-1}(1-\alpha_t^2)\lambda_d(\tilde{D}) \geq \frac{d^2(1-\alpha_t)^2}{(d+1)^2}\lambda_d(A_t)$, which is itself equivalent to showing that

$$\lambda_d(\tilde{D}) \geq \frac{(1-\alpha_t)(d-1)}{(1+\alpha_t)(d+1)} \lambda_d(A_t).$$

Using Lemma 2, we can prove the statement above by showing that

$$\hat{\varphi}_{\tilde{D}}\left(\frac{(1-\alpha_t)(d-1)}{(1+\alpha_t)(d+1)}\lambda_d(A_t)\right) = 1 - \frac{2(1+d\alpha_t)}{(d+1)(1+\alpha_t)} \sum_i \frac{\tilde{b}_i^2}{\lambda_i(A_t) - \frac{(1-\alpha_t)(d-1)}{(1+\alpha_t)(d+1)}\lambda_d(A_t)} \geq 0,$$

where $\tilde{b}_i$ is as defined in the proof of Lemma 3. The lowest possible value of the right-hand side of the equation above occurs when $\tilde{b}_d^2 = \lambda_d(A_t)$ and $\tilde{b}_i^2 = 0$ for $i \neq d$. Thus,

$$\hat{\varphi}_{\bar{D}}\left(\frac{(1-\alpha_t)(d-1)}{(1+\alpha_t)(d+1)}\lambda_d(A_t)\right) \geq 1 - \frac{2(1+d\alpha_t)}{(d+1)(1+\alpha_t)}\frac{1}{1-\frac{(1-\alpha_t)(d-1)}{(1+\alpha_t)(d+1)}} = 0,$$

proving Eq. (10). This equation immediately implies the weaker statement $\lambda_d(A_{t+1}) \geq \frac{d^2}{(d+1)^2}\lambda_d(A_t)$ since $\alpha_t \leq 0$.

We next prove a shallow-cut equivalent of Lemma 4. We argue that for a sufficiently small $k = k(d)$ such that if $\lambda_d(A_t) \leq k\epsilon^2$ and $x_t' A_t x_t > \frac{1}{4}\epsilon^2$, $\lambda_d(A_{t+1}) \geq \lambda_d(A_t)$. As in Lemma 4, one can take $k = \frac{1}{400d^2}$. To show this result, it is sufficient to prove that

$$\hat{\varphi}_{\bar{D}}\left(\frac{(d^2-1)}{d^2(1-\alpha_t^2)}\lambda_d(A_t)\right) \geq 0.$$

We can mimic the proof of Lemma 4 to obtain

$$\hat{\varphi}_{\bar{D}}\left(\frac{(d^2-1)}{d^2(1-\alpha_t^2)}\lambda_d(A_t)\right) \geq 1 - \frac{2(1+d\alpha_t)}{(1+d)(1+\alpha_t)}\left[\frac{4\sqrt{k}}{1-\frac{d^2-1}{d^2(1-\alpha_t^2)}} + \frac{1}{1-\frac{d^2-1}{d^2(1-\alpha_t^2)}\sqrt{k}}\right]. \tag{11}$$

Since we assumed that $\epsilon \geq 4d\delta$ and we know that $\sqrt{x_t' A_t x_t} \geq \frac{1}{2}\epsilon$, by the definition of α_t we have $\alpha_t = -\delta/\sqrt{x_t' A_t x_t} \geq -2\delta/\epsilon \geq -1/2d$. Since $\alpha_t \in [-1/2d, 0]$, the quantity inside the square brackets in Eq. (11) converges to 1 when k goes to infinity. The limit as k goes to infinity of the right-hand side of the inequality above is therefore $1 - \frac{2(1+d\alpha_t)}{(1+d)(1+\alpha_t)}$, which is strictly positive when $d > 1$ (as in the proof of Lemma 4, we ignore the trivial case of $d = 1$). We thus reach our desired result.

We have now proved that Lemmas 3 and 4 still apply in the noisy setting. We are thus ready to prove our theorem. Just as in Theorem 2, our core argument is that the volume of the ellipsoid decreases exponentially fast in the number of explore steps, and that Lemmas 3 and 4 together provide a bound on the smallest possible volume of the ellipsoid. If we use an explore price at step t, the following volume decrease bound applies under a shallow cut:

$$\frac{\text{Vol } E_{t+1}}{\text{Vol } E_t} \leq e^{-\frac{(1+d\alpha_t)^2}{5d}},$$

as shown in Eq. (3.3.21) of Grötschel et al. (1993). Since $\alpha_t \geq -1/2d$,

$$\frac{\text{Vol } E_{t+1}}{\text{Vol } E_t} \leq e^{-\frac{1}{20d}}.$$

We can therefore repeat the proof of Lemma 1, with the sole difference that we replace $e^{-n/2d}$ by $e^{-n/20d}$. We then obtain a bound on the number of explore steps N:

$$N \leq 20d^2 \ln\left(\frac{20R(d+1)}{\epsilon}\right).$$

By using $\epsilon = \max\{Rd^2/T, 4d\delta\}$, we obtain $N = O(d^2 \ln(\min\{T/d, R/\delta\}))$. □

The result of Theorem 3 immediately implies that for $\sigma = O(1/T\ln T)$, the regret of ShallowPricing is $O(Rd^2\ln(T/d))$. That is, under low noise, we recover the same regret bound as in the noiseless regime (Theorem 2).

Corollary 1. *Suppose* $\sigma = O\left(\frac{Rd}{T\ln T}\right)$. *Then, the worst-case regret of the* ShallowPricing *algorithm with parameter* $\epsilon = Rd^2/T$ *is* $O(Rd^2\ln(T/d))$.

Proof. Let δ be as defined in Eq. (7). By Eq. (8), there is at most probability $1/T$ that $|\delta_t| > \delta$ for some t. Therefore, the regret incurred by potentially removing θ from the uncertainty set is at most $RT/T = R$. The regret incurred by the explore steps is R times the number of explore steps as given by Theorem 3, which is equal to $O(Rd^2\ln(T/d))$ by the right choice of δ. The regret incurred by exploit steps is at most $T(\epsilon+\delta)$ since $\epsilon+\delta$ is the maximum loss from a sale per round. By the choices of ϵ and δ, this regret term is also bounded by $O(Rd^2\ln(T/d))$, completing the proof. □

However, under a high-noise setting (e.g., $\sigma = O(1)$), the performance of ShallowPricing deteriorates. In particular, adding a buffer to all exploit prices becomes too costly. To address this issue, one needs to adapt the algorithm to allow some learning to take place also during the exploitation periods. With this motivation in mind, we will show how to combine ShallowPricing with EXP4 to achieve a better regret bound in a high-noise setting. Before doing so, we turn our attention to EXP4 and show how to apply it to our feature-based pricing problem.

6.2. Applying EXP4 to Our Feature-Based Pricing Problem

We now focus on EXP4 of Auer et al. (2002), the oldest and best-known algorithm for adversarial contextual bandits. While the ideas we use originate from Auer et al. (2002), we will present them using the modern language in Section 4.2.1 of the survey by Bubeck and Cesa-Bianchi (2012). EXP4 is a regret-optimal general purpose bandit algorithm, but one that is computationally inefficient. In this subsection, we will instantiate EXP4 to our problem and study its regret performance. In Section 6.3, we will propose an algorithm that combines our ellipsoid technique with EXP4.

The generic setup of EXP4 comprises a space $\mathcal{X}$ of contexts, a space $\mathcal{A}$ of actions, and a set Π of policies. Each policy $\pi \in \Pi$ is a mapping $\pi : \mathcal{X} \to \mathcal{A}$. In each step t, the adversary chooses a context $x_t \in \mathcal{X}$ and a function $r_t(\cdot)$ mapping actions $a \in \mathcal{A}$ to rewards. The learner observes the context x_t but not the function $r_t(\cdot)$. The learner chooses an action a_t and obtains the reward $r_t(a_t)$. The regret is defined as:

$$\textsc{Regret} = \mathbb{E}\left[\max_{\pi\in\Pi}\sum_t r_t(\pi(x_t)) - \sum_t r_t(a_t)\right],$$

where the expectation is taken over the choice of a_t by the algorithm.

The EXP4 algorithm maintains weights $w_t(\pi)$ for each policy $\pi \in \Pi$, which are initialized as $w_1(\pi) = 1$ for all policies. For each t, a policy π is drawn with probability proportional to $w_t(\pi)$

and the algorithm chooses according to the recommendation $a_t = \pi(x_t)$ given by this policy, and then observes the reward $r_t(a_t)$ for the chosen action. Subsequently, the algorithm comes up with the following unbiased estimator of the function $r_t(\cdot)$ for each policy:

$$\tilde{r}_t(\pi) = \begin{cases} r_t(a_t) \cdot \dfrac{\sum_{\pi \in \Pi} w(\pi)}{\sum_{\pi:\ \pi(x_t)=a_t} w(\pi)} & \text{if } \pi(x_t) = a_t; \\ 0 & \text{otherwise,} \end{cases}$$

and uses $\tilde{r}_t(\pi)$ to update the weights according to:

$$w_{t+1}(\pi) = w_t(\pi) \cdot \exp(\eta \cdot \tilde{r}_t(\pi)),$$

for a given fixed parameter $\eta > 0$. The total regret guarantee is given by:

$$\textsc{Regret} \leq \frac{\ln(|\Pi|)}{\eta} + \frac{\eta}{2} \sum_t |\{\pi(x_t) :\ \pi \in \Pi\}|. \tag{12}$$

The regret bound in Eq. (12) is drawn from the last equation in the proof of Theorem 4.2 in the survey by Bubeck and Cesa-Bianchi (2012) (the original source of this result is Auer et al. (2002)). There are two minor differences between Eq. (12) and the one in Bubeck and Cesa-Bianchi. First, our parameter η is a constant while they allow it to change over time. Second, we replaced the total number of available actions with the number of actions that may be selected by any policy, that is, $|\{\pi(x_t) :\ \pi \in \Pi\}|$.

Since $\{\pi(x_t) :\ \pi \in \Pi\} \subseteq \mathcal{A}$, the last term is bounded by $|\mathcal{A}|$ leading to the overall bound of $\sqrt{2T\,|\mathcal{A}|\ln(|\Pi|)}$ given the appropriate choice of η. We refer to the excellent survey in Bubeck and Cesa-Bianchi (2012) for a modern version of the regret analysis of EXP4.

6.2.1. Instantiating EXP4 for the noiseless regime. We start by instantiating EXP4 for the noiseless case ($\delta_t = 0$ for all t) of the dynamic pricing problem. Both the action and context spaces are continuous in our original feature-based pricing problem. Thus, to get a reasonable guarantee, we will need to discretize them by balancing the error induced by the discretization and the size of the discretized action and context spaces.

For a fixed discretization parameter $\gamma \geq 0$, we define the discretization operator $\lfloor \cdot \rfloor_\gamma$ as follows. For any real number $y \in \mathbb{R}$, we let:

$$\lfloor y \rfloor_\gamma = \gamma \cdot \lfloor y/\gamma \rfloor.$$

For a vector $y \in \mathbb{R}^d$, we define $\lfloor y \rfloor_\gamma$ as the $\mathbb{R}^d$ vector obtained by applying $\lfloor \cdot \rfloor_\gamma$ to each component, i.e., $(\lfloor y \rfloor_\gamma)_i = \lfloor y_i \rfloor_\gamma$. Finally, for a set $K \subseteq \mathbb{R}^d$, we define $\lfloor K \rfloor_\gamma = \{\lfloor y \rfloor_\gamma :\ y \in K\}$.

Having defined the discretization operator, we are now ready to instantiate EXP4. For the remainder of this section, we assume $K_1 = [0,1]^d$ to keep the discretization arguments clean. We

associate a policy with each vector $\theta \in \lfloor [0,1]^d \rfloor_\gamma$ such that there are $|\Pi| = O(1/\gamma^d)$ policies. For every vector $\theta \in \lfloor [0,1]^d \rfloor_\gamma$, let policy π_θ associate x_t with the following price

$$\pi_\theta(x_t) = \lfloor \theta' x_t \rfloor_{\gamma\sqrt{d}} - \gamma\sqrt{d}.$$

This discretization ensures that policies always post prices that are integral multiples of $\gamma\sqrt{d}$ and thus, $|\mathcal{A}| \leq O((\gamma\sqrt{d})^{-1})$. The EXP4 guarantee in Eq. (12) implies that the performance of the algorithm is away from the performance of the best policy by at most $O\left(\sqrt{T\,|\mathcal{A}|\ln(|\Pi|)}\right) = O\left(\sqrt{\frac{T\sqrt{d}}{\gamma}\ln(1/\gamma)}\right)$. If $\theta \in \mathbb{R}^d$ is the true vector of weights, then the policy that prices according to the discretized policy $\pi_{\lfloor\theta\rfloor_\gamma}$ always sells and incurs a discretization loss of at most $O(\gamma\sqrt{d})$ per iteration with respect to the optimal policy, since $\lfloor\theta\rfloor_\gamma{}' x_t - \theta x_t \leq \|\theta - \lfloor\theta\rfloor_\gamma\| \leq \gamma\sqrt{d}$ so that $\theta' x_t \geq \pi_{\lfloor\theta\rfloor_\gamma}(x_t) \geq \theta' x_t - 2\gamma\sqrt{d}$.

We now need to select a discretization parameter γ that minimizes the sum of the learning loss incurred by EXP4, $O\left(\sqrt{\frac{T\sqrt{d}}{\gamma}\ln(1/\gamma)}\right)$, and the loss due to discretization, $O\left(T\gamma\sqrt{d}\right)$. Choosing $\gamma = (T\sqrt{d})^{-1/3}$, we obtain a regret bound of:

$$O\left(\sqrt{\frac{T\sqrt{d}}{\gamma}\ln\left(\frac{1}{\gamma}\right)}\right) + O\left(T\gamma\sqrt{d}\right) = \tilde{O}\left(T^{2/3}d^{1/3}\right).$$

This discretization is valid as long as $\gamma \in (0,1]$. Compared to EllipsoidPricing, the discretized version of EXP4 offers a better regret bound with respect to d, but a much worse regret bound with respect to T.

6.2.2. Instantiating EXP4 for the noisy regime. The key modification required to adapt the analysis above to the noisy regime is to change the definition of a policy as follows. Define $p^*(v) = \arg\max_p p \cdot \Pr(v + \delta \geq p)$ and for every discretized vector θ define:

$$\pi_\theta(x_t) = p^*\left(\lfloor \theta' x_t \rfloor_{\gamma\sqrt{d}} - \gamma\sqrt{d}\right).$$

To show that the same regret bound of $\tilde{O}\left(T^{2/3}d^{1/3}\right)$ also holds in the noisy case, all we need to do is to show that the total discretization loss is still $O(T\gamma\sqrt{d})$. The following lemma proves this.

Lemma 5. *Consider a random variable δ. Let $p_v^* = \arg\max_p[p \cdot \Pr(v + \delta \geq p)]$. Let also $R_v(p) = p \cdot \Pr(v + \delta \geq p)$. Then, if $v \in [\hat{v} - \epsilon, \hat{v} + \epsilon]$ then: $R_v(p^*_{\hat{v}-\epsilon}) \geq R_v(p_v^*) - 2\epsilon$.*

Proof. We first note that $R_v(p)$ is monotone in v:

$$R_v(p^*_{\hat{v}-\epsilon}) = p^*_{\hat{v}-\epsilon} \cdot \Pr(v + \delta \geq p^*_{\hat{v}-\epsilon}) \geq p^*_{\hat{v}-\epsilon} \cdot \Pr(\hat{v} - \epsilon + \delta \geq p^*_{v-\epsilon}) = R_{\hat{v}-\epsilon}(p^*_{\hat{v}-\epsilon}).$$

We next bound $R_{\hat{v}-\epsilon}(p^*_{\hat{v}-\epsilon})$:

$$R_{\hat{v}-\epsilon}(p^*_{\hat{v}-\epsilon}) \geq R_{\hat{v}-\epsilon}(p_v^* - (v - \hat{v} + \epsilon)) \geq (p_v^* - 2\epsilon)\Pr(\hat{v} - \epsilon + \delta \geq p_v^* - (v - \hat{v} + \epsilon)) \geq R_v(p_v^*) - 2\epsilon.$$

The first inequality follows from the optimality of $p^*_{\hat{v}-\epsilon}$, the second inequality from $v \in [\hat{v} - \epsilon, \hat{v} + \epsilon]$, and the third inequality from the monotonicity of $R_v(p)$. □

6.3. Combining ShallowPricing and EXP4

In this subsection, we introduce a new algorithm, EllipsoidEXP4, which combines ShallowPricing and EXP4. This new algorithm will recover the logarithmic regret with respect to T in the limit as the noise vanishes ($\sigma \to 0$); incur regret similar to EXP4, $\tilde{O}(T^{2/3})$, in high-noise settings ($\sigma = O(1)$); and obtain an intermediate regret performance in moderately noisy settings. Our results are in the spirit of earlier work that smoothly interpolates between deterministic and noisy regimes (Hazan and Kale 2011).[7] The main idea is to use the ellipsoid technique to prune the space of possible policies such that all policies output only a small set of actions per context. With that, we will exploit the last term in Eq. (12) to improve the regret guarantee.

Our algorithm will maintain both an uncertainty set K_t and a set of weights for each discretized point of the uncertainty set, i.e., for a fixed discretization parameter $\gamma > 0$, we will keep a weight $w(\theta)$ for each $\theta \in \lfloor K_t \rfloor_\gamma$. We initialize K_1 as before and $w(\theta) = 1$ for every $\theta \in \lfloor \gamma(K_1) \rfloor_\gamma$. At each point, $\overline{b}_t$ and $\underline{b}_t$ refer to the minimum and maximum possible values of $\theta' x_t$ based on the current uncertainty set. The algorithm has three parameters: a discretization term γ, a shallow cut margin ϵ, and an EXP4 update parameter η.

The EllipsoidEXP4 algorithm uses the same explore prices and ellipsoid updates as ShallowPricing. However, the exploit steps are replaced by EXP4 steps. Whenever an explore step occurs, EXP4 is reinitialized. Since we know from Theorem 3 that the number of explore steps is relatively small, the EXP4 reinitializations are not very costly in terms of regret. EllipsoidEXP4 proceeds as follows. In each period t:

- ShallowPricing exploration: If $\overline{b}_t - \underline{b}_t > \epsilon$, we price at $p_t = \frac{1}{2}(\overline{b}_t + \underline{b}_t)$ and update the uncertainty set according to the ShallowPricing rule. When this occurs, we restart the EXP4 algorithm by resetting to 1 the weights of all policies in $\lfloor \gamma(K_{t+1}) \rfloor_\gamma$.
- EXP4: If $\overline{b}_t - \underline{b}_t \leq \epsilon$, we proceed according to the EXP4 algorithm in the noisy regime with parameter η, by selecting $\theta \in \lfloor \gamma(K_t) \rfloor_\gamma$ with probability proportional to $w(\theta)$, choosing the price $p_t = \pi_\theta(x_t)$ (see the policy definition in Subsection 6.2.2) and updating the weights according to the EXP4 update rule. The uncertainty set is unchanged, i.e., $K_{t+1} = K_t$.

Theorem 4. *Suppose*[8] $K_1 = [0,1]^d$. *Then, the* EllipsoidEXP4 *algorithm with parameters* $\epsilon = O(\max\{d^{5/2}/T, d\sigma \ln(T)\})$, $\eta = \sqrt{\gamma d^{1/2} \ln(1/\gamma)/(\sigma T \ln(T))}$, *and* $\gamma = (d^{1/2} T^{-1} \sigma \ln(T))^{1/3}$ *incurs the following regret:*

$$O\left(d^{5/2} \ln(T/d) \cdot \left[1 + T^{2/3} d^{2/3} (\sigma \ln(T))^{1/3} \sqrt{\ln(T/\sigma)}\right]\right).$$

[7] There is also a line of research that interpolates between stochastic and adversarial bandits (see Bubeck and Slivkins 2012, Lykouris et al. 2018).

[8] This regret bound does not involve R since we assumed $K_1 = [0,1]^d$. Consequently, $R = \sqrt{d}$.

Proof. Recall that with probability $1-1/T$, $|\delta_t| \le \sigma \ln(T)$ for all t, so that ShallowPricing will never remove the true θ from the uncertainty set. There are at most $O(d^2 \ln(T/d))$ iterations in which we use ShallowPricing, so EXP4 is restarted at most as many times. In each consecutive run of EXP4, we can bound the regret by the bound of Eq. (12) plus the additional loss from discretization. By the condition that $\overline{b}_t - \underline{b}_t \le \epsilon$, the policies π_θ will suggest at most $\max(1, \epsilon/(\gamma\sqrt{d})) \le \max(1, \sqrt{d}\sigma \ln(T)/\gamma)$ actions (per round) by using the definition of ϵ. If all policies suggest the same action for all contexts, then the regret with respect to the best policy is equal to zero. Otherwise, we can use Eq. (12) to bound the regret with respect to the best action in $\lfloor \gamma(K_t) \rfloor_\gamma$ by:

$$\frac{\ln(\gamma^{-d})}{\eta} + \frac{\eta T}{2} \cdot \frac{\sqrt{d}\sigma \ln(T)}{\gamma} = O\left(\sqrt{T\gamma^{-1} d^{3/2} \sigma \ln(T) \ln(1/\gamma)}\right),$$

for $\eta = \sqrt{\gamma d^{1/2} \ln(1/\gamma)/(\sigma T \ln(T))}$. The best policy in $\lfloor \gamma(K_t) \rfloor_\gamma$ has regret at most $O(T\gamma\sqrt{d})$ with respect to the optimal policy due to discretization, so the total regret from a sequence of consecutive runs of EXP4 is at most:

$$O\left(\sqrt{T\gamma^{-1} d^{3/2} \sigma \ln(T) \ln(1/\gamma)} + T\gamma\sqrt{d}\right) = O\left(T^{2/3} d^{2/3} (\sigma \ln(T))^{1/3} \sqrt{\ln(T/(\sigma\sqrt{d} \ln T))}\right)$$

for $\gamma = (d^{1/2} T^{-1} \sigma \ln(T))^{1/3}$. This quantity can be bounded by:

$$O\left(T^{2/3} d^{2/3} (\sigma \ln(T))^{1/3} \sqrt{\ln(T/\sigma)}\right).$$

By the guarantee in Theorem 3, there are at most $O(d^{5/2} \ln(T/d))$ runs of ShallowPricing. Thus, there are also at most $O(d^{5/2} \ln(T/d))$ consecutive runs of EXP4, and hence the total regret is bounded by:

$$O\left(d^{5/2} \ln(T/d) \cdot [1 + T^{2/3} d^{2/3} (\sigma \ln(T))^{1/3} \sqrt{\ln(T/\sigma)}]\right). \quad \square$$

In the limit as the noise vanishes ($\sigma \to 0$), EllipsoidEXP4 recovers the performance of EllipsoidPricing. In a high-noise setting ($\sigma = O(1)$), it performs approximately as well as EXP4. In moderately-noisy settings such as $\sigma = O(1/\sqrt{T})$, EllipsoidEXP4 incurs a regret of $\tilde{O}(\sqrt{T})$, which is superior to EXP4's performance. The bounds we derived are potentially not tight. For example, whether a regret bound better than $\tilde{O}(T^{2/3})$ can be obtained in a high-noise setting under adversarial contexts is an open research question.

The reader should note that the technique of localizing the solution to a narrow region using ShallowPricing and then switching to a contextual-bandit algorithm is more broadly applicable, and EXP4 could be replaced by a more computationally efficient algorithm. For example, if the feature vectors are i.i.d., we can replace EXP4 by the stochastic gradient approach of Amin et al.

(2014) and obtain similar regret bounds with a better computational performance. Alternatively, one could replace EXP4 by one of the recent computationally-efficient approaches to contextual bandits by Syrgkanis et al. (2016a,b). This would come at a cost, however, since these algorithms yield a worse regret performance than EXP4. We want to emphasize that this is not a meta-algorithm in which one can plug a generic contextual-bandit algorithm, but rather a general design principle that can be applied to several existing algorithms. For example, we rely on the property that the regret depends on the number of actions that could potentially be the optimal action for any given context (as opposed to the total number of actions). This is fortunately true for all algorithms discussed above.

7. Extensions

In this section, we extend our results to non-linear market value models and to the case where the length of the horizon T is not known in advance.

7.1. Non-Linear Models

So far, we assumed that the market value follows a linear model of the form $v_t = \theta' x_t$. An alternative common model is the logistic regression: $v_t = [1 + \exp(\theta' x_t)]^{-1}$ — see Richardson et al. (2007) and Chakrabarti et al. (2008) for examples where market values are learned from data via logistic regressions. More generally, a basic set of features x_t is often transformed by a feature-map $\phi(\cdot)$ to capture correlations and non-linear dependencies on the features. In applications of hedonic pricing, popular models of market values are (i) the log-log model, i.e., $\ln v_t = \sum_i \theta_i \ln(x_{t,i})$ and (ii) the semi-log (or log-linear) model: $\ln v_t = \theta' x_t$. In all such cases, one can express: $v_t = f(\theta' \phi(x_t))$ for some given functions $f(\cdot)$ and $\phi(\cdot)$. Next, we argue that Theorem 2 can easily be extended to this more general setting:

Proposition 1. *Let f be a non-decreasing and continuous function with Lipschitz constant L over the domain $[-R, R]$. Denote $\bar{f} = f(R)$. Let $\phi(\cdot)$ be a feature map such that $\|\phi(x_t)\| \leq 1$ and let the market value take the form $v_t = f(\theta' \phi(x_t))$. Then, the* EllipsoidPricing *algorithm with parameter $\epsilon = \bar{f} d^2 / LT$ incurs regret $O(\bar{f} L d^2 \cdot \ln(RT/\bar{f} d))$.*

Proof. Denote by $\tilde{x}_t = \phi(x_t)$, so that $v_t = f(\theta' \tilde{x}_t)$. For every exploitation round, we know that the value of $\theta' \tilde{x}_t$ lies in an interval $I_t = [\underline{b}_t, \bar{b}_t]$ of length at most ϵ. The loss by pricing at $f(\underline{b}_t)$ is at most $f(\bar{b}_t) - f(\underline{b}_t) \leq L \cdot (\bar{b}_t - \underline{b}_t) \leq L\epsilon$. Using the trivial loss of $\bar{f}$ in each exploration round, we obtain:

$$\text{Regret} \leq TL\epsilon + \bar{f} \cdot 2d^2 \ln(20R(d+1)/\epsilon) \leq O(\bar{f} \cdot d^2 \ln(RT/\bar{f} d)),$$

where the second inequality follows from taking $\epsilon = \bar{f} d^2 / LT$. □

Note that our Lipschitz assumption is different but somewhat related to the one in the line of work on Lipschitz bandits (see, Kleinberg et al. 2013, and the references therein). In our setting, the unknown is a d-dimensional vector, whereas in Lipschitz bandits the entire mapping from contexts to values is unknown.

7.2. Unknown Horizon

An additional assumption that can be relaxed is the knowledge of the time horizon T. Note that when we set the EllipsoidPricing parameter $\epsilon = Rd^2/T$ in Theorem 2, we need to know the value of T in advance. By using the standard *doubling trick*[9] in online learning, one can make the algorithm agnostic in T at the expense of a constant factor. Consequently, this extends our result to the case where the value of T is unknown. We construct a sequence of phases of doubly exponential size: call phase 0 the first 2^{2^0} time steps, phase 1 the next 2^{2^1} steps and so on, i.e., phase k has 2^{2^k} time steps. In each phase k, we re-start the algorithm (forgetting all of the information gained in the past) and run it with $T = 2^{2^k}$. In other words, for each phase k, we decrease ϵ to $Rd^2/2^{2^k}$ and restart our algorithm.

Proposition 2. *By applying the* EllipsoidPricing *algorithm with $\epsilon = Rd^2/2^{2^k}$ in phase k, we obtain a total regret $O(Rd^2 \ln T)$, while being agnostic about the length of the horizon T.*

Proof. Given T time steps, let $\bar{k}$ be the minimum value such that $\sum_{k=0}^{\bar{k}} 2^{2^k} \geq T$. Therefore, for T time steps, the algorithm will have $\bar{k} \leq \lceil \log_2 \log_2 T \rceil$ phases. The total regret from all the time steps in phase k is at most $O(Rd^2 \ln(2^{2^k})) = O(Rd^2 2^k)$. Therefore, the total regret over all phases is at most $\sum_{k=0}^{\lceil \log_2 \log_2 T \rceil} O(Rd^2 2^k) = O(Rd^2 2^{\log_2 \log_2 T}) = O(Rd^2 \ln T)$. □

8. Computational Experiments

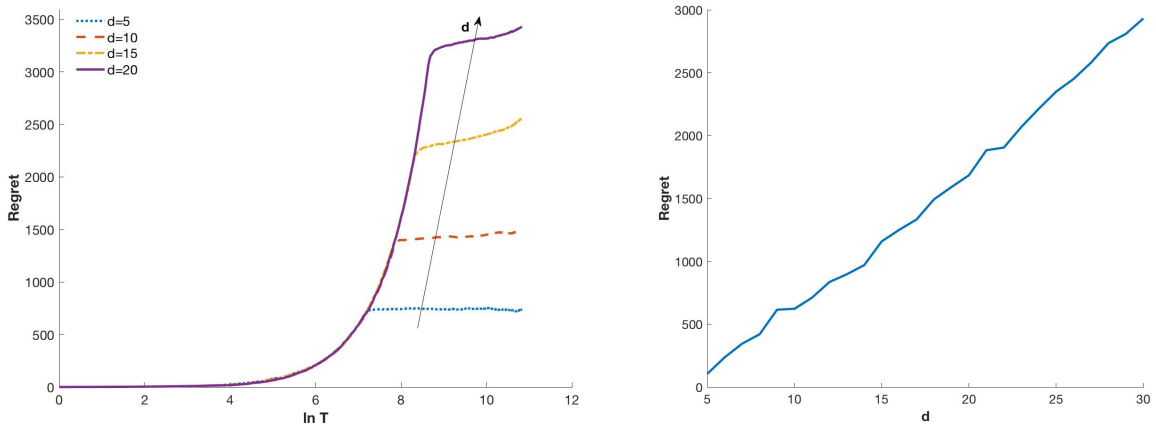
In this section, we computationally test the performance of EllipsoidPricing algorithm. We also compare its regret performance to EXP4's.

8.1. Regret as a Function of $\ln T$ and d

So far, we have considered a setting where nature adversarially selects the vectors of features x_t at each time, as well as the vector θ within a bounded set. Computing an actual optimal (minimax) policy for nature is a hard task, so we test our algorithm in a stochastic environment. We consider the case where nature selects both x_t and θ in an i.i.d. fashion.

We consider a setting where the vectors x_t and θ are drawn i.i.d. from a multivariate Gaussian distribution $N(0, I)$, with each component being replaced by its absolute value, and with the values normalized so that $||x_t|| = 1$ for all t. We also tested several other continuous distributions (e.g.,

[9] For settings like ours where the regret is logarithmic in T, the technique is sometimes called the *squaring trick* since the length of a phase is the square of the length of the previous phase (see Amin et al. 2011).



(a) $d \in \{5, 10, 15, 20\}$ (b) $T = 10,000$

Figure 8 Regret of the EllipsoidPricing algorithm as a function of $\ln T$ and d.

uniform and exponential) and the results in terms of regret happen to be very similar. We vary the value of T between 100 and 50,000 and the value of d between 5 and 30. For simplicity, we use $R = 1$. In Figure 8(a), we plot the regret as a function of $\ln T$ for different values of d; in Figure 8(b), we plot the regret as a function of d with $T = 10,000$. The driving force behind the shape of the regret function in Figure 8(a) is the fact that the scale of the x-axis is logarithmic. For small values of T, the curve looks exponential because the regret is approximately linear in T while the algorithm is mostly learning. When the algorithm switches to mostly exploiting (earning), which occurs for medium and large values of T, the curve becomes linear in $\ln T$, as predicted by our theorems. We also consider a setting where x_t are drawn i.i.d. from a Bernoulli distribution to represent binary features, where we obtain a similar regret as Figure 8 (the plots are omitted for conciseness).

8.2. Adaptability

In the previous cases, the algorithm explores until some threshold time, and then it mostly exploits. The separation between the exploration and exploitation phases follows from the fact that the vectors x_t are drawn from an i.i.d. distribution. We illustrate this phenomenon in Figure 9, where we plot the proportion of exploration rounds as a function of time intervals of length $T/20$. However, this is not always the case. As we mentioned earlier in the paper, our algorithm can explore and exploit without a clear separation between the phases. Depending on the amount of uncertainty in a specific direction, it can decide whether or not to explore. To illustrate this behavior, we test a situation where the setting evolves over time by changing the distribution of x_t after half of the time periods have elapsed.

In what follows, we show that our algorithm can adapt to dynamic environments. We consider two different settings, depicted in Figure 10. Figure 10(a) considers the case where in the first half of

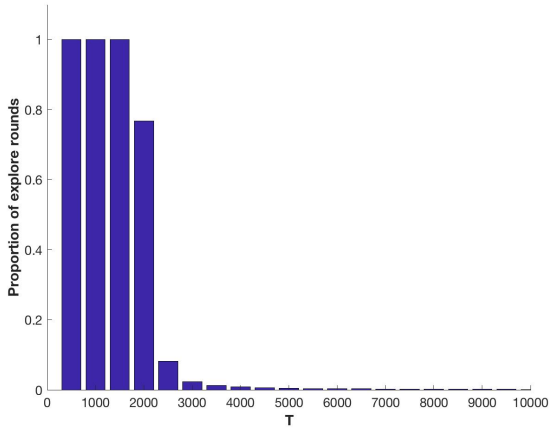
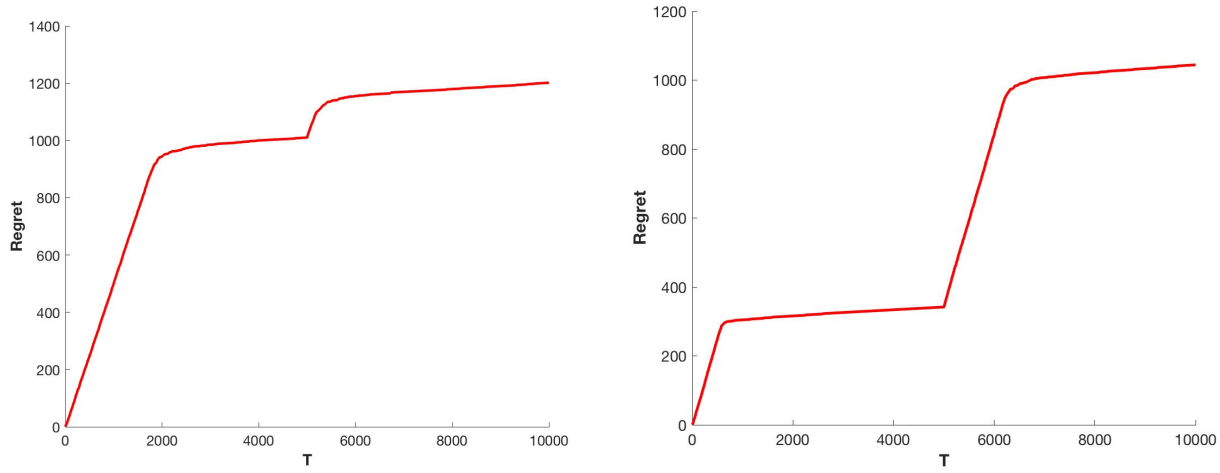


Figure 9 **Explore rounds versus exploit rounds for the EllipsoidPricing algorithm as a function of T for $d = 15$. We show the average results over 100 independent instances.**

the iterations (i.e., during the first $T/2$ time periods), the vectors of features are the absolute values of components drawn from normally distributed $N(0, I)$; and in the second half of the periods, the vectors of features are uniformly distributed $U[-1, 1]^d$ (in both cases, the vector x_t is normalized such that $||x_t|| = 1$ for all t).



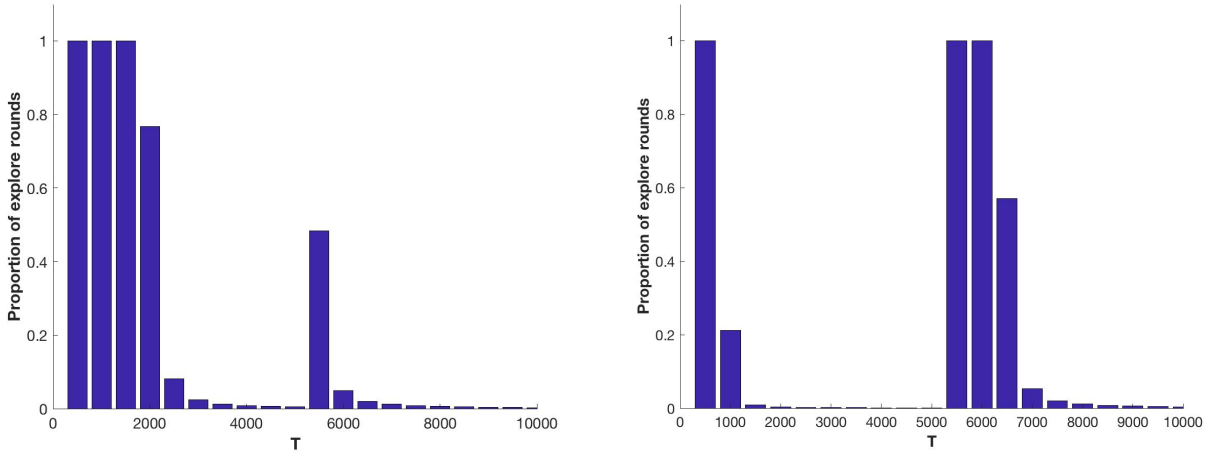
(a) Normal then uniform features.

(b) Half then all features.

Figure 10 **Regret of the EllipsoidPricing algorithm as a function of T for $d = 15$ when the distribution of the features changes at $T/2$. We show the average results over 100 independent instances.**

Figure 10(b) considers the case where the vector is random but in the first half of the iterations, the last half of the components are zero. In other words, we have random values in the first $d/2$ elements and 0 in the second half. After $T/2$, all the d elements of x_t are random. Both before and after, all features are drawn from the same normalized Gaussian distribution. One can see that in

the two different settings, the regret of our algorithm remains low, while adapting to the change in the distribution. In these cases, the algorithm will explore again when needed. Figure 11 shows the algorithm starting to explore again after the change in distributions at $T/2$, under the same settings as in Figure 10. This type of situations is very typical as the vectors of features can depend on external factors such as seasonality and trends.



(a) Normal then uniform features.

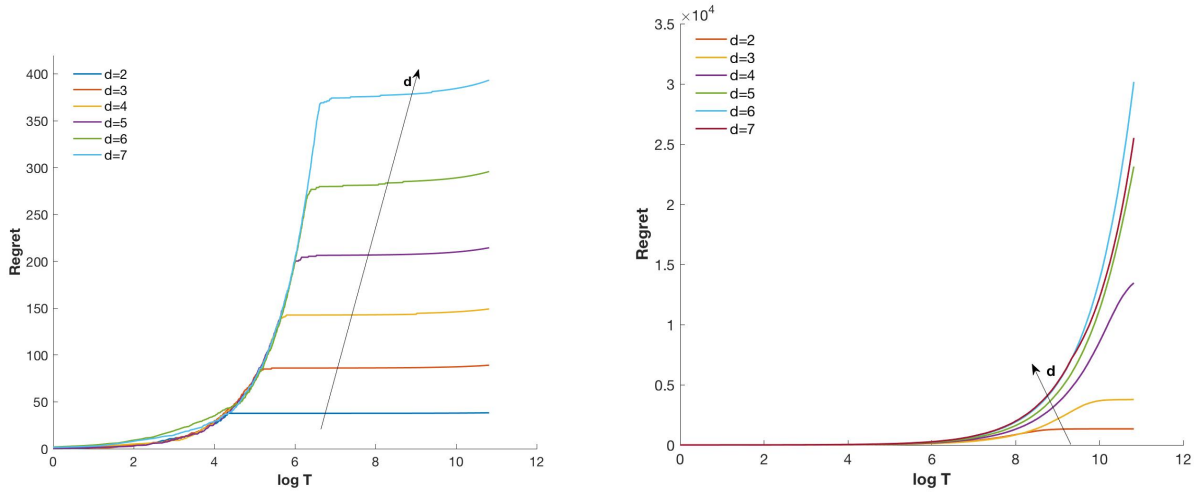
(b) Half then all features.

Figure 11 Explore versus exploit rounds for the EllipsoidPricing algorithm as a function of T for $d = 15$ when the distribution of the features changes at $T/2$. We show the average results over 100 independent instances.

8.3. Comparing Our Algorithm to a Benchmark from the Literature

We next compare the performance of our algorithm to EXP4, a general purpose regret-optimal algorithm for adversarial contextual bandits (Auer et al. 2002). We focus on low values of the dimension $d = 2, \ldots, 7$ given that EXP4 has poor computational performance in high dimensions. For computational convenience, given the discretization used in EXP4, we draw the parameters θ and x_t uniformly at random in $[0,1]^d$. The results of EXP4 depend on a parameter η which represents the learning rate. We plot the results for the best learning rate we could find for each instance.

As one can see from Figure 12, the EllipsoidPricing algorithm yields a significantly smaller regret when compared to EXP4 (the right panel has a y-axis scaled by 10^4). This is expected as EXP4 is a general purpose algorithm whereas EllipsoidPricing is tailored to our problem setting. It is still reassuring to observe that EllipsoidPricing is able to reduce significantly the regret by exploiting the structure of our problem. In addition, its running time is much lower, allowing us to solve the problem for settings with higher dimensions in reasonable timeframes.



(a) Regret of EllipsoidPricing. (b) Regret of EXP4 (the y-axis is scaled by 10^4).

Figure 12 Regret of EllipsoidPricing and EXP4 as a function of $\ln T$ for different values of d. Note that the scales of the y-axes are different in the two plots.

We next test the performance of EllipsoidPricing, ShallowPricing, and EXP4 in a noisy setting. Specifically, we consider an additive noise δ_t drawn i.i.d Gaussian with mean zero and standard deviation $\sigma = 0.1$, $T = 50,000$, and $d = 2, 3, 4, 5$ (as before, given EXP4's computational limitations, we restrict the dimension to be low). We compare the performance of EllipsoidPricing, ShallowPricing, and EXP4 by computing the difference in regret relative to EXP4, i.e., Regret(EXP4) - Regret(EllipsoidPricing) and Regret(EXP4) - Regret(ShallowPricing). The results are presented in Table 1. Both of our algorithms outperform EXP4 even in the noisy setting. We also varied some of the parameters such as T and σ and observed consistent results.

Table 1 Average regret difference relative to EXP4 for $T = 50,000$

	Regret(EXP4) - Regret(EllipsoidPricing)	Regret(EXP4) - Regret(ShallowPricing)
$d = 2$	12,450	11,715
$d = 3$	18,955	19,435
$d = 4$	23,855	26,365
$d = 5$	30,235	30,970

8.4. Numerical Insights

This section allowed us to test and validate computationally the performance of the algorithms proposed in this paper. We draw the following insights:

- Our results are robust to the distributions of both the vector θ and the vectors of features x_t. We tested several different distributions (both continuous and discrete) and observed that the magnitude of the regret attained by our algorithm is robust to the distribution.

- Our algorithm is able to adapt to the data. In particular, if the vectors of features vary in time and have a time-dependent distribution, the algorithm still estimates θ correctly and the regret remains small. This follows from the fact that our algorithm does not separate the exploration and exploitation phases, as in some other classical approaches. Instead, the algorithm always learns from new features and reacts accordingly.
- Our algorithm outperforms EXP4, a general purpose regret-optimal algorithm for adversarial contextual bandits for both noiseless and noisy settings.

9. Conclusions and Future Directions

In this paper, we considered the problem of pricing highly differentiated products described by vectors of features. The firm has to set the price of each product in an online fashion. The market value of each product is linear in the values of the features, and the firm does not initially know the values of the different features. Our goal was to propose an efficient online pricing method by balancing the exploration/exploitation tradeoff to achieve a low regret. We first considered a multi-dimensional version of binary search over polyhedral sets and showed that it has exponential worst-case regret with regard to the dimension of the feature space. We then proposed a modification of the prior algorithm where uncertainty sets are replaced by their Löwner-John ellipsoids. We showed that the algorithm we proposed has a worst-case regret that is quadratic in the dimension of the feature space and logarithmic in the time horizon.

We also proposed two variants of the algorithm that add robustness to noisy valuations: (1) ShallowPricing which is based on shallow cuts of an ellipsoid, allowing us to add a safety margin to each cut and (2) EllipsoidEXP4 which is a combination of ShallowPricing with the standard adversarial contextual-bandits algorithm EXP4. For EllipsoidEXP4, we showed a regret guarantee that (i) matches the bound of EllipsoidPricing as the noise vanishes, (ii) approximately matches the regret guarantee of EXP4 in high-noise settings, and (iii) leads to intermediate regret guarantees in moderately noisy environments.

We would like to end by discussing some future research directions. Closing the gap between our regret bound and the best-known lower bound of $\Omega(d \ln \ln T)$ is an interesting (and challenging) problem that we do not attempt to resolve in this paper. Understanding whether a better regret bound could be achieved in a setting with stochastic rather than adversarial features is another important open problem. A limitation of the EllipsoidEXP4 algorithm is its requirement to know a bound on the noise parameter σ. An interesting extension is to develop an approach for estimating σ. The case of an unknown σ with a gaussian noise is tractable under certain conditions. However, the general case of estimating a σ-subgaussian noise is left as an avenue for future research. One last direction would be to consider a setting where the parameter θ varies over time. For example,

one could consider a problem where, at each period, a vector θ_t takes a different value, but under the assumption of limited variation, that is, $\|\theta_{t+1} - \theta_t\| \leq \Delta_t$.

Acknowledgments

We would like to thank Kevin Jiao, who helped us implement the EXP4 algorithm, and Arash Asadpour for his insightful suggestions regarding the proof of Lemma 6. We also thank Mohsen Bayati, Dirk Bergemann, Omar Besbes, Vahab Mirrokni, Hamid Nazerzadeh, Georgia Perakis, Umar Syed, and the participants of the 2016 Google Market Algorithms Workshop and of the 2016 Utah Winter Operations Conference for their comments and suggestions. Part of this work was completed while the first two authors were hosted by Google Research in New York City. The authors thank Google Research for its generous support.

References

Abbasi-Yadkori, Yasin, Dávid Pál, Csaba Szepesvári. 2011. Improved algorithms for linear stochastic bandits. *Proceedings of NIPS*. 2312–2320.

Agarwal, Alekh, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, Robert E Schapire. 2014. Taming the monster: A fast and simple algorithm for contextual bandits. *Proceedings of ICML*.

Agrawal, Shipra, Nikhil R Devanur. 2015a. Linear contextual bandits with global constraints and objective. Working paper, Columbia University.

Agrawal, Shipra, Nikhil R Devanur. 2015b. Linear contextual bandits with knapsacks. Working paper, Columbia University.

Agrawal, Shipra, Nikhil R. Devanur, Lihong Li. 2016. An efficient algorithm for contextual bandits with knapsacks, and an extension to concave objectives. *Proceedings of COLT*. 4–18.

Amin, Kareem, Rachel Cummings, Lili Dworkin, Michael Kearns, Aaron Roth. 2015. Online learning and profit maximization from revealed preferences. *Proceedings of AAAI*.

Amin, Kareem, Michael Kearns, Umar Syed. 2011. Bandits, query learning, and the haystack dimension. *Proceedings of COLT*. 87–106.

Amin, Kareem, Afshin Rostamizadeh, Umar Syed. 2014. Repeated contextual auctions with strategic buyers. *Proceedings of NIPS*. 622–630.

Araman, Victor F, René Caldentey. 2009. Dynamic pricing for nonperishable products with demand learning. *Operations Research* **57**(5) 1169–1188.

Auer, Peter. 2003. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research* **3** 397–422.

Auer, Peter, Nicolo Cesa-Bianchi, Yoav Freund, Robert E Schapire. 2002. The nonstochastic multiarmed bandit problem. *SIAM journal on computing* **32**(1) 48–77.

Babaioff, Moshe, Shaddin Dughmi, Robert Kleinberg, Aleksandrs Slivkins. 2015. Dynamic pricing with limited supply. *ACM Transactions on Economics and Computation (TEAC)* **3**(1) 4.

Badanidiyuru, Ashwinkumar, Robert Kleinberg, Aleksandrs Slivkins. 2013. Bandits with knapsacks. *Proceedings of FOCS*. 207–216.

Badanidiyuru, Ashwinkumar, John Langford, Aleksandrs Slivkins. 2014. Resourceful contextual bandits. *Proceedings of COLT*. 1109–1134.

Bertsimas, Dimitris, Allison O'Hair. 2013. Learning preferences under noise and loss aversion: An optimization approach. *Operations Research* **61**(5) 1190–1199.

Bertsimas, Dimitris, Phebe Vayanos. 2015. Data-driven learning in dynamic pricing using adaptive optimization. Working Paper, MIT.

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

Besbes, Omar, Assaf Zeevi. 2015. On the (surprising) sufficiency of linear models for dynamic pricing with demand learning. *Management Science* **61**(4) 723–739.

Bland, Robert G, Donald Goldfarb, Michael J Todd. 1981. The ellipsoid method: A survey. *Operations Research* **29**(6) 1039–1091.

Boutilier, Craig, Relu Patrascu, Pascal Poupart, Dale Schuurmans. 2006. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence* **170**(8-9) 686–713.

Bray, Janna. 2017. The price is right. Airbnb (https://airbnb.design/smart-pricing-how-we-used-host-feedback-to-build-personalized-tools).

Broder, Josef, Paat Rusmevichientong. 2012. Dynamic pricing under a general parametric choice model. *Operations Research* **60**(4) 965–980.

Bubeck, Sébastien, Nicolo Cesa-Bianchi. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning* **5**(1) 1–122.

Bubeck, Sébastien, Aleksandrs Slivkins. 2012. The best of both worlds: stochastic and adversarial bandits. *Conference on Learning Theory*. 42–1.

Chakrabarti, Deepayan, Deepak Agarwal, Vanja Josifovski. 2008. Contextual advertising by combining relevance with click feedback. *Proceedings of WWW*. 417–426.

Chen, Yiwei, Vivek F Farias. 2013. Simple policies for dynamic pricing with imperfect forecasts. *Operations Research* **61**(3) 612–624.

Chu, Wei, Lihong Li, Lev Reyzin, Robert E Schapire. 2011. Contextual bandits with linear payoff functions. *Proceedings of AISTATS*. 208–214.

den Boer, Arnoud V, Bert Zwart. 2013. Simultaneously learning and optimizing using controlled variance pricing. *Management Science* **60**(3) 770–783.

Dudik, Miroslav, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, Tong Zhang. 2011. Efficient optimal learning for contextual bandits. *Proceedings of UAI*.

Golub, Gene H. 1973. Some Modified Matrix Eigenvalue Problems. *SIAM Review* **15**(2) 318–334.

Grötschel, Martin, László Lovász, Alexander Schrijver. 1993. *Geometric Algorithms and Combinatorial Optimization*. 2nd ed. Springer-Verlag.

Harrison, J Michael, N Bora Keskin, Assaf Zeevi. 2012. Bayesian dynamic pricing policies: Learning and earning under a binary prior distribution. *Management Science* **58**(3) 570–586.

Hazan, Elad, Satyen Kale. 2011. Better algorithms for benign bandits. *Journal of Machine Learning Research* **12**(Apr) 1287–1311.

Keskin, N Bora, Assaf Zeevi. 2014. Dynamic pricing with an unknown linear demand model: asymptotically optimal semi-myopic policies. *Operations Research* **62**(5) 1142–1167.

Khachiyan, L. G. 1979. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR* **244** 1093–1096.

Kleinberg, Robert, Tom Leighton. 2003. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. *Proceedings of FOCS*. 594–605.

Kleinberg, Robert, Aleksandrs Slivkins, Eli Upfal. 2013. Bandits and experts in metric spaces. *arXiv preprint arXiv:1312.1277* .

Lykouris, Thodoris, Vahab Mirrokni, Renato Paes Leme. 2018. Stochastic bandits robust to adversarial corruptions. *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 114–122.

Malpezzi, Stephen. 2003. *Hedonic pricing models: a selective and applied review*. Section in Housing Economics and Public Policy: Essays in Honor of Duncan Maclennan.

Milon, J Walter, Jonathan Gressel, David Mulkey. 1984. Hedonic amenity valuation and functional form specification. *Land Economics* **60**(4) 378–387.

Phillips, Robert Lewis. 2005. *Pricing and revenue optimization*. Stanford University Press.

Qiang, Sheng, Mohsen Bayati. 2016. Dynamic pricing with demand covariates. Working Paper, Stanford University.

Richardson, Matthew, Ewa Dominowska, Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. *Proceedings of the 16th international conference on World Wide Web*. ACM, 521–530.

Roth, Aaron, Aleksandrs Slivkins, Jonathan Ullman, Zhiwei Steven Wu. 2017. Multidimensional dynamic pricing for welfare maximization. *Proceedings of the 2017 ACM Conference on Economics and Computation*. ACM, 519–536.

Roth, Aaron, Jonathan Ullman, Zhiwei Steven Wu. 2016. Watch and learn: Optimizing from revealed preferences feedback. *Proceedings of STOC*.

Rothschild, Michael. 1974. A two-armed bandit theory of market pricing. *Journal of Economic Theory* **9**(2) 185–202.

Sirmans, Stacy, David Macpherson, Emily Zietz. 2005. The composition of hedonic pricing models. *Journal of Real Estate Literature* **13**(1) 1–44.

Syrgkanis, Vasilis, Akshay Krishnamurthy, Robert Schapire. 2016a. Efficient algorithms for adversarial contextual learning. *International Conference on Machine Learning*. 2159–2168.

Syrgkanis, Vasilis, Haipeng Luo, Akshay Krishnamurthy, Robert E Schapire. 2016b. Improved regret bounds for oracle-based adversarial contextual bandits. *Advances in Neural Information Processing Systems*. 3135–3143.

Toubia, Olivier, John Hauser, Rosanna Garcia. 2007. Probabilistic polyhedral methods for adaptive choice-based conjoint analysis: Theory and application. *Marketing Science* **26**(5) 596–610.

Toubia, Olivier, John R Hauser, Duncan I Simester. 2004. Polyhedral methods for adaptive choice-based conjoint analysis. *Journal of Marketing Research* **41**(1) 116–131.

Toubia, Olivier, Duncan I Simester, John R Hauser, Ely Dahan. 2003. Fast polyhedral adaptive conjoint estimation. *Marketing Science* **22**(3) 273–303.

Viappiani, Paolo, Craig Boutilier. 2009. Regret-based optimal recommendation sets in conversational recommender systems. *Proceedings of the third ACM conference on Recommender systems*. ACM, 101–108.

Wilkinson, James Hardy. 1965. *The Algebraic Eigenvalue Problem*, vol. 87. Clarendon Press Oxford.

Ye, Peng, Julian Qian, Jieying Chen, Chen-hung Wu, Yitong Zhou, Spencer De Mars, Frank Yang, Li Zhang. 2018. Customized regression model for airbnb dynamic pricing. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 932–940.

Online Appendix

To prove Theorem 1, we first prove the following combinatorial lemma.

LEMMA 6. *Let d be a multiple of 8 and $S_1, S_2, S_3, \dots$ be a sequence of uniformly-drawn random subsets of $\{1, \dots, d\}$ of size $d/4$. Then, $t_0 = \Omega(1.2^d)$:*

$$\Pr\left(|S_j \cap S_k| \leq d/8, \forall 1 \leq j < k \leq t_0\right) \geq 1/2.$$

Proof of Lemma 6. Let S_j and S_k be two randomly sampled subsets of the set $\{1, \dots, d\}$ of cardinality $d/4$. Consider a given feasible set S_k (a subset of $\{1, \dots, d\}$ of size $d/4$). Then, there exists exactly $\binom{d}{d/4}$ possibilities for the set S_j. The number of possibilities for the set S_j that share exactly i elements with the set S_k is given by the product of binomials $\binom{d/4}{i}\binom{3d/4}{d/4-i}$. Therefore, the number of possible sets S_j with more than $d/8$ elements from S_k is given by $\sum_{i=d/8+1}^{d/4}\binom{d/4}{i}\binom{3d/4}{d/4-i}$. Thus,

$$\Pr\left(|S_j \cap S_k| > d/8\right) = \sum_{i=d/8+1}^{d/4} \frac{\binom{d/4}{i}\binom{3d/4}{d/4-i}}{\binom{d}{d/4}} \leq O(1.69^{-d}), \tag{13}$$

where we will demonstrate the last inequality shortly. Now, consider a collection of $S_1, S_2, ..., S_{t_0}$ sets where $t_0 = 1.2^d$. Using the union bound, we can write:

$$\begin{aligned}
\Pr\left(|S_j \cap S_k| \leq d/8, \forall 1 \leq j < k \leq t_0\right) &= 1 - \Pr\left(\exists\ j < k \in \{1, ..., t_0\} \mid |S_j \cap S_k| > d/8\right) \\
&\geq 1 - \sum_{1 \leq j \leq t_0} \sum_{1 \leq k < j} \Pr\left(|S_j \cap S_k| > d/8\right) \\
&\geq 1 - (1.2^d)^2 \cdot O(1.69^{-d}) \\
&\geq 1/2 \qquad \text{for sufficiently high } d.
\end{aligned}$$

This proves the statement of the lemma.

We now prove the inequality in Eq. (13). From Stirling inequalities, $\sqrt{2\pi n}(n/e)^n \leq n! \leq e\sqrt{n}(n/e)^n$, we have

$$\binom{d}{d/4} \geq p_1(d)\frac{d^d}{(d/4)^{(d/4)}(3d/4)^{(3d/4)}} = p_1(d)1.755^d, \tag{14}$$

where we use $p_1(d)$ to represent a polynomial function of d. Our next step is to use the same technique to show that the numerator from Eq. (13), $\sum_{i=d/8+1}^{d/4}\binom{d/4}{i}\binom{3d/4}{d/4-i}$, is bounded by $p_2(d)1.038^d$, where $p_2(d)$ is a polynomial function of d. We define:

$$h(d) = \sum_{i=d/8+1}^{d/4} \binom{d/4}{i}\binom{3d/4}{d/4-i}.$$

Using the Stirling inequalities once more, we can bound the quantity above by

$$\begin{aligned} h(d) &= \sum_{i=d/8+1}^{d/4} \frac{(d/4)!}{i!(d/4-i)!} \frac{(3d/4)!}{(d/4-i)!(d/2+i)!} \\ &\leq \tilde{p}_2(d) \sum_{i=d/8+1}^{d/4} \frac{(d/4)^{(d/4)}}{i^i(d/4-i)^{(d/4-i)}} \frac{(3d/4)^{(3d/4)}}{(d/4-i)^{(d/4-i)}(d/2+i)^{(d/2+i)}}, \end{aligned}$$

for some polynomial $\tilde{p}_2(d)$. Note that we can bound the sum by $(d/4-d/8)$ times the highest value of i and therefore

$$h(d) \leq p_2(d) \max_{i\in\{d/8,\ldots,d/4\}} \frac{(d/4)^{(d/4)}}{i^i(d/4-i)^{(d/4-i)}} \frac{(3d/4)^{(3d/4)}}{(d/4-i)^{(d/4-i)}(d/2+i)^{(d/2+i)}},$$

with a slightly different polynomial $p_2(d)$ than before. We now replace i by yd for some $y \in [1/8, 1/4]$ to obtain the following bound:

$$\begin{aligned} h(d) &\leq p_2(d) \max_{y\in[1/8,1/4]} \frac{(d/4)^{(d/4)}}{(yd)^{(yd)}(d/4-yd)^{(d/4-yd)}} \frac{(3d/4)^{(3d/4)}}{(d/4-yd)^{(d/4-yd)}(d/2+yd)^{(d/2+yd)}} \\ &= p_2(d) \max_{y\in[1/8,1/4]} \left[\frac{(1/4)^{(1/4)}(3/4)^{(3/4)}}{y^y(1/4-y)^{(1/2-2y)}(1/2+y)^{(1/2+y)}}\right]^d. \end{aligned} \tag{15}$$

Now consider the function $g(y)$ defined as

$$g(y) = y^y(1/4-y)^{(1/2-2y)}(1/2+y)^{(1/2+y)}.$$

The second derivative of the logarithm of $g(y)$ is equal to

$$\frac{d^2 \ln g(y)}{dy^2} = \frac{-y^2+2.75y+.05}{(y-1)^2y(y+1)},$$

which is positive in the region $[1/8, 1/4]$. Therefore $\ln g(y)$ is strictly convex within this region and thus, has a unique minimizer. Using the first order condition, we can compute this minimizer to be $y = 0.169$. In addition, $g(0.169)$ evaluates to 0.549. Replacing this value in Eq. (15), we obtain:

$$h(d) \leq p_2(d) \left(\frac{.570}{.549}\right)^d = p_2(d)1.038^d. \tag{16}$$

The ratio between the two bounds from Eqs. (14) and (16) gives us

$$\frac{p_2(d)1.038^d}{p_1(d)1.755^d} = O(1.69^{-d}),$$

since the polynomials are absorbed by the exponential terms. This proves Eq. (13), completing our proof. □

Proof of Theorem 1. Consider a setting where $K_1 = [1,2]^d$. For $\Omega(a^d)$ steps, assume that nature draws a random subset S_t of $\{1,\ldots,\tilde{d}\}$ with size $\tilde{d}/4$, where $\tilde{d} = 8\lfloor d/8 \rfloor$. Nature chooses the feature vectors $x_t = \frac{1}{\sqrt{\tilde{d}/4}} I\{S_t\}$, i.e., the i-th coordinate of the vector x_t is $\frac{1}{\sqrt{\tilde{d}/4}}$ if $i \in S_t$, and 0 otherwise. We first show that with probability at least 1/2, the regret incurred over $\Omega(a^d)$ steps is at least $\Omega(Ra^d)$, where a is the constant 1.2 from Lemma 6. We will assume that d is a multiple of 8, and therefore use d instead of $\tilde{d}$.

We divide the proof into two cases depending on the value of ϵ.

1. Assume that $\epsilon < 0.5\sqrt{d}$ and consider the case $\theta = (1,1,\ldots,1)$. We now analyze the event where the pairwise intersection of sets S_t is at most $d/8$. In this case, we have:

$$\min_{\hat{\theta}\in K_1} \hat{\theta}' x_1 = \sqrt{d/4} = 0.5\sqrt{d} \qquad \text{and} \qquad \max_{\hat{\theta}\in K_1} \hat{\theta}' x_1 = \frac{2(d/4)}{\sqrt{d/4}} = \sqrt{d}.$$

The difference is equal to $0.5\sqrt{d}$ and is larger than ϵ, so that our algorithm will explore and set an explore price of $p_1 = 0.75\sqrt{d}$. Since $\theta' x_1 < p_1$, a sale does not occur, and the algorithm incurs a regret of $0.5\sqrt{d}$.

We next claim by induction that for every t, we have:

$$\min_{\hat{\theta}\in K_t} \hat{\theta}' x_t = 0.5\sqrt{d} \qquad \text{and} \qquad \max_{\hat{\theta}\in K_t} \hat{\theta}' x_t = \sqrt{d}.$$

As a result, the price is set to $p_t = 0.75\sqrt{d}$, no sale occurs, and the algorithm incurs a regret of $0.5\sqrt{d}$ in every period. The base case ($t = 1$) was shown above. Assume that the claim is true for t and we next show that it holds for $t+1$.

We have: $K_{t+1} = K_t \cap \{\theta' x_t \le 0.75\sqrt{d}\} = K_1 \cap_{s=1,2,\ldots t} \{\theta' x_s \le 0.75\sqrt{d}\}$. Note that for any $s = 1,2,\ldots,t$, we have: $(1,1,\ldots,1)' x_s = 0.5\sqrt{d}$ and hence, $\theta = (1,1,\ldots,1) \in K_{t+1}$. Therefore, we obtain:

$$\min_{\hat{\theta}\in K_{t+1}} \hat{\theta}' x_{t+1} = (1,1,\ldots,1)' x_{t+1} = 0.5\sqrt{d}.$$

Consider the vector $\widetilde{\theta}$ such that $\widetilde{\theta}_i = 2$ for $i \in S_{t+1}$, and $\widetilde{\theta}_i = 1$ otherwise. If we show that $\widetilde{\theta} \in K_{t+1}$, then we have:

$$\max_{\hat{\theta}\in K_{t+1}} \hat{\theta}' x_{t+1} \ge \widetilde{\theta}' x_{t+1} = \sqrt{d}.$$

Since the maximum over the initial set K_1 is also equal to $\sqrt{d}$, the above maximum cannot be larger than $\sqrt{d}$. The last step is to show that $\widetilde{\theta} \in K_{t+1}$. We know that $\widetilde{\theta} \in K_1$. In addition, we have for any $s = 1,2,\ldots,t$:

$$\begin{aligned}\widetilde{\theta}' x_s = \frac{1}{\sqrt{d/4}} \sum_{i\in S_s} [1 + I\{i \in S_{t+1}\}] &= \frac{1}{\sqrt{d/4}} \left[\frac{d}{4} + |S_s \cap S_{t+1}|\right] \\ &\le \frac{1}{\sqrt{d/4}} \left[\frac{d}{4} + \frac{d}{8}\right] = 0.75\sqrt{d},\end{aligned}$$

where the inequality follows from Lemma 6. Therefore, $\widetilde{\theta} \in K_{t+1}$.

For all $t = 1, 2, ..., k$, our algorithm incurs a regret of $0.5\sqrt{d}$. Recall that we have $k = \Omega(a^d)$ such steps, so that the total regret is given by: $0.5\sqrt{d} \cdot \Omega(a^d) = \Omega(\sqrt{d} \cdot a^d)$.

2. Assume that $\epsilon \geq 0.5\sqrt{d}$ and consider the case $\theta = (2, 2, \ldots, 2)$. In this scenario, our algorithm will always exploit (as the difference between the maximum and minimum is equal to $0.5\sqrt{d}$). The total regret is then: $(\sqrt{d} - 0.5\sqrt{d}) \cdot \Omega(a^d) = \Omega(\sqrt{d} \cdot a^d)$.

Note that in the argument above, we assumed $K_1 = [1, 2]^d$, which is an uncertainty set where $R = \max_{\theta \in K_1} \|\theta\| = 2\sqrt{d}$. If we replace the uncertainty set with $K_1 = [\alpha, 2\alpha]^d$, for some $\alpha > 1$, R would increase to $2\alpha\sqrt{d}$ and the regret would scale with α. Consequently, the regret incurred over $\Omega(a^d)$ steps is $\Omega(Ra^d)$.

We next show the $\ln T$ part of the regret. Recall that $K_1 = [1, 2]^d$ and $\theta = (1, 1, ..., 1)$. Let $\hat{T} = \Omega(a^d)$ be the period in which the sequence of steps outlined above ends. We first argue that the vectors z_i belong to the set $K_{\hat{T}}$, for all $i = 1, ..., d$, where z_i is a vector with 2 in dimension i and 1 in all other dimensions. To see this, we show that z_i satisfies $(z_i)'x_t \leq p_t$ for all $t = 1, ..., \hat{T}$. We have: $(z_i)'x_t = 2x_i + \sum_{j=2}^{d} x_j \leq \frac{4}{\sqrt{d}} + \frac{d}{4}\frac{2}{\sqrt{d}}$ using the way x_is are chosen in this construction. Recall that $p_t = 0.75\sqrt{d}$, and hence we obtain $(z_i)'x_t \leq \frac{4}{\sqrt{d}} + \frac{d}{4}\frac{2}{\sqrt{d}} \leq p_t$, for $d \geq 16$.

For periods $k = \hat{T} + i$ for $i = 1, ..., d$, we will assume that nature chooses the vectors $x_{\hat{T}+i} = e_i$. For $k = \hat{T} + 1$, we have

$$\min_{\hat{\theta} \in K_{\hat{T}+1}} \hat{\theta}' x_{\hat{T}+1} = \theta' e_1 = 1 \qquad \text{and} \qquad \max_{\hat{\theta} \in K_{\hat{T}+1}} \hat{\theta}' x_{\hat{T}+1} = (z_1)' e_1 = 2.$$

Therefore, $p_{\hat{T}+1} = 3/2$. Note that this does not eliminate any of the vectors z_i from the uncertainty set for $i > 1$. Repeating this argument d times, we will have added the inequalities $\theta_i \leq 3/2$ for all $i = 1, ..., d$ by step $k = \hat{T} + d$.

It remains to argue that $K_{\hat{T}+d+1} = [1, 3/2]^d$, which requires proving that no point in $[1, 3/2]^d$ was removed from the set during steps $k = 1, ..., \hat{T}$. This is equivalent to showing that for all $\hat{\theta} \in [1, 3/2]^d$, $\hat{\theta}' x_t \leq p_t = \frac{3}{4}\sqrt{d}$. It is sufficient to show this inequality for $\hat{\theta} = (3/2, 3/2, ...3/2)$, which is the worst value of $\hat{\theta}$. By the construction of x_t, $(3/2, 3/2, ...3/2)' x_t = \frac{3}{2}\sum_{i=1}^{d/4} \frac{2}{\sqrt{d}} = \frac{3}{4}\sqrt{d}$, satisfying the desired inequality.

With this construction, we have returned in period $\hat{T} + d + 1$ to the same stage we were in period 1, except that the uncertainty set has been reduced by half in all dimensions. We can now repeat the same argument by using the identical sequence of contexts, $x_t = \frac{1}{\sqrt{d/4}} I\{S_t\}$, and argue that the price will be $p_t = (3/8)\sqrt{d}$, resulting in no sales for $\Omega(a^d)$ steps, followed by the contexts $x_t = e_i$ for d steps, where we again reduce by half each dimension of the uncertainty set.

After repeating this entire argument k times, we are left with an uncertainty set equals to $[1, 1 + 2^{-k}]^d$. After $k = \log_2(1/\epsilon)$ iterations, the uncertainty set becomes $[1, 1 + 1/\epsilon]^d$. The regret

is bounded by (i) the loss due to explore periods, which is proportional to the number of explore steps, $a^d \log_2(1/\epsilon)$ plus (ii) the loss due to exploit periods, which corresponds to the maximum loss under an exploit price, $\epsilon\sqrt{d}$, multiplied by the number of exploit steps, $(T - a^d \log_2(1/\epsilon))$:

$$a^d \log_2(1/\epsilon) + \epsilon\sqrt{d} \cdot (T - a^d \log_2(1/\epsilon)).$$

The value of ϵ that minimizes this regret is $\epsilon = a^d/(T\sqrt{d})$, yielding a regret bound of $\Omega(a^d \ln(T\sqrt{d}/a^d))$. Replacing $a = 1.2$ with any number strictly between 1 and 1.2 yields the desired $\Omega(a^d \ln(T))$. □