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 and needs to price them in order 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 it 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 a question in online advertising, where impressions arrive over time and can be described by vectors of features. We first consider a multi-dimensional version of binary search over polyhedral sets, and show that it has exponential worst-case regret. 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 that is quadratic in the dimensionality 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 by using a technique called shallow cuts. Finally, we present computational experiments to illustrate the performance of our algorithm.

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

1. Introduction

The majority of models of dynamic pricing 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 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 online and are hugely differentiated from each other?

One setting where such a problem emerges is online advertising. Consider the problem faced by a web publisher (such as The New York Times) when selling impressions (user views) to advertisers. At each period, a new user visits the publisher's website, hence creating an opportunity for the publisher to sell an ad space on his website that is targeted towards that particular user. Because of targeting technologies such as web browser cookies, the publisher may have access to useful

information about each user. Therefore, each impression sold by the publisher is a unique product, with its own set of characteristics. An ad space that will be shown to a middle-age mother who lives in Chicago and was recently shopping online for a new laptop is likely to have a different market value than an ad space that will be shown to a teenage girl who lives in Shanghai whose main online interest is music.

The publisher's problem is to set prices for the impressions as they arrive. A priori, the seller does not know the market value of the different impressions. Since impressions are so highly differentiated, it is hopeless for the publisher to try to learn the value of each specific impression before extracting revenue. Instead, we propose that the seller use a hedonic pricing model, where the market value of each impression is a function of the market values of its features. This way, the seller needs to learn only the market values of the different features, thus borrowing strength from the similarity between impressions. In the context of online advertising, some of the natural features that sellers can utilize are demographic information (age, gender, location) and data that can be obtained from the browsing history (interests, online shopping history).

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 he 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 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 dimensionality of the feature space.

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 set. 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 set 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 in 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 at 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 a variant of the algorithm that is robust to noise. We call this variant 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, therefore, still obtain regret guarantees under bounded adversarial noise or i.i.d. Gaussian noise (see Section 6).

Online advertising is a very large market, and some of the largest corporations in the world, such as Google and Facebook, earn most of their revenues from online ad markets. A lot of the value created by online advertising is due to the fact that online ads can be far more targeted than traditional ads. Our paper is a step towards developing pricing algorithms that allow publishers to

monetize targeting. We would like to point out that most online ad markets are run as auctions, but our paper does not consider auctions, only posted prices. It would be natural to embed the algorithm we propose into auctions, where our algorithm would generate reserve prices. We note that finding reserve prices in markets for targeted ads is a challenging and important problem. When highly targeted, many ad auctions become thin, in the sense that there is only a single bidder, or that one bidder assigns a much higher value to the product relative to the other bidders. For obvious reasons, it is hard for a seller to extract revenues in thin auctions. Therefore, an algorithm that produces good reserve prices for targeted ads would generate significant value for publishing platforms.

Our model is applicable to several other online markets beyond advertising. 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 does not set prices but offer pricing suggestions to the hosts. Airbnb receives a fixed fraction of the revenue earned by the hosts, so that its incentives are aligned with the hosts' incentives. As in our model, if a given good (in this case, a one night stay in a home at a particular day) is not sold, it generates no revenue and becomes obsolete. The model and algorithm proposed in this paper could be used by a platform such as Airbnb in order to extract revenue and at the same time, learn the values of the different listing features. 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, 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 algorithm we propose in this paper may also be useful in such contexts.

2. Related Literature

Our work lies at the intersection of two literatures: dynamic pricing with learning and contextual bandits. 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 values 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) and Besbes and Zeevi (2015)) as well as non-parametric ones (e.g., Besbes and Zeevi (2009) and 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)). Cesa-Bianchi et al. (2013) observe that although the loss function induced by pricing problems is neither continuous nor convex, it is well-structured enough to enable algorithms with improved regret over general-purpose bandit algorithms. Relying on this observation, they provide an $\tilde{O}(T^{1/2})$ regret algorithm[1] for the problem of setting reserve prices in a second price auction environment. Another important dimension in pricing problems is that of limited supply. Badanidiyuru et al. (2013) study a problem where the seller has a fixed number of goods, so he must trade-off learning and earning not only across time but also across supply levels. They 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).

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 (2003) and has recently grown into a large literature (see, for example, Dudik et al. (2011) and Agarwal et al. (2014)). As in our paper, many models of contextual bandits (but certainly not all) assume that payoffs – or market values in our model – are linear in the feature vector (Chu et al. (2011), Abbasi-Yadkori et al. (2011) and Agrawal and Devanur (2015a)). Products having market values which are a function of their features is a typical assumption in the marketing literature, an assumption often referred to as hedonic pricing (see Milon et al. (1984), Malpezzi (2003) and Sirmans et al. (2005)).

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

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

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 $\widetilde{O}(T^{2/3})$. 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.

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) 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 the literature on learning from revealed preferences. 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. In addition, 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. Another somewhat 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 quite 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. (2004) and Bertsimas and O'Hair (2013)). The fact that we cannot choose directions of cuts makes our problem significantly harder than the problem considered by Toubia et al. (2004).

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. Unlike us, they assume many products arrive per period, all of which have the same set of features. The seller thus observes an aggregate demand rather than a single transaction per period. 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. They refer to this as an "astrology report" effect, where covariates introduce noise that drives exploration. Approaches based on least squares can be used in stochastic models, but not in adversarial models such as the model considered in this paper.

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 than or equal to the market value v_t, or rejects it, in which case the product is lost. The goal of the seller is to design a pricing policy to maximize his 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 knowledge about the market value function v.

Our model captures an online advertising problem where impressions (or user views) arrive one at a time and are described by a vector of features, such as the demographics and browsing history of the user. The publisher (or seller) chooses prices for the impressions as they arrive. If the price is equal to or below the market value of the impression, the impression sells for the posted price. If the price is too high, the impression disappears. The publisher can use prices not only to earn immediate revenue, but also to learn how the different features of the impressions affect the market price.

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 to the market value, i.e., $p_t \leq \theta' x_t$, a sale occurs and the seller collects 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 $K_{t+1} = K_t \cap \{\theta \in \mathbb{R}^d : \ \theta' x_t \geq p_t\}$ 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.[2]

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

[2] 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.

x_t in every 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 he knew the value of θ, and the second term is the actual revenue generated by the 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.

3.1. Alternative Models

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: non-linear and noisy valuations. A special case of non-linearity is addressed via a Lipschitz continuity argument, as discussed in Section 7. Noisy valuations, on the other hand, require more fundamental changes to our algorithm.

Noisy valuations are an important step towards a more realistic model of buyer's valuation, since they handle cases where no single model is a perfect prediction. We can model noise as the vector of weights θ_t being drawn in each step from an unknown but fixed distribution $\mathcal{F}$ and $v_t = \theta_t' x_t$. It is useful to isolate the expectation $\theta = \mathbf{E}[\theta_t]$ and write:

$$v_t = \theta' x_t + \delta_t,$$

where δ_t is a scalar random variable given by $\delta_t = (\theta_t - \theta)' x_t$. We study this model in Section 6. Our focus in Section 6 will be on the case with large signal-to-noise ratio, i.e., the noise term δ_t has a smaller magnitude relative to $\theta' x_t$. We now briefly discuss how to handle the case where the noise term is large (i.e., the same magnitude as $\theta' x_t$) or even fully adversarial.

One of the frameworks in the contextual bandits literature is the fully adversarial model, where in each step an adversary picks a vector θ_t. In this setting, the regret is typically redefined as the difference between the revenue obtained by the algorithm and the revenue generated via the best fixed θ, i.e.:

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

Kleinberg and Leighton (2003) show a lower bound of $\Omega(T^{2/3})$ regret which is inherited by our model, since their setting is a one-dimensional special case of our model. This lower bound (up to logarithmic factors) can be attained even in a multidimensional contextual setting such as ours via the algorithm of Agarwal et al. (2014). In particular, by an appropriate discretization argument, one can obtain a total regret of

$$\tilde{O}(T^{2/3}d^{1/3}). \tag{2}$$

The derivation details of Eq. (2) can be found in the Appendix.

In this paper, we focus in cases where the valuation of buyers is not completely adversarial enabling logarithmic guarantees on the regret. From a technical standpoint, those cases are of special interest since they cannot be handled by off-the-shelf contextual bandit algorithms.

4. A Multi-Dimensional Binary Search Algorithm

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 if not, $K_{t+1} = [\ell_t, p_t]$. 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 at that price, the seller 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:

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

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.

The binary search algorithm has low enough regret in T, $O(\ln T)$, and is pretty simple so that we can hope it generalizes 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 might never offer an opportunity for the seller to learn θ precisely. In our online advertising example, some features might represent properties of the impression that are rarely seen, such as whether the user is shopping for a private yacht. The seller may never see an impression with a non-zero entry on this specific feature, or might see too few to build a good estimate of its market value.

It can also happen that in all the examples seen by the seller, two of the features are correlated. Perhaps, a user who recently visited a hotel website is also very likely to have visited an airline website. In this case, it would be hard for the seller to disentangle the values of these two features. Moreover, if these two features are often correlated, it might not be worth the effort for the seller to try to disentangle them, since there is often a trade-off between learning and generating revenue.

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 he reaches a good estimate of θ to start exploiting. For example, perhaps T is equal to a year-long period and in the first $T/2$ periods, the warm months of the year, the seller received almost no skiing-related impressions. It does not make sense for the seller to wait until he starts receiving skiing-related impressions before he starts exploiting on other types of impressions. We refer the reader to Mirrokni et al. (2012) for an argument on why Internet advertising requires models that are robust to the sequence of features.

4.3. Explore and Exploit Prices

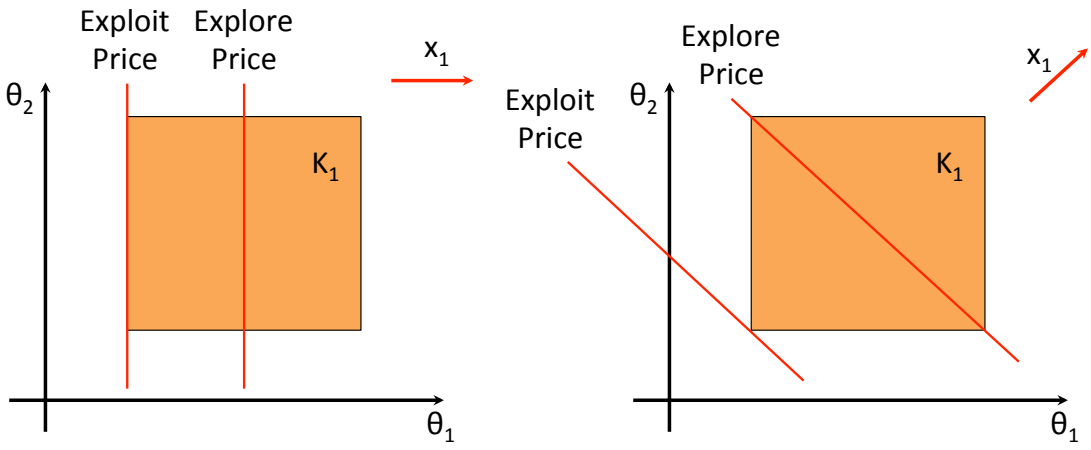
Based on our discussion so far, we know that we cannot hope to learn the value of θ exactly. Also, pre-determined exploration and exploitation phases do not seem to be adequate here. Instead, for each product that arrives, we will decide whether we exploit or not. In particular, we will exploit if we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

In order 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:

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

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 in Figure 1, note that the explore price always divides the feasible region into two parts, whereas the exploit price is always located at the boundary of



(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 when $x_1 = (1/\sqrt{2}, 1/\sqrt{2})$.**

the set. Note also 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 $K_{t+1} = K_t \cap \{\theta \in \mathbb{R}^d : \ \theta' x_t \geq p_t\}$. If not, 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. We prove this result using the following combinatorial lemma, the proof of which we defer to the Appendix.

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

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

Using Lemma 1, we next show that PolytopePricing has a worst-case regret that is exponential in d.

Theorem 1. *For any parameter $\epsilon > 0$, the algorithm* PolytopePricing *suffers worst-case regret $\Omega(Ra^d)$ with the constant $a = 1.2$.*

Proof. 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 zero otherwise. We will show that with at least probability $1/2$ the regret is at least $\Omega(Ra^d)$, where a is the constant 1.2 from Lemma 1. To keep notation cleaner, 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 $t = 1,2,\ldots,k$, 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 1. 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 as a function of d and R is $\Omega(Ra^d)$. $\square$

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

We chose to omit the dependency on T in the statement of Theorem 1 to highlight the exponential dependency on the dimension. The reader should notice, however, that a lower bound of $\Omega(a^d \ln T)$ can be obtained by interleaving sequences like the ones in the proof of Theorem 1 with a sequence of vectors in the coordinate directions that restore the knowledge set to the shape of a cube.

5. Ellipsoid Pricing

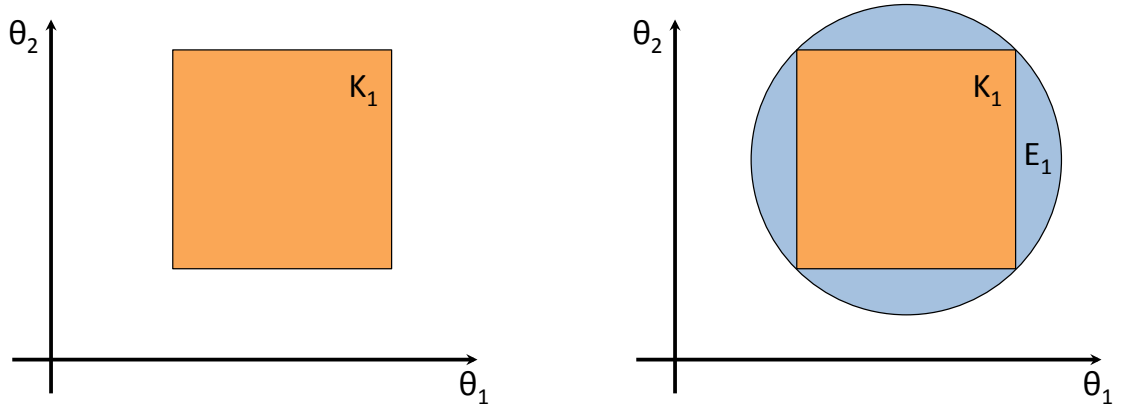
In this section, we modify the PolytopePricing algorithm from the previous section so as to achieve a regret that is polynomial (in fact, quadratic) in the dimension. As an added bonus, the algorithm becomes also simpler to implement and computationally cheap. The algorithm now requires only that we maintain a $d \times d$ matrix and that we 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 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:[3]

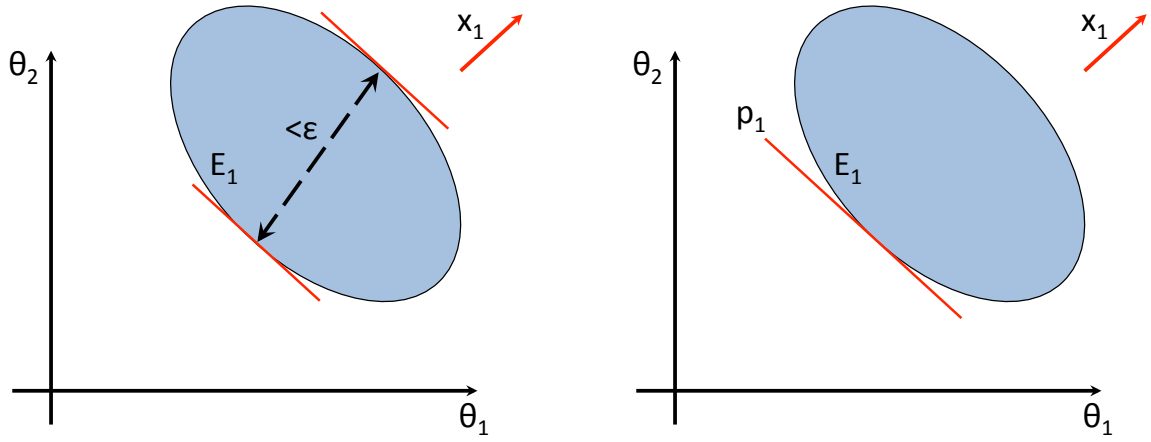
$$\underline{b}_t = \min_{\hat{\theta} \in E_t} \hat{\theta}' x_t \qquad \text{and} \qquad \bar{b}_t = \max_{\hat{\theta} \in E_t} \hat{\theta}' x_t. \tag{4}$$

[3] Note that we slightly abuse notation by reusing the variable names $\underline{b}_t$ and $\bar{b}_t$ 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.



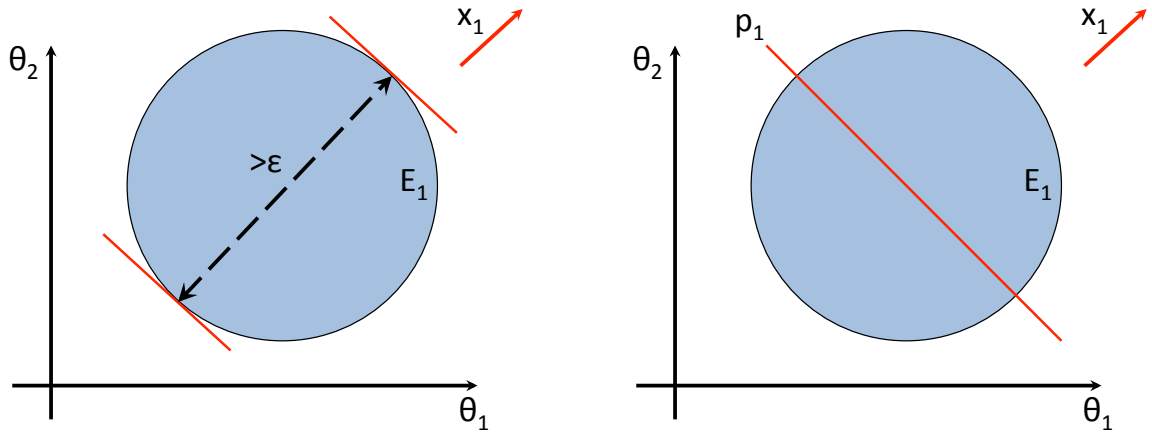
(a) Solve for the max and the min over E_1. (b) Compute the exploit price p_1.

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

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 $H_{t+1} = E_t \cap \{\theta \in \mathbb{R}^d : \ \theta' x_t \geq p_t\}$. If not, 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:

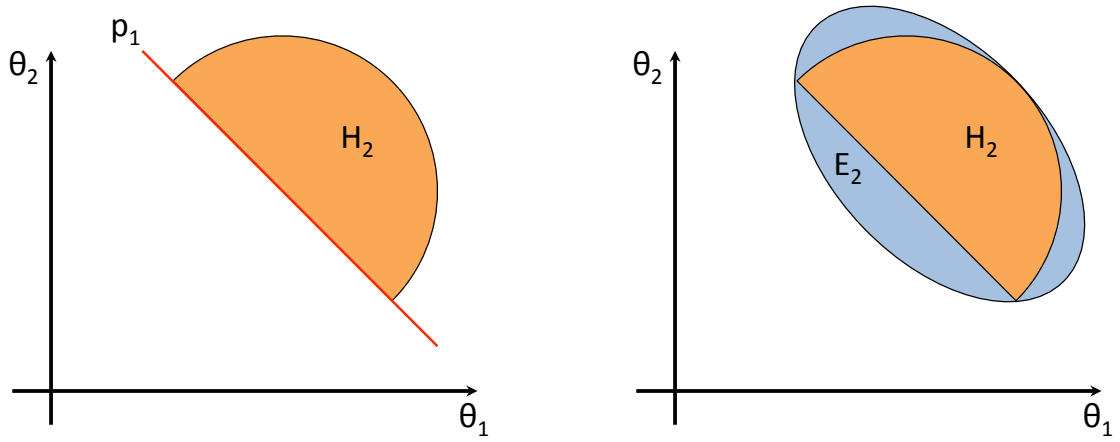
THEOREM 2. *The worst-case regret of the* ELLIPSOIDPRICING *algorithm with parameter* $\epsilon = Rd^2/T$ *is* $O(Rd^2 \ln(T/d))$.

We defer the proof of this theorem until Section 5.3. A remarkable fact is that efficiency is achieved by *enlarging* the uncertainty set. At the expense of adding candidate vectors $\hat{\theta}$ that are



(a) Solve for the max and the 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.

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.

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.

In order 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. As a result, we won't need to explore from that point onwards. In order 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 section 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:

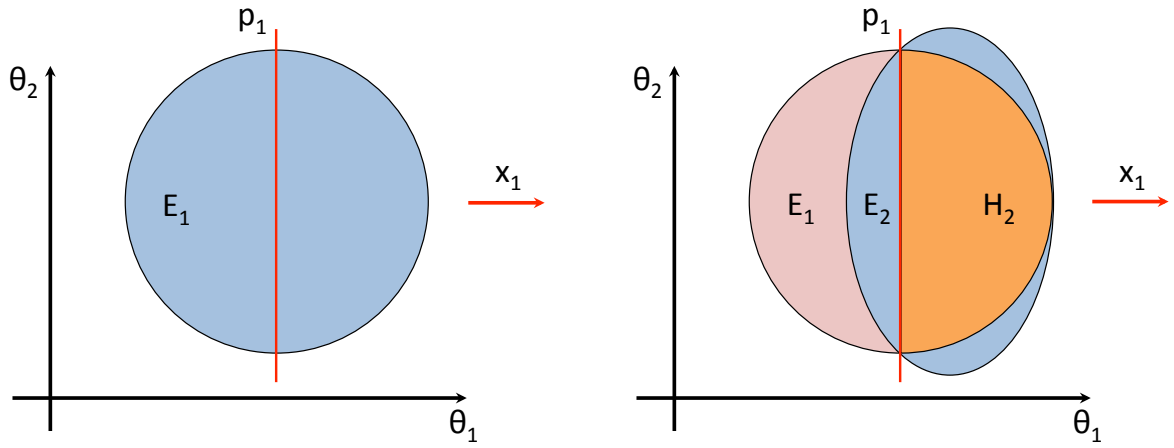
$$E(A, a) := \{\theta \in \mathbb{R}^d : \ (\theta - a)' A^{-1} (\theta - a) \leq 1\}.$$

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:

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

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 $\textsc{Vol}\ E(A)$ instead of $\textsc{Vol}\ E(A, a)$, 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



(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.

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:

$$\tilde{A} = \frac{d^2}{d^2-1}\left(A - \frac{2}{d+1}bb'\right). \tag{5}$$

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

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

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 $A = I$ (where 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 equation (5), we obtain that $\tilde{A}$ is the diagonal matrix with eigenvalue $\frac{d^2}{(d+1)^2} < 1$ in the 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.

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 equation (5). 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 keep track does not depend on T.

5.3. Regret Analysis for EllipsoidPricing

In order 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 2. 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 until Section 5.5. It is simple to see how Lemma 2 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, $\text{Regret} \leq NR + (T-N)\epsilon$. By Lemma 2 we have: $\text{Regret} \leq 2Rd^2 \ln(20R(d+1)/\epsilon) + T\epsilon$. Choosing $\epsilon = Rd^2/T$, the total regret becomes $\text{Regret} = O(Rd^2 \ln(T/d))$. $\square$

The core of our analysis consists of proving Lemma 2. 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 EllipsoidPricing is a low regret algorithm.

5.4. More Tools from Linear Algebra

In order 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:

$$\varphi_D(z) = \prod_j (\lambda_j - z) \cdot \hat{\varphi}_D(z) \quad \text{where} \quad \hat{\varphi}_D(z) = 1 - \beta \sum_i \frac{b_i^2}{\lambda_i - z}. \tag{6}$$

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. The intermediate value theorem then 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 eigenvalue of D, then:

$$\lambda_d - \beta b'b \leq \sigma_d \leq \lambda_d \leq \sigma_{d-1} \leq \lambda_{d-1} \leq \sigma_{d-2} \leq \lambda_{d-2} \leq \ldots \leq \lambda_2 \leq \sigma_1 \leq \lambda_1. \tag{7}$$

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

Lemma 3. *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 equation (6), 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 $\sigma_d \geq z$. $\square$

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 3 to show that the smallest eigenvalue doesn't decrease by much in any given iteration:

LEMMA 4. *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 equation (5), 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 3, 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 the maximal element. The equality follows from the fact that λ_d is the smallest eigenvalue. □

Lemma 4 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 more generally. Assume x_t is the eigenvector that corresponds to the smallest eigenvalue of an arbitrary A_t. Then, the smallest eigenvalue of A_{t+1} is equal to $\frac{d^2}{(d+1)^2}\lambda_d(A_t)$. Lemma 4 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 doesn't decrease.

LEMMA 5. *There exists a sufficiently small $k = k(d)$ such that if $\lambda_d(A_t) \le k\epsilon^2$ and $x_t' A_t x_t > \frac{1}{4}\epsilon^2$, then $\lambda_d(A_{t+1}) \ge \lambda_d(A_t)$, i.e., the smallest eigenvalue doesn't decrease after the update. In addition, one can take $k = \frac{1}{400d^2}$.*

Proof. In this proof, we assume $d \ge 2$. Note that the lemma is trivially true for $d = 1$. Using the same notation as in the proof of Lemma 4, we need to show that $\lambda_d(A_{t+1}) = \frac{d^2}{d^2-1}\sigma_d \ge \lambda_d(A_t)$, where σ_d is the smallest eigenvalue of $\tilde{D}$. To prove that $\sigma_d \ge \frac{d^2-1}{d^2}\lambda_d(A_t)$, it is sufficient to show that $\varphi_{\tilde{D}}\left(\frac{d^2-1}{d^2}\lambda_d\right) \ge 0$ by using Lemma 3. Note that $\varphi_{\tilde{D}}\left(\frac{d^2-1}{d^2}\lambda_d\right) \ge 0$ holds if and only if $\hat{\varphi}_{\tilde{D}}\left(\frac{d^2-1}{d^2}\lambda_d\right) \ge 0$ since $\frac{d^2-1}{d^2}\lambda_d < \lambda_d$. Therefore, the remainder of the proof focuses on showing that $\hat{\varphi}_{\tilde{D}}\left(\frac{d^2-1}{d^2}\lambda_d\right) \ge 0$.

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:

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

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

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

The last inequality follows from the facts that $\sum_i c_i^2 \le 1$ 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 $\hat{\varphi}_D\left(\frac{d^2-1}{d^2}\lambda_1\right) \ge 0$. 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 can conclude that taking $k = 1/400d^2$ is enough. □

The intuition behind Lemma 5 is as follows. Assume λ_d is sufficiently small ($\lambda_d \le k\epsilon^2$). 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 $\lambda_i \le \sqrt{k}\epsilon^2$) 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 2, which is the last missing piece of our argument:

Proof of Lemma 2. 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 the ellipsoid $\tilde{E}_n$. 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 5, when the smallest eigenvalue is below $\epsilon^2/400d^2$, it can't shrink further. Also, from Lemma 4, 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:

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

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

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

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 pure linear model with $v_t = \theta' x_t$. In this section, we extend the model to allow for idiosyncratic noise: $v_t = \theta' x_t + \delta_t$, where δ_t represents an error in our estimate of v_t. We consider two variants of this model. The first one is a robust optimization model, where we assume the noise is bounded. The second version is a probabilistic model, where we assume the noise is normally distributed.

6.1. Bounded Noise

We first assume that the noise term δ_t is bounded within an interval $[-\delta, \delta]$. We assume nature selects the noise term in each period according to an adversarial closed-loop policy. To account for this adversarial noise model, we need to modify our definition of regret to incorporate the noise term:

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

Note that in a model with noise, the EllipsoidPricing algorithm, as presented in Section 5, might incorrectly remove the true value of θ from the uncertainty set. The algorithm we propose next is designed to make our algorithm robust to bounded noise. We call the modified algorithm ShallowPricing. The essence of the technique is to make our cuts safer. For exploit steps, we propose using the price $p_t = \underline{b}_t - \delta$. This is the highest price we could choose and still guarantee a sale. For explore steps, we propose using the same price as before: $p_t = (\underline{b}_t + \bar{b}_t)/2$. However, we propose a modification on the update of the uncertainty set. The robust version of our algorithm takes into account that the true θ might be on the wrong side of the cut. When 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., $K_{t+1} = K_t \cap \{\theta \in \mathbb{R}^d :\; \theta' x_t \leq p_t + \delta\}$. 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 shows the difference between a central cut and a shallow cut.

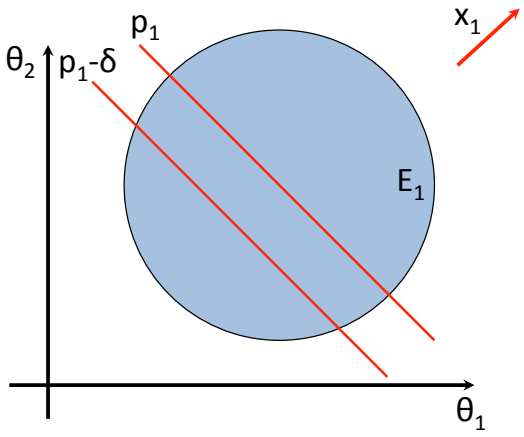


Figure 7 **If we remove the half-ellipsoid below p_1, we are performing a central cut of the ellipsoid E_1. If we remove only the subset below $p_1 - \delta$, we are performing a shallow cut of E_1.**

In order 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 zero represents a central cut of the ellipsoid. Our analysis does not involve the third kind of standard cut, 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 cuts has to be at least $-1/d$. With $\alpha_t \geq -1/d$, the following machinery allows us to compute the Löwner-John ellpsoids 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

$$A_{t+1} = \frac{d^2}{d^2-1}(1-\alpha_t^2)\left(A_t - \frac{2(1+d\alpha_t)}{(d+1)(1+\alpha_t)} b_t b_t'\right). \tag{8}$$

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. (8) is not all that different from Eq. (5). We can therefore adapt our analysis for central cuts to allow for shallow cuts. We are now ready to present the performance guarantee of our algorithm when tailored to an environment with bounded adversarial noise.

Theorem 3. *Under bounded noise, the worst-case regret of the* ShallowPricing(ϵ) *algorithm with parameter* $\epsilon = \max\{Rd^2/T, 4d\delta\}$ *is* $O(Rd^2 \ln(\min\{T/d, R/\delta\}) + d\delta T)$.

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

Let N be the number of exploration steps. The regret is bounded by:

$$\text{Regret} \leq NR + \epsilon T + 2\delta T, \tag{9}$$

where the $2\delta T$ term captures a per-period error of up to 2δ in pricing. The key step in our analysis is to bound N. To do so, we must first show that Lemmas 4 and 5 still apply in our noisy setting. We first 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 4. 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. (8). Perform the same change of basis as in the proof of

Lemma 4 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 3, 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 4. The lowest possible value of the right-hand side of the equation above takes place when $\tilde{b}_d^2 = \lambda_d(A_t)$ and $\tilde{b}_i^2 = 0$ for $i \neq d$. Thus,

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

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 now prove a shallow-cut equivalent of Lemma 5. 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 5, one can take $k = \frac{1}{400d^2}$. To show this result, it is sufficient to prove that

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

We can mimic the proof of Lemma 5 to obtain that

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

Since we assumed that $\epsilon \geq 4d\delta$ and we know that $\sqrt{x_t' A_t x_t} \geq \frac{1}{2}\epsilon$, we have by the definition of α_t that $\alpha_t = -\delta/\sqrt{x_t' A_t x_t} \geq -2\delta/\epsilon \geq -1/2d$. 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 5, we ignore the trivial case of $d = 1$). We thus reach our desired result. Note that we obtain the precise value of $k = 1/400d^2$ by numerical inspection, as in Lemma 5.

We have now proved that Lemmas 4 and 5 still apply in a 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 4 and 5 together provide a bound on the smallest volume possible 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 2, with the sole difference that we replace $e^{-n/2d}$ with $e^{-n/20d}$. We therefore obtain a bound on the number of explore steps N:

$$N \leq 20d^2 \ln\left(\frac{20R(d+1)}{\epsilon}\right). \tag{12}$$

Plugging in $\epsilon = \max\{Rd^2/T, 4d\delta\}$, we obtain that $N = O(d^2 \ln(\min\{T/d, R/\delta\}))$. Together with Eq. (9), we have

$$\text{REGRET} \leq O(Rd^2 \ln(\min\{T/d, R/\delta\}) + \max\{Rd^2, d\delta T\} + \delta T).$$

The first term inside the maximand is dominated by the first term inside the big O and can be removed. The last term inside the big O is dominated by the second term inside the maximand and can also be removed. Our expression thus simplifies to

$$\text{REGRET} \leq O(Rd^2 \ln(\min\{T/d, R/\delta\}) + d\delta T),$$

completing our proof. □

With bounded adversarial noise, we necessarily incur regret that is linear in T. This is the case even if θ is known a priori, as the seller would have to price products at $\theta' x_t - \delta$ to ensure the occurrence of sales. Fortunately, if the noise parameter δ is small, the regret expression in Theorem 3 is identical to the one in Theorem 2.

COROLLARY 1. *Assume $\delta \leq Rd/4T$. Then, the worst-case regret of the* SHALLOWPRICING(ϵ) *algorithm with parameter $\epsilon = Rd^2/T$ is $O(Rd^2 \ln(T/d))$.*

Proof. Under $\delta \leq Rd/4T$, Theorem 3 says that the regret of SHALLOWPRICING(ϵ) with $\epsilon = Rd^2/T$ is $O(Rd^2 \ln(T/d) + d\delta T)$. Under $\delta \leq Rd/4T$, we also have that $O(d\delta T) = O(Rd^2)$. □

6.2. Gaussian Noise

We now consider the case with $v_t = \theta' x_t + \delta_t$, where δ_t is an i.i.d. Gaussian noise with mean zero and standard deviation σ. Our regret metric for this setting still makes worst-case assumptions with regards to θ and the feature vectors, but makes a probabilistic assumption about the noise term δ_t:

$$\text{REGRET}(\Pi) \;=\; \max_{\theta \in K_1,\ X \in \mathbb{X}} \; \sum_{t=1}^{T} E_{\delta_t}\Big[\theta' x_t + \delta_t - p_t I\{\theta' x_t \geq p_t\}\Big].$$

Despite the noise no longer being bounded, we still propose using the ShallowPricing algorithm. While δ was a model primitive in Subsection 6.1, δ is now an additional parameter of the ShallowPricing(ϵ, δ) algorithm.

Under bounded noise, the ShallowPricing(ϵ, δ) algorithm never made mistakes, in the sense that θ was always inside the uncertainty set. With unbounded noise, there is a positive probability that ShallowPricing(ϵ, δ) will make a mistake, regardless of our choice of δ. We now prove a performance bound on ShallowPricing(ϵ, δ) under Gaussian noise.

Theorem 4. *Under Gaussian noise, the regret of the* ShallowPricing(ϵ, δ) *algorithm with parameters* $\epsilon = \max\{Rd^2/T, 4d\delta\}$ *and* $\delta = 2\sigma\sqrt{\ln(RT)}$ *is* $O\left(Rd^2 \ln(\min\{T/d, R/\delta\}) + d\delta T\right)$.

Proof. In the event that the error δ_t is in the range $[-\delta, \delta]$ in each time step, we obtain by Theorem 3 that the regret is bounded by $O(Rd^2 \ln(\min\{T/d, R/\delta\}) + d\delta T)$. We now reason about what happens when the noise falls outside this range. We will say that a mistake happens in time t, if δ_t does not lie in the range $[-\delta, \delta]$. We denote this event by Z_t. In any given round, the probability that a mistake occurs is:

$$\Pr(Z_t) = 2\Phi(-\delta/\sigma) \leq 2e^{-\delta^2/2\sigma^2},$$

where $\Phi(\cdot)$ is the cdf of a standard Gaussian and the inequality follows from the bound $2\Phi(z) \leq e^{-z^2/2}$ (see Chang et al. (2011)). The probability that some mistake happens is:

$$\Pr(\cup_t Z_t) \leq T2e^{-\delta^2/2\sigma^2} = Te^{-2\ln(RT)} = \frac{1}{R^2T},$$

where the inequality follows from the union bound, and the first equality follows from the definition of δ. Now, if a mistake occurs, we will use the crude bound of RT for the regret. Since this happens with probability at most $1/(R^2T)$, this only adds a constant term to the regret. □

As in the bounded noise case, if the noise is sufficiently small, we recover the regret expression from Theorem 2.

Corollary 2. *Assume* $\sigma \leq Rd/(T\sqrt{\ln(RT)})$. *Then, the regret of the* ShallowPricing(ϵ, δ) *algorithm with parameters* $\epsilon = Rd^2/T$ *and* $\delta = 2\sigma\sqrt{\ln(RT)}$ *is* $O(Rd^2 \ln(T/d))$.

Proof. When $\sigma \leq Rd/(T\sqrt{\ln(RT)})$, the term inside the logarithm in the regret bound of Theorem 4 becomes T/d. In addition, the second term becomes: $O(d\delta T) = O(d\sigma T\sqrt{\ln(RT)}) = O(Rd^2)$, and hence gets absorbed in the first. □

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 in Internet advertisement 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 $\phi(\cdot)$ in order 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 $\epsilon = \bar{f} d^2/LT$ has 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 $\epsilon = \bar{f} d^2/LT$. □

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. Using the standard *doubling trick*[4] 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.*

[4] 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)).

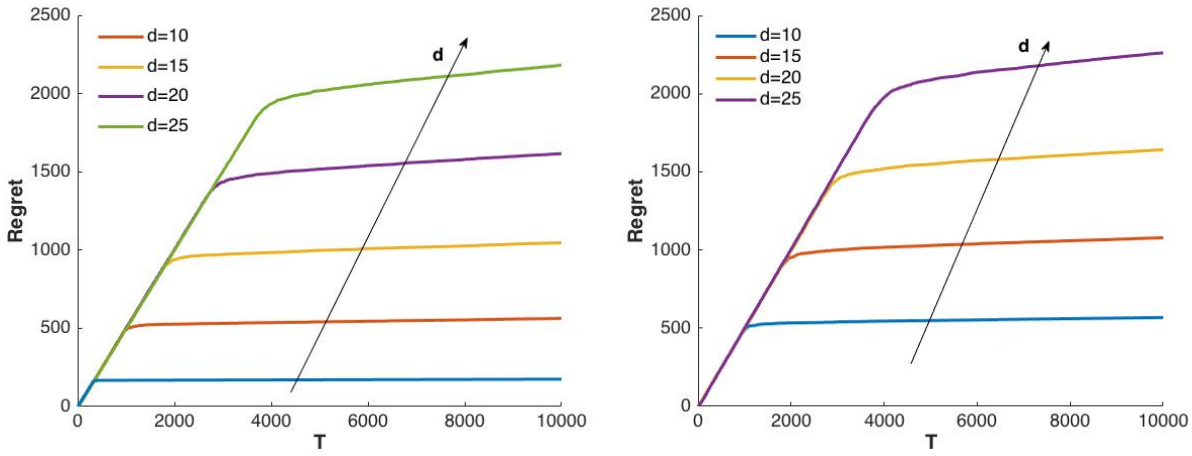
Proof. Given T time steps, let $\bar{k}$ be the minimum value such that $\sum_{k=0}^{\bar{k}} 2^{2^k} \geq T$. Therefore, for T time steps, the algorithm will have $\bar{k} \leq \lceil \log_2 \log_2 T \rceil$ phases. The total regret from all the time steps in phase k is at most $O(Rd^2 \ln(2^{2^k})) = O(Rd^2 2^k)$. 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 algorithm EllipsoidPricing developed in Section 5, as well as the algorithm ShallowPricing from Section 6 that allows for noisy valuations.

8.1. Regret as a function of T and d

In this paper, we considered an adversarial setting where nature selects the worst vectors of features x_t at each time, as well as the worst vector θ within a bounded set. Computing an actual optimal 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. Interestingly, we show that the regret in such an i.i.d. setting behaves similarly as in an adversarial world.



(a) Using continuous features. (b) Using discrete features.

Figure 8 Regret of the EllipsoidPricing algorithm as a function of T and d for continuous and discrete features.

We first consider a setting where the vectors x_t are drawn i.i.d. with a multivariate Gaussian distribution $N(0, I)$, with the values normalized so that $||x_t|| = 1$ for all t. We also tested several other continuous distributions (e.g., uniform and exponential) and the results in terms of regret happen to be very similar. We vary the value of T between 100 and 10,000 and the value of d between 10 and 25. For simplicity, we use $R = 1$. In Figure 8(a), we plot the regret as a function of T for different values of d. Even though the setting is i.i.d. (and not adversarial), we obtain a regret that is of similar magnitude to our bound. In the proof of Lemma 2 in Section 5.3, we

derived an upper bound on the regret, given by: $Rd^2[1+2\ln(\frac{20(d+1)T}{d^2})]$. For example, when $R=1$, $d=10$, $T=10,000$, this amounts to 2,099. Numerically, we obtain regret equal to 563.42, and hence of similar magnitude. We also consider a setting where x_t are drawn i.i.d. with a Bernoulli distribution. This case can correspond to binary features such as user gender or mobile versus desktop computer. In this case, we obtain a similar regret behavior (see Figure 8(b) for the case where the Bernoulli success probability is set to 0.6).

8.2. Estimation Error

In each exploration step of the EllipsoidPricing algorithm, we update the vector a_t, which corresponds to the center of the ellipsoid (see Section 5). The vector a_t can be interpreted as an estimate of the (unknown) vector θ at time t. Consequently, one interesting question is how the estimation error $||\theta - a_t||_2$ evolves as a function of T. In Figure 9, we plot the error in the θ estimate as a function of T for $d=15$ when x_t are normally distributed $N(0,I)$ (and normalized). One can see that the algorithm learns the true value of θ, and the error quickly converges to zero.

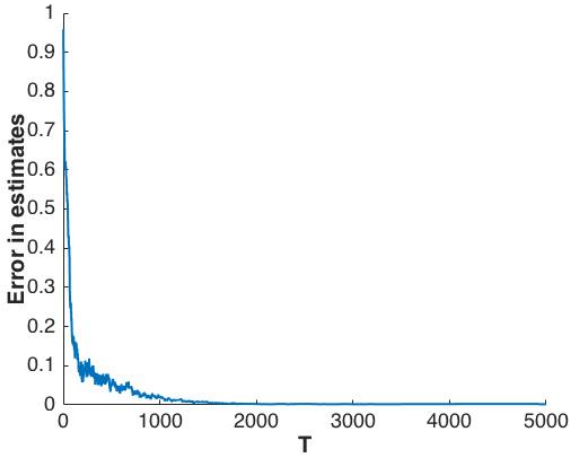


Figure 9 **Errors in estimates as a function of T for $d=15$.**

8.3. 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 randomly drawn. We illustrate this phenomenon in Figure 10, 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

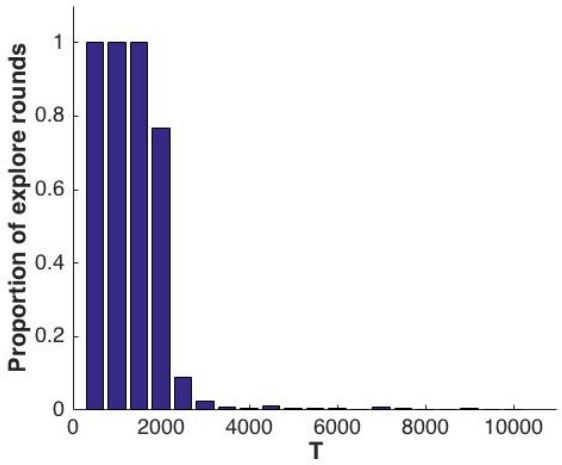
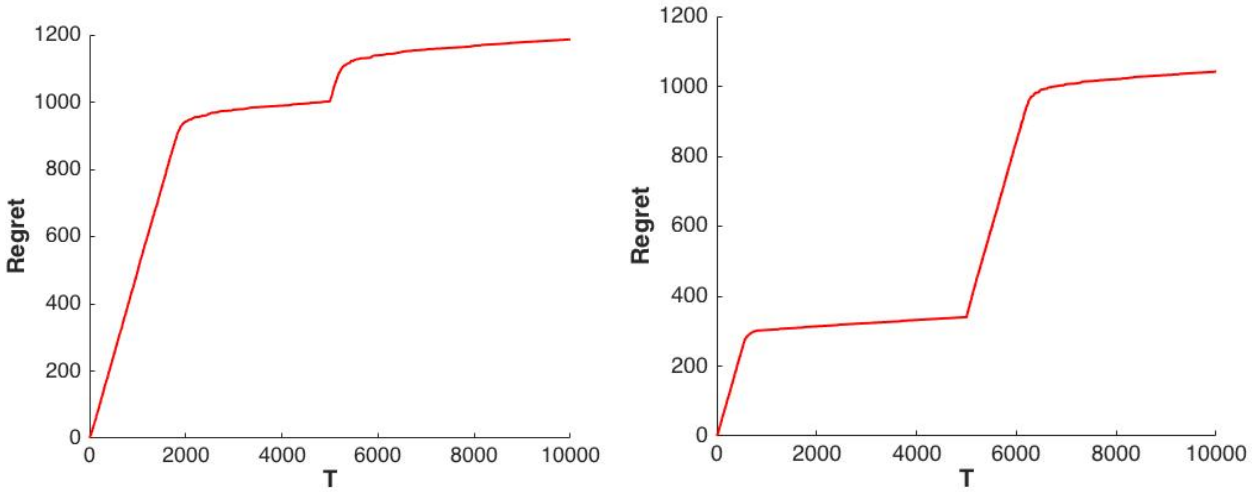


Figure 10 **Explore rounds versus exploit rounds for the EllipsoidPricing algorithm as a function of T for $d = 15$.**

situation where the setting evolves over time by changing the distribution of x_t after half of the time periods.

In what follows, we show that our algorithm can adapt to dynamic environments. We consider two different settings, depicted in Figure 11. Figure 11(a) considers the case where the first half of the iterations (i.e., during the first $T/2$ time periods), the vectors of features are normally distributed $N(0, I)$, and 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 11 **Regret of the EllipsoidPricing algorithm as a function of T for $d = 35$ when the distribution of the features changes at $T/2$.**

Figure 11(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 a 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 12 shows the algorithm starting to explore again after the change in distributions at $T/2$, under the same setting as in Figure 11(a). This type of situations is very typical as the vectors of features can depend on external factors such as seasonality and trends.

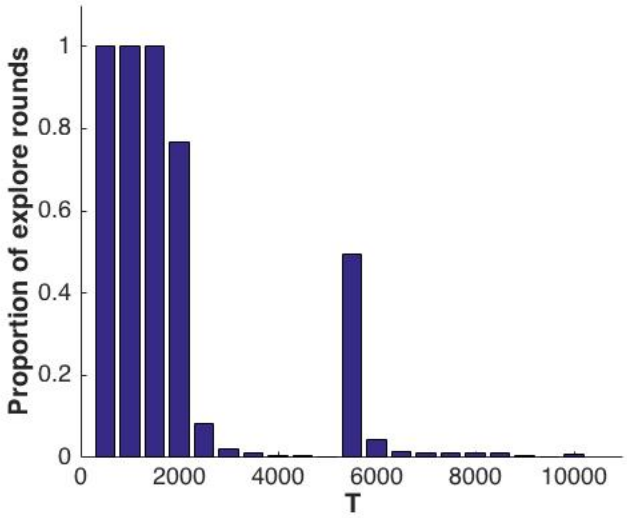
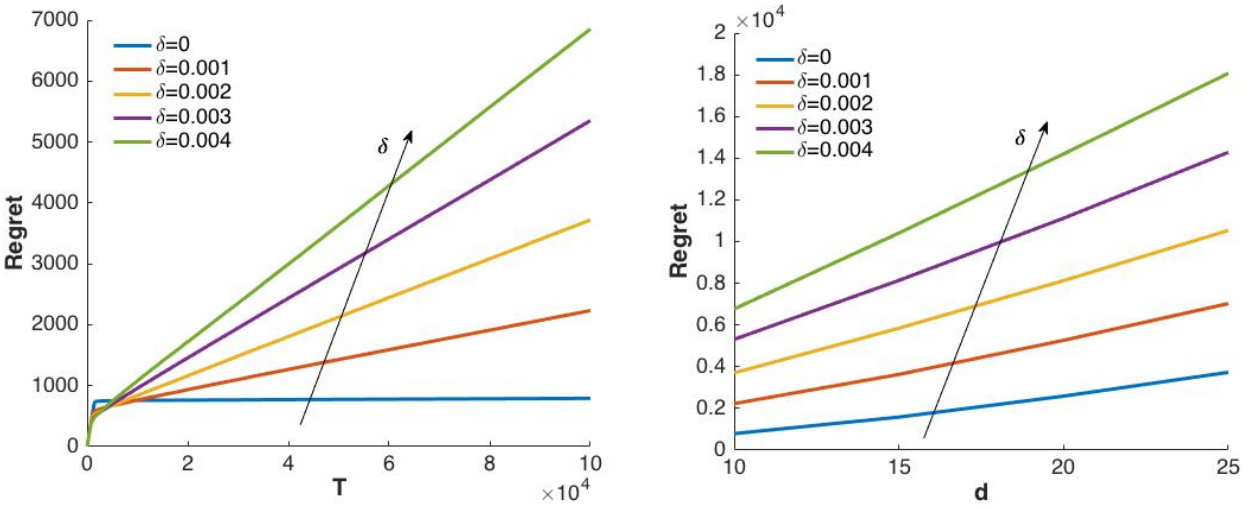


Figure 12 **Explore rounds versus exploit rounds for the EllipsoidPricing algorithm as a function of T for $d = 15$ using continuous features**

8.4. Testing the ShallowPricing algorithm

Finally, we test the version of our algorithm when the valuation is noisy (see Section 6). We assume that the additive noise is uniformly distributed $U[-\delta, \delta]$. In Figure 13, we plot the regret as a function of T and d for different values of δ. In Figure 13(a), we set $d = 15$ and vary T, whereas in Figure 13(b), we set $T = 100,000$ and vary d. One can see as expected, that the regret behaves as $O(Rd^2 \ln(\min\{T/d, R/\delta\}) + d\delta T)$ as we have shown in Theorem 3. On one hand, for small values of δ, we have a logarithmic dependence with respect to T and we retrieve the result of Corollary 1. On the other hand, when δ becomes large, we switch to a linear dependence.

Interestingly, when we consider Gaussian noise, we obtain a very similar behavior as Figure 13. In Figure 14, we set $\delta = 0.001$ and consider the case of an additive Gaussian noise. For any value of $\sigma \leq 0.1$, we obtain a regret plot similar to the one depicted in Figure 14.



(a) Regret as a function of T.

(b) Regret as a function of d.

Figure 13 **Regret of the ShallowPricing algorithm as a function of T and d for different supports of the noise.**

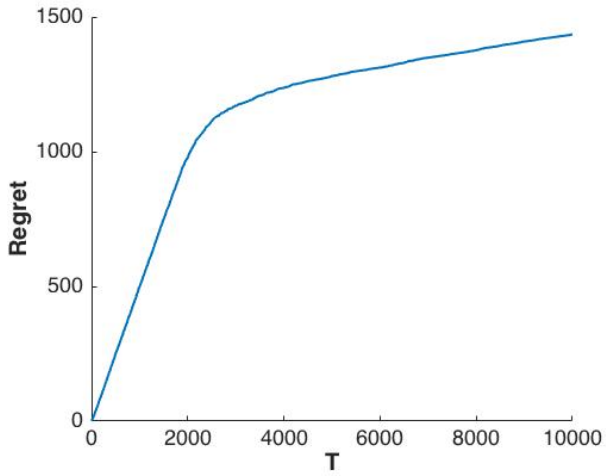


Figure 14 **Regret of the ShallowPricing algorithm as a function of T for $d = 15$ using continuous features and an additive Gaussian noise $N(0, \sigma)$, with $\sigma = 0.01$.**

8.5. 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:

- Even though our regret bound was derived for an adversarial setting, we observe that the regret achieved when the vectors of features are drawn i.i.d. is of similar magnitude.
- 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 works also for the case where the valuation is noisy (see Section 6). We tested different distributions of the additive noise (both bounded and unbounded), and observed a good regret performance.

9. Conclusions and Future Directions

In this paper, we considered the problem of pricing highly differentiated products described by vectors of features (e.g., impressions in online advertising). 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 is to propose an efficient online pricing method by balancing the exploration/exploitation tradeoff so as 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. 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 dimensionality of the feature space and logarithmic in the time horizon.

We would like to end by discussing some future research directions. An interesting direction would be to consider a setting where the parameter θ changes over time. For example, one could consider a problem where, at each time period, a vector θ_t takes a different value, but under the assumption of limited variation, i.e., $\|\theta_{t+1} - \theta_t\| \leq \Delta_t$. Another important future direction would be to explicitly embed our algorithm in a dynamic auction design problem. In an auction design setting, an algorithm like ours could be used to dynamically generate reserve prices for the sequence of auctions.

Acknowledgments

We thank Arash Asadpour, 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, ideas and suggestions. The authors would like to specially thank Arash Asadpour for his insightful suggestions regarding the proof of Lemma 1. 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.

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.

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

Cesa-Bianchi, Nicolo, Claudio Gentile, Yishay Mansour. 2013. Regret minimization for reserve prices in second-price auctions. *Proceedings of SODA*. 1190–1204.

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

Chang, Seok-Ho, Pamela C Cosman, Laurence B Milstein. 2011. Chernoff-type bounds for the gaussian error function. *IEEE Transactions on Communications* **59**(11) 2939–2944.

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.

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.

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.

Mirrokni, Vahab S, Shayan Oveis Gharan, Morteza Zadimoghaddam. 2012. Simultaneous approximations for adversarial and stochastic online budgeted allocation. *Proceedings of SODA*. 1690–1701.

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, 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.

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.

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

Appendix

Reduction to Contextual Bandits. Here, we show how to reduce our problem to a standard contextual bandit setting and, thus, obtain a regret bound for a version of our model with fully adversarial noise. More precisely, we prove Eq. (2) via the algorithm of Agarwal et al. (2014). In order to use this algorithm, we must first discretize the policy space, which in our problem corresponds to the set K_1. In what follows, we assume for simplicity that $R=1$. To achieve the desired regret bound, we discretize K_1 into a d-dimensional lattice with increments of $\eta = 1/(T\sqrt{d})^{1/3}$. This induces a policy space with cardinality $O(1/\eta^d)$. We also discretize the set of prices, restricting ourselves to multiples of $\eta\sqrt{d}$. Therefore, our set of prices has cardinality $O(1/\eta\sqrt{d})$. In this discretized space, a policy θ determines a price p_t as a function of x_t by first computing the value of $\theta' x_t$ and then rounding down the result to the nearest multiple of $\eta\sqrt{d}$. The maximum revenue loss per period from this discretization is $O(\eta\sqrt{d})$. Therefore, over the entire horizon of T periods, discretization costs us up to $O(T\eta\sqrt{d})$ regret.

We now apply the result of Agarwal et al. (2014) for this discretized class of policies, yielding a regret of $O\left(\sqrt{\frac{T}{\eta\sqrt{d}}\ln\left(\frac{1}{\eta^d}\right)}\right)$. Adding the regret from the discretization, we obtain a total regret of

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

where the equality follows from the definition of η.

Proof of Lemma 1. 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,

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

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 that

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

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

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

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

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

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. □