Nonparametric Bandits Leveraging Informational Externalities to Learn the Demand Curve

Ian N. Weaver
Yale School of Management, ian.weaver@yale.edu,

Vineet Kumar
Yale School of Management, vineet.kumar@yale.edu,

We propose a novel theory-based approach to the reinforcement learning problem of maximizing profits when faced with an unknown demand curve. Our method is based on multi-armed bandits, which are a collection of minimal assumption non-parametric models that balance exploration and exploitation for maximizing rewards while learning across arms. Specifically, we build on Gaussian process bandits, which represent a flexible non-parametric model that unlike other non-parametric alternatives also provide principled estimates of uncertainty. We leverage the informational externality inherent in price experimentation across arms (price levels) in two ways: correlation between demands correspond to closer price levels, and demand curves are weakly monotonically decreasing. Incorporating these informational externalities limits unnecessary exploration of certain prices and performs better (characterized by greater profitability or reduced experimentation) than currently advanced approaches like UCB (with partial identification), or baseline Gaussian process bandits. Additionally, our method can be used in conjunction with methods like partial identification. Across a wide spectrum of true demand distributions our algorithm demonstrated a significant increase in rewards, most notably for right-skewed underlying WTP distributions where current approaches tend to underperform. Our algorithm performed consistently achieving between 96.9% and 99.1% of the optimal rewards depending on the simulation setting.

Key words: Multiarmed Bandits, Reinforcement Learning, Pricing

1. Introduction

We propose a new method based on reinforcement learning using multi-armed bandits (MABs) to efficiently learn an unknown demand curve. Our algorithm is non-parametric method based on Gaussian process Thompson sampling that incorporates microeconomic theory to converge towards optimal pricing and profits with significantly less experimentation than current methods. We use weak restrictions on the monotonicity of demand, which creates an informational externality between the outcomes of arms in the MAB. Our method exploits this informational externality to achieve higher profitability with the same level of experimentation as extant methods. The proposed method has several other

advantages relative to state of the art approaches, and can also be used in a complementary manner with recent approaches like partial identification (Misra et al. 2019).

In practice, the demand curve is only known in the price range that an established product is sold. The demand curve is generally unknown elsewhere with pricing mistakes being largest for new products that differ most from past products (Huang et al. 2022). A telling example is that of the Atlanta Falcons who in 2016 announced they would be dramatically slashing concessions to untested prices to improve brand equity. When asked how they projected the volume of sales to change, the CEO of the ownership group of the Falcons, Steve Cannon replied, "It could be a 10 percent bump, it could be a 30 percent bump, who knows."[1] The next season sales volume for food rose 50%.[2] Even with a sophisticated marketing team there is no perfect substitute for price experimentation about a particular product. Overall, McKinsey estimates that "30 percent of the thousands of pricing decisions companies make every year fail to deliver the best price."[3]

Knowledge of the demand curve is a primary starting point for managers to implement marketing strategies such as pricing, promotions, or even distribution. As firms do not know much about the demand curve for new and unknown products, those that do experiment often learn demand using trial and error (Furman and Simcoe 2015). Even in categories that are well known, demand can undergo significant changes in time. To learn demand at different price points, a simple approach is to use a balanced experiment (also called A/B testing) and randomly allocate consumers across a set of different prices. However, firms across a wide range of industries are reluctant to do much price experimentation (Ariely 2010) as managers worry that price experimentation can confuse or alter customers' expectations for uncertain gains. Even when field experiments are run, additional internal pressures often lead to managers not running them long enough making detecting effects difficult (Hanssens and Pauwels 2016). These factors imply that a method which can identify and learn the critical part of the demand curve efficiently from minimal experimentation can be quite valuable in consequently identifying the optimal price points

[1] https://www.ajc.com/sports/the-economics-the-falcons-new-cheap-stadium-food/8xvH1bAYTewjU2KQoc5HyN/

[2] https://www.washingtonpost.com/sports/2019/03/06/were-evangelists-this-why-atlanta-falcons-are-selling-hot-dogs/

[3] https://www.mckinsey.com/business-functions/growth-marketing-and-sales/our-insights/using-big-data-to-make-better-pricing-decisions

for products. This differs from advertising where many companies are willing to experiment (Pfeffer and Sutton 2006, Sahni and Nair 2020, Huang et al. 2018, Simester et al. 2009).

Managers are right to be wary of using a balanced experiment (A/B testing) for price experimentation as it can be quite wasteful. Why explore prices that obviously give poor profits as much as prices that experimentally give higher profits? Not only does it lead to a profit loss for the firm, but testing an obviously non-optimal price gives up an opportunity to test a different price. This is why the pricing problem is particularly suited to MABs which simultaneously learn while earning. They deal with the classic trade-off between *exploitation* (current payoff) and *exploration* (learning additional information) as the agent tries to maximize their rewards over some horizon.

In our demand estimation setting, we expand on canonical bandits by leveraging two distinct but related sources of *informational externality* across arms. First, we know that demand at closer price points are more likely to be similar compared to demand at more distant price points. This feature of the pricing problem, which we term information externality, implies that knowing the demand at a focal price point helps in learning about demand not only at that price, but also at other price points. Such learning is more likely to be greater for prices close to the focal price, and less for prices further away from it. However, rather than specifying this informational externality as a model restriction, we use a Gaussian process based model that allows it to be learned in a flexible manner from the data. Second is the characterization from microeconomic theory that demand curves are monotonic. Thus, the quantity demanded at a focal price must be *weakly lower* than the demand at all prices lower than the focal price. Incorporating these related sources of informational externality potentially helps us learn demand more efficiently and accurately, and forms the focus of this paper.

The primary advantage of our method is efficiency in learning the optimal price in a flexible non-parametric manner using reinforcement learning when the firm has no prior information about demand. If a firm is willing to undertake adaptive experimentation to learn demand, but faces a high cost of experimentation and wants to minimize that, our method helps the firm to a higher profitability than current approaches across a robust range of true underlying WTP distributions. Alternatively, if a firm can undertake significant experimentation, our method converges to the optimal profits significantly faster than alternative state-of-the-art approaches. A second advantage is that since our method

is based on first principles of microeconomic theory, it can be applied to any attribute with a vertical quality-like attribute, not just prices. More broadly, any domain-specific restrictions that can be imposed based on prior conceptual knowledge can be incorporated into this framework. Third, hyperparameter tuning is quick and automatic allowing the bandit to run in real time without pre-experimentation. Fourth, our method is based on Gaussian processes and has the advantage that is provides an estimate of uncertainty, along with point estimates. In fact, the entire posterior distribution can be characterized to provide a complete characterization of uncertainty around the learned demand curve. Fifth, an important aspect is that we do not require any knowledge about the market or consumer characteristics, unlike recent advances like UCB-PI, which requires some knowledge of a consumer's segment membership (Misra et al. 2019). Finally, our method can be used in conjunction with partial identification approaches, e.g. UCB-PI, making a hybrid approach potentially complementary.

Our method is non-parametric reinforcement learning, and is most appropriately situated in the class of other non-parametric learning models, including Upper Confidence Bound (UCB), Thompson sampling (TS) and Gaussian process Thompson sampling (GPTS), which we detail in turn. First is the classic A/B testing (balanced experiment) approach, which follows the "learn, then earn" strategy, or exploration followed by exploitation. Next is the class of reinforcement learning methods, which can be characterized as following a strategy of exploration and exploitation simultaneously. In this category is the well-known UCB algorithm (Auer 2002, Auer et al. 2002), which is more likely to explore the arms with higher payoffs, but also explores based on how many times each arm has been chosen. The idea is that when an arm has only been played infrequently, its payoffs may not have been accurately characterized relative to others, and thus trying the arm may result in learning more valuable information. The other class of algorithms is based on Thompson sampling (Thompson 1933), which involves a Bayesian updating of the rewards distribution corresponding to each of the arms as they are played. We base our method on GPTS. The algorithm works by modeling a GP on the prior data and then using Thompson sampling to pick a price. The key distinction here is this takes advantage of the information externality so instead of updating a single arm each round, the GP modeling the entire demand distribution is updated each time an arm is tried.

A Gaussian process (GP) is an infinite-dimensional collection of random variables, where any finite subset has a multivariate normal distribution. It is specified by a mean function and covariance function (a matrix of covariances), where the covariance represents the dependence between arms and is required to be positive semidefinite (Williams and Rasmussen 2006). Any valid kernel can be used to specify the covariance, e.g. radial basis, constant, squared exponential etc. (Srinivas et al. 2009). Specifically, GPs are distributions over random functions belonging to some class, where functions can be drawn according to some specified probability distribution over the class of functions. Similar to how distributions can be used to specify how scalar (or vector) values can be drawn from real numbers, GPs specify how functions can be drawn randomly from classes of functions. An important feature is that the functions can be non-parametric. Each function in the class represents a mapping from points in the support to a scalar or a vector. In our pricing application, each function represents a possible demand curve, which is a mapping from prices (across a specified support) to quantities, both of which are scalars. The GP in our application thus represents a distribution over possible demand curves. A major feature of GPs is that they are able to account for the information externality across the different arms as the entire distribution can be updated after the addition of only a data point at a single price.

The other core idea of this paper is the weak monotonicity of demand curves. As noted previously (Slivkins 2019, Misra et al. 2019), when using bandits in pricing incorporating monotonicity is important because it specifies a particular dependency between the different arms that would be ignored otherwise. This occasionally leads bandits oblivious to this dependency to make poor decisions from noisy data that would obviously be incorrect to the human eye (see Appendix EC.1 for a simple example illustrating this idea). One particularly notable contribution incorporating monotonicity into bandits is the partial identification method (Misra et al. 2019). Partial identification formalizes the notion that the rewards from a specific arm (price) can be dominated by another arm. Critically this method relies on both the assumption of individual weakly decreasing demand curves and the availability of highly informative segmentation data. Informative segmentation allows for the price experimentation data from a particular segment to be used to estimate the demand bounds for the entire segment. The demand bounds are then aggregated across

all segments to obtain the corresponding aggregated rewards bounds for each price. Dominated prices can then be eliminated if the lower bound for one price is higher than the upper bound for a different price. As segmentation becomes increasingly uninformative, the demand bounds approach the entire support at which point it would provide no new additional information. In contrast, our algorithm uses a weaker assumption of requiring a weakly decreasing demand curve only at the aggregate level, and it does not need informative segmentation to leverage the gains from the monotonicity assumption; the assumption is modeled directly into the sampling from the GP. As the mechanism is different it is possible to use partial identification as a complement to our GP-based approach.

A major challenge in using GPs for pricing problems is that we may obtain non-monotonic demand functions. This issue is exacerbated by the fact that a GP models the entire demand distribution meaning particularly noisy data at one price could affect the demand estimation at another. Therefore, a method that can obtain monotonic demand curves from a GP is highly valuable; however, specifying monotonicity in a GP is not trivial. While rejection sampling can be used to obtain a monotonic draw on the set of prices being considered, there is no guarantee that this can be done quickly especially as this set of prices gets larger. Instead, we specify monotonicity by building up a function interpolating its value from the sum of its derivatives at nearby points. For decreasing functions, the first derivative is always negative at every point. Since sign restrictions are easy to impose relative to shape restrictions like monotonicity, we are able to then restrict the first derivative of demand to be negative, leading to our method drawing only monotonic demand curves from the GP.

We next provide an overview of how our approach works. First, the demand distribution for each arm (price level) is specified by a prior. Then our algorithm uses Thompson sampling (TS) to obtain a draw from the GP which gives a demand at each price. This draw is then scaled to the reward distribution (either by the revenue or profit of each arm) and the arm with the highest reward draw is played. The realized purchase decision from playing the focal arm is then used to update the GP allowing for the demand distribution to be updated across all arms, and not just for the focal arm. However, when we obtain a demand draw for each arm, the resulting demand curve could well be non-monotonic. We implement monotonicity when drawing the underlying demand curve, by first specifying the underlying demand curve as the sum of its first derivative at nearby points. We then

impose sign restrictions on the derivatives, so that the obtained demand draws result only in monotonically decreasing functions, corresponding to a downward-sloping demand curve. The prior on the demand distributions also specifies the uncertainty (or noise), and we use weak theoretical restrictions to specify the distribution of noise. As noise varies across price (for example, a price where 10% of customers purchase will have less noisy data then a price where 50% of customers purchase), we also specify a model where the noise specification is heterogeneous allowing for more flexibility on the uncertainty of demand. If a specific arm has higher uncertainty, the algorithm may be marginally more likely to explore either that arm or nearby arms, since the informational externality property allows for learning from nearby arms.

In a series of simulations across a range of settings, our algorithm outperforms a set of benchmarks. We found that our algorithm gained higher expected total rewards than UCB, TS, UCB-PI, and GPTS (without monotonic sampling). Averaged across three simulation settings (repeated 1000 times each), our algorithm had a percent increase in expected rewards of 59.6%, 36.0%, 10.1% and 3.1% over using UCB, TS, UCB-PI, and GPTS respectively. There was massive heterogeneity in these results depending on the underlying distribution with the biggest boost occurring for right-skewed distributions. This is because the benchmarks tend to over-explore higher prices meaning they perform well for left-skewed distributions but poorly for right-skewed distributions. Meanwhile, our algorithm leverages monotonicity to limit over-exploration of higher prices leading to more consistent results; on the simulations we ran our algorithm achieved between 96.9% and 99.1% of the optimal rewards depending on the underlying distribution. Additionally, our algorithm can be combined with the partial identification method, but we found that the additional benefits were quite small only further increasing the expected rewards from 0.01% to 0.16%.

Contribution: This paper proposes a combination of reinforcement learning and economic theory by enforcing monotonicity on non-parametric bandits. More generally, this method can be easily adapted to any bandit situation where the underlying data has a vertical attribute. We provide a practical, quick, implementable algorithm with minimal assumptions that outperformed benchmarks in a series of simulations. Most importantly, it does not depend on a restrictive parametrized model and hyperparameter tuning is automatic and fast enough that the algorithm could be deployed in industry.

There are two main contributions. First, we provide a method that builds upon Gaussian process bandits to account for economic theory imposing the general property that demands curves are downward sloping. We specify an algorithm that efficiently obtains only monotonic, downward-sloping demand curves throughout the experimentation process. Our approach results in significantly higher profits relative to state-of-the-art methods in the MAB literature while having a lower variance between trials. Additionally, this increase in efficiency means that a larger number of prices can be included in the consideration set. These are important managerial considerations given the reluctance to do pricing experiments, as compared to (for example) advertising experiments. Second, we provide a detailed discussion around hyperparameter tuning including a new variant which deals with the heterogeneity in noise pertinent to the pricing scenario and leads to further increases in algorithmic efficiency. More broadly, we show that in any situation where data has some general known form a priori, that these constraints can be combined with the flexibility of non-parametric bandits to improve empirical performance.

Our proposed method has specific limitations. First, to use our method requires the firm to be willing to experiment with prices. Some firms may be reluctant to do so at all, and in such cases our method clearly in not useful. Second, our method requires all the assumptions of A/B testing, namely that each consumer enters the experiment once and not multiple times, and there is no temporal variation in the WTP distribution as we conduct the experiment. Third, while our approach is based on first principles, and a weak restriction of monotonicity, there are cases where such an assumption might not hold (for example, in some cases where price strongly serves as a signal of quality demand could increase with price). Fourth, while we have demonstrated that our method works to more efficiently learn the critical parts of the demand curve across a wide variety of distributions used in prior research, we cannot exhaustively enumerate and test all possible distributions. Finally, GPs can get computationally complex and intractable as the number of observations increases. In our application, however, this is less of a concern since the experimentation occurs over the bounded support of price levels, and more importantly the experimentation is performed over the number of discrete prices, rather than the number of consumers.

2. Literature Review

This section aims to provide an overview of dynamic pricing as well as multi-armed bandits. The goal is to provide context for the many avenues of research in this broad field. Many of these works are discussed in greater detail elsewhere in the paper.

Dynamic Pricing: Traditionally, many papers in marketing, economics, and operations make strong assumptions about the information that a firm has regarding product demand. The strongest assumption used is that the firm can make pricing decisions based on knowing either the demand curve or the underlying distribution of customers' valuations. (Oren et al. 1982, Rao and Bass 1985, Tirole 1988). A relaxation of this assumption is to assume that firms know the demand only up to a parameter, used in some of the earlier works on learning demand through price experimentation (Aghion et al. 1991, Rothschild 1974). More typically in marketing, where the focal point is to understand the impact of various marketing variables on purchase decision, the consumer utility function is written as a sum of observable characteristics including price and then estimated from data (Zhang and Chung 2020, Jindal et al. 2020, Huang et al. 2022).

However, a truly robust pricing policy must be able to incorporate all possible demand curves making nonparametric approaches the gold standard. In economics, nonparametric approaches tend to account for state changes between periods (Bergemann and Schlag 2008, Handel and Misra 2015). While this does have benefits, it tends to not be analytically tractable (for example, Handel and Misra (2015) consider a stylized two-time period model). A more tractable nonparametric approach in the operations literature is to have an *exploration* phase followed by an *exploitation* phase (Besbes and Zeevi 2009), however this assumes that the state does not change between the exploration and exploitation phases. While the algorithm in Besbes and Zeevi (2009) corresponds to a balanced experiment, further refinements to the "learn, then earn" approach have been made (Cheung et al. 2017, Wang and Hu 2014).

For targeted dynamic pricing and couponing, hidden Markov models (Zhang et al. 2014), machine learning (Dube and Misra 2022), and reinforcement learning (Liu 2022) have been used. Of particular interest, Dubé and Misra showed the importance of price experimentation in industry by running a balanced experiment at ZipRecruiter to train their model. The results were so prominent that the firm implemented the best uniform price policy leading to an increase in profits of 55%. Other nonparametric approaches include MAB

methods, which are discussed in more detail in the next section. More exhaustive reviews on dynamic pricing can be found in literature reviews by Aviv and Vulcano (2012) and den Boer (2015).

Multi-armed Bandits: Multi-armed bandit methods are used across many fields including computer science, statistics, operations and marketing. They have many uses beyond pricing including advertising (Schwartz et al. 2017), website optimization (Hill et al. 2017, Hauser et al. 2009), recommendation systems (Kawale et al. 2015), and arcade games (Osband et al. 2016).

Two of the most fundamental bandit algorithms are Upper Confidence Bounds (UCB) and Thompson sampling (TS). The main difference is that UCB (Auer et al. 2002) is a deterministic nonparametric approach, while TS (Thompson 1933, Chapelle and Li 2011) is a non-deterministic parametric approach; full descriptions of these algorithms can be found in Section 4. Variants of both these approaches have been applied to dynamic pricing (Ferreira et al. 2018, Misra et al. 2019).

Notably, UCB and TS have been compared to "learn, then earn" benchmarks. Multiple papers find that bandits outperformed "learn, then earn" over a fixed number of trials even when the ex-post optimal exploration time in "learn, then earn" was used (Ferreira et al. 2018, Misra et al. 2019). However, the comparison is uneven as "learn, then earn" approaches limit the number of rounds where experimentation is possible, while bandits have no such limitations. This distinction is important for practical purposes as companies may only be able to commit to limited experimenting before choosing a fixed price (a common situation as discussed in Cheung, Simchi-Levi, and Wang 2017). While it may seem that "learn, then earn" algorithms are tailored for these situations, the vast range of new bandit methods and the lack of research comparing "learn, then earn" and bandit methods under limited experimentation conditions warrants further research.

The main limitation of TS and UCB in pricing is that they consider rewards to be independent between arms ignoring the information from the underlying demand curve. One attempt to incorporate this information with a UCB algorithm is by Misra, Schwartz, and Abernethy (2019). They modify UCB by filtering out dominated arms by leveraging segmentation and individual-level weakly decreasing demand curves. The key limitation is that it relies on *segmentation* data, which is not needed in our algorithm.

Other bandit algorithms have been created to allow for an unknown function correlating arms including Lipschitz bandits (Bubeck et al. 2011, Kleinberg et al. 2008), Asymptotic Randomised Control bandits (Cohen and Treetanthiploet 2021), and Gaussian process bandits (Srinivas et al. 2009). However, these methods do not restrict the types of underlying demand functions to be weakly decreasing.

Gaussian Processes: In recent years, Gaussian processes have become one of the most prominent tools for Bayesian machine learning (Williams and Rasmussen 2006) with one of the main advantages being they not only predict means but also the uncertainties surrounding these estimates. Taking advantage of these properties, Gaussian process bandits were introduced by Srinivas et al. (2009) through the *GP-UCB* algorithm. This algorithm fits a Gaussian process to the data and then picks a test point using a UCB-like decision rule based on the estimated mean and variance at each test point. One difficulty with GP-UCB is that performance is dependent on choosing an acquisition function which balances the trade-off between exploration and exploitation. The performance of each acquisition function varies greatly dependent on the underlying distribution (Hoffman et al. 2011), which is not known a priori. Thompson sampling provides a Bayesian approach to automatizing this trade-off and has been proven to have equivalent bounds to GP-UCB (Russo and Van Roy 2014).

An application of Gaussian process bandits to dynamic pricing is by Ringbeck and Huchzermeier (2019).[4] Their paper combines Gaussian process bandits with inventory constraints which are common in the dynamic pricing operations literature (Besbes and Zeevi 2009, Ferreira et al. 2018). Our paper differs in two ways. First, we incorporate the weakly decreasing nature of demand curves by providing a principled way to sample Gaussian process bandits so that only weakly decreasing estimates are obtained. Second, we provide a characterization of automatic hyperparameter tuning,[5]. We found tuning had such a varying effect on results that we provide a new tuning method specific to dynamic pricing.

3. Model Preliminaries

The general setup and assumptions are common to all benchmarks as well as the novel algorithm presented in this paper.

[4] The only application known to the authors.

[5] There is no mention of hyperparameter tuning in Ringbeck and Huchzermeier (2019)

Assumptions: For each consumer we assume that they have stable preferences, budget, and outside options. This first set of assumptions are typical to any field experiment, and necessary for results of an experiment to be applicable after the experiment has ended. This rules out dynamic situations where customers may change over time or customers' current decisions are greatly affected by future beliefs. Such situations include strategic consumers (Nair 2007), learning (Erdem and Keane 1996, Yu et al. 2016) and stockpiling (Ching and Osborne 2020, Hendel and Nevo 2006). These assumptions are typically implicit in most field experiments, e.g. in the advertising literature (Hoban and Bucklin 2015, Lambrecht et al. 2018, Gordon et al. 2019). For example, if strategic consumers seeing a discount believe the company is experimenting and might discount even more at a later time, then the results would not accurately reflect the treatment effect.

Additional assumptions consistent with much of the MAB pricing literature are that the firm is a monopolist[6] (Kleinberg and Leighton 2003, Besbes and Zeevi 2009, Misra et al. 2019), does not price discriminate (Besbes and Zeevi 2009, Ferreira et al. 2018, Ringbeck and Huchzermeier 2019, Misra et al. 2019), and does not have inventory constraints.[7] Our algorithm can be extended to include inventory constraints, which are common to the Operations literature (Besbes and Zeevi 2009, Ferreira et al. 2018, Ringbeck and Huchzermeier 2019), by incorporating a maximization constraint to the algorithm. Additionally, we assume that firms do not know anything about consumer valuations a priori. While this is standard within the MAB literature, the standard pricing literature often assumes that shape or parametric class of the demand curve is known (Rao and Bass 1985, Nair 2007).

The key additional assumption that our paper makes is that the aggregate demand curve is weakly decreasing; this is implemented when a demand draw is taken from the GP. In contrast, Misra et al. (2019) assume weakly decreasing demand curves at the individual level, which are then aggregated across segments to obtain demand bounds at each price. While a fairly standard economic assumption, violations of demand curve monotonicity are possible even in a stable environment. For example, when firms are actively responding to price changes in real-time, the algorithms can end up running correlated experiments

[6] This assumption can actually be weakened as results will hold as long as the algorithm is deployed in a stable environment where competitors do not change prices strategically in response to real-time changes and firms are not worried about future competitive entries (Rubel 2013).

[7] Inventory constraints are necessary when inventory is limited, such as in clothing. When the product is limited it may be best to forgo selling to a customer in lieu of another customer with a higher WTP. For products without production constraints (e.g. a Netflix subscription) this is not an issue.

(Hansen et al. 2021) where raising the price can lead to an increase in demand as a result of a change in competitor behavior. Another reason why a consumer might not purchase at a low price even though they would at a higher price is because too low a price could signal issues regarding the quality or authenticity of a product. We expect that these violations are likely to occur at very low or very high prices which are generally outside the set of reasonable prices that a company would be willing to test. Further, our assumption is less restrictive as a small number of customers with non-monotonic demand curves within the range of test prices does not necessarily violate assumptions about the monotonicity of the aggregate demand curve (Quah 2000).

3.1. Model Setup

The MAB problem applied to pricing without inventory constraints has three components: actions, rewards, and a policy. The first component, actions, refers to the set of prices from which the firm can choose. Prior to the experiment, the firm selects a finite set of K ordered prices $P = \{p_1, ..., p_K\}$ where $p_1 < p_2 < ... < p_K$, where prices are scaled so that $0 \leq p_k \leq 1 \ \forall p_k \in P$. If $\{\tilde{p_1}, ..., \tilde{p_K}\}$ are the unscaled prices then the set of scaled prices can be created by dividing any price by the largest price in the set; that is $p_k = \tilde{p_k}/\tilde{p_K}$. At each time-step t of the experiment the firm chooses a price p_k from P.

The second component, rewards, refers to the profits that a firm makes at each purchase opportunity. The firm faces an unknown true demand $D(p)$, and the true profit is given by $\pi(p) = pD(p)$. The true profit is not observed, and instead the firm observes noisy realizations of profits for each price p_k. Considering the data at each price separately, define n_{kt} to be the number of times that price p_k (arm k) has been chosen through time t. The profit realizations at a given price p_k at time t can be denoted $\pi_{k,1}, \pi_{k,2}, ..., \pi_{k,n_{kt}}$ with a corresponding sample mean of $\bar{\pi}_{kt} = 1/n_{kt} \sum_{\tau=1}^{n_{kt}} \pi_{k,\tau}$.

The final component is a policy, Ψ, which is a decision-making rule that picks a price each round using the history from the past rounds. In this situation the history can be written as $H_t = \{S_t = (s_{1t}, ..., s_{Kt}), N_t = (n_{1t}, ..., n_{Kt})\}$ where s_{kt} denotes the number of purchases for an action k through time t and n_{kt} denotes the total number of times action k was tested through time t. Formally, in round t the policy picks a price using the history from the past $t-1$ rounds: $p_{k_t} = \Psi(P, H_{t-1})$. What distinguishes various MAB algorithms is how this policy is defined.

To summarize, there is an unknown distribution of consumer valuations for a product. In each round, the firm shows a price and then the customer makes a purchase decision which is observed by the firm. Following Misra et al. (2019), we will assume that the consumer can only buy 0 or 1 units of a product and that the company can change the price every 10 customers.[8] A summary of the notation is given in Table 1.

Table 1 Summary of Bandit Notation

Notation	Description	Formula
Ψ	Policy for dynamic pricing (i.e. decision rule)	
k	Action index: the set of actions consists of K choices	$k \in \{1, 2, ..., K\}$
t	Time-step: denotes the t-th customer of the price experiment	
k_t	Action chosen at time t: this is dependent on the set of actions, policy and past history	$k_t = \Psi(P, H_{t-1})$
p_k	Scaled prices	$p_k \in P = \{p_1, ..., p_K\}$ where $0 \leq p_k \leq 1 \; \forall p_k$
n_{kt}	Number of times price p_k has been tested through time t	
s_{kt}	Number of purchases at price p_k through time t	
H_t	History from past t rounds of experiment	$H_t = \{S_t = \{s_{1t}, ..., s_{Kt}\}, N_t = \{n_{1t}, ..., n_{Kt}\}\}$
$D(p_k)$	Demand at price p_k	
$\bar{D}_t(p_k)$	Sample mean of demand through time t of price p_k	
π_{k_t}	Profit realized when price p_k was tested in time period t	
$\bar{\pi}_{kt}$	Sample mean profit through time t of price p_k	

The empirical performance of different algorithms will be measured by comparing the total rewards over numerous simulations, as well as the variance of the rewards, with algorithms having higher average total rewards and lower variance considered as superior. A detailed discussion of the performance metrics (and their relationship to regret, which may be more familiar to some readers) can be found in Appendix EC.2.

4. Benchmarks

The benchmark algorithms come from the most prominent papers in machine learning, operations, and marketing research on dynamic pricing using MABs. They include UCB (Auer 2002), Thompson sampling (Ferreira et al. 2018) and UCB-PI (Misra et al. 2019). We will compare a standard implementation of Gaussian process Thompson sampling (Srinivas

[8] Alternatively one could specify that prices can be changed every X minutes and have customers arrive according to a Poisson distribution. Both setups are just approximations of how the algorithm may be deployed in industry.

et al. 2009), henceforth called *GPTS*, with these benchmarks and then assess the additional benefit of requiring monotonic sampling, allowing for heterogeneity in noise, and combining our algorithm with partial identification. Complete details on these benchmark algorithms can be found in Appendix EC.3.

Thompson sampling (*TS*) is a randomized Bayesian parametric approach. For each action, a reward distribution is specified a priori and updated by the history of past trials (Thompson 1933). In each round, an action is chosen according to the probability that it is optimal given the history of past trials. The easiest implementation is to take a sample from each distribution during each round and pick the arm that gives the highest payoff in the sample.

In contrast, the Upper Confidence Bound policy (*UCB*) is a deterministic non-parametric approach popularized because it is proven to asymptotically have the best possible performance in terms of achieving the lowest maximum regret[9] (Lai and Robbins 1985, Agrawal 1995, Auer et al. 2002). This is subtly different from empirical performance. The appeal of UCB is that it provides provable guarantees; the proof states that UCB has the lowest possible bound for expected regret, but it does not rule out the possibility that another algorithm will empirically have a lower regret (even if the bound is larger). There is empirical and theoretical evidence to suggest that greedy algorithms will empirically outperform UCB especially for large number of arms (Bayati et al. 2020).

UCB uses a formula that scores each action in any given round and the highest scoring action is picked. It is a deterministic algorithm implying that given a particular history from past rounds, it will always choose the same action. The scoring rule is the sum of an exploitation and exploration term. The exploitation term is the sample mean of past rewards at a given action, which informs which actions have previously had higher payoffs.

The exploration bonus, meanwhile, is increasing in how uncertain we are about the sample mean for an action; that is, it decreases with the number of times an action has been chosen. A UCB variant, known as *UCB-Tuned*, considers the variance of the rewards received at an arm directly with higher variances corresponding to higher exploration bonuses. It is shown to empirically perform better than UCB (Auer et al. 2002). Applying UCB to pricing requires an additional variant as UCB typically assumes that each

[9] For those unfamiliar with the concept of regret see Appendix EC.2.

action has the same rewards support. For pricing, however, the reward is dependent on the price chosen so a simple fix is UCB-Price-Tuned which is simply UCB-Tuned where the exploration bonus is scaled by the price (Misra et al. 2019).

When applied to pricing, UCB can be combined with *partial identification* by leveraging segmentation information and weakly decreasing demand curves to eliminate dominated prices from consideration. This prevents UCB from potentially picking these bad prices leading to a significant performance improvement detailed in a recent state-of-the-art approach (Misra et al. 2019). This algorithm is known as *UCB-PI* and also has a tuned variant called *UCB-PI-Tuned*.

For partial identification to work, it requires segmentation. Specifically, this means that a company can sort customers into segments of known sizes and every customer within a segment will have WTP bounded within a range (this range is not assumed but rather estimated within the algorithm). This information allows for demand bounds to be estimated for the entire segment using purchase decision data from customers who have participated in the experiment; the demand bound estimates are then updated for a particular segment every time a customer *from that segment* makes a purchase decision. In creating segments, the trade-off is that while a larger number of more accurate segments gives tighter bounds it also takes longer to get data on all the segments.

These demand bounds are then aggregated across all segments to obtain demand bounds (and their corresponding reward bounds) at each price. If the upper reward bound at a particular price is less than the lower reward bound at another price, then that price is dominated and removed from consideration. UCB-Tuned is then run on the set of non-dominated prices rather than the original set of prices.

The process of estimating demand bounds for a segment using purchase data utilizes weakly decreasing demand curves at the individual level. The idea is that the customer decision at a given price provides information at what would have occurred had the customer been shown a different price. Specifically, if a customer bought at price p then they would purchase at all prices lower than p. Similarly, if a customer did not buy at price p then they would not purchase at any price above p. Thus it is possible to calculate the highest price where all consumers in segment s purchase and lowest price where no consumer from segment s purchase, which establish the demand bounds for a particular segment (full details can be found in Appendix EC.3.3.1).

Critically, the performance boost of partial identification depends on the quality and accuracy of the segmentation. The idea of the method is that if customers within a segment have a similar WTP and segment sizes are known then the purchase decision information from a few customers can be used to provide information about a portion of the underlying WTP distribution. As segmentation becomes less uninformative and the range of valuations for each segment approach the entire support (that is, there is only one segment because segments are indistinguishable), then the ability to learn something about the underlying WTP distribution diminishes to 0. More technical details can be found in Appendix EC.3.3.2).

In contrast, a key contribution of our method is it does not require segmentation to leverage the importance of economic structure from weakly decreasing demand curves. However in our algorithm, an arm cannot be completely removed from consideration; there is always a chance of any price arm winning a draw, even if the chance is infinitesimally small. This means that, like with UCB, our algorithm can be combined with the partial identification method for potentially further gains. We examine the question of quantifying the additional value added by including segmentation.

5. Gaussian Process Bandits

The fundamental algorithm used in this paper is Gaussian process Thompson sampling (Srinivas et al. 2009) adapted to dynamic pricing. This provides a flexible non-parametric method for modeling the underlying demand function, which is an improvement over methods that do not consider correlation between arms.

5.1. Overview of Gaussian Processes

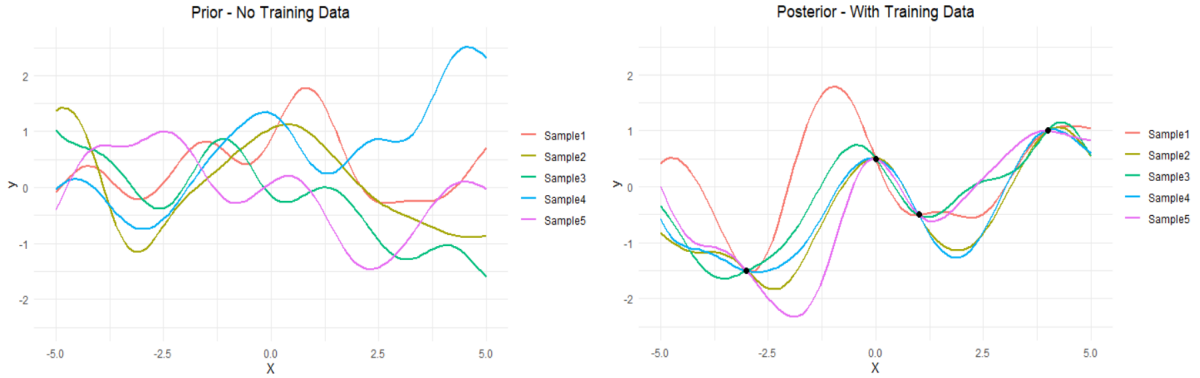
A Gaussian process (GP) is a stochastic process (collection of random variables) such that every finite subset of random variables has a multivariate Gaussian distribution. Intuitively any point $\mathbf{x} \in \mathbb{R}^d$ being assigned a random variable $f(\mathbf{x})$, then a Gaussian process can be thought of as a *probability distribution over possible functions.*

Importantly, Bayesian inference can be used on GPs. A typical Bayesian approach would be to estimate a distribution over the parameters of a parametric function, but instead Gaussian processes can be used to infer a distribution over functions directly. That is, a Gaussian process can be used to define a prior over functions and as data is observed, the posterior distribution of functions will change. GPs are particularly useful as they are

flexible enough to model any continuous underlying demand function (like a non-parametric method such as KNN) while providing a measure of uncertainty about the estimates like a parametric model.

To illustrate, consider a situation where the goal is to learn a function f at some test points X^* from some potentially noisy data $\mathcal{D} = \{X, \mathbf{y}\} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$ where each $\mathbf{x}_i \in \mathbb{R}^d$ and each $y_i \in \mathbb{R}$. The output data y_i gives a noisy signal of the true value of f at x_i; that is, $y_i = f_i + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$. For ease of exposition we will set $\mu(\mathbf{x}) = 0$.[10] A prior probability distribution over the space of functions is set, which changes when training data is incorporated. For example, Figure 1 illustrates a situation with noiseless data,[11] which means every function drawn from the posterior distribution must go through the data points. In situations with noise it would be less clear cut, but the range of uncertainty around the data point would become smaller than for areas where there is no data.

Figure 1 Random Samples from Gaussian Process With and Without Training Data



Notes. Lines represent 5 random samples from the GP in both the prior and the posterior. In the prior, the mean was set to 0. For both the prior and posterior the RBF kernel was used with hyperparameters $l = 1, \sigma_f = 1, \sigma_y = 0.0001$. There were 101 test points $X^* = \{-5, -4.8, ..., 5.8, 5\}$. The five samples from the posterior distribution are drawn from the GP with training data $X = \{-3, 0, 1, 4\}$, and $Y = \{-1.5, 0.5, -0.5, 1\}$.

Formally, the assumption is that f is jointly Gaussian distributed and completely defined by its mean, $\mu(\mathbf{x})$, and covariance function $k(\mathbf{x}, \mathbf{x}')$ such that $f \sim \text{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')$. The mean and covariance function are defined as follows (Williams and Rasmussen 2006):

$$\mu(x) = \mathbb{E}[f(\mathbf{x})] \tag{1}$$

[10] If the prior mean function is non-zero, when $f \sim GP(\mu, k)$ the function $f' = f - \mu$ is a zero-mean Gaussian process $f' \sim GP(0, k)$. Hence, using observations from the values of f, one can subtract the prior mean function values to get observations of f', and do the inference on f'. Finally, after obtaining the posterior on $f'(X^*)$ one can simply add back the prior mean $\mu(X^*)$ to the posterior mean to obtain the posterior on f.

[11] A very small amount of noise is added for computational purposes.

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}')] \tag{2}$$

The covariance function $k(\mathbf{x}, \mathbf{x}')$, also known as a *kernel*, controls the shape of the function underlying data. Some examples include linear, cosine and Laplacian kernels, but a particularly common and robust kernel used for bandits (Srinivas et al. 2009) is the Radial Basis Function (RBF), which is a particular case of the more general class of Matérn kernels. The RBF kernel is defined as follows:

$$k^{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 e^{-\frac{1}{2l^2}(\mathbf{x}-\mathbf{x}')^2} \tag{3}$$

where σ_f and l are hyperparameters. The vertical span of the function is described by σ_f, while l describes how quickly the correlation between two points drops as the distance between them increases (Shahriari et al. 2015). As the kernel completely controls the shape of the function, it is imperative that they are tuned correctly or else the posterior GP will provide poor estimates.

A final hyperparameter is the noise parameter, σ_y^2, which is a measure of how noisy the training outputs y_i are for a function f. Recall that, $y_i = f_i + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$. This value can either be supplied (for example, a scientific instrument would have a known precision error) or estimated with the shape hyperparameters. However, in general, leading GP researchers suggest using domain knowledge when possible as it might be difficult to disentangle shape and noise parameters (Murray 2008). For example, consider two close input points (x-axis) that have very different outputs (y-axis). One possible explanation is the data is accurate and the GP needs shape parameters that permit sufficiently high variation to allow for large output differences from nearby inputs. Alternatively, it could be that the true outputs are actually close together but that the data is very noisy; in this case the previous shape parameters would be overfitting.

The kernel can be used to compute a covariance matrix $K(X^*, X^*)$ containing the covariance between all sets of test points as well as a covariance matrix (either $K(X^*, X)$ or $K(X, X^*)$) between training and test cases. Then, the joint distribution of the training data X and the test points X^* can be written as follows (equation (2.21) in Williams and Rasmussen (2006)):

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}^* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_y^2 I & K(X, X^*) \\ K(X^*, X) & K(X^*, X^*) \end{bmatrix}\right) \tag{4}$$

where $\mathbf{f}^* = f(X^*)$ is a random variable denoting the Gaussian process posterior prediction. It then follows from equations (2.22 - 2.24) in Williams and Rasmussen (2006) that

$$\mathbf{f}^*|X, \mathbf{y}, X^* \sim N(\mu(\mathbf{f}^*), \text{Cov}(\mathbf{f}^*)) \text{ where} \tag{5}$$

$$\mu(\mathbf{f}^*) = K(X^*, X)[K(X, X) + \sigma_y^2 I]^{-1}\mathbf{y} \text{ and} \tag{6}$$

$$\text{Cov}(\mathbf{f}^*) = K(X^*, X^*) - K(X^*, X)[K(X, X) + \sigma_y^2 I]^{-1}K(X, X^*) \tag{7}$$

5.2. Overview of Gaussian Process Bandits

Gaussian processes can be combined with bandits such as UCB and TS (Srinivas et al. 2009). The main difference is that instead of using the raw data directly, a posterior GP is estimated before using a UCB scoring rule or Thompson sampling. The advantage is that a GP estimates the entire function over the support, when raw data does not use information from nearby arms. This can minimize incorrect "exploitation" in early rounds, while still learning the true rewards over the long run.

A common question is why is estimating a GP necessary instead of some other non-parametric method? What is the advantage of having a posterior distribution of functions rather than just a posterior mean? The issue is an estimated posterior mean does not accurately reflect the uncertainty in the data the way a GP or using the raw data does. This can cause the algorithm to get stuck and under explore leading to inaccurate results. In Srinivas et al. (2009), they tested a decision rule optimizing the rewards from only the posterior GP mean estimates, rather than the entire distribution, and found it was "too greedy too soon and tends to get stuck in shallow local optima" (p. 4).

Gaussian processes know what they do not know, which allows the uncertainty around the estimates to be incorporated into the decision making process in a principled manner. Whereas other machine learning methods might possibly provide higher accuracy estimates for the mean, they are not as competent at knowing to what certainty the prediction is true. As a result, they are ill-suited for bandit algorithms where controlling the exploration-exploitation trade-off is crucial. This is why a method like GPs is needed where the entire distribution of functions is estimated when creating a smoothing alternative to the raw data.

The basis of the approach used in this paper is *GPTS* which combines Gaussian processes with Thompson sampling (summarized in Algorithm 1). Each round, a posterior GP is estimated from the data and then randomly sampled. The arm that gives the highest reward is chosen, the result is observed and the process is repeated.

Algorithm 1: GPTS

```
Choose a set of prices, and set necessary priors.
for t = 1, 2, ... do
    Estimate GP from sample data
    Obtain random draw D* from posterior GP at test points
    Play price p_k := arg max_{p_k} p_k D*(p_k)
    Observe purchase decision and update history
    Perform Bayesian update
end
```

6. GPTS adapted to Pricing

This section contains several aspects pertinent to running GPTS for a pricing experiment. We define the notation relevant to GPTS in pricing, detail how model hyperparameters are tuned, and discuss how our approach deals with heteroscedastic noise. Finally, we specify how the theoretical restriction of monotonicity of demand curves is incorporated into the GPTS pricing model.

Notation: After round t, the training data consists of a set of t data points representing the prices tried and rewards received in each trial. More formally, the input training data are the prices tried $P_t = \{p_{k_1}, ..., p_{k_t}\}$ with corresponding profits received as the output training data $\boldsymbol{y} = \{\pi_{k_1}, ..., \pi_{k_t}\}$. While the test points can be any points within the support of P, so firms are typically interested in learning the demand at particular prices,[12] for simplicity we set the test points to be P. A separate discussion beyond the scope of this paper is how to choose P. While firms may be reticent in testing a large number of prices, large differences in demand between consecutive prices can be potentially problematic when fitting a GP.

[12] For example, many firms price products ending in 99 cents (Stiving and Winer 1997) because of left-digit bias (Bizer and Schindler 2005), which states that consumers focus more on the left-digits of a price (dollars) than the cents. One recent study, estimated that rise in price from \$4.99 to \$5.00 had an equivalent effect on demand to a rise of 15-25 cents (Strulov-Shlain 2019). Additionally, unusual or unfamiliar prices can be confusing to customers, which can also be detrimental to business.

Table 2 Summary of GP Notation

Description	Notation in Pricing Setting
Input training data	$P_t = \{p_{k_1}, ..., p_{k_t}\}$
Output training data	$\boldsymbol{y} = \{\pi_{k_1}, ..., \pi_{k_t}\}$
Test points	$P = \{p_1, ..., p_K\}$
Training data noise	$\boldsymbol{\sigma_y^2}$
RBF kernel	$k^{\text{RBF}}(p_i, p_j) = \sigma_f^2 e^{-\frac{1}{2l^2}(p_i - p_j)^2}$
RBF hyperparameters	$\{\sigma_f, l\}$
Covariance function (kernel) evaluated at two points	$k(p_i, p_j)$
Covariance matrix between all sets of test points	$K(P, P)$
Covariance matrix between training and test cases	$K(P_t, P)$ or $K(P, P_t)$

Tuning Hyperparameters: As the kernel controls the function shape, a crucial practical step in GP bandits implementation is the selection of hyperparameters. An efficient, accurate and automatic method is needed for picking hyperparameters for the algorithm to be used in industry. To recap, there are two types of hyperparameters that need to be picked: shape parameters for the kernel, $\{\sigma_f, l\}$, and the noise parameter, σ_y^2, which dictates how noisy the data is. There are two main methods for tuning the hyperparameters. The first is the typical machine learning method of estimating the GP on training data and testing a grid of hyperparameters to see which performs best on the test data. Another standard process is to choose values of the hyperparameters that maximize the likelihood of the data given a model. Mathematically, this is equivalent to minimizing the negative log marginal likelihood (equation 2.30 in Williams and Rasmussen (2006)).

$$\log \text{prob}(\boldsymbol{y}|P_t) = -\frac{1}{2}\boldsymbol{y}^T(K(P_t, P_t) + \boldsymbol{\sigma_y^2}I)^{-1}\boldsymbol{y} - \frac{1}{2}\log|K(P_t, P_t) + \boldsymbol{\sigma_y^2}I)| - \frac{t}{2}\log(2\pi) \quad (8)$$

where t denotes the number of data observations (i.e. the number of rounds in a bandit).

For our baseline implementation, we choose to specify the noise parameters directly. Purchase decisions are binary, which implies that we can characterize the upper bound of the variance at any price. When prices are set in $p \in [0, 1]$, the maximum variance in the purchase probability is 0.25 which occurs when the true purchase probability is 0.5. Thus, a conservative approach without estimating the noise parameter would be set it to 0.25. While this will be too conservative for some prices, using noise parameters that are too small could lead to a lack of search and potentially poor results. We call the implementation of GPTS where the hyperparameters are estimated in this manner *GPTS-Homo*, to reflect that the noise for the sample data is homogeneous across price levels.

A computational issue that arises in fitting a posterior Gaussian process is that the matrix inversion is $\mathcal{O}(n^3)$, implying that it does not scale well to larger datasets. This issue

is particularly problematic in calculating the hyperparameters which require many calculations of the negative log marginal likelihood for the minimization algorithm to converge. However, in our setting, we are able to avoid this problem since only prices from a limited, finite support P are tested, allowing for significant simplification. Rather than treating each customer decision as a separate data point, the sample purchase probabilities at each price in P can be used. This reduces the cardinality of the training data from the round t to instead be the number of prices being tested $|P|$, thus resolving the computational complexity of the algorithm becoming much slower over time. Complete details along with to deal with precision float errors can be found in Appendix EC.4.

Heteroscedasticity: We have so far assumed that the data is homoscedastic; that is, the noise is the same regardless of the price. Mathematically this means $\bar{D}_t(p_k) = D(p_k) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$. However, we know that in our application (and others) that this is unlikely to be true. The variance in the observation data is much higher at a price where half the customers purchase versus a price where almost no customer is purchasing.

There are a few potential remedies. The first is to use a heteroscedastic Gaussian process method (Goldberg et al. 1997, Kersting et al. 2007). The problem with these methods is they are overconfident in areas with little data and high variance. For example, if there is an area with few data points, but they all have similar values we cannot know whether that is because the true variance is small or because we have not sampled enough. Thus, finding the most likely heteroscedastic GP (Kersting et al. 2007) is unsuitable for the bandit scenario. As the bandit explores only as much as is necessary there will be areas with few data points prone to underestimation of the noise leading to inaccurate results if a "most likely" heteroscedastic Gaussian process method were used.

Another approach would be to model the error by specifying some structural form. In our case that means the noise of the data is dependent on how likely a customer is to purchase. In general, we can write $\sigma_y^2 = g(D(p))$ for some unknown function g. However, noise depends on the underlying distribution which is completely unknown, making it unlikely that any suitable candidates for $g(\cdot)$ exist.

Since no structural form can be specified, we propose a new solution to the issue of heteroscedastic noise in pricing. The first step is to estimate the GP using homoscedastic noise, which produces reasonable results erring on the side of caution.[13] A demand draw is

[13] The problem with the homoscedastic specification is that it will overestimate the variance in some areas.

then taken from this GP. As the purchase decision is Bernoulli then a noise estimate can be calculated from the demand draw for each p in the test set as $\tilde{D}(p)(1-\tilde{D}(p))$. Another GP is then derived using this estimation as the noise input (tuning the shape hyperparameters per usual). We call the implementation of GPTS where the hyperparameters are estimated in this manner *GPTS-Hetero*, reflecting a flexible heteroscedastic specification.

6.1. Monotonicity

We now specify how to obtain a weakly decreasing demand curve (function) from the GP. Recall that the baseline GP allows for any function, and does not impose any restrictions on the shape. A simple approach to monotonicity would just be to use rejection sampling and sample the GP until a weakly decreasing draw is obtained. There are two broad problems with this approach. First, there is no guarantee that a weakly decreasing draw will be found expediently, and second, the probability of obtaining such a draw decreases as the number of test points increases and as the number of observations decreases. Specifically, since monotonicity must be satisfied locally at each point as well as globally, there is no alternative to checking this condition at all the price levels. In cases with many arms where the sample means are highly non-monotonic, the probability of finding a monotonic draw can become vanishingly slim. To ensure that a weakly decreasing draw can be obtained from a GP in an expedient manner in all cases, we develop a method from first principles. We build upon the approach from Maatouk and Bay (2017), where we can estimate the GP as a function of its derivatives. The crucial property that the derivative of a GP is also a GP allows us to transform the sampling problem of monotonicity. Specifically, a decreasing monotonic function can be characterized as a function whose first derivative is negative at all points. That is, the new problem is equivalent to sampling from a truncated multivariate normal distribution, which lends itself to quick estimation procedures such as Gibbs sampling. Another advantage of this principled approach is that the function is guaranteed to be monotonic not just at the discrete price levels forming the support, but also at any intermediate price where no experimentation is performed. The only assumption needed is that the demand function is differentiable with a continuous derivative.

Basis Functions: The first step is to estimate the demand function using a collection of functions h_j known as the interpolation basis. The demand function will be estimated by linearly interpolating between the function values at knots spaced over the support. Following the notation of Maatouk and Bay (2017), let $u_j \in [0,1], j = 0,1,...,N$ denote

equally spaced knots on $[0,1]$ with spacing $\delta_N = 1/N$ and $u_j = j/N$. The interpolation basis is defined as

$$h_j(p) = h\left(\frac{p-u_j}{\delta_N}\right) \text{ where} \tag{9}$$

$$h(p) = (1-|p|)\mathbb{1}(p \in [-1,1]) \tag{10}$$

Then for any continuous function $D:[0,1]\to\mathbb{R}$ the function

$$D_N(\cdot) \approx \sum_{j=0}^{N} D(u_j)h_j(\cdot) \tag{11}$$

approximates D by linearly interpolating between the function values at the knots u_j (see Appendix EC.5.1 for a visualization of the basis functions).

One key property of the interpolation basis is that as the gap between the evenly spaced knots becomes infinitesimally small, the distance between the estimation and the true function converges to 0. This is unsurprising because as the gap between knots becomes infinitesimally small, the gaps where the continuous function are unknown become infinitesimally small. A proof is provided in Appendix EC.5.2.

The purpose of using this estimation technique is to create a principled way of sampling a weakly decreasing demand curve. The key idea is to write the demand function in terms of its derivatives. Then obtaining a decreasing function is equivalent to obtaining a sample of test points where each derivative is negative. Thus, the problem has been reduced to sampling from a truncated multivariate normal distribution, which lends itself to efficient sampling methods such as Gibbs sampling. Explicitly, the demand function can be written in terms of its intercept, derivatives and basis functions as shown in Proposition 1. A proof is provided in Appendix EC.5.3.

PROPOSITION 1. *Assuming the demand function $D:[0,1]\to\mathbb{R}$ is differentiable with a continuous derivative (i.e. $D\in C^1([0,1])$), then it can be estimated by its intercept and derivatives by the following equation:*

$$D(p) \approx D(0) + \sum_{j=0}^{N} D'(u_j)\int_0^p h_j(x)dx \tag{12}$$

While this works for all class C^1 functions on the support, we additionally assume that this unknown demand function D is weakly decreasing meaning that is belongs to a subset B defined as follows:

$$B \coloneqq \{D \in C^1([0,1]) : D'(p) \leq 0, p \in (0,1)\} \tag{13}$$

That is, D belongs to the subset of functions where the derivative is never positive at any value of p.

6.2. Monotonic GPTS

As the integral of the basis functions h can be calculated a priori, equation (12) gives a formula to estimate a demand function D from its intercept as well as the derivatives at the knots (in our case the test points). We leverage the crucial property that the joint distribution of values and their derivatives are also a GP, implying that both the intercept and the derivatives can be estimated using a single GP (see Appendix EC.5.4 for details). Once the GP has been estimated, Gibbs sampling can be used to acquire a draw where every derivative is non-negative. By substituting this draw into equation (12), a monotonically decreasing draw for D is obtained.

Algorithm 2: GPTS-Mono

```
Choose a set of prices, test prices, kernel, knots and set necessary priors
Calculate the integrals of the basis functions at the knots
for t = 1, 2, ... do
    Estimate the posterior GP of derivatives and the intercept using experiment history
    Obtain a draw of negative derivatives and the intercept using Gibbs sampling
    Estimate demand draw, D*, using equation (12)
    Choose arm k with price p_{k_t} := arg max_{p_k} p_k D*(p_k)
    Observe purchase decision and update history
    Perform Bayesian update
end
```

When the demand sample is estimated using a draw of negative derivatives from the Gaussian process, we affix "mono" to the algorithm name such as *GPTS-Homo-Mono* or *GPTS-Hetero-Mono* depending on how the GP is fitted to the data. The complete algorithm is detailed in Algorithm 2. An overview of all algorithms discussed can be found in Table 3.

Table 3 Overview of All Algorithms

Algorithm	Bayesian	Heteroscedastic Noise	Implements Monotonicity	Dependence Across Arms	Partial Identification
UCB					
UCB-Tuned		✓			
TS	✓	✓			
UCB-PI-Tuned		✓	✓		✓
GPTS-Homo	✓			✓	
GPTS-Homo-Mono	✓		✓	✓	
GPTS-Hetero-Mono	✓	✓	✓	✓	
GPTS-Hetero-Mono-PI	✓	✓	✓	✓	✓

7. Empirical Performance: Simulations

We evaluate our proposed algorithm and benchmarks using a series of simulations using the setup in Misra et al. (2019). Each simulation has the following structure. First, at each time period, a customer with WTP drawn from an unknown distribution arrives from a large pool of potential customers with unit demand for the product. Second, the customer observes the price set by the algorithm, and will only purchase if their valuation for the product is greater than that price. The outcome (purchase or not) information is observed by the algorithm, which then updates its price. While it is possible to change prices every period (i.e. for each customer), in practice this is too difficult, so to better represent industry procedures we change prices every 10 customers (equivalent to 500 price changes over 5000 customers). A total of $K = 100$ normalized prices ranging from 0.01 to 1 in 1 cent increments are tested.

While our algorithm does not require knowledge of customer segments or a specific customer's segment membership, UCB-PI requires segmentation information. To allow for comparison between GPTS-Mono and UCB-PI, we use the setup from Misra et al. (2019) where each customer belongs to one of $S = 1000$ segments; the size of the segments, ψ_s are drawn from a simplex on the uniform distribution. Additionally, each segment's midpoint valuation, v_s, is drawn from the true underlying distribution and every customer within this segment has a valuation within $\delta = 0.1$ of the midpoint. We further specify the distribution of valuations within the segment to be $[v_s - \delta, v_s + \delta]$. We use a scaled Beta(2,2) to specify the how valuations are distributed within a segment.[14]

The last requirement to run these algorithms is a method of initialization. While TS and GPTS can run immediately using distribution priors, UCB variants require that each

[14] This has the nice property of having the entire support within the segment range while having more weight on values closer to the midpoint.

arm has been tested before the algorithm can compute bounds. This can be done by either testing every price once (or 10 times in our case) or by choosing a prior that encourages exploration (we can assume that every untested price has been tested once and the trial was a success). A full discussion of the initialization methods used can be found in Appendix EC.6.1).

To reasonably show the effectiveness of our algorithm, we want to show that performance is empirically robust across a wide spectrum of unknown true distributions of consumer valuations. As in Misra et al. (2019), we consider the following distributions.

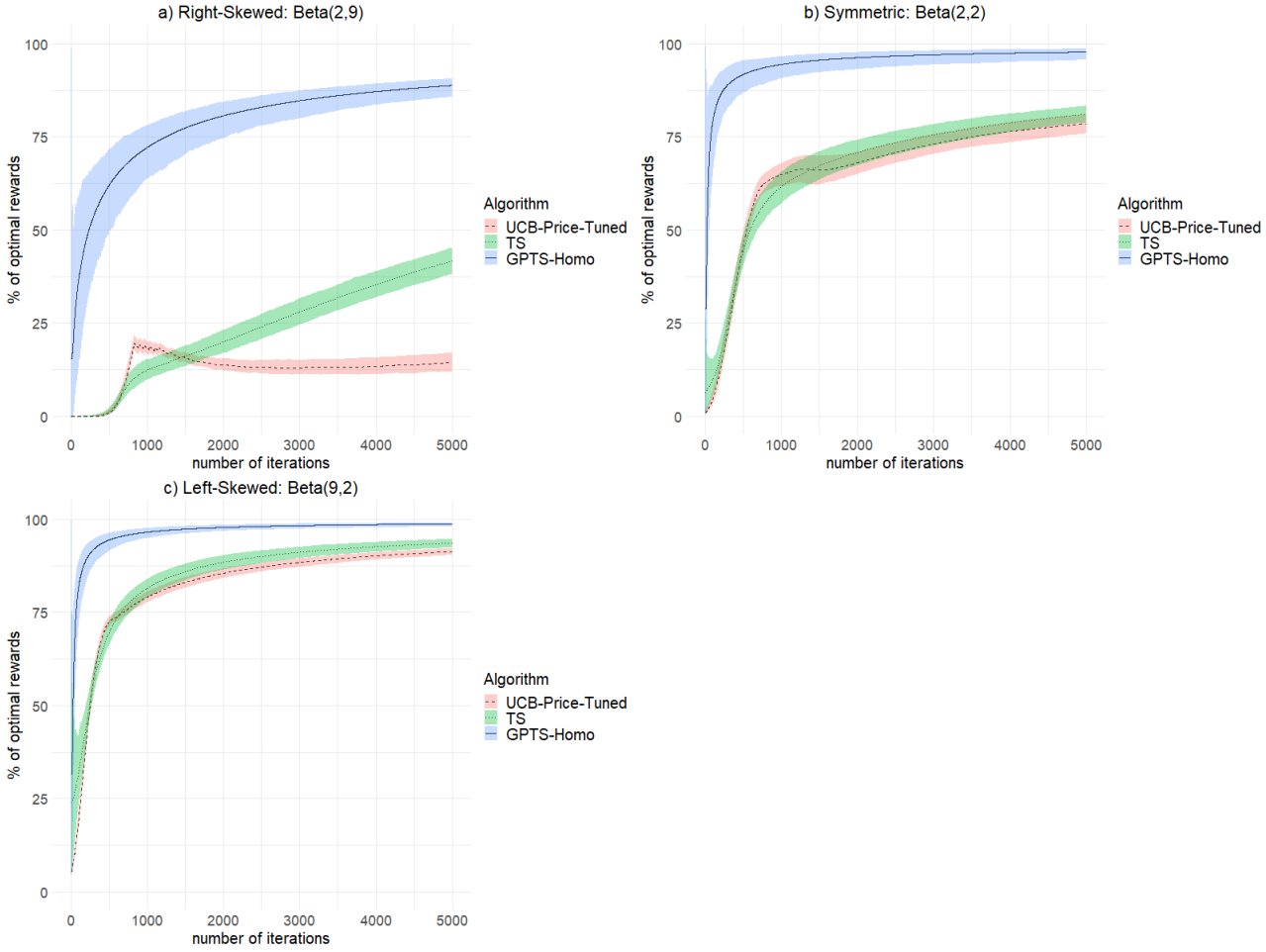
1. Right-skewed Beta distribution – Beta(2,9)
2. Smmetric Beta distribution – Beta(2,2)
3. Left-skewed Beta distribution – Beta(9,2)

Overall, the distributions considered are unimodal distributions with a right, left and symmetric skew. A full graphical description of the willingness to pay, the demand curve and the profit curve for each simulation setting can be found in Appendix EC.6.2.

Baseline Algorithms. The first set of results compares the baseline versions of each of the algorithms: UCB-Price-Tuned, TS, and GPTS-Homo. This comparison will provide evidence of the importance of the first information externality; i.e. how the dependency across arms is important for pricing bandits. Each algorithm was run $n_{sim} = 1000$ separate times for $T = 5000$ iterations. The empirical performance is shown in Figure 2. As there is a large variation between reward outcomes in individual trials induced by randomness from the WTP draws, expected rewards calculated from the prices played are used instead of the actual rewards. The shaded areas show the 95% range of the expected rewards across the n_{sim} simulations.

In each of our simulation settings, GPTS-Homo outperformed TS and UCB-Price-Tuned handily. This provides evidence for the first information externality, and shows that allowing for dependency across arms is important in pricing bandits. GPTS-Homo obtained between 88.9% and 98.8% of optimal rewards depending on the true distribution of WTP, while UCB-Price-Tuned obtained between 14.4% and 91.4%. The difference in results are even more pronounced after 1000 iterations where GPTS-Homo has already honed in on the set of most profitable prices while TS and UCB-Price-Tuned have barely gained enough information to stop using the priors at each price. In settings where there is some implicit cost of experimentation, this difference is likely to be highly valuable.

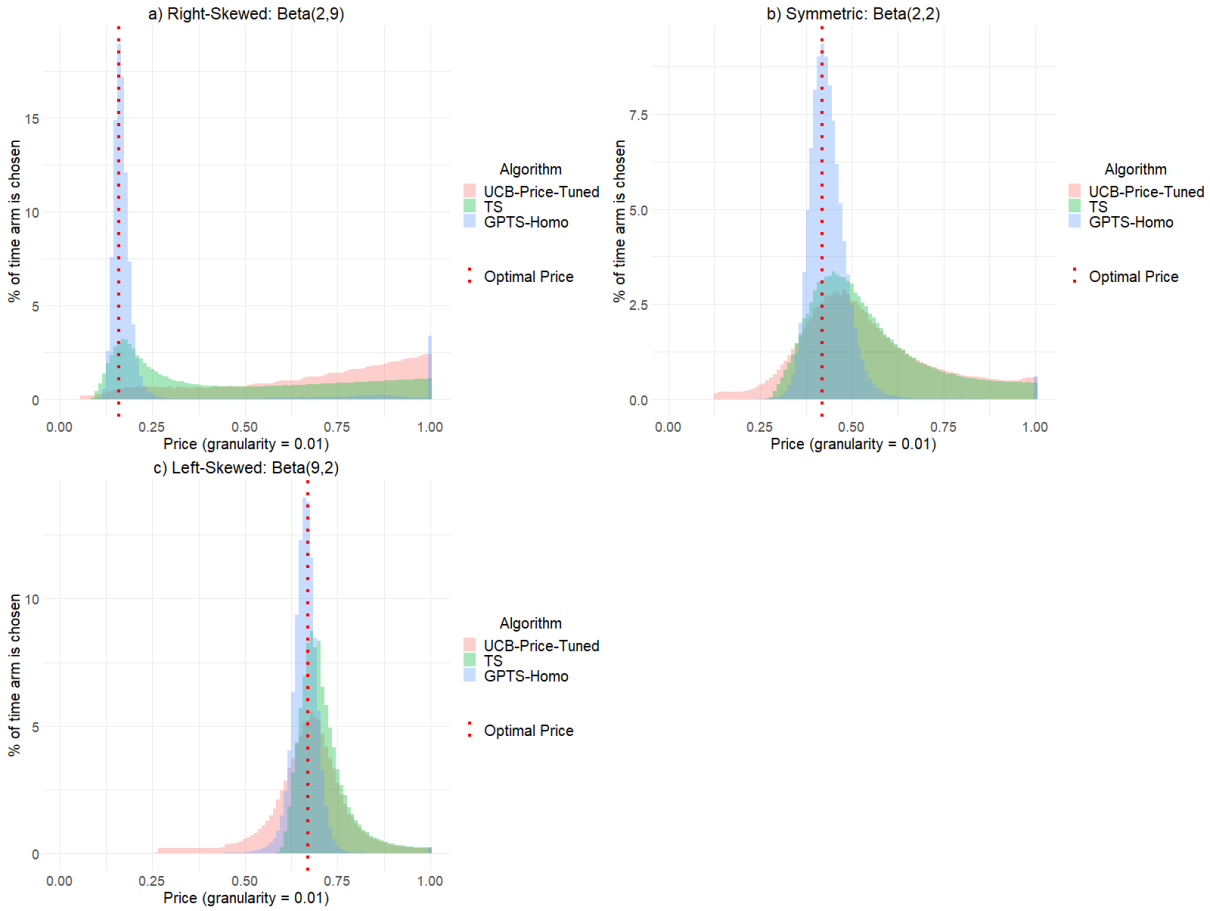
Figure 2 Percent of Optimal Rewards (Mean and 95% Range)



Notes. The lines are the means of the expected percentage of optimal rewards while the shaded regions are the 95% range of the expected rewards across the 1000 simulations. The results include any burn-in periods.

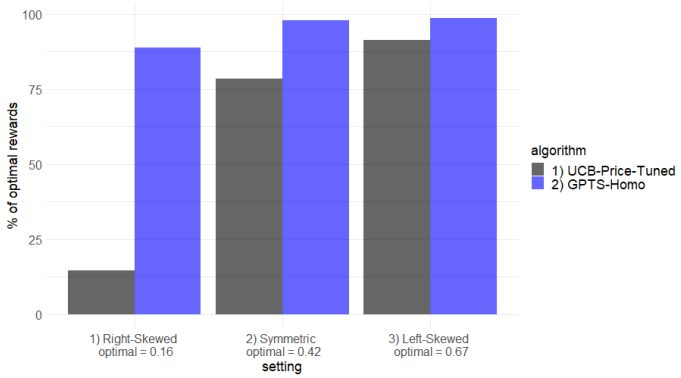
The large discrepancies between the algorithms success depend on where the optimal price is located. All the algorithms tend to over-explore higher prices because there are larger possible reward bounds at higher prices. For example, a 10% error range at $p = 0.6$ is larger than the entire range of possible rewards at $p = 0.05$. For this reason, TS and UCB-Price-Tuned perform very poorly when the optimal price is low (compared to the support of prices) because they expend considerable time ruling out potentially profitable high prices. This effect is particularly noticeable in Figure 2a where UCB-Price-Tuned performs worse than the burn-in, even after 5000 simulations, because it spends the entire time exploring the poor performing high prices. This effect was mitigated for GPTS-Homo because it models the entire demand curve at once instead of treating arms independently. As a result, it can more quickly dismiss a range of low performing high prices as it does not have to assess each one individually. The end result is the expected percent of optimal

Figure 3 **Histogram of Prices Played**



Notes. The histogram is for 5000 iterations. Results are averaged over 1000 simulations.

Figure 4 **Percent of Optimal Rewards**



rewards is 88.9% for GPTS-Homo while just 14.4% for UCB-Price-Tuned. Overall, the simulations illustrate a pattern that the performance gap between GPTS-Homo and UCB-Price-Tuned closes as the true unknown optimal price increases (Figure 4).

GPTS with Monotonicity. While GPTS-Homo provides empirical evidence for the first information externality, the focal point of our paper is to explore the second information externality: monotonicity. To assess the impact of incorporating monotonicity, GPTS-Homo and GPTS-Homo-Mono are compared.

Figure 5 shows the distribution of expected rewards after 5000 iterations for 1000 simulations of GPTS-Homo and GPTS-Homo-Mono. While the monotonic version did better in every scenario, once again the biggest gains come from the underlying right-skewed distribution. Including monotonicity led to a 8.0% increase in rewards for the Beta(2,9) distribution while the increases were minimal for the symmetric (0.35%) and left-skewed (0.13%) distributions. Despite the potentially small differences, every scenario is statistically significant with the Wilcoxon signed-rank test (see Table 4). A paired test is used because the n-th simulation for GPTS-Homo and GPTS-Homo-Mono uses the same draw from the underlying WTP distribution, and algorithm performance is highly correlated depending on the draw.

Table 4 Comparison of GPTS-Homo and GPTS-Homo-Mono (Wilcoxon signed-rank test)

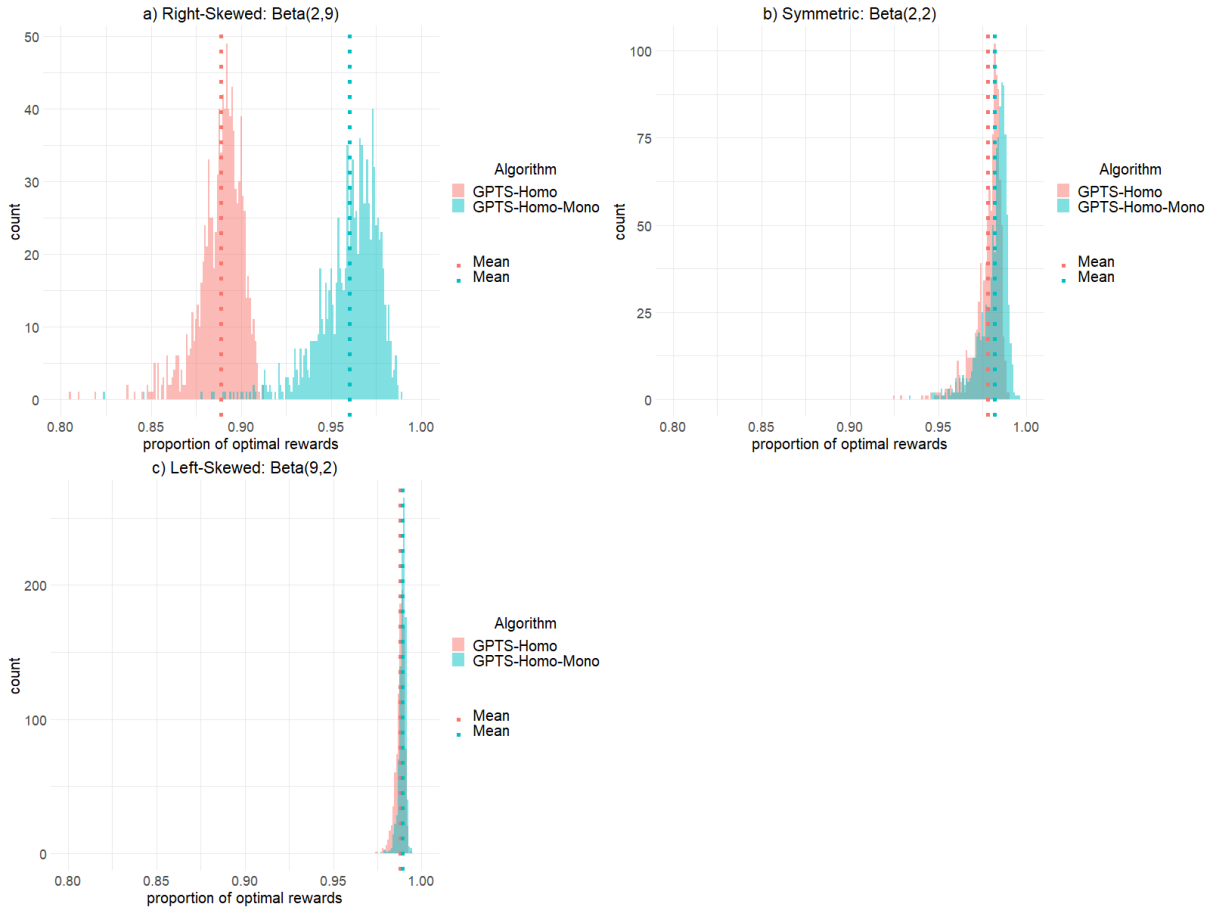
True WTP Distribution	Optimal Price	Algorithm	Expected % of Optimal Rewards (mean)	p-value	Avg Reward Increase
Beta(2, 9)	\$0.16	GPTS-Homo GPTS-Homo-Mono	88.86% 96.00%	$< 0.0001^{***}$	8.03%
Beta(2, 2)	\$0.42	GPTS-Homo GPTS-Homo-Mono	97.81% 98.16%	$< 0.0001^{***}$	0.35%
Beta(9, 2)	\$0.67	GPTS-Homo GPTS-Homo-Mono	98.78% 98.92%	$< 0.0001^{***}$	0.13%

Notes. Each algorithm was run for 1000 simulations for each WTP distribution. The WTP draws are identical across algorithms (that is, the j-th customer in simulation n will have the same valuation regardless of which algorithm was used to choose prices).

Overall, this provides strong evidence for the importance of the second information externality (from microeconmic theory) that including monotonicity in the bandit empirically leads to gains in rewards. The question remains of how and why is there such a discrepancy in performance boost dependent on the underlying distribution.

The mechanism by which enforcing monotonicity increases profits is better illuminated in Figures 6 and 7. Figure 6 shows the difference in cumulative profits over time, which reveals that the difference is most pronounced in early rounds and shrinks as the number of iterations increases. This suggests that the advantage of including monotonicity happens during early iterations and that the algorithms perform similarly in later iterations.

Figure 5 Histogram of Rewards across Simulations

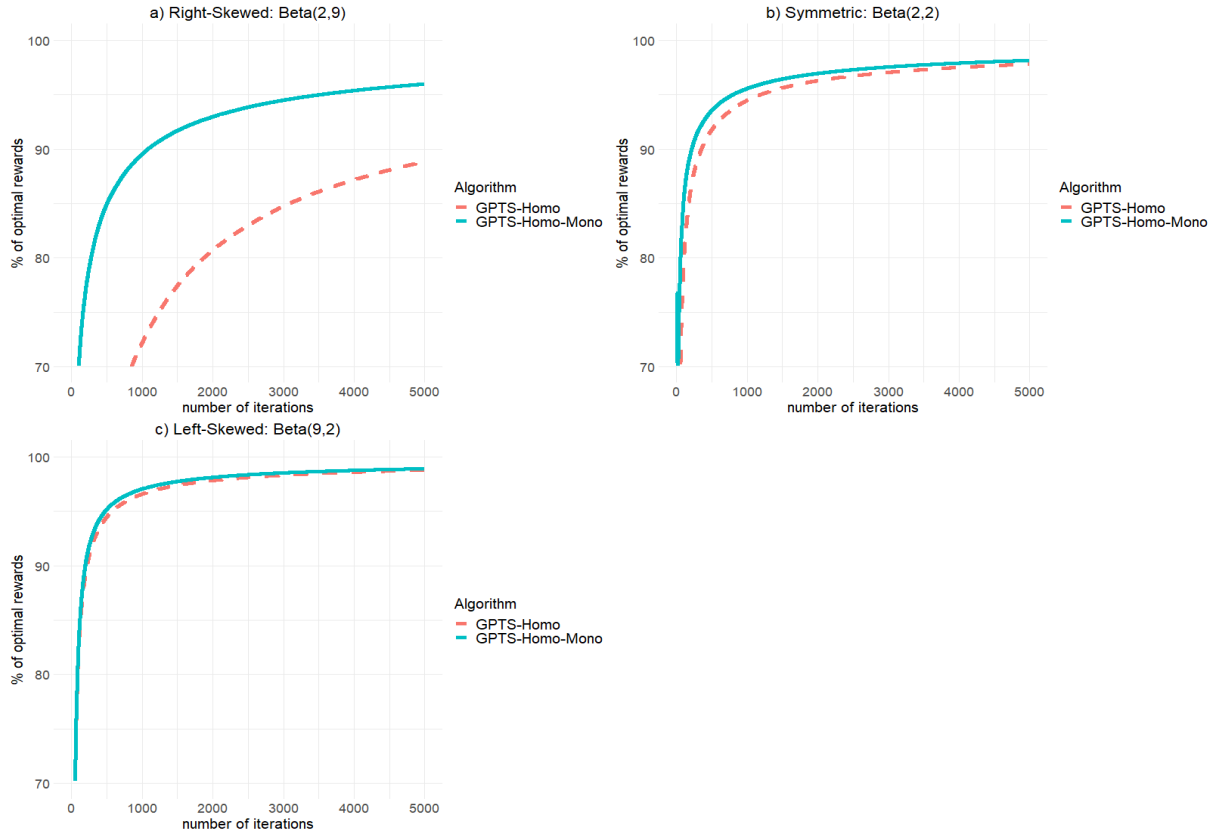


Notes. These are the expected proportion of optimal rewards after 5000 iterations. The count is over 1000 simulations. The dotted line is the mean across 1000 simulations.

Consider the improvement in rewards with limited experimentation (at 1,000 iterations) for the right-skewed distribution; it increases from 72% (without monotonicity) to 90%, which represents a 25% improvement. Figure 7 shows the histogram of prices played, which shows that in each setting the optimal price was chosen most often and that this peak was higher for GPTS-Homo-Mono. Importantly it shows a dramatic reduction in the number of times "bad" prices around \$1.00 were chosen, resulting in an increase in rewards.

To summarize, the largest problem of not considering monotonicity is because in early iterations when prices have not been tested, the data can often be quite noisy. This can lead to demand draws that end on an upwards slope on the support of prices leading to \$1.00 being chosen disproportionately (Figure 7). As more data is gained, the problem becomes less frequent resulting in GPTS-Homo and GPTS-Homo-Mono performing similarly in later rounds. An upwards portion of the demand curve violates monotonicity meaning such a draw ending on an upwards curve would not be possible in GPTS-Homo-Mono, which

Figure 6 Cumulative Percent of Optimal Rewards



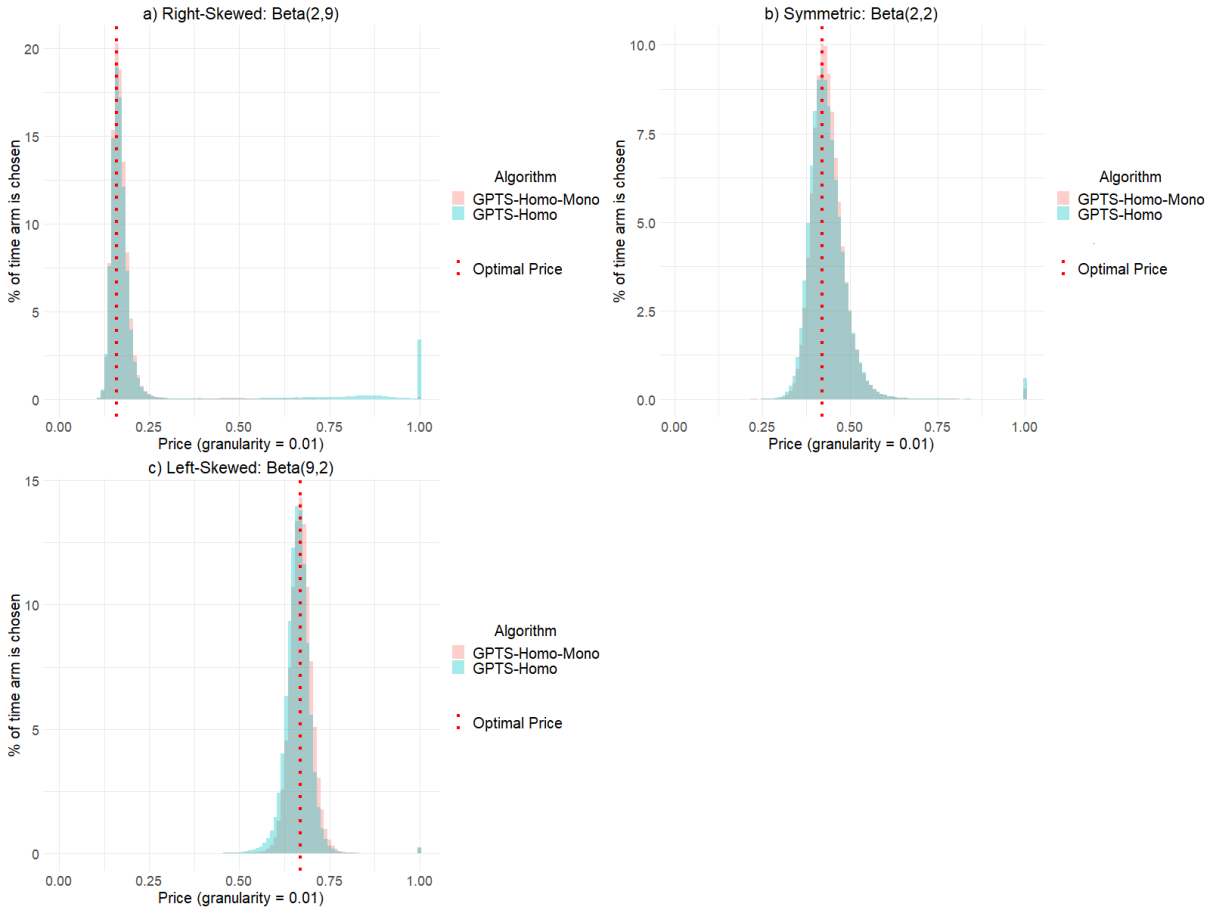
Notes. This represents the average (over 1000 simulations) expected percentage of optimal rewards cumulatively from 1 to 5000 iterations.

leads to a substantial decrease in the number of times \$1.00 is chosen allowing a better price to be tested.

This also explains why the benefit is greatest for the right-skewed case and minimal for the left-skewed case. The reward is the demand scaled by price, so if the optimal is a very low price then the bounds at high low-reward prices need to be very small before they stop being explored; this process requires a lot of data. However, including monotonicity can bypass the need for these bad draws because it considers that the vast majority of these draws violate the monotonicity of demand curves. Meanwhile, in the left-skewed case the optimal price is a lot higher meaning the amount of exploration needed to stop exploring high low-reward prices is much lower; the number of bad draws that can be averted by considering monotonicity is thus limited.

GPTS with Heteroscedasticity. While the majority of the improvement comes from incorporating the information externalities, further fine-tuning is possible. The algorithms presented thus far consider noise to be homoscedastic which is not true for pricing bandits.

Figure 7 Histogram of Prices Played

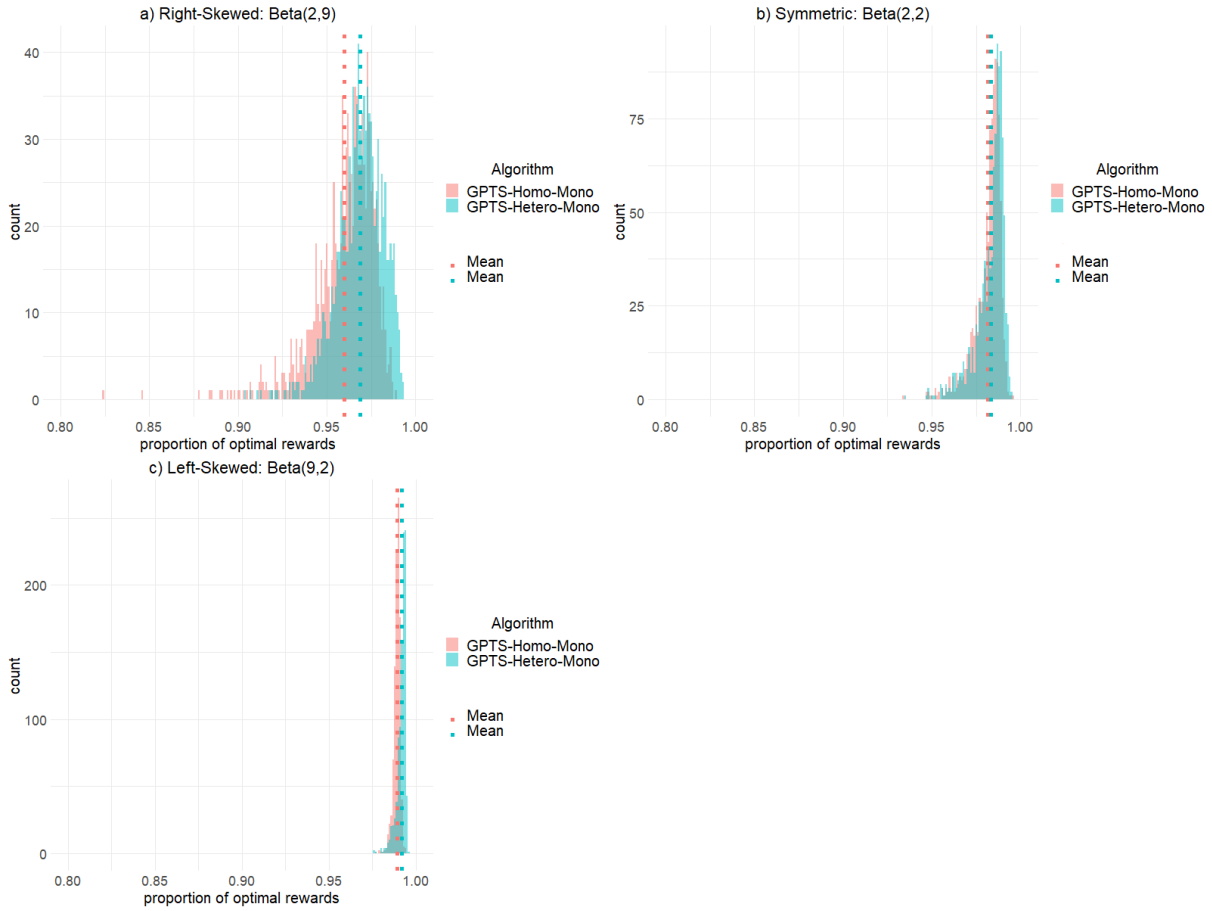


Notes. The histogram is for 5000 iterations. Results are averaged over 1000 simulations.

Practically, this leads to oversampling for prices where there is little response variation; most notably this happens at \$1.00 so allowing for heteroscedascity would allow for less experimentation. GPTS-Hetero-Mono allows for the noise to vary between arm and provides a small boost to rewards - an additional 0.15% to 0.89% depending on the underlying WTP distribution. While the gains from heteroscedasticity are not large, the difference in every setting was significant using the Wilcoxon signed-rank test.

Hybrid: Combining GPTS with Partial Identification. We next consider a hybrid approach, combining our algorithm with the partial identification method from Misra et al. (2019). This hybrid algorithm leads to a significant increase in rewards using the Wilcoxon signed-rank test, though the gains are quite small; rewards increased by just 0.01% to 0.16% depending on the setting. Figure 9 shows that the histogram of simulation results is near identical for GPTS-Hetero-Mono with and without partial identification. It also

Figure 8 Histogram of Rewards across Simulations



Notes. These are the expected proportion of optimal rewards after 5000 iterations. The count is over 1000 simulations. The dotted line is the mean across 1000 simulations.

Table 5 Comparison of GPTS-Homo-Mono and GPTS-Hetero-Mono (Wilcoxon signed-rank test)

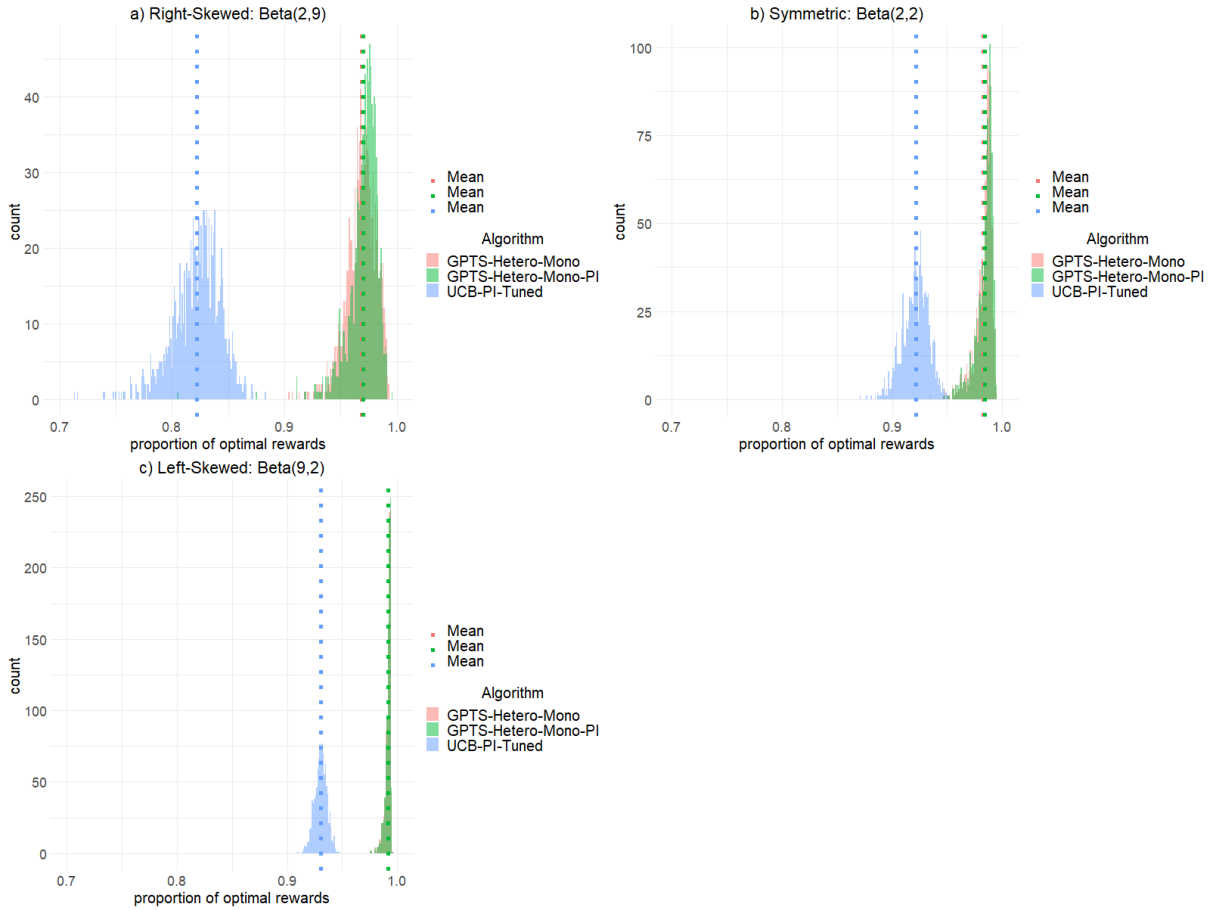
True WTP Distribution	Optimal Price	Algorithm	Expected % of Optimal Rewards (mean)	p-value	Avg Reward Increase
Beta(2, 9)	\$0.16	GPTS-Homo-Mono	96.00%	$< 0.0001^{***}$	0.89%
		GPTS-Hetero-Mono	96.85%		
Beta(2, 2)	\$0.42	GPTS-Homo-Mono	98.16%	$< 0.0001^{***}$	0.15%
		GPTS-Hetero-Mono	98.31%		
Beta(9, 2)	\$0.67	GPTS-Homo-Mono	98.91%	$< 0.0001^{***}$	0.27%
		GPTS-Hetero-Mono	99.18%		

Notes. Each algorithm was run for 1000 simulations for each WTP distribution. The WTP draws are identical across algorithms (that is, the j-th customer in simulation n will have the same valuation regardless of which algorithm was used to choose prices).

shows how GPTS-Hetero-Mono with partial identification greatly outperforms UCB with partial identification.

We might expect such results for a couple of reasons. First, the Wilcoxon signed-rank test manages to find significance for such small differences because partial identification

Figure 9 Histogram of Rewards across Simulations



Notes. These are the expected proportion of optimal rewards after 5000 iterations. The count is over 1000 simulations. The dotted line is the mean across 1000 simulations.

Table 6 Comparison of GPTS-Hetero-Mono with and without Partial Identification (Wilcoxon signed-rank test)

True WTP Distribution	Optimal Price	Algorithm	Expected % of Optimal Rewards (mean)	p-value	Avg Reward Increase
Beta(2, 9)	\$0.16	Without PI With PI	96.85% 97.00%	$< 0.0001^{***}$	0.16%
Beta(2, 2)	\$0.42	Without PI With PI	98.31% 98.41%	$< 0.0001^{***}$	0.11%
Beta(9, 2)	\$0.67	Without PI With PI	99.18% 99.19%	$< 0.0001^{***}$	0.01%

Notes. Each algorithm was run for 1000 simulations for each WTP distribution. The WTP draws are identical across algorithms (that is, the j-th customer in simulation n will have the same valuation regardless of which algorithm was used to choose prices).

can only improve an algorithm; it removes dominated prices from consideration. Thus, the addition of partial identification will improve algorithm performance but the question is by how much. Our results show that there is very little to gain from using segmentation and partial identification when our algorithm is being used. This is because we already

manage to incorporate the monotonicity constraint without use of segmentation. Generally the prices that are removed from consideration have a very low probability of being picked anyway so there is very little difference between the performance of GPTS-Hetero-Mono with and without partial identification.

8. Conclusion and Future Research

We have proposed a method to achieve efficient and robust learning of the demand curve using reinforcement learning informed by microeconomic principles. It is important to have a non-parametric method that is able to learn arbitrary demand curves, which the method proposed in this paper can accommodate. Theory is usefully incorporated in multiple ways in our model: as a dependency across the rewards of arms (where closer arms are more likely to have greater dependence), and by enforcing a monotonicity restriction on the realized demand function. This allows for uncertainty to be reduced for all arms from a single purchase decision, and rules out the possibility of noisy data leading to demand curves draws that violate economic theory. Our approach is based on Gaussian processes, which provide a flexible nonparametric basis for learning of the reward distributions. In practice, the method does not need any prior information on consumers (e.g. segment membership) and operates in real time across a wide variety of demand distributions.

Our method is especially useful for managers who have limited time for experimentation. In every simulation setting, our algorithm achieved 90% of optimal profits after just 1000 iterations, which is something no benchmark managed within 5000 iterations. This reduction in needed experimentation time should make price experimentation more palatable to managers; less experiments means potential for monetary loss is minimized and fewer customers are impacted. Additionally, our method potentially allows for a greater set of prices to be tested as we can obtain demand draws at prices not tested in the experiment. As a finer grid of prices can get closer to the true optimal, this could not only increase gains during the experiment but have a long-lasting profit impact if upon termination of the experiment a particular price is chosen for the long-run.

There are several avenues to further extend the applicability of this research. First, the method could be adapted to situations with either multiple units purchased or even multiple products, wherein consumer choice of one product could also have an informational externality in inferring the distribution of valuations for other related products.

Second, the underlying demand is time-varying, which can create new challenges in specifying monotonicity. Third, if experimentation of price levels and demand responses across competing firms is available, this would be a naturally important setting. More broadly, the efficient learning of unknown demand curves with minimal impact of experimentation remains a widely shared goal across a number of markets, and we believe this present research contributes by closely connecting theory to develop such algorithms.

Competing Interests

All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. The authors have no funding to report.

References

Aghion P, Bolton P, Harris C, Jullien B (1991) Optimal learning by experimentation. *The review of economic studies* 58(4):621–654.

Agrawal R (1995) Sample mean based index policies by o (log n) regret for the multi-armed bandit problem. *Advances in Applied Probability* 27(4):1054–1078.

Agrawal S, Goyal N (2012) Analysis of thompson sampling for the multi-armed bandit problem. *Conference on learning theory*, 39–1.

Ariely D (2010) Why businesses don't experiment. *Harvard business review* 88(4).

Auer P (2002) Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3(Nov):397–422.

Auer P, Cesa-Bianchi N, Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47(2-3):235–256.

Aviv Y, Vulcano G (2012) Dynamic list pricing. *The Oxford handbook of pricing management*.

Bayati M, Hamidi N, Johari R, Khosravi K (2020) Unreasonable effectiveness of greedy algorithms in multi-armed bandit with many arms. *Advances in Neural Information Processing Systems* 33:1713–1723.

Bergemann D, Schlag KH (2008) Pricing without priors. *Journal of the European Economic Association* 6(2-3):560–569.

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

Bizer GY, Schindler RM (2005) Direct evidence of ending-digit drop-off in price information processing. *Psychology & Marketing* 22(10):771–783.

Bubeck S, Munos R, Stoltz G, Szepesvári C (2011) X-armed bandits. *Journal of Machine Learning Research* 12(May):1655–1695.

Chapelle O, Li L (2011) An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, 2249–2257.

Cheung WC, Simchi-Levi D, Wang H (2017) Dynamic pricing and demand learning with limited price experimentation. *Operations Research* 65(6):1722–1731.

Ching AT, Osborne M (2020) Identification and estimation of forward-looking behavior: The case of consumer stockpiling. *Marketing Science* 39(4):707–726.

Cohen S, Treetanthiploet T (2021) Correlated bandits for dynamic pricing via the arc algorithm. *arXiv preprint arXiv:2102.04263* .

Cohen SN, Treetanthiploet T (2020) Asymptotic randomised control with applications to bandits. *arXiv preprint arXiv:2010.07252* .

den Boer AV (2015) Dynamic pricing and learning: historical origins, current research, and new directions. *Surveys in operations research and management science* 20(1):1–18.

Dube JP, Misra S (2022) Personalized pricing and consumer welfare. *Journal of Political Economy* URL `http://dx.doi.org/10.1086/720793`.

Erdem T, Keane MP (1996) Decision-making under uncertainty: Capturing dynamic brand choice processes in turbulent consumer goods markets. *Marketing science* 15(1):1–20.

Ferreira KJ, Simchi-Levi D, Wang H (2018) Online network revenue management using thompson sampling. *Operations research* 66(6):1586–1602.

Furman J, Simcoe T (2015) The economics of big data and differential pricing. *The White House President Barack Obama* .

Gittins J (1974) A dynamic allocation index for the sequential design of experiments. *Progress in statistics* 241–266.

Goldberg PW, Williams CK, Bishop CM (1997) Regression with input-dependent noise: A gaussian process treatment. *Advances in neural information processing systems* 10:493–499.

Gordon BR, Zettelmeyer F, Bhargava N, Chapsky D (2019) A comparison of approaches to advertising measurement: Evidence from big field experiments at facebook. *Marketing Science* 38(2):193–225.

Handel BR, Misra K (2015) Robust new product pricing. *Marketing Science* 34(6):864–881.

Hansen KT, Misra K, Pai MM (2021) Frontiers: Algorithmic collusion: Supra-competitive prices via independent algorithms. *Marketing Science* 40(1):1–12.

Hanssens DM, Pauwels KH (2016) Demonstrating the value of marketing. *Journal of marketing* 80(6):173–190.

Hauser JR, Urban GL, Liberali G, Braun M (2009) Website morphing. *Marketing Science* 28(2):202–223.

Hendel I, Nevo A (2006) Measuring the implications of sales and consumer inventory behavior. *Econometrica* 74(6):1637–1673.

Higham NJ (2002) Computing the nearest correlation matrix—a problem from finance. *IMA journal of Numerical Analysis* 22(3):329–343.

Hill DN, Nassif H, Liu Y, Iyer A, Vishwanathan S (2017) An efficient bandit algorithm for realtime multivariate optimization. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1813–1821.

Hoban PR, Bucklin RE (2015) Effects of internet display advertising in the purchase funnel: Model-based insights from a randomized field experiment. *Journal of Marketing Research* 52(3):375–393.

Hoffman M, Brochu E, De Freitas N, et al. (2011) Portfolio allocation for bayesian optimization. *UAI*, 327–336 (Citeseer).

Huang J, Reiley D, Riabov N (2018) Measuring consumer sensitivity to audio advertising: A field experiment on pandora internet radio. *Available at SSRN 3166676* .

Huang Y, Ellickson PB, Lovett MJ (2022) Learning to set prices. *Journal of Marketing Research* 59(2):411–434.

Jindal P, Zhu T, Chintagunta P, Dhar S (2020) Marketing-mix response across retail formats: the role of shopping trip types. *Journal of Marketing* 84(2):114–132.

Kawale J, Bui HH, Kveton B, Tran-Thanh L, Chawla S (2015) Efficient thompson sampling for online matrix-factorization recommendation. *Advances in neural information processing systems*, 1297–1305.

Kersting K, Plagemann C, Pfaff P, Burgard W (2007) Most likely heteroscedastic gaussian process regression. *Proceedings of the 24th international conference on Machine learning*, 393–400.

Kleinberg R, Leighton T (2003) The value of knowing a demand curve: Bounds on regret for online posted-price auctions. *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, 594–605 (IEEE).

Kleinberg R, Slivkins A, Upfal E (2008) Multi-armed bandits in metric spaces. *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 681–690.

Lai TL, Robbins H (1985) Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics* 6(1):4–22.

Lambrecht A, Tucker C, Wiertz C (2018) Advertising to early trend propagators: Evidence from twitter. *Marketing Science* 37(2):177–199.

Liu X (2022) Dynamic coupon targeting using batch deep reinforcement learning: An application to livestream shopping. *Marketing Science* forthcoming.

Maatouk H, Bay X (2017) Gaussian process emulators for computer experiments with inequality constraints. *Mathematical Geosciences* 49(5):557–582.

Misra K, Schwartz EM, Abernethy J (2019) Dynamic online pricing with incomplete information using multiarmed bandit experiments. *Marketing Science* 38(2):226–252.

Murray I (2008) Introduction to gaussian processes. *Dept. Computer Science, University of Toronto* .

Nair H (2007) Intertemporal price discrimination with forward-looking consumers: Application to the us market for console video-games. *Quantitative Marketing and Economics* 5(3):239–292.

Oren SS, Smith SA, Wilson RB (1982) Nonlinear pricing in markets with interdependent demand. *Marketing Science* 1(3):287–313.

Osband I, Blundell C, Pritzel A, Van Roy B (2016) Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 4026–4034.

Pfeffer J, Sutton RI (2006) Evidence-based management. *Harvard business review* 84(1):62.

Quah JKH (2000) The monotonicity of individual and market demand. *Econometrica* 68(4):911–930.

Rao RC, Bass FM (1985) Competition, strategy, and price dynamics: A theoretical and empirical investigation. *Journal of Marketing Research* 22(3):283–296.

Rebonato R, Jäckel P (2011) The most general methodology to create a valid correlation matrix for risk management and option pricing purposes. *Available at SSRN 1969689* .

Ringbeck D, Huchzermeier A (2019) Dynamic pricing and learning: An application of gaussian process regression. *Available at SSRN 3406293* .

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

Rubel O (2013) Stochastic competitive entries and dynamic pricing. *European Journal of Operational Research* 231(2):381–392.

Russo D, Van Roy B (2014) Learning to optimize via posterior sampling. *Mathematics of Operations Research* 39(4):1221–1243.

Russo D, Van Roy B, Kazerouni A, Osband I, Wen Z (2017) A tutorial on thompson sampling. *arXiv preprint arXiv:1707.02038* .

Sahni NS, Nair HS (2020) Does advertising serve as a signal? evidence from a field experiment in mobile search. *The Review of Economic Studies* 87(3):1529–1564.

Schwartz EM, Bradlow ET, Fader PS (2017) Customer acquisition via display advertising using multi-armed bandit experiments. *Marketing Science* 36(4):500–522.

Shahriari B, Swersky K, Wang Z, Adams RP, De Freitas N (2015) Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE* 104(1):148–175.

Simester D, Hu Y, Brynjolfsson E, Anderson ET (2009) Dynamics of retail advertising: Evidence from a field experiment. *Economic Inquiry* 47(3):482–499.

Slivkins A (2019) Introduction to multi-armed bandits. *arXiv preprint arXiv:1904.07272* .

Srinivas N, Krause A, Kakade SM, Seeger M (2009) Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995* .

Stiving M, Winer RS (1997) An empirical analysis of price endings with scanner data. *Journal of Consumer Research* 24(1):57–67.

Strulov-Shlain A (2019) More than a penny's worth: Left-digit bias and firm pricing. *Available at SSRN 3413019* .

Sutton RS, Barto AG (2018) *Reinforcement learning: An introduction* (MIT press).

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

Tirole J (1988) *The theory of industrial organization* (MIT press).

Wang Z, Hu M (2014) Committed versus contingent pricing under competition. *Production and Operations Management* 23(11):1919–1936.

Williams CK, Rasmussen CE (2006) *Gaussian processes for machine learning*, volume 2 (MIT press Cambridge, MA).

Yu M, Debo L, Kapuscinski R (2016) Strategic waiting for consumer-generated quality information: Dynamic pricing of new experience goods. *Management Science* 62(2):410–435.

Zhang JZ, Netzer O, Ansari A (2014) Dynamic targeted pricing in b2b relationships. *Marketing Science* 33(3):317–337.

Zhang L, Chung DJ (2020) Price bargaining and competition in online platforms: An empirical analysis of the daily deal market. *Marketing Science* 39(4):687–706.

Electronic Companion Supplement

EC.1. A Simple Example Illustrating Importance of Considering Monotonicity

What is the issue with algorithms that do not incorporate monotonicity into their decision process of choosing a price? As a simple example, let us assume that customers (unknown to the algorithm) have valuations uniformly distributed between 0 and 1, which means the underlying demand is $D(p) = (1-p)$ and the optimal price is $p = 0.5$. Customers only buy either one or zero units of a product. Let the set of prices be $P = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, with the simulation lasting for 500 time-steps. Every iteration the customer is shown a price picked by the algorithm, they make a purchase decision, and then this information is used to update the algorithm for the next iteration. The algorithm being tested is UCB-Price-Tuned (see EC.6) which is UCB algorithm adapted to price that on average performs quite well.

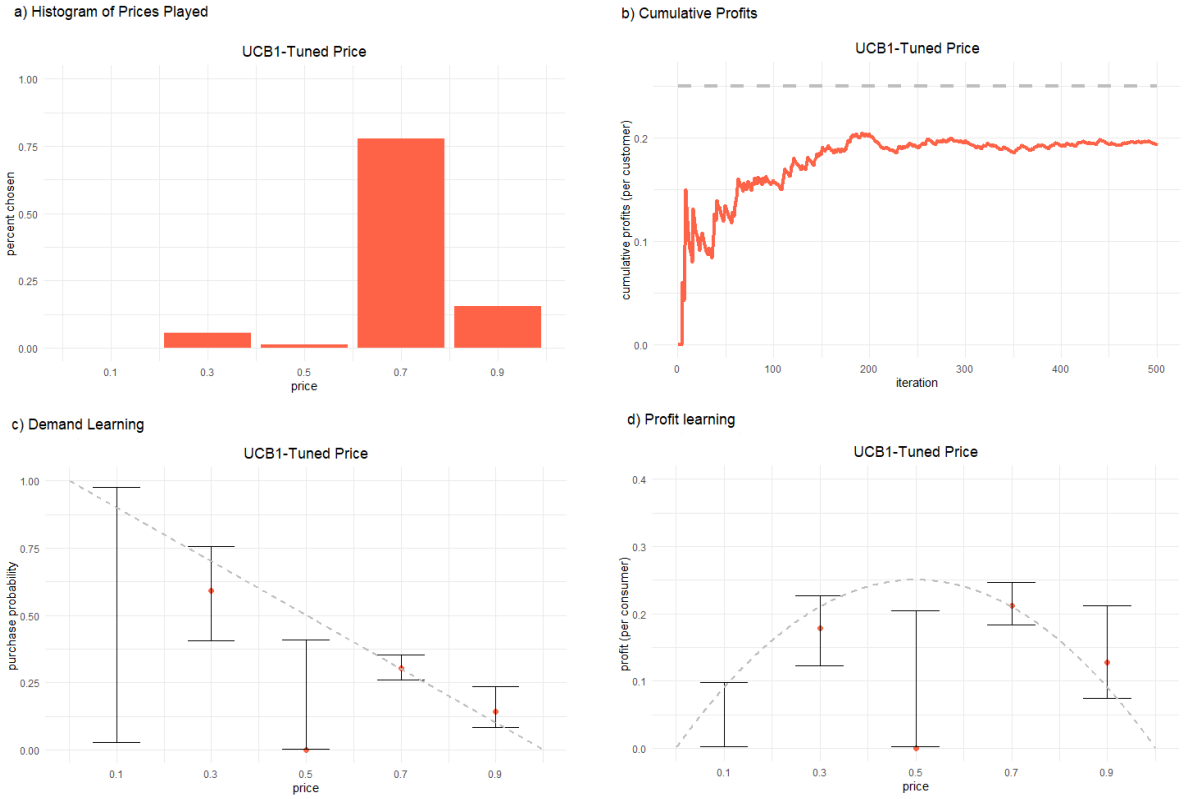
While UCB-Price-Tuned generally performs well, there is large variance among outcomes between individual simulations. Figure EC.1 shows one particular simulation where UCB-Price-Tuned performs poorly. While the algorithm will eventually settle on $p = 0.5$[15] as the number of iterations goes to infinity,[16] the question is whether these early round sub-optimal choices are preventable? Not only does it cause short-term losses, but it may lead a company to incorrectly setting the wrong price if the experiment is ended prematurely.

The main culprit for the bad results is ignoring that the true demand curve should be monotonically decreasing even if the sample demand means are not. In this example, we were unlucky in that no customer purchased at price $p = 0.5$ the first 6 times the price was tried. This led to the bandit "exploiting" other prices with higher sample profits. However, from basic economic intuition we know that the demand at 0.7 and 0.9 can not be higher than at 0.5 and that the algorithm is not choosing 0.5 enough in early rounds. What is not immediately obvious, though, is how to ameliorate this issue. Creating an algorithm that balances knowing that the true underlying demand curve is decreasing with the sample demand means is the main aim of this paper.

[15] In this trial, $p = 0.5$ is tested again in iteration 508 and is chosen nearly exclusively thereafter (1499 of the next 1500 iterations).

[16] UCB is guaranteed to achieve the optimal (Auer 2002)

Figure EC.1 UCB1-Tuned Price (Unlucky Draw)



Notes. The optimal price is $p = 0.5$. a) Shows histogram of prices played in the experiment. b) Shows cumulative profit per customer over the 500 iterations - dotted grey line is the true optimal profit level. c) Shows the mean and 95% credible intervals for purchase probability at each price tested - dotted grey line shows the true purchase probability. d) Shows the mean and 95% credible intervals for profit at each price tested - dotted grey line shows the true expected profit

EC.2. Performance Metrics

Those familiar with the bandit research may be more familiar with the concept of *regret*. Regret is defined as the difference between profits under full knowledge versus the expected profits from the policy in question (Lai and Robbins 1985). Formally, the cumulative regret of policy Ψ through time t is

$$\begin{aligned} \text{Regret}(\Psi, P, t) &= \mathbb{E}\left[\sum_{\tau=1}^{t}(\pi^* - \pi_{k_\tau}|\Psi, P, H_{\tau-1})\right] \\ &= t(\pi^*|P) - \mathbb{E}\left[\sum_{\tau=1}^{t}(\pi_{k_\tau}|\Psi, P, H_{\tau-1})\right] \end{aligned} \tag{EC.1}$$

where P is the set of prices being considered, π^* is the ex post maximum expected profit in a given round, and π_{k_τ} is the profit realized when price p_k is played in time period τ where $k_\tau = \Psi(P, H_{\tau-1})$.

Regret is a popular metric as it lends itself nicely to theoretical proofs. For example, UCB has been proven to be the asymptotically best possible performing policy in terms of achieving the *lowest maximum* regret (Lai and Robbins 1985, Agrawal 1995, Auer 2002). Alternatively known as the *minimax regret*, this criterion combines two desirable properties of bandits; not only is it better for an algorithm to have lower regret, but it is better if there is less variance between trial runs.

Empirically, we think an intuitively more clear objective is to maximize the expected total reward (Gittins 1974, Cohen and Treetanthiploet 2020). Like regret, this is also an ex-post measure as calculating the expected total reward requires knowledge of the true reward distribution. In comparison, for field experiments the true rewards distribution is not known meaning only the observed total reward can be used as a comparison metric.

Formally, in our simulations the goal is to pick a decision rule, Ψ, that picks a sequence of prices from consideration set, P, that maximizes the total expected profit

$$\mathbb{E}\left[\sum_{\tau=1}^{t}\pi_{k_t}|P\right] \tag{EC.2}$$

EC.3. Details of Benchmarks

EC.3.1. Thompson Sampling

Thompson sampling (TS) is a randomized Bayesian parametric approach. For each action, a reward distribution is specified a priori and updated by the history of past trials. In each round, an action is picked according to the probability that it is optimal given the history of past trials. That is,

$$\text{Prob}(p_k|H_{t-1}) = \text{Prob}(E[\pi_{k,t}|p_k] > E[\pi_{k,t}|p_{k'}], \forall p_{k'} \neq p_k|H_{t-1}) \tag{EC.3}$$

The easiest implementation is to take a sample from each distribution each round and pick the one that gives the highest payoff.

One problem is that the true rewards distribution is not known a priori; however, as purchase decision is a binary decision then the structure is known in our application.[17] For Bernoulli bandits, beta distributions are used (Chapelle and Li 2011, Agrawal and Goyal 2012). Commonly used in Bayesian statistics, beta distributions have nice properties for binary bandits as they form a family of continuous probability distributions on the interval

[17] For an introduction to non-Bernoulli bandits see Russo et al. (2017).

$(0,1)$. The mean of Beta(α,β) is $\alpha/(\alpha+\beta)$ and as α and β get higher, the concentration around the mean grows. For a Bernoulli trial, the posterior distribution is simply Beta$(\alpha+1,\beta)$ if the trial was a success and Beta$(\alpha,\beta+1)$ if the trial was a failure.

To implement Thompson sampling for Bernoulli bandits, Beta$(1,1)$ is typically used as an uninformative prior as it is the uniform distribution on $(0,1)$. For notional simplicity, we set α to be denoted by S for number of successes and β to be denoted by F for number of failures. Each round a draw is taken from the beta distribution of each arm and the arm with the highest score (after being scaled by the prices) is chosen; the prior for the arm chosen is then updated with either a success of a failure. The entire algorithm is summarized below.

Algorithm 3: Thompson Sampling for Bernoulli bandits in Pricing

For each action $k=1,2,...,K$ set $S_k=0, F_k=0$

for $t=1,2,...$ **do**

  For each action k, sample $\theta_k(t)$ from Beta(S_k+1, F_k+1)

  Play price $p_{k_t} := \arg\max_{p_k}\ p_k\theta_k(t)$

  Observe profit π_{k_t}

  If $\pi_{k_t}=0$, then $F_k=F_k+1$, else $S_k=S_k+1$

end

EC.3.2. Upper Confidence Bounds

The *Upper Confidence Bound* (UCB) policy (Agrawal 1995, Auer 2002) is a deterministic non-parametric algorithm which is the optimal solution in minimizing the maximum regret. UCB uses a formula that scores each action in any given round and the highest scoring action is picked. The focal scoring rule is *UCB1*, which consists of two terms: a sample mean of past profits and an exploration bonus.

$$p_k^{\text{UCB1}} = \arg\max_{p_k\in P}\left(\underbrace{\bar{\pi}_{kt}}_{\text{Sample Mean}} + \underbrace{\sqrt{\frac{2\log(t)}{n_{kt}}}}_{\text{Exploration Bonus}}\right) \tag{EC.4}$$

Here, $\bar{\pi}_{kt}$ is the sample mean profit of playing action k through time t and n_{kt} is the number of times action k has been played by time t. The exploration is decreasing in the number of times an action k has been played and increasing in the number of time-steps t with a unit change in the denominator dominating a unit change in the numerator.

UCB overlooks a key component, which is it does not account for the variance of the rewards for each arm; there is less need to explore an arm that always gives the same reward compared to an arm with varied outcomes. This adjustment is accounted for in a second algorithm called *UCB-Tuned*, which empirically performs better (Auer et al. 2002).

$$\begin{aligned} p_k^{\text{UCB1-Tuned}} &= \arg\max_{p_k \in P} \left(\bar{\pi}_{kt} + \sqrt{\frac{\log(t)}{n_{kt}} \min\left(\frac{1}{4}, V_{kt}\right)} \right) \\ V_{kt} &= \left(\frac{1}{n_{kt}} \sum_{\tau=1}^{n_{kt}} \pi_{k\tau}^2 \right) - \bar{\pi}_{kt}^2 + \sqrt{\frac{2\log t}{n_{kt}}} \end{aligned} \tag{EC.5}$$

In general it assumed that each action has the same support, which is not the case for pricing where the reward is dependent on the price chosen; more precisely, $\pi_{k_t} \in [0, p_k]$. One simple solution is to scale the exploration bonus by the price, p_k, which empirically improves performance (Misra et al. 2019). We call this new algorithm where the exploration bonus is scaled by price *UCB-Price-Tuned*.

$$\begin{aligned} p_k^{\text{UCB-Price-Tuned}} &= \arg\max_{p_k \in P} \left(\bar{\pi}_{kt} + p_k \sqrt{\frac{\log(t)}{n_{kt}} \min\left(\frac{1}{4}, V_{kt}\right)} \right) \\ V_{kt} &= \left(\frac{1}{n_{kt}} \sum_{\tau=1}^{n_{kt}} \pi_{k\tau}^2 \right) - \bar{\pi}_{kt}^2 + \sqrt{\frac{2\log t}{n_{kt}}} \end{aligned} \tag{EC.6}$$

To summarize the UCB algorithm and its variants, the following pseudo-code is provided.[18]

Algorithm 4: UCB-Type Algorithms

```
Test each price once and update history
for t = k+1, k+2, ... do
    Play price p_{k_t} with highest index (equation EC.4, EC.5 or EC.6 depending on the UCB-variant)
    Observe profit π_{k_t} and update history
end
```

EC.3.3. UCB Partial Identification

EC.3.3.1. Description of UCB-PI. UCB-PI requires that the firm can identify to which of the S segments each customer belongs a priori (this information is not used to

[18] There are multiple methods for initializing the algorithm. The simplest is to just try each price once. Another approach is to set priors to encourage exploration. This involves temporarily setting each arm to have been tested once with the result in every case being a success. Once an arm is actually tested, this prior is replaced with the actual result.

price discriminate). The size of each each segment, ψ_s, is either known a priori (or updated each trial using customer identification). The size of the segments are normalized so that $\sum_{s\in S}\psi_s = 1$.

At any time step t, the first goal is to estimate the lower and upper bound of WTP for each segment s. Using the purchase data at time t, define $p^{\min}_{s,t}$ as the highest price where all consumers in segment s purchased and $p^{\max}_{s,t}$ as the lowest price where no consumer from segment s purchased. Given the assumption of weakly decreasing individual demand curves, the maximum WTP in a segment, $v^{\max}_s$ is the same as the lowest price where no consumer from the segment would purchase (likewise the minimum WTP in a segment, $v^{\min}_s$ is the highest price where all consumers in the segment would purchase). However, $p^{\max}_{s,t}$ ($p^{\min}_{s,t}$) is a downward biased estimator of $v^{\max}_s$ ($v^{\min}_s$) as it is likely the case that the consumer with the highest (lowest) valuation in a segment has not been observed by time t. A full description of how the valuation range, $[\hat{v}^{\min}_{s,t}, \hat{v}^{\max}_{s,t}]$, is estimated for each segment using the consumer purchase data can be found in Misra et al. 2019.

The second step is to find the demand and profit bounds for the entire population of customers; this is done by aggregating across segments. Defining $F_s(\cdot)$ to be the true cumulative density of all valuations within a segment s, then the aggregate demand can be written as

$$D(p_k) = \sum_{s\in S}\psi_s(1 - F_s(p_k)) \tag{EC.7}$$

While $F_s(p_k))$ is not observed, the pricing assumption can be used to obtain lower and upper bounds for demand. From the assumption, $F_s(p_k) = 0$ if p_k is less than the lower bound of valuations for segment s, $\hat{v}^{\min}_{s,t}$. Similarly, $F_s(p_k) = 1$ if p_k is greater than the upper bound of valuations for segment s, $\hat{v}^{\max}_{s,t}$. Thus, the possible range of demand at a given price p_k is

$$[\sum_{s\in S}\mathbb{1}(\hat{v}^{\min}_{s,t} \geq p_k), \sum_{s\in S}\mathbb{1}(\hat{v}^{\max}_{s,t} \geq p_k)] \tag{EC.8}$$

To obtain the profit bounds at a particular price, the demand bounds are scaled by the price.

$$LB_t(\pi(p_k)) = p_k \sum_{s\in S}\psi_s\mathbb{1}(\hat{v}^{\min}_{s,t} \geq p_k) \tag{EC.9}$$

$$UB_t(\pi(p_k)) = p_k \sum_{s\in S}\psi_s\mathbb{1}(\hat{v}^{\max}_{s,t} \geq p_k) \tag{EC.10}$$

The final step is to eliminate completely dominated actions. That is if the upper bound of a particular price is less than the lower bound of another price, then that price should never be chosen. That is p_k should never be chosen if $UB(\pi(p_k)) \leq \max_l LB(\pi(p_l))$. Additionally, the exploration bonus is scaled by $2\hat{\delta}$, the maximum of the estimated range of the valuations across all segments, giving the final decision rule.

$$p_k^{UCB-PI-Tuned} = \arg\max_{p_k \in P} \begin{cases} \bar{\pi}_{kt} + 2p_k\hat{\delta}\sqrt{\frac{\log(t)}{n_{kt}}\min\left(\frac{1}{4}, V_{kt}\right)} & \text{if } UB_t(\pi(p_k)) > \max_l LB_t(\pi(p_l)) \\ 0 & \text{if } UB_t(\pi(p_k)) \leq \max_l LB_t(\pi(p_l)) \end{cases}$$

$$V_{kt} = \left(\frac{1}{n_{kt}}\sum_{\tau=1}^{n_{kt}} \pi_{k\tau}^2\right) - \bar{\pi}_{kt}^2 + \sqrt{\frac{2\log t}{n_{kt}}} \tag{EC.11}$$

After a price is tested, the result is observed and the history is updated before the next round. A complete summary of the entire algorithm is outlined in Algorithm 5.

Algorithm 5: UCB-PI-Tuned

```
Test each price once and update history
for t = k+1, k+2, ... do
    Estimate lower and upper valuation bounds in each segment
    Calculate lower and upper aggregate demand/profit bounds
    Eliminate dominated actions (those prices where upper profit bound is lower than another
      price's lower profit bound)
    Pick price p_{k_t} using a scaled UCB-Tuned policy on the set of non-dominated actions
    Observe profit π_{k_t} and update history
end
```

EC.3.3.2. UCB-PI-Tuned Converges to UCB-Price-Tuned as Segmentation Becomes Uninformative. Completely uninformative segmentation means the highest price, 1, and lowest price, 0, is possible in each segment. Consequently, $v_{s,t}^{\min} = 0$ and s and $v_{s,t}^{\max} = 1$ for each segment. From the equation for the lower bound, equation EC.9, it follows that the lower bound at every single price in the support $[0, 1]$ is 0 and so no price can be dominated. Additionally, for a completely uninformative segment the midpoint will be 0.5 with $\delta = 0.5$. Applying these changes reduces the UCB-PI-Tuned decision rule (equation EC.11) to the UCB-Price-Tuned decision rule (equation EC.6).

EC.4. Practical Implementation Issues with GPTS

EC.4.1. Reducing Cardinality of Training Set

A large practical issue with using GPTS is that tuning the hyperparameters is costly as matrix inversion is $\mathcal{O}(n^3)$. Generally, the input training data is the set of prices tried $P_t = \{p_{k_1}, ..., p_{k_t}\}$, and the corresponding output training data are the rewards realized $\boldsymbol{y} = \{\pi_{k_1}, ..., \pi_{k_t}\}$. However, as set of test prices is picked a priori, then the dimension of the training data can be reduced as each price tested has to come from this set. Accordingly, the input training data is the set of prices that have been tested and the output training data are the corresponding sample means of the rewards at those prices. That is, at time t when every price in the test set has been tried, the input training data is $P = \{p_1, ..., p_K\}$ and the output training data is $\{\bar{\pi}_{1t}, ..., \bar{\pi}_{Kt}\}$ where $\bar{\pi}_{kt} = 1/n_{kt} \sum_{\tau=1}^{n_{kt}} \pi_{k,\tau}$. With this adjustment, the dimension of the training data can never be larger than the test set, allowing the algorithm run at the same pace regardless of the number of observations.

The one overlooked detail is that $\boldsymbol{\sigma}_{\boldsymbol{y}}^2 = \{\sigma_{1t}^2, ..., \sigma_{Kt}^2\}$ will be dependent on the number of times that a price has been tried. As sample variance scales by the number of observations, then simply the noise parameter becomes $\boldsymbol{\sigma}_{\boldsymbol{y}}^2 = \{\sigma_{1t}^2/n_{1t}, ..., \sigma_{Kt}^2/n_{Kt}\}$ where n_{kt} is the number of times that arm k has been played by round t. Overall, these adjustments make an unnoticeable difference compared to using each observation individually in the training data.

EC.4.2. Floating Point Precision Errors

One very common error when running GPTS is floating point precision errors. This prevents matrices from being inverted using Cholesky decomposition as there are quite often trivially small negative eigenvalues. This is a common problem well studied in finance (Higham 2002) and the solution is to create an algorithm that finds the nearest covariance matrix to the *almost* covariance matrix. We use the approach devised by Rebonato and Jäckel (2011).

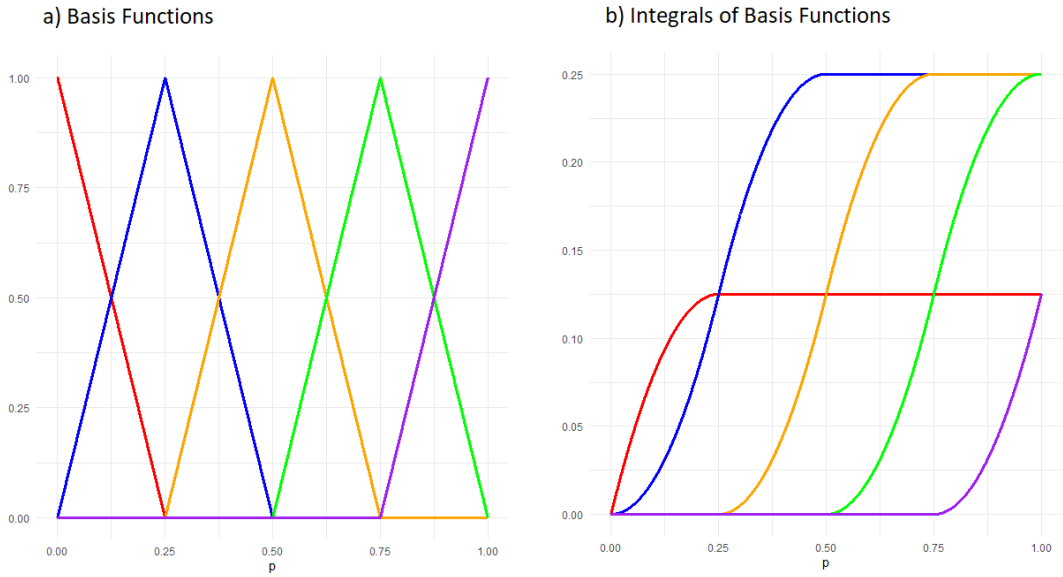
To summarize the algorithm, as a covariance matrix is positive semi-definite it can be written as $C = MDM^{-1}$ where D is the diagonal matrix of eigenvalues and M is the matrix of column eigenvectors. Each iteration the negative eigenvalues in D are replaced with a small positive value (we use $1e^{-10}$) to create D^+. The new covariance becomes $C^+ = MD^+M^{-1}$. This process continues iteratively until C^+ has all positive eigenvalues.

EC.5. Further Details on Monotonic GPTS

EC.5.1. Visualizing Basis Functions

Consider an example where $N = 4$ meaning there are 5 equally spaced knots $\{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$. Figure EC.2 shows the five basis functions along with their corresponding integrals.

Figure EC.2 Plot of Basis Functions and their Integrals (5 knots)



For any continuous function $f : [0,1] \to \mathbb{R}$ the function $\tilde{f}(\cdot) \approx \sum_{j=0}^{N} f(u_j) h_j(\cdot)$ approximates f by linearly interpolating between the function values at the knots u_j. That is, each point of the estimate $\tilde{f}(\cdot)$ is just the weighted linear interpolation between two knots. For example, estimating x at $\frac{3}{8}$ gives $\tilde{f}(3/8) = (1/2)f(1/4) + (1/2)f(1/2)$.

EC.5.2. Basis Estimation Converges to True Function

Recall μ_j are the knots, N is the number of knots, $\delta_n = 1/N$ is the distance between the equally spaced knots, and $h_j(p)$ are the basis functions as defined in equations 9 and 10.

LEMMA EC.1. *For every $p \in [0,1]$ the basis functions sum to 1. That is, $\sum_{j=0}^{N} h_j(p) = 1$.*

LEMMA EC.2. *A basis function $h_j(p)$ is equal to 0 when $p - \mu_j > \delta_N$.*

PROPOSITION 2. *The distance between a continuous function D and its estimation via basis functions $D_N(p) = \sum_{j=0}^{N} D(\mu_j) h_j(p)$ converges to 0 as $N \to \infty$.*

Proof:

$$|D_N(p) - D(p)| = \left| \sum_{j=0}^{N} D(\mu_j) h_j(p) - D(p) \right| \tag{EC.12}$$

$$= \left| \sum_{j=0}^{N} h_j(p)(D(\mu_j) - D(p)) \right| \quad \text{by Lemma 1} \tag{EC.13}$$

$$\leq \sum_{j=0}^{N} h_j(p) \left| D(\mu_j) - D(p) \right| \quad \text{by Triangle Inequality} \tag{EC.14}$$

$$\leq \sum_{j=0}^{N} \sup_{p'-p \leq \delta_N} |D(p') - D(p)| \quad \text{by Lemma 1 and 2} \tag{EC.15}$$

By continuity of D as $(p' - p) \to 0$ then $D(p') - D(p) \to 0$. Thus as $N \to \infty$,

$$|D_N(p) - D(p)| \xrightarrow[N\to\infty]{} 0 \tag{EC.16}$$

COROLLARY EC.1. *The distance between a continuous function D and its estimation using its derivatives $D_N(p) = D(0) + \sum_{j=0}^{N} D'(u_j) \int_0^p h_j(x)dx$ converges to 0 as $N \to \infty$.*

EC.5.3. Proof of Proposition 1

From Proposition 2, a continuous demand D can be estimated via basis functions as $D(p) = \sum_{j=0}^{N} D(\mu_j)h_j(p)$. As, by assumption, the derivative of D is also continuous then the same formula can be used to estimate D':

$$D'(p) \approx \sum_{j=0}^{N} D'(u_j)h_j(p) \tag{EC.17}$$

Additionally, by the Fundamental Theorem of Calculus

$$D(p) - D(0) = \int_0^p D'(t)dt \tag{EC.18}$$

Substituting into (EC.17) into (EC.18) gives

$$D(p) \approx D(0) + \int_0^p \sum_{j=0}^{N} D'(u_j)h_j(t)dt \tag{EC.19}$$

$$D(p) \approx D(0) + \sum_{j=0}^{N} D'(u_j) \int_0^p h_j(t)dt \quad \square \tag{EC.20}$$

EC.5.4. Gaussian Process Estimation of Derivatives and Intercept

Typically, the Gaussian process posterior prediction is obtained using the following equations for the mean and covariance.

$$\mu(D^*) = K(P, P_t)[K(P_t, P_t) + \sigma_y^2 I]^{-1}\pi_{\mathbf{t}} \text{ and}$$

$$\text{Cov}(D^*) = K(P, P) - K(P, P_t)[K(P_t, P_t) + \sigma_y^2 I]^{-1}K(P_t, P) \tag{EC.21}$$

where $P_t = \{p_{k_1}, ..., p_{k_t}\}$ is a vector of prices tried (input training data), $\pi_t = \{\pi_{k_1}, ..., \pi_{k_t}\}$ is a vector of rewards obtained (output training data), $P = \{p_1, ..., p_K\}$ is the set of prices under consideration (test points), σ_y^2 is the noise parameter,[19] $K(\cdot, \cdot)$ is the covariance matrix given by the RBF kernel, and D^* is the GP posterior estimation of the demand function.

The basis function method requires the estimation of the intercept and the derivatives at each of the prices in the consideration set. More formally, the goal is to estimate the posterior mean and covariance for $\{D(0), D'(p_1), ..., D'(p_k)\}$. Temporarily ignoring the intercept, the key is that the derivatives of a GP are also a GP. This means that D'^*, the posterior vector of derivatives of D^*, is

$$D'^* \sim N\left(\frac{d}{dp}\mu(D^*), \frac{d}{dp}\operatorname{Cov}(D^*)\right) \tag{EC.22}$$

Note as the values of D^* are only at our test points P then the derivative only needs to be calculated with respect to P. This means that to to estimate the posterior mean and covariance for $\{D(0), D'(p_1), ..., D'(p_k)\}$, the only necessary needed changes are to calculate the derivatives of the kernel function with respect to the test points.

Claim 1. *The derivative of the covariance between a derivative test point at p_i^* and an observation at p_j is*

$$\frac{\partial k(p_i^*, p_j)}{\partial p_i^*} = \frac{\sigma_f^2}{l^2}(p_j - p_i^*)\exp\left(\frac{-(p_i^* - p_j)^2}{2l^2}\right) \tag{EC.23}$$

Claim 2. *The derivative of the covariance between an observation at p_i and a derivative test point at p_j^* is*

$$\frac{\partial k(p_i, p_j^*)}{\partial p_j^*} = \frac{\sigma_f^2}{l^2}\left(p_i - p_j^*\right)\exp\left(\frac{-(p_i - p_j^*)^2}{2l^2}\right) \tag{EC.24}$$

Claim 3. *The derivative of the covariance between a derivative test point at p_i^* and another derivative test point at p_j^* is*

$$\frac{\partial^2 k(p_i^*, p_j^*)}{\partial p_i^* \partial p_j^*} = \frac{\sigma_f^2}{l^4}\left(l^2 - (p_i^* - p_j^*)^2\right)\exp\left(\frac{-(p_i^* - p_j^*)^2}{2l^2}\right) \tag{EC.25}$$

[19] This can be thought of as either a scalar or a vector depending on whether the specification is homoscedastic or heteroscedastic.

EC.6. Simulation Details

EC.6.1. Initialization

Before data is collected, priors are needed on arms to to allow for the various algorithms to be calculated. While this has little effect on long-term behavior, a bad initialization strategy can lead to unnecessary poor results in the short term.

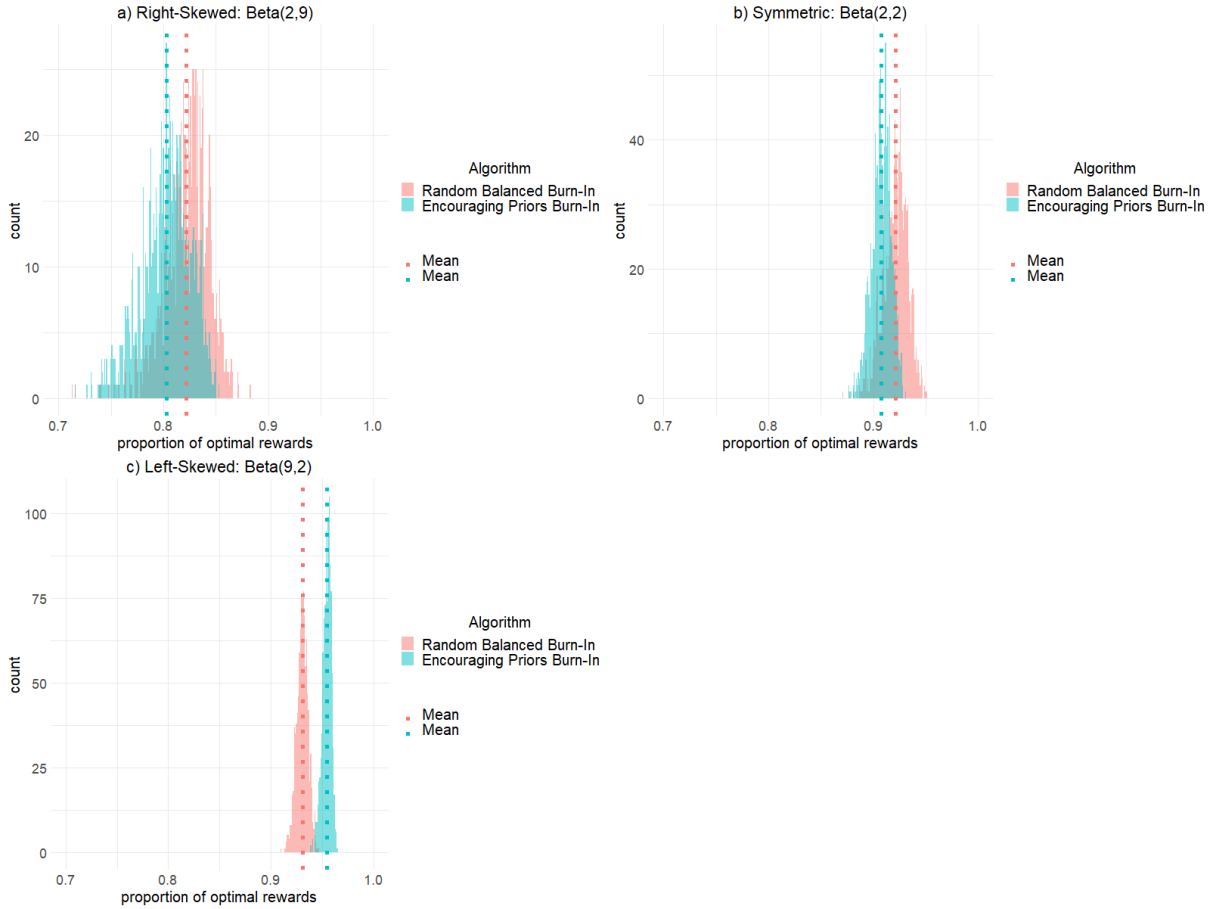
EC.6.1.1. Thompson sampling. As is standard for Thompson sampling with binary outcomes (Russo et al. 2017), an uninformative prior of Beta(1,1) is used for the demand at each price being tested. As information becomes available the prior is updated at the price tested.

EC.6.1.2. Gaussian process Thompson sampling. A GP is completely specified by its mean and covariance matrix. We use the RBF kernel which means that a prior GP can be completely specified by a mean and hyperparameters σ_f and l. Arbitrarily, we choose the mean to be 1 and set $\sigma_f = 0.7$ and $l = 0.4$, but the values do not matter; it would be equally valid to choose the first price at random. Once even one data point becomes available, the mean and hyperparameters can be estimated from the data and a draw across the entire support of prices being considered can be made.

EC.6.1.3. UCB. UCB presents a larger challenge as some assumption needs to be made about the arm for the UCB decision rule to be calculated. One common approach is to try each arm once (or 10 times in our setting which is how long it takes to change the price) in a balanced burn-in before starting the algorithm (Sutton and Barto 2018). However, in the pricing setting there is a strictly more efficient method which is to assume that each price has been tested once at that a customer purchased at each price. These encouraging priors will lead to the UCB method picking prices in a descending fashion. Because of the nature of the pricing setting where demand will be scaled by the price, it may be possible to eliminate lower prices before trying them if the upper bound of the profit is less than the lower bound at a higher price. However, it is impossible to eliminate a higher price from data from a lower price, because even if the demand is 1 it could be that the demand is also 1 at this higher price. Thus, this method of setting encouraging priors where it is assumed that each price has been tested once where each trial resulted in a purchase is more efficient than testing each price once.

EC.6.1.4. UCB-PI. The largest problem with initialization is for the UCB-PI method. The initialization process is not described in Misra et. al (2019) and our testing found no method which was superior in all the trial conditions. While the initialization method described in the UCB section works, the partial identification method gains information from trying varied prices which allows for both high and low dominated prices to be removed from consideration.

Figure EC.3 Histogram of Rewards across Simulations



Notes. These are the expected proportion of optimal rewards after 5000 iterations. The count is over 1000 simulations. The dotted line is the mean across 1000 simulations.

For this reason, we consider an additional initialization process where there is a burn-in consisting of a randomized balanced experiment where each price is tried once[20] except the price is skipped if it has already been removed from the consideration set by the partial identification method. After the burn-in the UCB-PI decision rule is used.

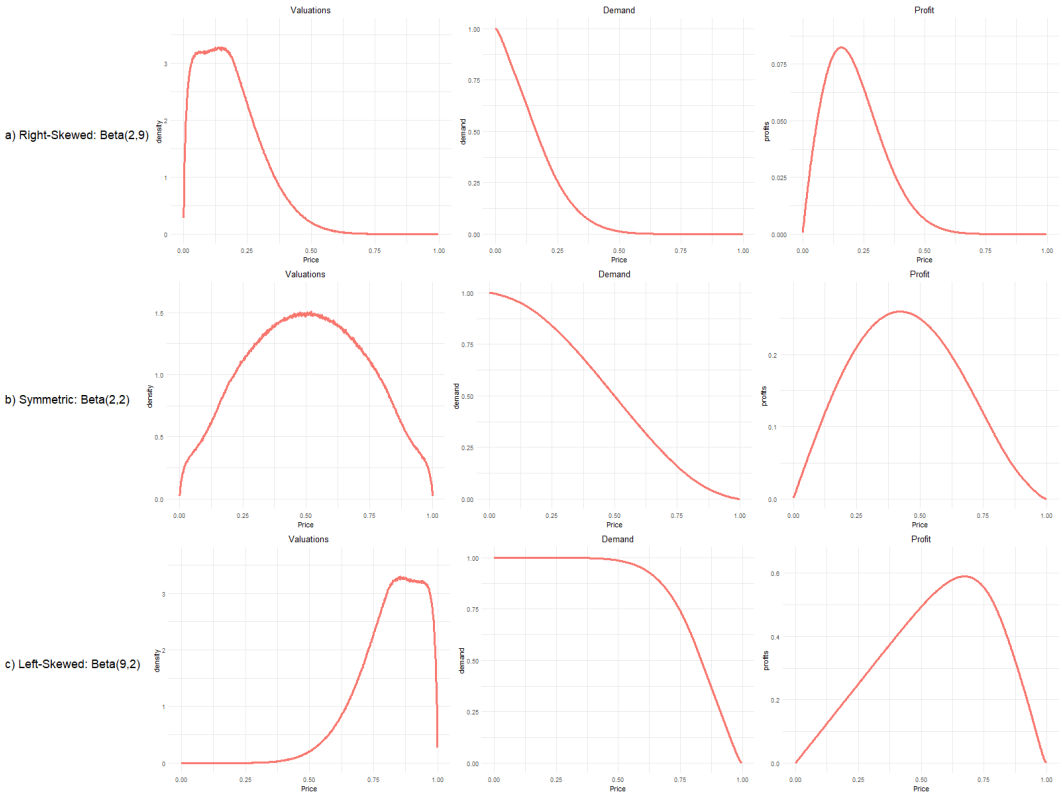
[20] 10 times in our setting

Which initialization method works better depends on the underlying true WTP. Assuming demand is 1 for each price tends to lead to higher profits for distributions where the optimal price is high (such as Beta(9,2)), while the randomized balanced experiment works better for distributions where the optimal price is low or central (such as Beta(2,9) or Beta(2,2)). The results for the two methods are compared below. For consistency, the results in the main body of the paper use the same initialization method for both UCB and UCB-PI, which is to use encouraging priors assuming demand is 1 for all prices that have yet to be tested.

EC.6.2. Willingness To Pay Distributions

The true willingness to pay, and associated demand and profits for each test case are shown in Figure EC.4.

Figure EC.4 True Underlying Distributions



Notes. The first column shows the underlying willingness to pay distribution for each of the test cases. The second column shows the true demand given these valuations, while the third column gives the associated profits.