Dynamic Pricing & Learning: An Application of Gaussian Process Regression

Daniel Ringbeck & Arnd Huchzermeier

WHU – Otto Beisheim School of Management, Burgplatz 2, 56179 Vallendar, Germany
Daniel.Ringbeck@whu.edu • Arnd.Huchzermeier@whu.edu

Abstract

We consider the problem of offering an optimal price when demand is unknown and must be learned by price experimentation – a variant of the multi-armed bandit problem. In each period, the retailer must decide on a price while using knowledge already acquired to weigh the benefits of exploring other prices versus maximizing revenue. Other algorithms proposed for such problems either learn the price–demand relation for each price separately or determine the parameters of an assumed parametric demand function. We instead combine Gaussian process regression with Thompson sampling as a nonparametric learning algorithm that can learn any functional relation between price and demand. This GP-TS algorithm scales significantly better with the number of price vectors than do existing approaches, yet it makes no restrictive assumptions on the price–demand relationship's functional form. We show how to apply the algorithm to finite inventory settings that consider both single and multiple products and also to settings in which exogenous contextual information can affect prices. One advantage of the algorithm is that its performance depends not on the number of price vectors over all products but only on the number of products considered. For each setting considered here, we benchmark our approach to existing algorithms. The GP-TS algorithm's learning performance is far superior to that of its peers, especially when there are increases in the number of products to learn concurrently. Finally, we propose extensions that enable the application of our algorithm when demand follows a Bernoulli or Poisson distribution.

Keywords: dynamic pricing, revenue management, demand learning, Gaussian processes

1 Introduction

In today's online commerce sector, the sheer amount of information available – when combined with the speed of information processing – enables retailers to make real-time decisions. Increasing the capacity to process information allows a retailer to enhance the customer's service experience and also to increase the efficiency of selling its products. The latter task is especially relevant in the context of pricing decisions for products with short life cycles and limited inventory. Products of this type are commonly sold in the fashion industry, where a rapidly rotating product assortment can give the retailer a substantial competitive advantage. For such products, retailers must optimize prices despite a lack of experience regarding their price–demand relationship. Research has documented that price decisions based on analytics can result in significant gains in revenue (Ferreira et al. 2016). In the case of multiple products or vertically integrated supply chains, the retailer could also face the problem of managing the revenue of a network with interdependent resources. With a total size of $2,842 billion, the global online retailing market accounts for a considerable share of the overall retail sector (eMarketer 2018).

It is only natural that the online pricing problem has attracted the attention of researchers in the operations research community. The literature refers to such problems as dynamic pricing with learning under inventory constraints, and several works have inspired a surge of interest in the topic. The focus of this research is the development of learning algorithms that optimize the prices of products under inventory constraints and *without* prior knowledge of price–demand relations. Thus the retailer must experiment with prices to gain knowledge of the price–demand relationship and, at the same time, optimize revenue. In the multi-armed bandit (MAB) literature, this problem is known as the "exploration–exploitation trade-off". Scholars in the MAB field typically assume neither the existence of any interrelations among prices nor a simple parametric relationship between price and demand.

It is worth noting that, in related fields such as machine learning, Bayesian methods have garnered significant research interest. Such methods are now widely applied in the fields of physics, geostatistics, and biology. One algorithm that has received much attention is Gaussian process regression, which is a nonparametric Bayesian approach to function approximation that yields uncertainty estimates in addition to closed-form predictions. Gaussian processes (GPs) use kernels to "virtually" approximate any functional relationship. The availability of uncertainty estimates facilitates the process of combining GPs with acquisition functions to determine a learning problem's next "query point". Those uncertainty estimates make GPs extremely useful for solving MAB problems.

We shall introduce Gaussian processes to the study of dynamic pricing and the field of operations research more generally. Toward those ends, we combine GPs with existing research on dynamic pricing and learning under inventory constraints (see e.g., Ferreira et al. 2018). Such a combination of machine learning techniques with existing algorithms in the MAB literature is an emerging field of research (Misra et al. 2019). In particular, we develop an outright Bayesian optimization framework that can be adapted to any variant of the pricing problem when used in conjunction with Monte Carlo Markov chain (MCMC) estimation. This approach is both highly flexible and nonparametric, so it performs well even when – which is often the case in practice – the price–demand relation is nonlinear and dependent on other contextual factors. Rather than fully exploring all possible price vectors independently, which becomes infeasible as the number of products increases, our *Gaussian process Thompson sampling* (GP-TS) algorithm scales more efficiently. Moreover, our approach can be easily implemented and is not a "black box" model.

To assess the performance of the GP-TS algorithm, we compare it to state-of-the-art methods from operations research. More specifically, we conduct three numerical experiments that reveal our approach to be superior to existing methods. Especially in problems with multiple products, our algorithm learns to identify the optimal price vector much more rapidly than do the other algorithms. It is consequently our considered opinion that Gaussian processes constitute a valuable addition to dynamic pricing research as well as to the set of modeling techniques employed by both marketing and operations researchers.

The rest of our paper proceeds as follows. Section 2 reviews the relevant dynamic pricing literature and the MAB literature before introducing Gaussian processes. In Section 3, we introduce our algorithm and discuss its properties. Section 4 is devoted to testing the algorithm in several numerical experiments, and Section 5 describes some extensions of the GP-TS algorithm to problem settings that are non-Gaussian. We conclude in Section 6 with a discussion of our study's findings, its limitations, and possible directions for future research.

2 Literature Review

In this section, we review the literature that is relevant to our work. After surveying related literature in the dynamic pricing domain, we review – in the broader field of multi-armed bandit problems – the research that is most closely connected to this paper. We then cover the literature on Gaussian process regression – a machine learning algorithm that we employ to model the online network revenue problem.

2.1 Dynamic Pricing with Demand Learning

The ever-increasing availability of data on real-time demand has heightened the relevance of dynamic pricing problems that involve learning about *online* demand – especially in the retail industry. Dynamic pricing with learning, which is a subset of the literature on dynamic pricing, can be further segmented into settings that consider (respectively) infinite and finite levels of inventory. Because our interest is in the finite inventory setting, we refer the reader to extensive literature reviews by Aviv and Vulcano (2012) and den Boer (2015); those reviews incorporate works that consider settings of the infinite inventory type.

Much as in most earlier works in the field, Araman and Caldentey (2009) and Farias and van Roy (2010) adopt a dynamic programming approach in a setting where customers' willingness to pay is known but where the *unknown* market size follows a Poisson process. Chen et al. (2014) employ self-adjusting heuristics that are used during an exploration phase at the beginning of the sales season; they establish strong theoretical bounds for their algorithm in both parametric and nonparametric cases. Wang et al. (2014) study a single-product setting with finite inventory and continuous prices. Making several assumptions about the properties of the underlying demand function allows these authors to develop a continuously learning algorithm. Although the works just cited are all relevant in the broader context of the research field of dynamic pricing with learning, their model assumptions differ markedly from the ones that we make. Hence we shall next review papers that are more relevant to our particular research context.

Besbes and Zeevi (2009) initiate a stream of literature in which the goal is to learn the optimal static price in a setting with incomplete information. They propose to divide the selling season into an exploration phase and an exploitation phase (and thus provide the basis for later research of Chen et al. 2014). During the *exploration* phase, all prices are explored a pre-defined number of times. In each period of the subsequent *exploitation* phase, the optimal prices for the rest of the selling season are computed and offered. Besbes and Zeevi (2012) extend their previous work to a revenue management setting for a full network, where multiple products share the same resources. One evident weakness of their approach is that it relies heavily on the choice of a reasonable length for the exploration phase. Another methodological difficulty is that the exploration phase's price experimentation explores all prices to the same extent and so often "overexplores" poor price choices, especially when the number of feasible prices is large. Ferreira et al. (2018) model the online network revenue management problem as a MAB problem. These authors exploit heuristics commonly used for this class of problems to manage the exploration–exploitation trade-off, combining the prices suggested by those heuristics with an optimization problem to account for inventory constraints.

Since our approach is most nearly related to these two works and since it also contributes to research on the MAB problem, we next review that problem in more detail.

2.2 Thompson Sampling and the Multi-Armed Bandit Problem

The Thompson sampling (TS) algorithm was initially developed for optimizing the allocation of experimental effort among different treatments in clinical trials (Thompson 1933). Russo et al. (2018) recounts that the academic literature on this approach, which is also known as "posterior sampling" and "probability matching", ignored TS for decades; in recent years, however, scholarly interest in TS has proliferated. Thomson sampling in its general form is a randomized Bayesian approach to solving sequential decision problems, of which multi-armed bandit problems are an illustrative example. In MAB problems, the agent must choose an action x out of K actions and must do so repeatedly over a time horizon T. The agent's goal is to maximize the reward r he receives for undertaking a sequence of actions $x_1, x_2, x_3, \ldots, x_T$. However, the agent does not know which action generates the highest rewards; thus his reward y for action x is stochastic, and it follows the unknown distribution $q_\theta(\cdot|x)$. The agent can achieve his goal only by balancing, in each time period t, the exploration of unknown actions – that is, learning the distribution parameters θ – against exploiting the best-known action. The performance of learning algorithms in the context of this problem is measured by *regret.* "Cumulative" regret is the difference between the best possible expected reward (as when the true distribution, θ, is known a priori) and the expected reward achieved by an algorithm with *no* knowledge of θ. Hence expected regret, $R(T, \theta)$, can be expressed as

$$\mathbb{E}[R(T,\theta)] = \mathbb{E}\Bigg[\sum_{1}^{T} r^*(t|\theta) - r^S(t|\theta)\Bigg]. \tag{1}$$

In this expression, $r^*(t|\theta)$ denotes the best possible reward in time period t when θ is known; $r^S(t|\theta)$ denotes the reward obtained by following a strategy S for choosing actions. In the MAB literature, theoretical bounds on the regret that can be achieved by an algorithm usually depend on T and, in the case of discrete sets of actions, on the number K of actions from which the agent can choose. Bubeck and Cesa-Bianchi (2012) show that the best possible lower bound in terms of T is $\Omega(\sqrt{T})$; in terms of T and K, it is $\Omega(\sqrt{KT})$.

Thompson sampling balances the exploration–exploitation trade-off by incorporating the prior distribution in addition to knowledge obtained by the agent and computing the posterior probability distribution of the exact value θ, $\hat{\theta}$. In each time period, a sample is obtained from this posterior distribution and then the algorithm chooses an action that maximizes the expected outcome, $\mathbb{E}_{q_{\hat{\theta}}}[y_t|x_t = x]$. After applying action x_t, the agent observes y_t and updates her belief, $\hat{\theta}$. Over time, that belief approximates θ and so the samples of $\hat{\theta}$ always generate the highest value for the best action. At this point, then, the agent automatically ceases to explore undesirable actions. Algorithm 1 summarizes this general form of TS in pseudo-code (see Russo et al. 2018 for a thorough introduction to TS and its variants).

Algorithm 1: Thompson Sampling (TS)

```
for $t = 1, \ldots, T$ do
    Sample $\hat{\theta}$.
    $x_t \longleftarrow \text{argmax}_{x \in \mathbb{A}} \mathbb{E}_{q_{\hat{\theta}}}[y_t|x_t = x]$.
    Apply $x_t$ and observe $y_t$.
    $\hat{\theta} \leftarrow \mathbb{P}_{p,q}(\theta \in \cdot|x_t, y_t)$.
```

Thompson sampling is just one of many decision algorithms developed for the MAB problem. Alternatives include simple "greedy" heuristics, ε-greedy strategies, strategies based on the upper confidence bound (Auer et al. 2002; Badanidiyuru et al. 2013), and algorithms that divide T into exploration and exploitation phases (Besbes and Zeevi 2009, 2012) as well as variants of all these approaches. Most notably, Chapelle and Li (2011) and Scott (2010) demonstrate the superior performance of TS in empirical settings; Kaufmann et al. (2012) and also Agrawal and Goyal (2013) establish theoretical performance guarantees for TS that are similar to those of its alternatives.

The TS algorithm has since been applied to a variety of other settings, such as A/B testing (Graepel et al. 2010), online advertising (Graepel et al. 2010; Agarwal 2013; Agarwal et al. 2014), Monte Carlo tree search (Bai et al. 2013), recommendation systems (Kawale et al. 2015), website optimization (Hill et al. 2017), marketing (Schwartz et al. 2017), and hyperparameter tuning (Kandasamy et al. 2018). It is clear that TS can be applied flexibly to many different kinds of problems. Another way to use TS consists of applying it in the context of a larger machine learning algorithm, thereby taking advantage of Thompson sampling's desirable behavior in online learning settings. For instance, TS has been applied to active learning problems with neural networks (Lu and van Roy 2017) and deep reinforcement learning (Osband et al. 2017). Today, companies such as Adobe, Amazon, Facebook, Google, LinkedIn, Microsoft, Netflix, and Twitter make use of this algorithm to enhance their operations (Graepel et al. 2010; Scott 2010; Agarwal 2013; Agarwal et al. 2014; Scott 2015; Hill et al. 2017). In operations research, Ferreira et al. (2018) build on the theoretical findings of Russo and van Roy (2014) and apply TS to a dynamic pricing problem with learning in a finite inventory setting. Ferreira et al. find an upper regret bound of $O\left(\sqrt{TK \log K}\right)$ for their algorithm – a bound indicative of theoretical performance guarantees that are close to the lower bound. As mentioned previously, we propose to combine TS with a nonparametric regression algorithm from the machine learning literature: Gaussian process regression.

2.3 Gaussian Processes

Here we briefly introduce Gaussian processes (GPs) and their application to learning problems; see Rasmussen and Williams (2008) and Shahriari et al. (2016) for additional details on these processes. Formally, a GP is a generalization of the Gaussian probability distribution that represents a distribution over a space of functions. Gaussian processes can also be viewed as a marginalized, kernel-based version of Bayesian linear regression. The main advantage of GPs is in their application to nonparametric Bayesian optimization, under which they can approximate highly nonlinear functions. Consider a collection of n points for $x = x_1, \ldots, x_n \in \mathbb{R}^d$ that are assumed to follow a functional relationship $f = f_{1:n}$ and observations $y = y_{1:n}$ that are assumed to be normally distributed given f and the Gaussian noise factor σ^2. In a GP, the assumption is that f is not only jointly Gaussian distributed but also completely defined by its mean and covariance functions – respectively, $m(x)$ and $k(x, x')$ – such that $f \sim \mathrm{GP}(m(x), k(x, x'))$. These mean and covariance functions are

$$\begin{aligned} m(x) &= \mathbb{E}[f(x)] \quad \text{and} \\ k(x, x') &= \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]. \end{aligned} \tag{2}$$

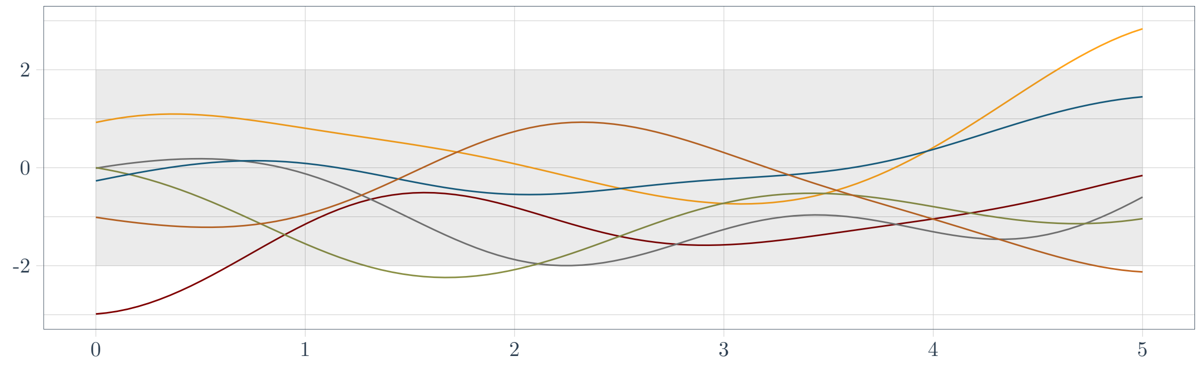
The covariance function $k(x, x')$ is the defining element of any GP model in that it represents a model's assumptions about the shape of the function underlying the data. A wide range of such functions – which are known as *kernels* and include (among others) linear, cosine, and Matérn varieties – are available for application to GPs (Rasmussen and Williams 2008). One common choice is the squared exponential covariance function, a special case of the Matérn kernel that is defined as follows:

$$k(x_i, x_j) = \exp\left\{ -\frac{(x_i - x_j)^2}{2l^2} \right\} \tag{3}$$

for l an exogenous length scale parameter (Shahriari et al. 2016). Given the covariance function k, we can compute a matrix $\Sigma(x, x)$ containing covariance terms between all test points x such that $\Sigma_{ij} = k(x_i, x_j)$ for any two points i and j; it is therefore possible to sample quasi-continuous functions $f \sim \mathcal{N}(m(x), \Sigma(x, x))$ from a GP prior. Note that $m(x)$, in this case, corresponds to the mean of the prior distribution, which we define as $m_0(x) = 0$ to simplify this introductory discussion.

In Figure 1 we plot samples from a two-dimensional GP with squared exponential kernel. The

Figure 1: Samples from an Illustrative Prior Distribution of a Gaussian Process



displayed Gaussian process is centered at a mean of 0, and the lines represent samples from the distribution of functions. The smoothness of these lines depends on our choice of the covariance function and also on its parameters. In the process of learning, we incorporate the function values at *training data* points to estimate the posterior probability distribution over the space of functions described by this Gaussian process. Given the prior and training data points $D_t = \{x_{1:t}, y_{1:t}\}$, the joint distribution of the data and the test points x can be written as

$$\begin{pmatrix} y_t \\ f(x) \end{pmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \Sigma_t + \sigma^2 I & k(x_{1:t}, x) \\ k(x_{1:t}, x)^T & k(x, x) \end{bmatrix}\right), \tag{4}$$

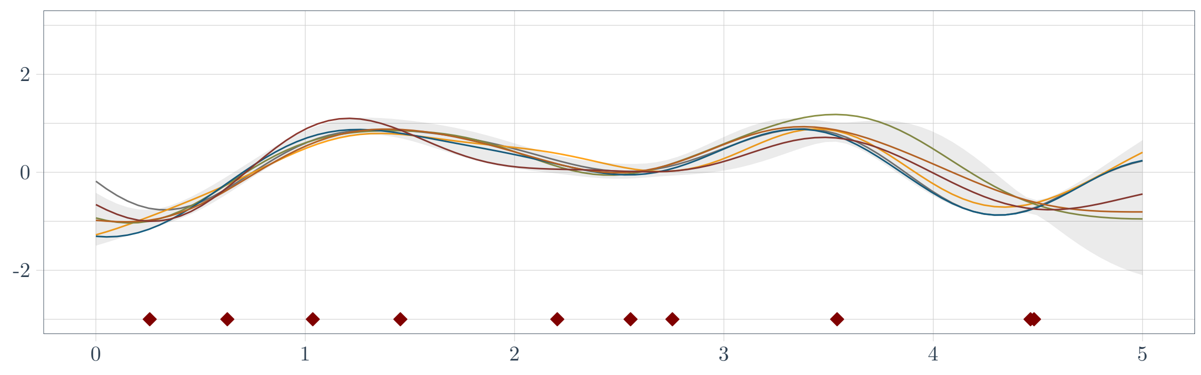
where $\Sigma_t = \Sigma(x_{1:t}, x_{1:t})$ and $\mu_0 = 0$ are as defined before. Then $f(x)$ – the posterior probability distribution of the test points x given data D_t – is defined as:

$$\begin{aligned} &f(x)|D_t \sim N(m_t(x), \sigma_t^2(x)) \quad \text{for} \\ &m_t(x) = k(x_{1:t}, x)^T(\Sigma_t + \sigma^2 I)^{-1} y_{1:t} \quad \text{and} \\ &\sigma_t^2(x) = k(x, x) - k(x_{1:t}, x)^T(\Sigma_t + \sigma^2 I)^{-1} k(x_{1:t}, x). \end{aligned} \tag{5}$$

Evaluating the posterior mean m_t and variance σ_t^2 at all test points x, we obtain a complete estimation of $f(x)$.

Any learning task requires an estimate not only of the expected reward but also of the amount that can be learned for each action taken while assessing the exploration–exploitation trade-off. Apart from being nonparametric, a key advantage of GPs is their ability to give measures of expected values, $m_t(x_i)$, and uncertainties, $\sigma_t^2(x_i)$, for any point x_i. This is why GPs are ideal for modeling the exploration–exploitation trade-off that underlies multi-armed bandit problems. In Figure 2 we consider the same GP as in Figure 1; however, the latter includes several samples at different points on the horizontal axis (marked by the nine red squares). We can use the prior and these data to compute the posterior distribution. This figure's plot reveals that the GP model adapts to the data in terms of both its mean prediction and the certainty of that prediction, where certainty is represented by the (shaded) 90% confidence intervals.

Figure 2: Samples from an Illustrative Posterior Distribution of a Gaussian Process



The ability of Gaussian processes to generate predictions of both value and uncertainty makes them well suited for the task of solving MAB-related problems. In practice, GPs have been applied to a diverse set of such problems; applications include robotics (Lizotte et al. 2007), interactive user interfaces (Brochu et al. 2010a), reinforcement learning (Brochu et al. 2010b), environmental monitoring (Marchant and Ramos 2012), and automatic machine learning (Snoek et al. 2012). Works related to marketing tend to focus on recommendation systems (Vanchinathan et al. 2014) or A/B testing (Scott 2010, 2015). In these contexts, a selection strategy uses expected values and uncertainties to determine the next *query point*. In the machine learning literature, such a strategy is often referred to as utilizing an acquisition function, $\alpha(\cdot|D_t)$. Gaussian processes can be combined with three different types of such approaches: (i) improvement-based policies, (ii) optimistic policies, and (iii) information-based policies. *Improvement-based* policies measure the probability that querying a given point yields an outcome superior to the best-known solution (Jones et al. 1998). *Optimistic* policies are the upper–confidence bound policies described previously. Srinivas et al. (2012) combine such policies with GPs and establish tight theoretical performance bounds for several covariance functions. Finally, *information-based* policies include the Thompson sampling heuristic discussed in Section 2.2. When combined with a Gaussian process, Thompson sampling consists of sampling from that process and then choosing the maximum of the sample: $x_{t+1} = \text{argmax}_{x \in \mathbb{A}} \alpha(x|D_t)$, where $\mathbb{A}$ is the set of points that are available for sampling.

Figure 3: Thompson Sampling as an Acquisition Function for a Gaussian Process

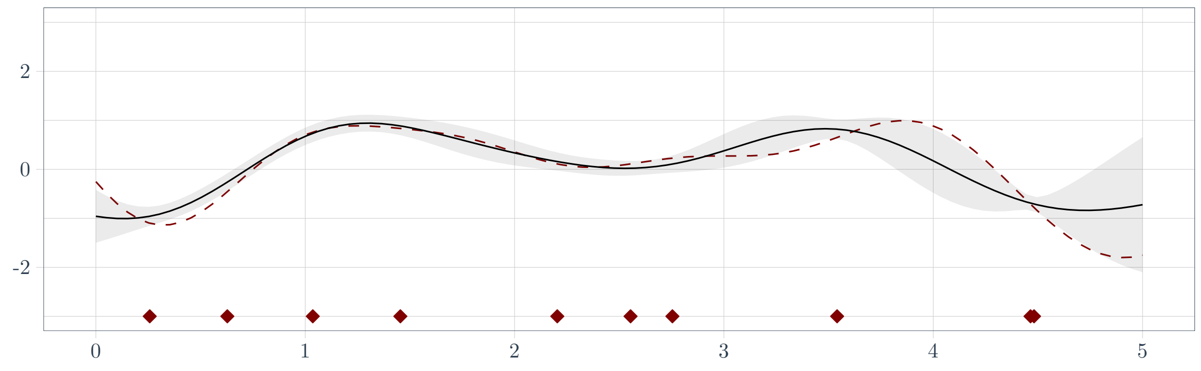


Figure 3 illustrates how TS can be used as an acquisition function for the GP plotted in Figures 1 and 2. As in any sequential decision problem, we must now choose the next point on the horizontal axis that will locate the highest point (of the true function) on the vertical axis. This figure's dashed red line represents a sample, from the posterior, that is determined in light of both the prior and the set of all previous sample points. We remark that the acquisition function, when sampled from the posterior, is likely to deviate from the mean function (the figure's plotted black line) in underexplored areas. Such a sample is therefore expected to differ significantly from the mean, which encourages exploration. The TS algorithm entails always choosing the *maximum* of the sample obtained from the posterior.

3 Gaussian Process Thompson Sampling for Dynamic Pricing

In this section, we describe our dynamic pricing model, introduce its components, and critically review its assumptions. The Gaussian process Thompson sampling (GP-TS) algorithm developed here improves on the shortcomings of previous work in this field. We shall elaborate on the advantages of combining Gaussian processes with Thompson sampling via comparison of the GP-TS algorithm with extant approaches to solving multi-armed bandit problems.

3.1 Dynamic Pricing with Learning and Limited Inventory

Problem Formulation. We consider the case of a retailer with a network revenue management problem involving multiple products, indexed by $i = 1, \ldots, N$, that are sold over a limited sales season comprising T periods indexed by $t = 1, \ldots, T$. Our assumption in this setup is that, for each product, the retailer has no knowledge of the relationship between its unit demand and its price. In other words, the retailer must learn the demand function *over the selling season* to optimize its total revenue. Selling these products consumes M limited resources, which are indexed by $j = 1, \ldots, M$. When products are mapped to resources, the implication is that product i consumes a_{ij} units of resource j; we use I_j to denote the inventory level of that resource. In each time period t, the retailer decides on a price for each product and then offers its products to customers at the prices so determined.

Formally, the retailer can choose from a discrete set $\{p_1, p_k, \ldots, p_K\}$ of k price vectors that each has length N. It follows that $p_{1,k}$ represents the price assigned to the first product in price vector k. Since the retailer chooses price vector $p_{\hat{k}}$ in time period t, we define $p(t) = p_{\hat{k}}$. Next, customers observe the chosen price vector $p(t)$ and make their respective purchase decisions. We assume that the retailer observes the *aggregate* unit demand for product i, or $d_i(p_i(t))$, and that demand for all products can be represented by the *random* variable d_i whereby we assume the parameters of the underlying probability distributions are unknown to the retailer. Besides, we assume the existence of a functional relationship – between price and demand – that need not follow any parametric model. Our final assumption is that of demand independence for different periods; in other words, we suppose that the MAB problem is a myopic one. The retailer receives revenue $\sum_{i=1}^{N} d_i p_i(t)$, after which inventory I_j is reduced by the amount $\sum_{i=1}^{N} d_i a_{ij}(t)$ for each resource j. Any demand not served by $I_i(t)$ is lost; that is, $d_i(t) = \min\{d_i(t), \min_j\{I_j(t)/a_{ij}\}\}$. The retailer sequentially chooses a series of price vectors $p(t)$ with the goal of maximizing the total revenue generated by all products over the entire selling season.

Algorithm. We now propose a method for combining Gaussian processes with Thompson sampling for dynamic pricing under inventory restrictions. Algorithm 2 describes this procedure in pseudo-code. In each time period t, we first estimate GPs for each product individually; thus, we incorporate the price interdependencies that may exist among any products. In the next step, all price vectors K are evaluated by sampling demand $\hat{d}_k(t)$ from the GPs such that $\hat{d}_i k \sim \mathrm{GP}_i(D^i_{t-1})$. Akin to the approach developed by Ferreira et al. (2018), for example, these demand samples $\hat{d}(t)$ – which cover all products and price vectors – serve as inputs to a linear program for optimizing overall revenue in the presence of inventory constraints.

The main advantage of this problem formulation is its flexibility: all kinds of linear relations between resources M and products N can be incorporated, which means that the model is suitable for network revenue optimization tasks (Gallego and van Ryzin 1994). The inventory constraints of resource j, denoted c_j, are updated during each time period such that $c_j(t) = \frac{I_j(t-1)}{T-t+1}$ – a fraction that represents the available inventory of resource j per period for the remaining time periods. The optimal solution to $\mathrm{LP}(\hat{d}(t), c(t))$ is a distribution over price vectors K, which are constrained only by the condition $\sum_{k=1}^{K}(x_k) <= 1$ (for a discussion of this issue, see Ferreira et al. 2018). Hence offering the optimal price distribution entails choosing a price vector $p_k(t)$ with the optimal

Algorithm 2: Gaussian Process Thompson Sampling (GP-TS)

```
for $t = 1, \dots, T$ do
    for $i = 1, \dots, N$ do
        1. Estimate GPs: Compute $\mathrm{GP}_i(D^i_{t-1})$, where
           $D^i_{t-1} = \{[p(1), d_i(1)], \dots, [p(t-1), d_i(t-1)]\}$.
    2. Sample demand from GPs: Sample $\hat{d}(t) = \{\hat{d}_{ik}(t)\}$ for each price vector $k$ by
       sampling demand for product $i$ from $\mathrm{GP}_i$ at $p_{ik}$.
    3. Optimize prices: Solve the linear program
       $$\begin{aligned} \mathrm{LP}(\hat{d}(t), c(t))\text{:}\ \max_x \quad & \sum_{k=1}^{K}\left(\sum_{i=1}^{N} p_{ik}\hat{d}_{ik}(t)\right)x_k \\ \text{subject to} \quad & \sum_{k=1}^{K}\left(\sum_{i=1}^{N} a_{ik}\hat{d}_{ik}(t)\right)x_k <= c_j(t)\ \forall j \in M, \\ & \sum_{k=1}^{K}(x_k) <= 1, \\ & x_k >= 0\ \forall k \in K. \end{aligned} \tag{6}$$
       The optimal solution to $\mathrm{LP}(\hat{d}(t))$ is $x(t) = (x_1(t), \dots, x_K(t))$.
    4. Offer optimal price distribution: Offer $p_k(t)$ with probability $x_k(t)$ and $p_\infty$ with
       probability $1 - \sum_{k=1}^{K}(x_k)$.
    5. Update history: Observe demand $d(t)$ and then update $D_t = D_{t-1} \cup \{p(t), d(t)\}$.
```

probability $x_k(t)$. So that we can simplify this constraint from $\sum_{k=1}^{K}(x_k) = 1$, we introduce a shut-off price p_∞ that is offered with probability $1 - \sum_{k=1}^{K}(x_k)$ and guarantees zero demand. In the final step, the retailer observes demand $d(t)$ and then updates the history D_t with both $d(t)$ and the offered price vector $p(t)$.

Discussion. In common with many other publications in the field of dynamic pricing and learning, our formulation relates to the multi-armed bandit problem. Following the terminology of this wider field of research in decision theory, the "arms" in our setting amount to price choices while observed demand corresponds to "rewards". A significant disadvantage of many works in this area is that they assume the arms (and hence, our case, the prices) to be independent. Recall from Section 2.2 that the regret bound $O\big(\sqrt{TK \log K}\big)$ derived by Ferreira et al. (2018) depends on the number K of possible price vectors, and their bound is close to the general bound $\sqrt{TK}$ derived by Bubeck and Cesa-Bianchi (2012). These results are impressive, but the model assumptions made by those authors are unrealistic in dynamic pricing settings. First of all, considering all price vectors to be independent requires that one ignores any functional price–demand relationship. In the second place, the number K of price vectors to explore will grow exponentially for problems with many products. Yet it is no more realistic to assume continuous prices (as in Araman and Caldentey 2009; Besbes and Zeevi 2009; Farias and van Roy 2010) and/or a simple linear relationship (as in Keskin and Zeevi 2014; cf. Talluri and van Ryzin 2006).

In contrast, we consider discrete price sets and use Gaussian processes to model this relationship in a nonparametric way. More specifically, we assume that the MAB problem of interest has dependent arms and so allows for dependence among prices – and thereby accelerates learning as the model translates, in each period, the knowledge gained from exploring one price into similar prices. For GPs in combination with TS, it is known that regret – depending on the numbers N and K of (respectively) products and price vectors – is only weakly bounded by $O\big(\sqrt{\log T^{N+1} \log K}\big)$ (Srinivas et al. 2012; Russo and van Roy 2014). This property of GPs encapsulates the advantage of Algorithm 2: instead of exploring all price vectors K individually, it is bounded by *logged* factors related to N and K. As a result, the algorithm scales more efficiently as the number of products increases. Thus we incorporate price–demand interdependence in a non-parametric and extremely flexible way, which allows us to maintain the more realistic (and favorable) assumption of discrete price sets. Moreover, our algorithm could also be applied in a continuous price setting if we increase the number of feasible prices so that the intervals between them become infinitesimal.

3.2 GP-TS in Contextual Settings

The setting considered so far may not be realistic – that is, because it supposes demand to be independent of any external factor *other* than price. We, therefore, render the model formulation more adaptable to conditions in the real world by introducing contextual information, by which we mean exogenous information available to the retailer before its price decision. Examples of contextual information in this setting include seasonally changing demand, changes in customer behavior, and competitors' actions. In the MAB literature, incorporating such information into the learning model yields a "contextual bandit problem". In what follows, we describe an adaptation of Algorithm 2 for such problems.

Problem Formulation. Here the contextual setting extends the previous definition of our problem by including a *context vector* $\zeta \in Z$, where Z represents a discrete feature space. At the start of each time period t, the retailer observes the current context vector $\zeta(t)$ and incorporates that information into the learning process before offering a price vector $p(t)$. For any context ζ and price vector p_k, we assume that demand is randomly distributed according to a known distribution but with unknown parameters. Each context vector ζ has a length N that indicates the particular context z_i of each product i; note that ζ_i can itself be a vector describing the specific context of i. So given the context of product i in period t, or $\zeta_i(t)$, we use $d_{ik}(t|\zeta_i(t))$ to denote the estimated mean demand of that product under price vector p_k. Suppose, for example, that the context $\zeta_i(t)$ a binary variable that is set to 1 when a product i (say, a sunscreen lotion) is in high seasonal demand and is set to 0 otherwise. The presence of such information complicates the task of learning because then the retailer, when given any price vector, must learn the demand distribution separately for each context. We now propose to extend our GP model by including contextual information as a new dimension of the Gaussian process.

Algorithm. Our contextual Algorithm 3, to which we refer as "GP-TS Contextual", is based on Algorithm 2 but features several distinctions. In Step 1, for instance, the GPs are estimated on data that incorporate not only past demand and past price offers but also past contextual information. Thus a step has been added – before sampling from the GP – during which the retailer observes the current context vector $\zeta(t)$. Since the resulting GP is of dimension $|\zeta_i| + N + 1$, it follows that the sampling in Step 3 occurs under price vector p_{ik} and in context $\zeta_i(t)$. However, there are no changes in either Step 4 (the optimization problem) or Step 5 (offering the optimal price distribution). Finally, updating the price history now includes storing the current context vector $\zeta(t)$.

Algorithm 3: Contextual Gaussian Process Thompson Sampling (GP-TS Contextual)

```
for $t = 1, \dots, T$ do
    for $i = 1, \dots, N$ do
        1. Estimate GPs: Compute $\mathrm{GP}_i(D^i_{t-1})$, where
           $D^i_{t-1} = \{[p(1), \zeta(1), d_i(1)], \dots, [p(t-1), \zeta(t-1), d_i(t-1)]\}$.
    2. Observe context: Observe the current context vector $\zeta(t)$.
    3. Sample demand from GPs: Sample $\hat{d}(t) = \{\hat{d}_{ik}(t|\zeta_i(t))\}$ for each price vector $k$ by
       sampling the demand for product $i$ from $\mathrm{GP}_i$ under $p_{ik}$ and in context $\zeta_i(t)$.
    4. Optimize prices: Solve the linear program
       $$\begin{aligned} \mathrm{LP}(\hat{d}(t|\zeta(t)), c(t)):\ \max_x \quad & \sum_{k=1}^{K}\Big(\sum_{i=1}^{N} p_{ik}\hat{d}_{ik}(t|\zeta(t))\Big)x_k \\ \text{subject to} \quad & \sum_{k=1}^{K}\Big(\sum_{i=1}^{N} a_{ik}\hat{d}_{ik}(t|\zeta(t))\Big)x_k <= c_j(t)\ \forall j \in [M], \\ & \sum_{k=1}^{K}(x_k) <= 1, \\ & x_k >= 0\ \forall k \in [K]. \end{aligned} \qquad (7)$$
    The optimal solution to $\mathrm{LP}(\hat{d}(t|\zeta(t)))$ is $x(t) = (x_1(t), \dots, x_K(t))$.
    5. Offer optimal price distribution: Offer $p_k(t)$ with probability $x_k(t)$ and $p_\infty$ with
       probability $1 - \sum_{k=1}^{K}(x_k)$.
    6. Update history: Observe demand $d(t)$ and then update $D_t = D_{t-1} \cup \{p(t), \zeta(t), d(t)\}$.
```

Our next task is to compare the performance of this algorithm – on three simulated problems – with the performance of some suitable alternatives.

4 Numerical Experiments

In this section, we test Algorithm 2 and compare it to relevant benchmark algorithms across three different problem settings. We start by considering a single-product setting with Gaussian distributed demand and inventory constraints, after which we extend that first experiment to a multi-product setting. Finally, we test our algorithm in settings that involve contextual factors as well as multiple products.

Algorithms. We implement Algorithm 2 and compare it with the three dynamic pricing algorithms best suited for this setting: the TS-Fixed (Ferreira et al. 2018), the TS-Update (Ferreira et al. 2018), and the BZ (Besbes and Zeevi 2012). Our GP-TS algorithm is a direct implementation of Algorithm 2.

The first algorithm proposed by Ferreira et al. (2018) is the TS-Fixed, which combines Thompson sampling with a single (and static) inventory optimization step based on the theoretically available inventory for each time period. In each period, pricing decisions are optimized based on sampled demand. The TS-Update is the second algorithm developed by Ferreira et al. (2018); in each round, it combines Thompson sampling with inventory optimization based on current inventory levels. Because we assume that the distribution of demand is Gaussian, we implement Thompson sampling for TS-Fixed and TS-Update while using a Gaussian distribution as its prior. We also assume that the Gaussian distribution's variance is known a priori, which facilitates updating the prior distribution. The BZ algorithm is the first one proposed by Besbes and Zeevi (2012). It first divides the selling season into exploration and exploitation phases; then this algorithm explores all prices in the former phase and, in the latter, exploits the knowledge so acquired during the rest of the selling season. We follow the authors in setting the parameter τ, which is used to calibrate the division of the selling season into these two phases, to $T^{2/3}$.

Demand Function. To describe how the price offered is related to observed product demand, we assume the following functional relationship that depends on three exogenous parameters (a, b, and c), the price, p, and the current product, i:

$$d(p) = a + b \times (1 + i) \times p - c \times p^2 + \gamma. \tag{8}$$

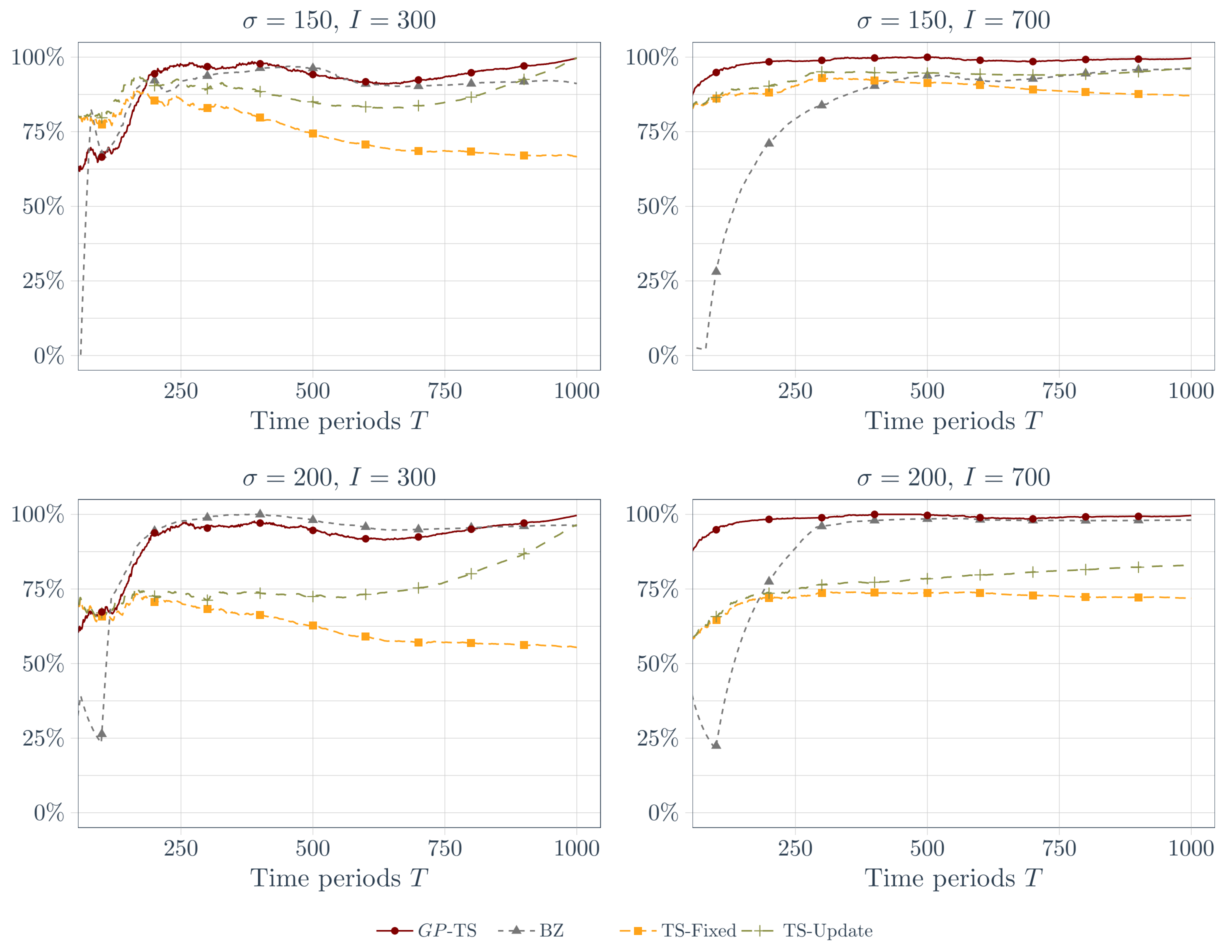
This demand function results in a price sensitivity that is increasing in the price, and the parameter γ reflects context-dependent shifts in the demand function. In our numerical experiment we put $a = 3000$, $b = 4$, $c = 10$, and $\gamma = 100$ as demand parameters. Here i represents the current product's index and reflects this convention: the first product's demand function is parameterized with $c = 20$, the second product with $c = 30$, and so on. To incorporate stochastic variation, we add a Gaussian noise term with variance σ.

4.1 Single-Product Setting

In this first simulation, we consider a retailer that markets a single product in a finite selling season. In this case, the retailer need only consider the product's stock of inventory as a resource. We assume that inventory scales as a function of the length of the selling season, so that $I = \alpha T$, and we consider two settings: one with $\alpha = 300$ and one with $\alpha = 700$. The retailer assesses the effect of pricing decisions on the Gaussian-distributed quantity of goods sold in each period. Thus the demand of customers is Gaussian distributed; its mean is a function of the price, and each price exhibits the same fixed variance, σ. We consider two values of the demand's variance σ – namely, $\sigma = 150$ and $\sigma = 200$. Finally, we assume a set of five possible product prices that range from 1 to 100; in particular, $K = \{1.00, 25.75, 50.50, 75.25, 100.00\}$.

Results. Figure 4 presents the results of 200 simulations over a selling season consisting of 1,000 time periods, where each graph shows the performance of all algorithms over the entire season. The most suitable performance measures for demand learning problems are the revenue generated per round as a percentage of that generated under the optimal price offering – that is, the optimal offering determined with prior knowledge of the demand function. Thus a performance of 100% implies that the revenue achieved by the algorithm is equal to the optimal revenue when the demand function is known a priori. The key performance measure of a learning algorithm is how *rapidly* the algorithm converges to 100%. Results are more robust when such performance per time period is averaged over all 200 simulations. Each graph depicts a different combination of values for inventory I and variance σ.

Figure 4: Performance Comparison of Dynamic Pricing Algorithms – Single-Product Example



We find that GP-TS is the best-performing algorithm – over all variances and inventory levels – in that it reaches the optimal performance of 100% the most quickly. This algorithm performs exceptionally well when inventory constraints are less binding. We remark that, in a simple setting such as this (with only five prices to explore), the benchmark algorithms can be expected to perform equally (or nearly as) well because the GP-TS's ability to learn functional relationship is most advantageous when there is a large number K of price vectors to explore. The second-best algorithm is BZ, whose performance seems to suffer only in the case of a small variance and less restrictive inventory. The TS-Update algorithm is only the third best, as its performance is drastically compromised when σ is large. Finally, TS-Fixed performs the worst of all tested algorithms. The reason is that, since this algorithm does not update its inventory optimization, it cannot loosen constraints when the algorithm's proposed strategy yields a suboptimal amount of sold inventory during the early exploration rounds; this hypothesis is confirmed by the inventory efficiency reported in the Appendix. Encouraged by these results for a "simple" single-product setting, we shall next investigate a more elaborate setting that involves multiple products.

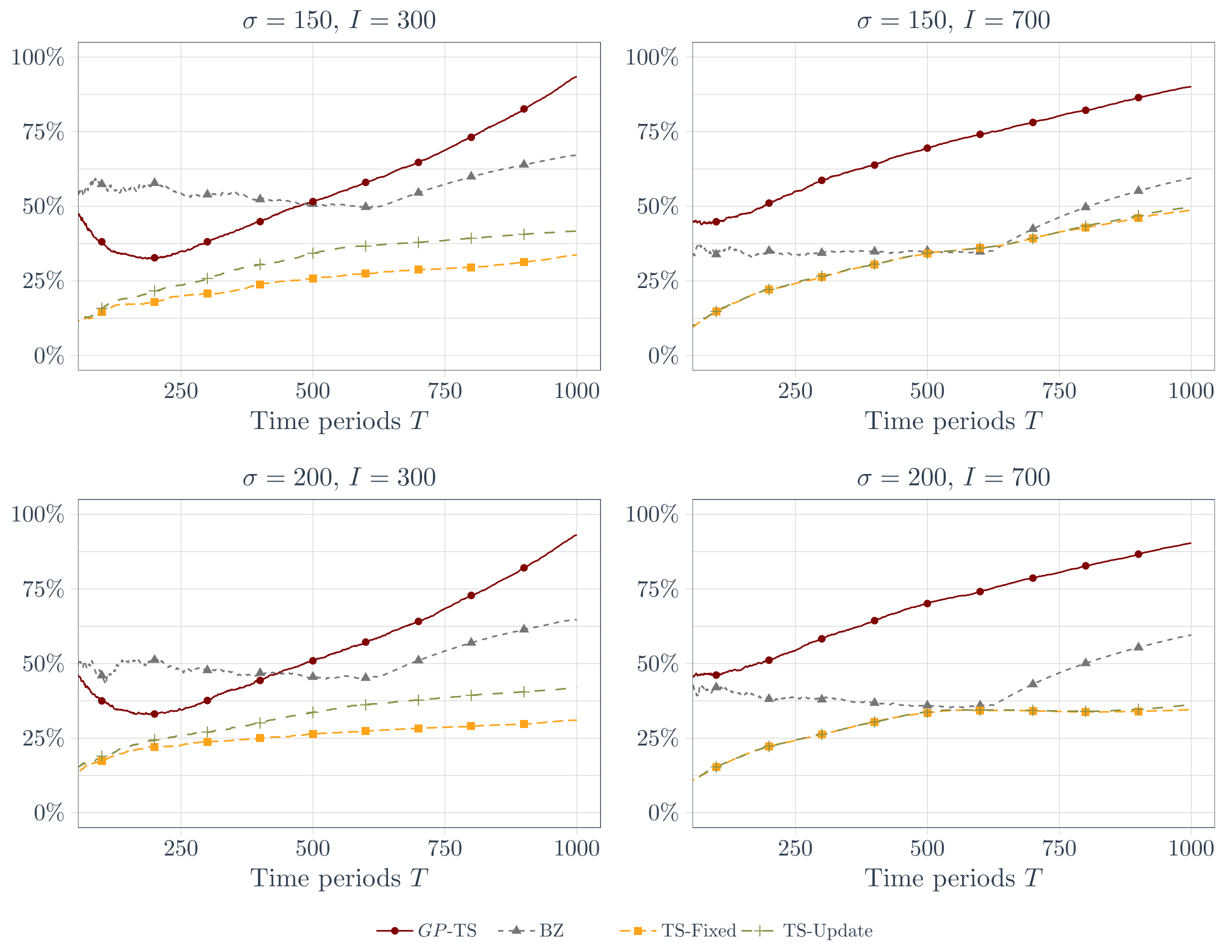
4.2 Multi-Product Setting

In our second experiment, we extend the previous experiment to multiple products. Suppose a retailer sells four products, each with limited inventory. We continue to assume that inventory scales with the length of the selling season, and we consider the same inventories as before except that now $\alpha = [300, 300, 300, 300]$ and $\alpha = [700, 700, 700, 700]$ for each product. Feasible price combinations form a grid of all possible combinations of the four products' prices. For each product, we again allow five different prices that are evenly distributed on the interval from 1 to 100; thus we consider the grid $K = \{[1.00, 1.00, 1.00, 1.00], \ldots, [100.00, 100.00, 100.00, 100.00]\}$. Customer demand is still assumed to follow the same demand distribution with Gaussian distributed noise. For values of the variance we again assume $\sigma = 150$ and $\sigma = 200$.

Results. Just as in the single-product example, we plot the performance of each algorithm over the selling season in four graphs that represent the possible combinations of our assumed parameter values for inventory and variance. Also, we continue to measure performance by comparing the algorithms' reward with that obtained under an optimal pricing strategy in which the demand function was known a priori. All reported results are based on 200 simulations over 1,000 time periods.

As compared with the single-product setting, all four algorithms take longer to learn the optimal price during the selling season; see Figure 5. Only GP-TS approaches the optimal price distribution. This outcome is not surprising given that the number of price vectors to explore grows exponentially

Figure 5: Performance Comparison of Dynamic Pricing Algorithms – Multi-Product Example



with the number of products. The advantages of GP-TS are much more apparent in this setting, and they reflect this algorithm's exploring the price–demand relationship for each product separately – in view of an underlying, functional relationship – while the other three algorithms must explore each price vector.

Among those three others, BZ performs better than either of the Thompson sampling algorithms. And reprising its behavior in the single-product case, GP-TS is more robust (than is any other algorithm tested here) to settings with higher demand volatility and again benefits from looser inventory restrictions. It is noteworthy that neither TS-Update nor TS-Fixed reach even half of the optimal revenue per time period during the first 1,000 periods. This result is not surprising when one considers that TS explores all possible price vectors randomly – in contrast to BZ, for instance, under which the length of the exploration period restricts exploration. In the Appendix, we provide additional evidence for these results: in terms of optimal revenue achieved, GP-TS exhibits performance that is 50% to 60% better than the other algorithms.

4.3 Contextual Setting

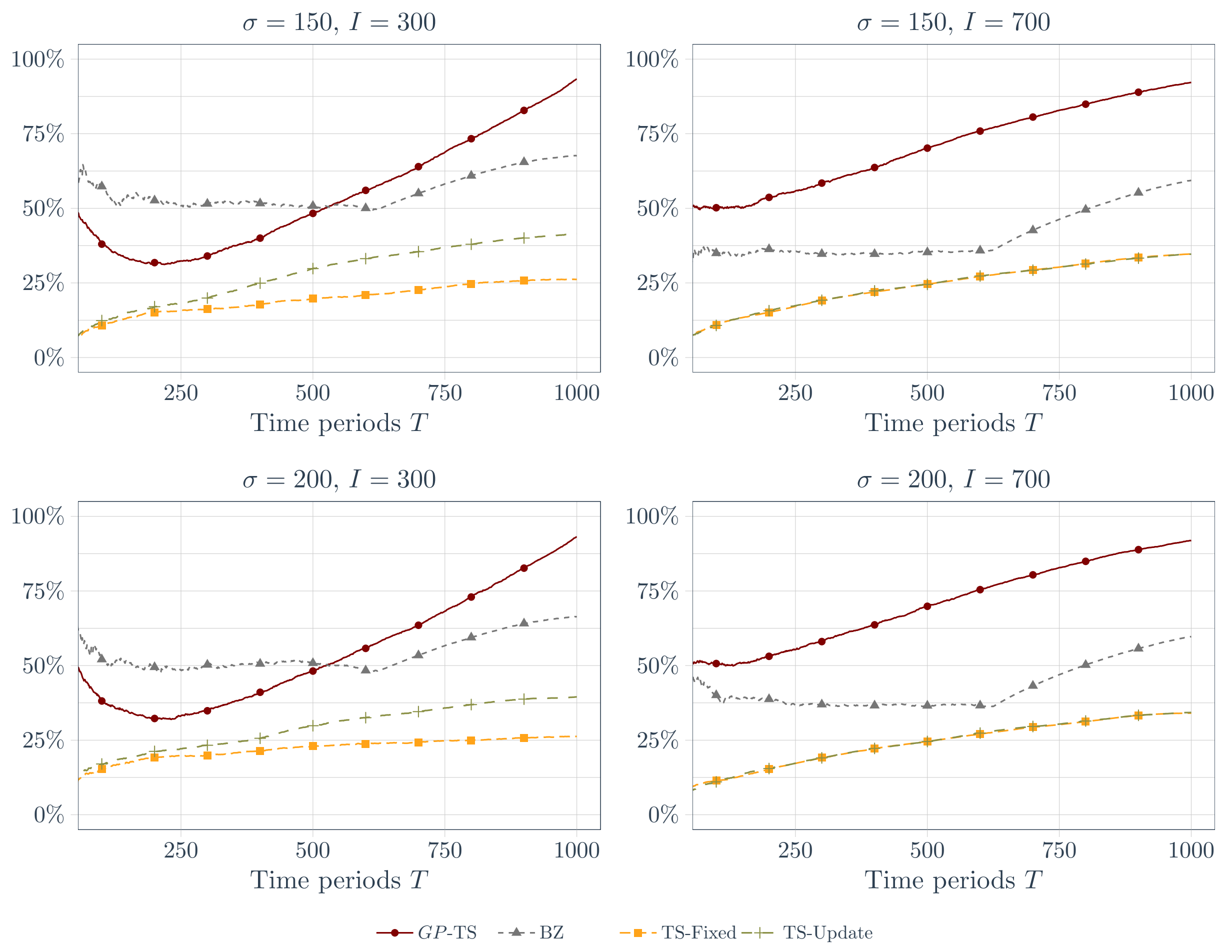
In both of the settings just described, we abstracted from the presence of contextual information. To test Algorithm 3, we adapted the multi-product setting to include a binary contextual vector. Hence the demand function in this setup changes drastically depending on a randomly assigned binary context, which serves to emulate volatile exogenous factors that affect demand. One example of such a factor is the weather; thus, for instance, the context could be "bad weather" that leads to greater demand for products offered by online retailers.

Problem Setting. We define a contextual vector ζ, where $|\zeta| = N$ and $\zeta_i \in \{0,1\}$ for all $i \in \zeta$. This context vector is sampled randomly at the beginning of each period. When $\zeta(t) = 1$, the demand function is changed by the factor γ and so results in a different optimal solution to the pricing problem. It follows that the given algorithm must learn the optimal solution while considering all manifestations of the context vector. Our simulation implements the simple case in which ζ is not only binary but also equal for all products in a given time period: $\zeta_i(t) = \zeta_j(t)$ for all $i, j \in \zeta(t)$.

Results. Because we are modifying the aforementioned multi-product setting, here we employ procedures similar to those used in that setting to conduct the simulations and analyze each algorithm's performance. Figure 6 presents our results for each of the four *contextual* multi-product settings.

We can see from these graphs that, in this setting as well, the GP-TS algorithm is the best at learning; in fact, its performance is as good as – or even slightly better than – in the case

Figure 6: Performance Comparison of Dynamic Pricing Algorithms – Contextual Setting



of a *noncontextual* multi-product setting. Furthermore, GP-TS is the only one of our four tested algorithms that converges to the optimal price distribution. For not only must the other algorithms explore each price vector individually (as in the previous experiment), now they also must learn demand for each price vector *and* context. That is, knowledge of a price vector accrues from past offerings made in combination with the same context vector. Of the other algorithms, BZ remains the best. Its performance in the first half of the selling season is superior even to that of GP-TS, but thereafter BZ fails to converge to the optimal pricing strategy. This deficiency might reflect that the BZ algorithm does not update inventory. Both TS-Update and TS-Fixed are characterized by slow learning. See the Appendix for an overview of all these results.

5 Extensions

The previous experiments showcase the performance of our algorithms for problems in which the distribution of demand is assumed to be Gaussian. In this section, we instead consider two other demand distributions that are commonly considered for dynamic pricing problems, after which we discuss how our GP-TS algorithm can be adapted to suit such settings.

5.1 GP-TS with Bernoulli Demand

Many real-world dynamic pricing situations involve binary decision problems. In the online retailing sector, for instance, pricing decisions can be made for each customer individually. In that case, setting the optimal price is equivalent to finding the price associated with the highest *purchase probability* – in other words, to estimating the parameters of a Bernoulli distribution. Because Algorithm 2 is designed for Gaussian probabilities, it is not applicable in Bernoulli demand settings.

However, GPs can be adapted to any non-Gaussian likelihood via link functions. In the case of Bernoulli demand, the logit function can serve as a link that is used to project the Gaussian likelihood onto a 0–1 interval. The downside of adopting this approach is that one *loses* the closed-form solution delivered by GP regression – that is because a conjugate prior is not available. However, two strategies often used to compute the *posterior* distribution are the Laplace approximation and Monte Carlo Markov chain (MCMC) estimation (Rasmussen and Williams 2008).

Extending the GP-TS algorithm to Bernoulli demand settings would enable its use in large-scale e-commerce applications. Because solutions derived using the MCMC procedure take longer to compute, a useful modification might be to apply a "batch" learning approach whereby learning occurs periodically and not after each customer arrival. Apart from pricing problems, this model setting applies to many other decisions faced in online retailing (e.g., product recommendations).

5.2 GP-TS with Poisson Demand

Another frequently observed variation is demand that exhibits a Poisson distribution. In terms of modeling demand, the advantages of a Poisson distribution are apparent: this distribution is well suited to describing low counts of (e.g., weekly) product sales of a particular product among an extensive collection of substitute products. Hence it is hardly surprising that a Poisson process is often used in the dynamic pricing literature (Besbes and Zeevi 2012; Ferreira et al. 2018). Adapting GP-TS to Poisson demand follows an approach similar to the one described previously for the Bernoulli demand setting. Diggle et al. (1998) estimate the posterior distribution by combining the MCMC approach with an exponential link function. As before, however, the drawback of using MCMC is the computational effort needed to re-estimate the Gaussian process.

6 Conclusion

We have shown that GPs are a powerful tool for capturing, in a flexible way, price–demand relationships in dynamic pricing problems. In this section, we summarize the implications of results obtained here, identify some limitations of our approach, and elaborate on related research opportunities.

6.1 Implications

We study finite-inventory, dynamic pricing problems in which a retailer must learn demand throughout the selling season and simultaneously maximize its revenue. The literature in this field, which is closely related to the broader domain of MAB problems, considers either independent, discrete prices or a parametric demand function and continuous prices. We develop an approach that (i) considers the more realistic case of discrete price sets, (ii) allows for any functional relationship between price and demand, and (iii) incorporates an optimization model that allows for the modeling of network revenue problems.

The algorithm we propose builds on a Gaussian process regression – a learning algorithm that has received considerable research attention over the past decade. In combination with Thompson sampling, which is a powerful heuristic for MAB problems, our algorithm considers functional relationships and inventory constraints while dynamically balancing the exploration–exploitation trade-off. We show that the design of our GP-TS algorithm allows it to scale efficiently with the number of price vectors to explore. This feature contrasts with any existing approach that explores each price vector independently; the reason is that such methods deliver poor results when the number of products increases. In numerical experiments undertaken in both the single- and multi-product settings, we show that GP-TS is superior to other tested algorithms, which consider all prices independently. Perhaps the most notable feature of our algorithm is that, in settings with multiple products, it benefits from being able to learn functional relationships. Also, GP-TS can manage available inventory more efficiently than is possible with other algorithms. These advantageous properties should shape research on revenue management and also have an impact on other operations problems that can be viewed from a MAB perspective.

6.2 Limitations

Our approach to dynamic pricing by way of Gaussian processes does have a few limitations. First, estimating Gaussian processes is computationally expensive and therefore infeasible for larger samples or in the context of online decisions that must be made rapidly (Shahriari et al. 2016). This shortcoming might be overcome via the adoption of a batch learning approach in which the covariance matrix of the Gaussian process is updated only after pre-specifed time intervals or by using "sparse" Gaussian processes (Lázaro-Gredilla et al. 2010). The latter reduces computational effort by estimating the process for a pre-defined number of spectral points that best represent all samples. Another limitation is that GP-TS performance suffers in higher dimensions because this algorithm is linearly dependent on the number of dimensions (Brochu et al. 2010b). Approaches that tackle this problem include combining GPs with relevance detection over those dimensions and replacing GPs with so-called random forests, which are well suited to handle high-dimensional problems. However, such approaches are largely beyond the realm of MAB research and are more appropriately classified as an aspect of machine learning research.

6.3 Future Research

The findings presented in this paper suggest several potential avenues for future research. First, variants of Gaussian process models that are capable of overcoming the limitations just described could be applied to the dynamic pricing and learning problem. Interested scholars could, for instance, make use of recent developments in the machine learning literature mentioned above that allow for problems in higher dimensions or with a large number of samples.

Second, the flexibility of modern Bayesian methods could well inspire a re-assessment of the assumptions underlying commonly studied dynamic pricing and learning problems. Especially useful would be a detailed investigation of how best to relax the limiting assumption that customer behavior is myopic. Research on this topic would also be related to a broader literature on the *drifting* bandit problem, in which the behavior of agents changes over time. Another possibility is that the expectations of agents who know about the history of offered prices could be modeled as a variant of the contextual bandit problem.

Third, researchers could apply our approach to continuous pricing problems and thus compare its predictive accuracy with that of algorithms specifically developed for such a setting. Finally, the effectiveness of demand learning algorithms that incorporate inventory constraints could be tested in practice. In general, the increased interest in Bayesian optimization methods and, in particular, Gaussian processes should have an impact on the methodology applied in the operations research and quantitative marketing communities.

References

Agarwal, D., 2013, "Computational advertising." In *Proceedings of the 22nd ACM international conference on information & knowledge management - CIKM '13*, ed. by Q. He, A. Iyengar, W. Nejdl, J. Pei, and R. Rastogi, 1585–1586, New York, New York, USA: ACM Press.

Agarwal, D., B. Long, J. Traupman, D. Xin, and L. Zhang, 2014, "Laser: a scalable response prediction platform for online advertising." In *Proceedings of the 7th ACM international conference on Web search and data mining - WSDM '14*, ed. by B. Carterette, F. Diaz, C. Castillo, and D. Metzler, 173–182, New York, New York, USA: ACM Press.

Agrawal, S., and N. Goyal, 2013, "Further Optimal Regret Bounds for Thompson Sampling." In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, ed. by C. M. Carvalho and P. Ravikumar, 31:99–107, Proceedings of Machine Learning Research, Scottsdale, Arizona, USA: PMLR.

Araman, V. F., and R. Caldentey, 2009, "Dynamic Pricing for Nonperishable Products with Demand Learning," *Operations Research* 57 (5): 1169–1188.

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

Aviv, Y., and G. Vulcano, 2012, "Dynamic list pricing." In *The Oxford handbook of pricing management.*

Badanidiyuru, A., R. Kleinberg, and A. Slivkins, 2013, "Bandits with Knapsacks." In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 207–216, IEEE.

Bai, A., F. Wu, and X. Chen, 2013, "Bayesian Mixture Modelling and Inference based Thompson Sampling in Monte-Carlo Tree Search." In *Advances in Neural Information Processing Systems 26*, ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, 1646–1654, Curran Associates, Inc.

Besbes, O., and A. Zeevi, 2012, "Blind Network Revenue Management," *Operations Research* 60 (6): 1537–1550.

— . 2009, "Dynamic Pricing Without Knowing the Demand Function: Risk Bounds and Near-Optimal Algorithms," *Operations Research* 57 (6): 1407–1420.

Brochu, E., T. Brochu, and N. de Freitas, 2010a, "A Bayesian Interactive Optimization Approach to Procedural Animation Design." In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 103–112, SCA '10, Goslar Germany, Germany: Eurographics Association.

Brochu, E., V. M. Cora, and N. de Freitas, 2010b, *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning.*

Bubeck, S., and N. Cesa-Bianchi, 2012, “Regret analysis of stochastic and nonstochastic multi-armed bandit problems,” *Foundations and Trends in Machine Learning* 5 (1): 1–122.

Chapelle, O., and L. Li, 2011, “An Empirical Evaluation of Thompson Sampling.” In *Advances in Neural Information Processing Systems 24*, ed. by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, 2249–2257, Curran Associates, Inc.

Chen, Q., S. Jasin, and I. Duenyas, 2014, “Adaptive Parametric and Nonparametric Multi-Product Pricing via Self-Adjusting Controls,” *SSRN Electronic Journal.*

den Boer, A. V., 2015, “Dynamic pricing and learning: Historical origins, current research, and new directions,” *Surveys in Operations Research and Management Science* 20 (1): 1–18.

Diggle, P. J., J. A. Tawn, and R. A. Moyeed, 1998, “Model-based geostatistics,” *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 47 (3): 299–350.

eMarketer, 2018, *Retail e-commerce sales worldwide from 2014 to 2021 (in billion U.S. dollars): Retrieved from statista.com.*

Farias, V. F., and B. van Roy, 2010, “Dynamic Pricing with a Prior on Market Response,” *Operations Research* 58 (1): 16–29.

Ferreira, K. J., B. H. A. Lee, and D. Simchi-Levi, 2016, “Analytics for an Online Retailer: Demand Forecasting and Price Optimization,” *Manufacturing & Service Operations Management* 18 (1): 69–88.

Ferreira, K. J., D. Simchi-Levi, and H. Wang, 2018, “Online Network Revenue Management Using Thompson Sampling,” *Operations Research* 66 (6): 1586–1602.

Gallego, G., and G. van Ryzin, 1994, “Optimal Dynamic Pricing of Inventories with Stochastic Demand over Finite Horizons,” *Management Science* 40 (8): 999–1020.

Graepel, T., J. Q. Candela, T. Borchert, and R. Herbrich, 2010, “Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft’s bing search engine.” In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, ed. by J. Fürnkranz and T. Joachims, 13–20, Haifa, Israel: Omnipress.

Hill, D. N., H. Nassif, Y. Liu, A. Iyer, and S. Vishwanathan, 2017, “An Efficient Bandit Algorithm for Realtime Multivariate Optimization.” In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining - KDD '17, ed. by S. Matwin, S. Yu, and F. Farooq, 1813–1821, New York, New York, USA: ACM Press.

Jones, D. R., M. Schonlau, and W. J. Welch, 1998, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization* 13 (4): 455–492.

Kandasamy, K., A. Krishnamurthy, J. Schneider, and B. Poczos, 2018, "Parallelised Bayesian Optimisation via Thompson Sampling." In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, ed. by A. Storkey and F. Perez-Cruz, 84:133–142, Proceedings of Machine Learning Research, Playa Blanca, Lanzarote, Canary Islands: PMLR.

Kaufmann, E., N. Korda, and R. Munos, 2012, "Thompson Sampling: An Asymptotically Optimal Finite-Time Analysis." In *Algorithmic Learning Theory*, ed. by N. H. Bshouty, G. Stoltz, N. Vayatis, and T. Zeugmann, 199–213, Berlin, Heidelberg: Springer Berlin Heidelberg.

Kawale, J., H. H. Bui, B. Kveton, L. Tran-Thanh, and S. Chawla, 2015, "Efficient Thompson Sampling for Online Matrix-Factorization Recommendation." In *Advances in Neural Information Processing Systems 28*, ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, 1297–1305, Curran Associates, Inc.

Keskin, N. B., and A. Zeevi, 2014, "Dynamic pricing with an unknown demand model: Asymptotically optimal semi-myopic policies," *Operations Research* 62 (5): 1142–1167.

Lázaro-Gredilla, M., J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, 2010, "Sparse Spectrum Gaussian Process Regression," *Journal of Machine Learning Research* 11 (Jun): 1865–1881.

Lizotte, D. J., T. Wang, M. H. Bowling, and D. Schuurmans, 2007, "Automatic Gait Optimization with Gaussian Process Regression." In *Proceedings of IJCAI*, 7:944–949.

Lu, X., and B. van Roy, 2017, "Ensemble Sampling." In *Advances in Neural Information Processing Systems 30*, ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 3258–3266, Curran Associates, Inc.

Marchant, R., and F. Ramos, 2012, "Bayesian optimisation for intelligent environmental monitoring." In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2242–2249.

Misra, K., E. M. Schwartz, and J. Abernethy, 2019, "Dynamic Online Pricing with Incomplete Information Using Multiarmed Bandit Experiments," *Marketing Science* 38 (2): 226–252.

Osband, I., D. Russo, Z. Wen, and B. van Roy, 2017, "Deep exploration via randomized value functions," *arXiv preprint arXiv:1703.07608*.

Rasmussen, C. E., and C. K. I. Williams, 2008, *Gaussian processes for machine learning*, 3rd ed., Adaptive computation and machine learning, Cambridge, Mass.: MIT Press.

Russo, D. J., B. van Roy, A. Kazerouni, I. Osband, and Z. Wen, 2018, "A Tutorial on Thompson Sampling," *FNT in Machine Learning (Foundations and Trends in Machine Learning)* 11 (1): 1–96.

Russo, D., and B. van Roy, 2014, "Learning to Optimize via Posterior Sampling," *Mathematics of Operations Research* 39 (4): 1221–1243.

Schwartz, E. M., E. T. Bradlow, and P. S. Fader, 2017, "Customer Acquisition via Display Advertising Using Multi-Armed Bandit Experiments," *Marketing Science* 36 (4): 500–522.

Scott, S. L., 2010, "A modern Bayesian look at the multi-armed bandit," *Applied Stochastic Models in Business and Industry* 26 (6): 639–658.

— . 2015, "Multi-armed bandit experiments in the online service economy," *Applied Stochastic Models in Business and Industry* 31 (1): 37–45.

Shahriari, B., K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, 2016, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proceedings of the IEEE* 104 (1): 148–175.

Snoek, J., H. Larochelle, and R. P. Adams, 2012, "Practical Bayesian Optimization of Machine Learning Algorithms." In *Advances in Neural Information Processing Systems 25*, ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, 2951–2959, Curran Associates, Inc.

Srinivas, N., A. Krause, S. M. Kakade, and M. W. Seeger, 2012, "Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting," *IEEE Transactions on Information Theory* 58 (5): 3250–3265.

Talluri, K. T., and G. J. van Ryzin, 2006, *The theory and practice of revenue management*, vol. 68, Springer Science & Business Media.

Thompson, W. R., 1933, "On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples," *Biometrika* 25 (3/4): 285.

Vanchinathan, H. P., I. Nikolic, F. de Bona, and A. Krause, 2014, "Explore-exploit in top-N recommender systems via Gaussian processes." In *Proceedings of the 8th ACM Conference on Recommender systems - RecSys '14*, ed. by A. Kobsa, M. Zhou, M. Ester, and Y. Koren, 225–232, New York, New York, USA: ACM Press.

Wang, Z., S. Deng, and Y. Ye, 2014, “Close the Gaps: A Learning-While-Doing Algorithm for Single-Product Revenue Management Problems,” *Operations Research* 62 (2): 318–331.

Appendix: Simulation Results Table

Table 1 gives the results of all numerical experiments. In this table, we compare the algorithms across all settings in terms of two metrics: inventory efficiency (IE) and percentage of achieved optimal revenue (OR). Inventory efficiency is computed by dividing total revenue by sold inventory and then comparing that quotient to the optimal revenue per inventory unit; this metric indicates how efficiently the algorithm converted inventory into revenue. The percentage of achieved optimal revenue is defined as the algorithm's predicted total revenue for the entire selling season *divided by* the optimal revenue over that same season. This metric summarizes the algorithm's overall performance.

Table 1: Overview of Numerical Experiment Results

Setting	Algorithm	$I = 300$				$I = 700$			
		$\sigma = 150$		$\sigma = 200$		$\sigma = 150$		$\sigma = 200$	
		OR	IE	OR	IE	OR	IE	OR	IE
Single Product	BZ	.91	.96	.96	.96	.96	.96	.98	.98
	TS-Fixed	.67	.99	.55	.99	.87	.99	.72	.99
	TS-Update	.99	.99	.97	.99	.96	.99	.83	.99
	GP-TS	.99	.99	.99	.99	.99	.99	.99	.99
Multi-Product	BZ	.67	.68	.64	.65	.59	.60	.60	.60
	TS-Fixed	.34	.40	.31	.43	.49	.54	.35	.42
	TS-Update	.42	.42	.42	.43	.50	.55	.36	.43
	GP-TS	.93	.94	.93	.94	.90	.96	.90	.96
Contextual	BZ	.68	.68	.66	.67	.59	.60	.60	.60
	TS-Fixed	.26	.41	.26	.43	.35	.37	.34	.36
	TS-Update	.35	.43	.34	.42	.41	.37	.40	.36
	GP-TS	.93	.94	.93	.94	.92	.97	.92	.96