Dynamic Pricing with Online Data Aggregation and Learning

Gianmarco Genalti gianmarco.genalti@polimi.it
Politecnico di Milano

Marco Mussi marco.mussi@polimi.it
Politecnico di Milano

Alessandro Nuara alessandro.nuara@mlcube.com
ML cube

Nicola Gatti nicola.gatti@polimi.it
Politecnico di Milano

Abstract

In several e-commerce scenarios, pricing long-tail products effectively is a central task for the companies, and there is broad agreement that Artificial Intelligence (AI) will play a prominent role in doing that in the next future. Nevertheless, dealing with long-tail products raises major open technical issues due to data scarcity which preclude the adoption of the mainstream approaches requiring usually a huge amount of data, such as, *e.g.*, deep learning. In this paper, we provide a novel *online learning* algorithm for dynamic pricing that deals with *non-stationary* settings—due to, *e.g.*, the seasonality or adaptive competitors—, and is very *efficient* in terms of the need for data thanks to assumptions—such as, *e.g.*, the monotonicity of the demand curve in the price—that are customarily satisfied in long-tail markets. Furthermore, our dynamic pricing algorithm is paired with a clustering algorithm for the long-tail products which aggregates similar products such that the data of all the products of the same cluster are merged and used to choose their best price. We first evaluate our algorithms in an offline synthetic setting, comparing their performance with the state of the art and showing that our algorithms are more robust and data-efficient in long-tail settings. Subsequently, we evaluate our algorithms in an online setting with more than $8,000$ products, including popular and long-tail, in an A/B test with humans for about two months. The increase of revenue thanks to our algorithms is about 18% for the popular products and about 90% for the long-tail products.

1. Introduction

The long-tail business model is pervasive in e-commerce. In particular, the long tail is a business strategy allowing companies to get a significant profit by selling low volumes of *hard-to-find* items to many customers instead of selling exclusively large volumes of a small portion of *popular* items (Brynjolfsson et al., 2011). The importance of the long tail is well known and investigated in several works since 2004. We mention just a few seminal works, *e.g.*, by Anderson (2004, 2006). On the one hand, dealing effectively with the long tail is technically challenging as data per product are extremely scarce. Most importantly, such a data scarcity precludes the adoption of several Artificial Intelligence (AI) tools of great success, such as, *e.g.*, deep learning, thus leaving the problem of designing suitable tools open. On the other hand, effective long-tail optimization is crucial for a company. Indeed, the revenue from the long tail usually represents a significant portion of the company's revenue (*e.g.*, up to 33%). Furthermore, the competition with other companies on the long tail is weaker than that on the popular products due to the difficulties in optimizing the pricing. Therefore, an effective optimization can lead to a significant increase of revenue for every company.

In this paper, we focus on real-world long-tail scenarios that are usually *non-stationary* due to the *seasonality* and/or competitors' *adaptive* behaviors. We design an *online learning algorithm* for dynamic pricing, namely *DynaLT* (Dynamic pricing for the Long Tail), which updates the estimates on the demand curve sample by sample and makes decisions to balance the customary machine learning trade-off between *exploitation* and *exploration*. We assume that the process to learn is stochastic. Such an assumption is reasonable even in the presence of adaptive competitors since, as we observed in our experimental analysis, the competitors rarely change the prices of long-tail products. Technically speaking, we use historical data to capture seasonality, and we combine them with a sliding window to

forget old data. The main challenges due to the long tail we face are two. The first challenge concerns the design of learning algorithms that are robust and efficient when data are scarce. More precisely, when a small amount of data are available, the observation of a new sample can dramatically change the shape of the estimated demand curve. In this case, robustness is crucial to avoid significant variations of the algorithm outputs. Similarly, data efficiency is of paramount importance in non-stationary settings to effectively track the changes and limit the delay in the learning process. We force the *monotonicity* of the demand curve learned by the algorithm to address this challenge. Remarkably, this assumption commonly holds with the long-tail products and allows better robustness (as new samples cannot dramatically change the shape of the demand curve learned by the algorithm) and data efficiency (as a sample at a given price provides information to many other prices). We force monotonicity by resorting to a specific class of Bernstein polynomials when estimating the demand curve. The second challenge concerns the design of algorithms capable of clustering the products such that every product of the same cluster will be priced with the same policy. In this case, there are two critical issues. The former is that the clustering cannot be based only on transaction data as data are too scarce. The second is that the common approaches assign some long-tail products to a popular product, which may be inefficient in practice due to the different market dynamics. The peculiarity of our clustering algorithm resides in exploiting similarities among products discovered from textual data describing the products, and it provides an explainable clustering by decision-tree approaches.

We first evaluate our algorithms in an offline synthetic setting, comparing their performance with the state-of-the-art and showing that our algorithms are more robust and data-efficient in the long-tail settings, thus supporting the need to adopt monotonicity in practice. Subsequently, we evaluate our algorithms in a real-world online setting with more than 8,000 products, including popular and long-tail, in an A/B test with humans for about two months. Remarkably, in this experiment, we obtain a revenue increase of about 18% for the popular products and 90% for the long-tail products.

2. Application domain and Motivation

Industrial Context Our work has been conducted in collaboration with an Italian e-commerce website selling more than $20,000$ different products composed of non-perishables consumables. Notice that in this case the assumption of monotonicity of the demand curve trivially holds as these products are not luxury, Veblen, or Giffen. The e-commerce website adopts the drop-shipping business model. Thus, it is not subject to warehousing costs and can suggest/recommend many different products to the customers, including long-tail products leading to rare yearly transactions. In particular, 75% of the products provide about 590 KEuros corresponding to about 10% of the total e-commerce turnover, and the number of units sold for these products in 2021 is smaller than 10. Furthermore, about 50% of the products available in the catalog present no order in 2021. In addition to long-tail products, the e-commerce website also sells a small number of popular products with a high turnover, which needs to be priced adequately. To simplify the business processes, the e-commerce website management required the design of a single algorithm to perform pricing on long-tail and popular products coherently. The adoption of a single algorithm for both kinds of products is due to the simplicity in its management and to maintain fairness w.r.t. customers, as argued by Haws and Bearden (2006); Garbarino and Lee (2003). The objective function to maximize is the *total profit*.

Related Works Many works deal with the problem of learning the demand curve and then choosing the best price given the curve estimated by the algorithm.[1] The Multi-Armed Bandit (MABs) problem is a theoretical framework addressing the exploration-exploitation trade-off by providing theoretical guarantees. This framework has been largely employed to tackle the dynamic pricing task. The seminal work resorting to the MAB framework for dynamic pricing is provided by Rothschild (1974). This approach has been extended in multiple directions. First, Kleinberg and Leighton (2003) tackle the problem of dealing with continuous-demand functions by proposing a discretization of the price values that provides theoretical guarantees on the algorithm's regret. Subsequently, Trovò et al. (2015, 2018) and Misra et al. (2019) exploit that, in many practical settings, the demand function is monotonically decreasing in the price to design novel algorithms outperforming the classical MAB policies empirically. We also mention the work by Mueller et al. (2019) which faces the multi-product pricing scenario through a contextual MAB algorithm. This approach has strong theoretical guarantees in stationary and non-stationary environments, though it does not deal specifically with long-tail products. To the best of the authors' knowledge, no work specifically adopts the MAB framework to tackle the long-tail setting.

1. For a comprehensive analysis of the dynamic-pricing literature, refer to Narahari et al. (2005); Bertsimas and Perakis (2006); Den Boer (2015).

3. Problem Formulation

We study the scenario in which an e-commerce website sells a set $\mathcal{J}$ of non-perishable products with unlimited availability. We assume that a textual description and transaction data are available for all the products $j \in \mathcal{J}$ sold in the past. At every time t, we aim to set a percentage margin (from now on, the margin) $m_{jt} \in \mathcal{M}_j$, where $\mathcal{M}_j$ is the finite set of feasible values for the margin of an item $j \in \mathcal{J}$. Such a margin m_{jt} is defined as:

$$m_{jt} := \frac{p_{jt} - c_j}{c_j}, \tag{1}$$

where p_{jt} and c_j are the selling price and the acquisition cost for product j at time t, respectively. Finally, we denote with $v_{jt}(m_{jt})$ the actual number of sales (volumes) for an item j at time t when choosing margin m_{jt}.

The objective function the e-commerce website aims to maximize is the *total profit*. Formally, the maximization problem is:[2]

$$m^*_{jt} = \arg\max_{m_{jt} \in \mathcal{M}_j} f_{jt}(m_{jt}), \tag{2}$$

where:

$$f_{jt}(m_{jt}) := m_{jt}\, c_j\, v_{jt}(m_{jt}). \tag{3}$$

Given a policy π returning at day t a margin value m_{jt} for each product $j \in \mathcal{J}$, we define the pseudo-regret over time $t \in T$ as:

$$R(\pi) := \sum_{t \in T} f_{jt}(m^*_{jt}) - \mathbb{E}\left[\sum_{t \in T} f_{jt}(m_{jt})\right],$$

where $f_{jt}(m^*_{jt})$ is the expected value provided by a clairvoyant algorithm choosing the optimal margin for each product, formally $m^*_{jt} \in \arg\max_{m_{jt} \in \mathcal{M}_j} f_{jt}(m_{jt})$. Intuitively, the notion of regret provides a measure of the cumulative loss of our policy π w.r.t. the (clairvoyant) policy choosing at each time t the optimal margin maximizing $f(\cdot)$. Thus, our goal is the minimization of the pseudo-regret $R(\pi)$, which is equivalent to the task of maximizing of the cash flow margin accumulated over time.

4. Pricing Single Products

To model the demand curve for each product j, we use the transaction data aggregated over a time interval of one week. These data consist in the aggregated average margin $m_{j\tau}$ and amount of units sold $v_{j\tau}$, for every product j and each week τ.

Seasonality Motivated by our seasonality analysis,[3] we factorize the dependence of the volumes on seasonality and margin with two different, independent functions. In particular, we define the *adjusted volumes* $\bar{v}_{j\tau}$ of product j at time τ as $\bar{v}_{j\tau} := v_{j\tau}\, s_{j\tau}$, where $s_{j\tau}$ is a coefficient, independent of the chosen margin, representing the seasonality for product j at time τ. This coefficient is estimated from historical data as discussed in Appendix A.1. The above factorization allows a dramatic reduction of the samples needed to have a stable estimate of the demand curve.

Non-stationary demand In addition to seasonality effects, the market can be non-stationary due to trends (*e.g.*, contractions or expansions) and adaptive behaviors of the competitors. These effects change the dependency of volumes on margin over time. We deal with these kinds of non-stationarity sources by adopting a sliding window that discards outdated data for the estimation of the volumes. More precisely, we use data coming from the last Y years for the estimation of the seasonality coefficients $s_{j\tau}$, while we adopt a sliding window of N weeks for the estimation of the adjusted volumes. Notice that, when N is small, the trend effect can be considered to be negligible, and the demand curve is sufficiently stable. In particular, the sliding window size is chosen to find the best trade-off to balance issues due to the non-stationary environment and trend w.r.t. the model's sample request to face noise and outliers (see Section 6.1).

2. Let us remark that this problem also applies to a generic convex combination of turnover and total profit.
3. See Appendix A.1

4.1 Bayesian Estimation of the Demand Curve

We aim to find the best margin for a product j using its transaction data. Our estimation algorithm is based on a *Bayesian Linear Regression* (BLR, Tipping, 2001). In such a regression model, we build an estimate $\hat{d}_j(\cdot)$ of the demand function for product j as a linear combination of the basis function taken as input, formally: $\hat{d}_j(m) = \sum_{h=0}^{Z} \theta_h \, \phi_h(m)$, where θ_h represents the h-th weight distribution and $\phi_h(m)$ represents the h-th basis function of the margin $m \in \mathcal{M}_j$. Notice that the BLR method returns a distribution $\hat{d}_j(m)$ for each margin $m \in \mathcal{M}_j$, which allows the adoption of a MAB approach in the following step of the algorithm. To increase data efficiency and robustness of the learning process in long-tail scenarios, we force our regression to return a monotonic non-increasing demand curve in the margin. Such an assumption is reasonable in our setting, as the products are non-perishable consumables and therefore they are not luxury, Veblen, or Giffen (Dougan, 1982; Kemp, 1998).

The demand curve estimation is performed using data collected during $\tau \in \mathcal{T} := \{t - N, \dots, t - 1\}$, *i.e.*, pairs $(m_{j\tau}, \bar{v}_{j\tau})$ of input margins $m_{j\tau}$ and output seasonally adjusted volumes $\bar{v}_{j\tau}$. To force the monotonicity of the estimated demand curve $\hat{d}(\cdot)$, we use a specific transformation of the standard *Bernstein polynomials* (Bernstein, 1912; Lorentz, 1953a) as basis functions in combination with a non-negative prior distribution for the θ_h parameters. Formally, the Bernstein polynomials of degree Z are composed by $Z + 1$ functions, defined as:

$$b_{h,Z}(m) = \binom{Z}{h} m^h (1 - m)^{Z-h}, \quad h = \{0, \dots, Z\}, \tag{4}$$

where $\binom{Z}{h}$ is the binomial coefficient. Notice that the choice of Bernstein polynomials allows us to model any demand function satisfying mild assumptions. More precisely, Bernstein polynomials converge to any function satisfying boundedness and continuity in a given range for a sufficiently large degree Z of the polynomials (Lorentz, 1953b).

Defining the row vector $B_Z(m) := \big[b_{0,Z}(m),\ b_{1,Z}(m),\ \dots,\ b_{Z,Z}(m)\big]$, a monotonic version $\phi_h(m)$ of the original basis functions is obtained as follows (McKay Curtis and Ghosh, 2011; Wilson et al., 2020):

$$\phi_h(m) := B_Z(m) \cdot (I_{Z+1} - S_{Z+1})^{-1} \cdot \mathbb{I}_h, \quad h = \{0, \dots, Z\}, \tag{5}$$

where I_{Z+1} is the identity matrix of order $Z + 1$, S_{Z+1} is the square matrix of dimension $Z + 1$ with all 1 in the *superdiagonal* ($s_{i,i+1} = 1$ for each i, 0 otherwise) (Olver and Shakiban, 2019), and $\mathbb{I}_h$ is the indicator operator selecting h-th element of the vector. Since Bernstein polynomials are defined over the support $[0, 1]$, we re-scale values over this range. From now on, for the sake of presentation and w.l.o.g., we assume that the margins are s.t. $\mathcal{M}_j \subseteq [0, 1]$.

To guarantee that the resulting demand function $\hat{d}_j(\cdot)$ is monotone, we use both the basis $\phi_h(\cdot)$ as defined in Equation (5) and a prior distribution for the θ_h parameters having a non-negative support. Therefore, we use the Lognormal distribution (Wilson et al., 2020) to model the parameters θ_h. Formally, we have the following:

$$\theta_h \sim \mathcal{LN}(\mu_h, \sigma_h), \quad \forall h \in \{0, \dots, Z\},$$

where $\mathcal{LN}(\mu_h, \sigma_h)$ denotes the *Lognormal* distribution with mean μ_h and standard deviation σ_h. Finally, the model fitting is performed using an *Variational Inference* approach provided in Blei et al. (2017), fitting a new parameter distribution using the last N available samples, *i.e.*, relying on the data $\big\{(m_{j\tau}, \bar{v}_{j\tau})\big\}_{\tau \in \mathcal{T}}$.

4.2 Exploration Strategy

Our problem can be naturally formulated as an online learning problem, where the goal is to balance the acquisition of information on the stochastic functions properly while, at the same time, maximizing the cumulative reward. The procedure addressing the exploration/exploitation at best is summarized in Figure 1. In particular, we resort to a sampling procedure similar to *Thompson Sampling* (TS, Agrawal and Goyal, 2012; Kaufmann et al., 2012). By construction, a Bayesian model provides in output a Lognormal probability distribution of the posteriors on the weights, which is used to drive the exploration in the learning process. Formally, we sample from the posterior distribution of BLR weights, retrieving a single realization of the posterior binding margins with the demand curve ($\hat{d}(m_{jt})$).

According to the MAB framework, we choose the best arm over a finite set of possible margins (representing the arms) $\mathcal{M}_j$. We can compute the value of the expected objective function $\hat{f}(m_{jt}), \forall m_{jt} \in \mathcal{M}_j$, which is the counterpart of

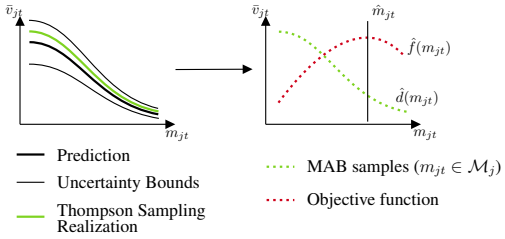


Figure 1: Optimal margin $\hat{m}_{jt}$ estimation process.

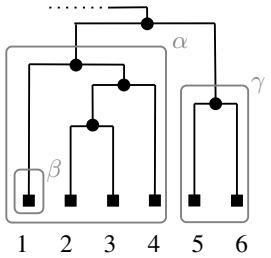


Figure 2: Example of a tree structure.

Equation 3 computed with the estimated demand function $\hat{d}(\cdot)$:[4]

$$\hat{f}(m_{jt}) = m_{jt}\,\hat{d}(m_{jt}). \tag{6}$$

The optimal margin $\hat{m}_{jt}$ is the best arm, corresponding to:

$$\hat{m}_{jt} = \underset{m_{jt}\in\mathcal{M}_j}{\arg\max}\ \hat{f}(m_{jt}), \tag{7}$$

where $\hat{f}(m_{jt})$ is the objective function estimated using demand curve $\hat{d}(\cdot)$, the latter coming from TS over the model.

5. Pricing Long-Tail Products

The algorithms proposed in Section 4 cannot be directly applied to long-tail products since the available data are not sufficient to produce a reliable estimate of the demand curve. The commonly adopted approach to applying to a long-tail product the same margin used for a popular product presenting similar characteristics may lead to wrong business decisions. This is mainly because the competition over long-tail and popular products is different, which, in its turn, can lead to different optimal margins. We deal with long-tail products by aggregating *similar* products subject to the constraint that the aggregated data are sufficient to produce an accurate estimation of the corresponding demand curve. Then, we apply our bandit pricing algorithm to each aggregation of products singularly. In the following sections, we describe the steps of our algorithm.

5.1 Distance Estimation

In this step, we exploit textual information to estimate the similarities among the products. This kind of information is indeed the only information available in large-scale e-commerce websites regarding the products. Initially, our algorithm removes the stop-words from the textual description (*i.e.*, recurring words such as, *e.g.*, "the" and "that"), as done in the work by Wilbur and Sirotkin (1992). Subsequently, the algorithm encodes into vectors the products' textual descriptions using *Term Frequency – Inverse Document Frequency* (TF-IDF) (Luhn, 1957; Jones, 1972). After that, it computes a distance matrix $\mathcal{D} = \left[d_{jk}\right]_{j,k\in\mathcal{J}}$, in which every entry provides the distance d_{jk} between each pair of vectors obtained using TF-IDF. Such a matrix expresses the similarities among the products. Additional technical details are provided in Appendix A.2.

5.2 Tree Structure Generation

In this step, we generate a binary tree structure based on the products' similarities by applying the *hierarchical clustering* approach proposed by Murtagh (1983) to the products and the corresponding distance matrix $\mathcal{D}$.[5] In this tree

4. Recall that the demand curve is no longer the expected volumes curve due to aggregation (see Section 5.4) and seasonality adjustment process.
5. We remark that the application of the hierarchical clustering algorithm by Murtagh (1983) requires the choice of a distance metric and a linkage method, *e.g.*, a method to compute the distance of two clusters. We adopt the metric induced by $\mathcal{D}$, and we opt for the use of the single linkage, as suggested by Ding and He (2002).

structure, every terminal node (*i.e.*, leaf) corresponds to a product $j \in \mathcal{J}$, and each non-terminal node corresponds to an aggregation of products, which we call *meta-products*. More precisely, a meta-product is the aggregation of those products whose leaves are reachable in the subtree whose root is the meta-product. Formally, we define a meta-product $\mathcal{K}$ as the set of products j present in the corresponding subtree. Figure 2 depicts an example of the tree structure resulting from the application of the above clustering approach over 6 products, in which products (terminal nodes) are depicted as squares, and meta-products (non-terminal nodes) are depicted as circles. In this example, the meta-product $\alpha = \{1, 2, 3, 4\}$ is the aggregation of the products 1, 2, 3, and 4. We remark that such a tree structure provides an explainable way to describe the similarities among the products whose interpretation is crucial in real-world applications, as it directly shows which products and aggregations are similar. Notice that, while all the non-terminal nodes are in principle meta-products, our algorithm works with only a subset of them chosen as discussed in the next step and discards the remaining ones.

5.3 Product Aggregation Strategy

In this step, the algorithm chooses the specific subset of meta-products to be priced. The rationale is to return a set of *minimal* meta-products, each populated with a sufficient amount of data to obtain an accurate demand curve estimation.

For every product j, we define a vector $\mathbf{s}_j := (s_{jt-N}, \dots, s_{jt-1})$, whose elements $s_{j\tau} = 1$ if at least a unit of the product j has been sold at time τ, $s_{j\tau} = 0$ otherwise. Similarly, given a meta-product α, we define a vector $\mathbf{s}_\alpha := (s_{\alpha t-N}, \dots, s_{\alpha t-1})$, obtained as $\mathbf{s}_\alpha := \oplus_{j\in\alpha}\mathbf{s}_j$, where $\oplus$ is the bit-wise "or" operation of the vectors corresponding to the products j belonging to α. Notice that $s_{\alpha\tau} = 1$ if at time τ at least a unit of at least one product belonging to the meta-product α has been sold. The condition stating that the amount of data for a meta-product α are sufficient is that the number of time points for which there is at least a sale of meta-product α is at least $q\,N$, where $q \in (0, 1]$ is a parameter that we can tune. The above condition can be evaluated by computing the sum of the elements of $\mathbf{s}_\alpha$ and comparing it with $q\,N$.

Finally, the choice of the meta-products is performed as follows. Starting from each product j we check the above condition, and if it is satisfied, the product j is chosen as a meta-product. An example of this case is represented by the product 1 in Figure 2. If the condition does not hold on the single product, we traverse upward the nodes of the aforementioned tree structure, and stop as soon as the above condition is satisfied. In this case, the meta-product corresponding to the non-terminal node is selected to build the demand model for the product j. Notice that the minimality principle we adopt is motivated by the need for balancing between the bias and variance of the demand curve estimates. Indeed, merging additional products to a minimal meta-product would most likely increase the bias of the demand curve estimated for each product therein, while providing only a negligible benefit in terms of variance reduction.

5.4 Meta-product Demand Estimation and Pricing

In this step, the algorithm estimates the demand curve of each selected meta-product and prices the corresponding products. Let us consider a meta-product aggregating the set of products $\mathcal{K} \subseteq \mathcal{J}$ and the corresponding sale statistics pairs $(m_{k\tau}, \bar{v}_{k\tau})$ for all products $k \in \mathcal{K}$ and time $\tau \in \mathcal{T}$. We compute the demand curve of a meta-product using the overall volume $\bar{v}_{\mathcal{K}\tau}$ for a specific time τ and the corresponding average margin $m_{\mathcal{K}\tau}$ used to get such a volume at time τ. The above quantities are computed, for each $\tau \in \mathcal{T}$, as:

$$\bar{v}_{\mathcal{K}\tau} := s_{\mathcal{K}\tau} \sum_{k\in\mathcal{K}} v_{k\tau}, \qquad m_{\mathcal{K}\tau} := \sum_{k\in\mathcal{K}} m_{k\tau} \cdot \frac{v_{k\tau}}{\sum_{h\in\mathcal{K}} v_{h\tau}},$$

where the average margin is computed averaging the products margins weighted by their seasonality-adjusted volumes, and the coefficient $s_{\mathcal{K}\tau}$ is computed similarly to its single-product counterpart (details are provided in Appendix A.1). The estimated demand function and the final selected margin $\hat{m}_{\mathcal{K}t}$ for the meta-product are computed using the same procedure described in Section 4, *i.e.*, using margins $m_{\mathcal{K}\tau}$ as input of the regression model and volumes $\bar{v}_{\mathcal{K}\tau}$ as output, as well as the selection of the margin. Indeed, once the above conversion has been applied, the meta-product data are of the same nature as the one of a single product j and, therefore, are processed in the same way. Finally, the selection of the margin for a product j is provided by the margin of the meta-product $\mathcal{K}$ including j with the smallest cardinality.

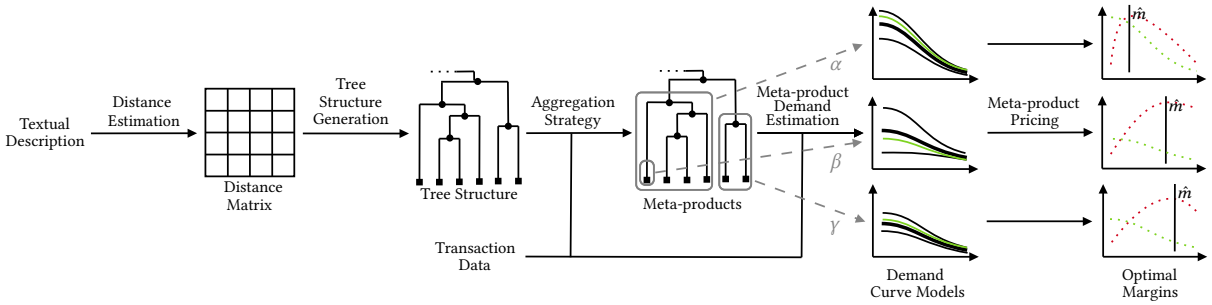


Figure 3: Scheme of the *DynaLT* algorithm.

For instance, in Figure 2, the margin of product 2 is selected using meta-product α, while product 1 using the margin corresponding to meta-product β.

A visual representation of the overall algorithm described in the previous sections is provided in Figure 3. The process starts from the textual description of each product and, thanks to these information, builds the tree structure. Subsequently, using the transaction data available, it builds the meta-products, estimates the corresponding demand functions, and, finally, it provides a margin to apply to each product in the catalog.

6. Experiments

In this section, we evaluate the empirical performance of our algorithms. Initially, we show how the resort to monotonic bandits improves the pricing performance. To do that, we use an offline setting whose optimal solution is known. Subsequently, we describe the application of our algorithm to a real-world long-tail setting.

6.1 Pricing Single Products

We compare our algorithm, namely *DynaLT*, with a BLR approach not exploiting the monotonicity, denoted with *NM-BLR*, where we use the Normal prior for the parameters θ_h and Bernstein's polynomials in Equation (5) as basis functions. A detailed description of the setting is deferred to Appendix A.3. We compare the two algorithms in terms of empirical regret $\hat{R}(\pi)$, *i.e.*, the empirical counterpart of the regret $R(\pi)$. Results are averaged over 15 independent runs for each algorithm (standard deviation is reported in brackets).

Noise and Outliers First, we study how *DynaLT* and *NM-BLR* methods are affected by the variation of the standard deviation of the noise of the volumes $v_{j\tau}$ and the introduction of outliers, *i.e.*, the presence of customers performing significantly larger orders than usual. In what follows, outliers are modeled as using, with probability o, a different noise term $\epsilon' \sim \mathcal{N}(0, \sigma')$ having $\sigma' = 10\,\sigma$. In particular, we investigate scenarios with $\sigma \in \{0.001, 0.005, 0.01\}$ and $o \in \{0\%, 10\%, 20\%\}$. The algorithms have been run over a time horizon of $|T| = 100$ weeks.

The empirical regret $\hat{R}(\pi)$ obtained with the two methods are summarized in Table 1 (the smaller, the better). On average, *DynaLT* outperforms its non-monotone counterpart *NM-BLR* on every setting. Overall, as expected, the performance of the two algorithms degrades as the standard deviation of the noise σ and the outlier percentage o increase. Without outliers, *DynaLT* is significantly better than *NM-BLR* for each value of the noise, and the improvement increases as the noise gets larger, with an improvement in terms of regret from $\approx 20\%$ for $\sigma = 0.001$ to $\approx 41\%$ for $\sigma = 0.01$. Conversely, with outliers, the advantage of using *DynaLT* is significant only for small values of noise standard deviation, *e.g.*, $\sigma = 0.001$ and $\sigma = 0.05$, leading to a reduction of the regret in the range $[12\%, 23\%]$. Finally, with both a large noise standard deviation ($\sigma = 0.01$) and outliers ($o = 10\%$ and $o = 20\%$), the performance of the two techniques are comparable.

Table 1: $\hat{R}(\pi)$ in the presence of noise and outliers.

Noise std σ		Outlier percentage o	0%	10%	20%
0.001		*DynaLT*	6.05 (0.12)	7.92 (0.2)	8.91 (0.25)
		NM-BLR	7.51 (0.04)	10.43 (0.08)	11.61 (0.14)
0.005		*DynaLT*	9.36 (0.29)	18.17 (0.81)	22.09 (1.09)
		NM-BLR	16.34 (0.42)	21.88 (0.55)	25.15 (0.68)
0.01		*DynaLT*	16.0 (0.27)	35.51 (1.74)	36.75 (1.56)
		NM-BLR	27.34 (0.6)	37.71 (1.64)	37.12 (1.21)

Table 2: $\hat{R}(\pi)$ in the presence of non-stationarities.

Window Size N		Number of Changes c	1	2	3
20		*DynaLT*	4.16 (0.54)	8.86 (1.85)	6.51 (0.56)
		NM-BLR	4.82 (0.8)	12.8 (2.17)	10.27 (0.23)
30		*DynaLT*	4.55 (0.8)	9.49 (2.15)	6.82 (0.49)
		NM-BLR	4.84 (0.84)	12.45 (1.75)	12.75 (0.27)
40		*DynaLT*	3.95 (0.54)	7.46 (1.54)	6.14 (0.65)
		NM-BLR	4.32 (0.22)	9.87 (1.69)	11.85 (0.69)

Non-stationarities Second, we evaluate our algorithm in a non-stationary setting. To do that, we simulate some changes in the environment due to, *e.g.*, a new competitor or a new product. This is done by abruptly changing the product volume function at specific time points. The specific shapes of the different volume functions are provided in Appendix A.3. In particular, we introduce $c \in \{1, 2, 3\}$ abrupt changes occurring at evenly spaced time points (over the entire time horizon). Notice that, since the underlying demand functions are different for different values of c, the regrets corresponding to these scenarios cannot be directly compared. Even in this case, we evaluate the impact of exploiting the monotonicity w.r.t. a traditional demand function estimation method (*NM-BLR*). Furthermore, we analyze the impact of changing the sliding window length $N \in \{20, 30, 40\}$. In this experiment, the empirical regret $\hat{R}(\pi)$ is computed w.r.t. a clairvoyant policy that knows when the changes in the environment would occur, and its value has been averaged over 15 independent runs. The algorithms have been run over a time horizon of $|T| = 120$ weeks.

The empirical regrets are reported in Table 2 (the smaller, the better). Even in this scenario, the performance provided by *DynaLT* is, on average, better than those of *NM-BLR*. However, the difference is significant only in the setting with $c = 3$. This suggests that monotonicity allows for a better estimate of the demand functions, especially if the environment changes frequently. Moreover, the performance for *DynaLT* achieved with different window sizes do not change significantly, suggesting that this method is less sensitive to changes in the window size.

6.2 Pricing Long-tail Products

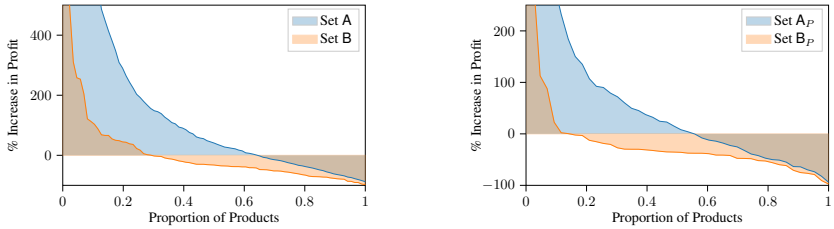
The *DynaLT* has been used for two months on a real-world e-commerce website, comparing its performance with the pricing strategy previously used by the business managers.[6]

Setting In this test, we have a catalog $\mathcal{J}$ of $7,826$ products, with a turnover of 2.50 MEuros per year and a cumulative net margin of 0.53 MEuros (according to 2021 statistics). We divide the products into two sets defined by e-commerce specialists, according to both technical and market aspects, to set a proper A/B testing procedure. Specifically, $2,132$ products have been priced by the company's experts, while $5,694$ have been priced by *DynaLT*. From now on, we refer to the former one as the *control set* B and to the latter as the *test set* A. Only $\approx 3\%$ of the products got on average at least 1 sale per week during 2021. We refer to this subset of products as *popular*, while the remaining ones are addressed as *long-tail*. Based on this classification, we further divide each one of the A and B sets into two subsets containing only *popular* products (A_P and B_P, respectively) and *long-tail* ones (A_{LT} and B_{LT}, respectively). The pricing process for the two above tests has been conducted over a test period $|T| = 8$ weeks, from November 22, 2021, to January 16, 2022. Since it is not possible to compute the regret $\hat{R}(\pi)$ of the strategies in a real-world scenario, we evaluate the different pricing schemes in terms of their profits. Formally, the performances of the two strategies have been evaluated using the *rate of the profits* between the analyzed period and the one obtained in a control period C, from November 23, 2020, to January 17, 2021. Let us define the total profit achieved by *DynaLT* as:

$$M(\mathsf{A}, T) := \sum_{t \in T} \sum_{j \in \mathsf{A}} v_{jt} m_{jt} c_j, \tag{8}$$

where m_{jt} is chosen by *DynaLT*. Similarly, we can define the profit $M(\mathsf{B}, T)$ achieved by human experts in the period T over the set B, and the profits $M(\mathsf{A}, C)$ and $M(\mathsf{B}, C)$ of *DynaLT* on the set A and human expert on the set B,

6. Further details about the e-commerce website have been retained due to NDA.



(a) Profit increase for the products in set A (blue) and B (orange).

(b) Profit increase per single popular product in set A_P (blue) and set B_P (orange).

Figure 4: Profit increase by product.

respectively, over the period C. The performance metric we adopt is:

$$G := \frac{M(\mathsf{A}, T)}{M(\mathsf{A}, C)} \frac{M(\mathsf{B}, C)}{M(\mathsf{B}, T)}. \tag{9}$$

Intuitively, G is greater than 1 if *DynaLT* increases the profit obtained during the period T w.r.t. period C more than human experts did. The *DynaLT* hyperparameters are set using historical data from 2016 to 2021.

Results *DynaLT* algorithm over the set A provides an increase of $\approx 5\%$ in terms of profit during the period T w.r.t. period C. Instead, the choice of the experts over the set B provided a reduction of the profit of $\approx 25\%$. Therefore, the performance index G is ≈ 1.4. Figure 4a represents the increases in profit for each product in settings A (blue area) and B (orange areas). While the set A records an increase in profit w.r.t. last year in $\approx 63\%$ of the products, set B achieves a positive performance in $\approx 29\%$ of the products. This suggests that the improvement provided by *DynaLT* is due to a better pricing strategy over a large number of products. As mentioned before, both products' sets A and B are mostly constituted by long-tail products. Indeed, $5,481$ out of $5,694$ products in A, and $2,078$ out of $2,132$ products in B are long-tail products. We will refer to the aforementioned subsets of long-tail products A_{LT} and B_{LT}, respectively. The performance index G computed over the new sets (using A_{LT} and B_{LT} in the definition in place of A and B, respectively) is $G_{LT} = 1.91$, suggesting that the pricing of long-tail products is significantly improved thanks to *DynaLT*. In the case of popular products, the improvement is smaller as we have $G_P = 1.18$. Nonetheless, in Figure 4b the profit increment for the popular products in the set A_P occurs for $\approx 55\%$ of the popular products, while in set B_P only in 14% of the cases we have an improvement.

7. Conclusions

In this paper, we propose *DynaLT* to manage the complex task of pricing products in an e-commerce scenario in the presence of long-tail products. Indeed, this kind of product commonly constitutes the majority of the ones present in a catalog, but automatic pricing methods are usually unable to handle them due to the scarcity of their transaction data. We propose a modeling approach based on the demand curve's properties, which can speed up the demand curve learning process, and an aggregation strategy to automatically group products with too little data. The modeling approach has been tested on synthetically generated data to show the advantages of including the monotonicity property, and the overall *DynaLT* has been implemented in a real-world e-commerce website, showing that its application increases the profits on average of 18% w.r.t. what is gained by manually pricing the products. In future, we will investigate the integration of advertising and recommendation strategies in *DynaLT* to understand whether the product presentation to a user may improve the performance. We will also study more complex user models, *i.e.*, assuming that the buyer is strategic w.r.t. the price choice.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283. USENIX Association, 2016.

Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory (COLT)*, volume 23, pages 39.1–39.26. JMLR Workshop and Conference Proceedings, 2012.

Chris Anderson. The long tail, 2004. URL https://www.wired.com/2004/10/tail/.

Chris Anderson. *The long tail: Why the future of business is selling less of more*. Hachette Books, 2006.

Serge Bernstein. Démonstration du théorème de weierstrass fondée sur le calcul des probabilités. *Communications of the Kharkov Mathematical Society*, -:1–2, 1912.

Dimitris Bertsimas and Georgia Perakis. Dynamic pricing: A learning approach. In *Mathematical and computational models for congestion charging*, pages 45–79. Springer, 2006.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Erik Brynjolfsson, Yu Hu, and Duncan Simester. Goodbye pareto principle, hello long tail: The effect of search costs on the concentration of product sales. *Management Science*, 57(8):1373–1386, 2011.

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

Chris Ding and Xiaofeng He. Cluster merging and splitting in hierarchical clustering algorithms. In *International Conference on Data Mining (ICDM)*, pages 139–146. IEEE Computer Society, 2002.

William R Dougan. Giffen goods and the law of demand. *Journal of Political Economy*, 90(4):809–815, 1982.

Ellen Garbarino and Olivia F Lee. Dynamic pricing in internet retail: effects on consumer trust. *Psychology & Marketing*, 20(6):495–513, 2003.

Kelly L Haws and William O Bearden. Dynamic pricing and consumer fairness perceptions. *Journal of Consumer Research*, 33(3):304–311, 2006.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *International conference on Algorithmic Learning Theory (ALT)*, pages 199–213. Springer, 2012.

Simon Kemp. Perceiving luxury and necessity. *Journal of economic psychology*, 19(5):591–606, 1998.

Robert Kleinberg and Tom Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *Symposium on Foundations of Computer Science (FOCS)*, pages 594–605. IEEE, 2003.

George G. Lorentz. *Bernstein Polynomials*. University of Toronto Press, 1953a.

George G. Lorentz. Degree of approximation. In *Bernstein Polynomials*, chapter 1.6, pages 19–23. University of Toronto Press, 1953b.

Hans Peter Luhn. A statistical approach to mechanized encoding and searching of literary information. *Journal of research and development*, 1(4):309–317, 1957.

S McKay Curtis and Sujit K Ghosh. A variable selection approach to monotonic regression with bernstein polynomials. *Journal of Applied Statistics*, 38(5):961–976, 2011.

Kanishka Misra, Eric M Schwartz, and Jacob Abernethy. Dynamic online pricing with incomplete information using multiarmed bandit experiments. *Marketing Science*, 38(2):226–252, 2019.

Jonas W Mueller, Vasilis Syrgkanis, and Matt Taddy. Low-rank bandit methods for high-dimensional dynamic pricing. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 1–11. NeurIPS Proceedings, 2019.

Fionn Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The computer journal*, 26(4): 354–359, 1983.

Yadati Narahari, CVL Raju, K Ravikumar, and Sourabh Shah. Dynamic pricing models for electronic business. *sadhana*, 30(2):231–256, 2005.

Peter J. Olver and Chehrzad Shakiban. Practical linear algebra. In *Applied Linear Algebra*, chapter 1.7, pages 52–53. Springer, 2019.

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

Michael E Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of machine learning research*, 1(Jun):211–244, 2001.

Francesco Trovò, Stefano Paladino, Marcello Restelli, and Nicola Gatti. Multi-armed bandit for pricing. In *European Workshop on Reinforcement Learning (EWRL)*, pages 1–9. -, 2015.

Francesco Trovò, Stefano Paladino, Marcello Restelli, and Nicola Gatti. Improving multi-armed bandit algorithms in online pricing settings. *International Journal of Approximate Reasoning*, 98:196–235, 2018.

W John Wilbur and Karl Sirotkin. The automatic identification of stop words. *Journal of information science*, 18(1): 45–55, 1992.

Ander Wilson, Jessica Tryner, Christian L'Orange, and John Volckens. Bayesian nonparametric monotone regression. *Environmetrics*, 31(8):e2642, 2020.

Appendix A. Implementation Details

The implementation of the *DynaLT* algorithm for the experiments presented in Section 6 has been done using Python 3. More specifically, the Bayesian Regression Model is implemented using TensorFlow Probability library (Abadi et al., 2016). In what follows, we provide the implementation details to allow the replicability of the experiments, *i.e.* the seasonality estimation (Appendix A.1), the distance estimation procedure (Appendix A.2), and the synthetic environment creation (Appendix A.3). Finally, we discuss the algorithm running time (Appendix A.4).

A.1 Seasonality

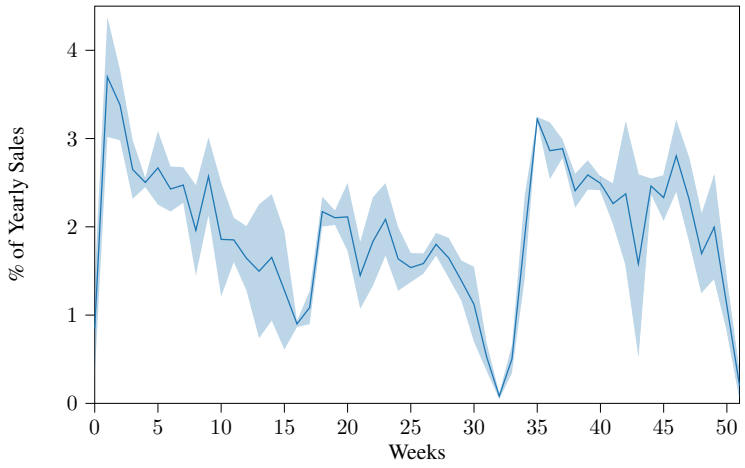


Figure 5: Percentage of sales over the 52 weeks in a year (standard deviation is depicted as semitransparent areas).

Figure 5 shows that, even if the seasonality effect is relevant, it is stable across years since the standard deviation bounds provided as semitransparent areas are small. Therefore, we model it as a multiplicative factor $s_{j\tau}$ for each product j at time τ such that we can compute seasonality adjusted volumes as $\bar{v}_{j\tau} := v_{j\tau} \cdot s_{j\tau}$.

The seasonality term $s_{j\tau}$ is estimated in a data-driven way using data coming from a set of previous years $\mathcal{Y}$. We denote with v_{jwy} the volume for product j corresponding to a week of the year $w \in \{1, \dots, 52\}$ and a year $y \in \mathcal{Y}$. At first, let us compute for a product j the proportion of the volumes sold in a specific week w for a year y, formally:

$$\hat{v}_{jwy} = \frac{v_{jwy}}{\sum_{i \in \mathcal{W}} v_{jiy}}. \tag{10}$$

The seasonality factor $s_j(w)$ for a specific week w is computed as follows:

$$s_j(w) = \frac{1}{\sum_{y \in \mathcal{Y}} \hat{v}_{jwy}/Y + H}\ , \tag{11}$$

where H is a shrinkage factor, and $Y := |\mathcal{Y}|$ is the cardinality of the set $\mathcal{Y}$. Finally, the correction factor $s_{j\tau}$ is equal to the $s_j(w)$ for the week w of the year corresponding to time τ.

The same procedure is applicable for meta-product by using the aggregated volumes of the product therein, *i.e.*, for meta-product $\mathcal{K}$:

$$v_{\mathcal{K}wy} = \sum_{k \in \mathcal{K}} v_{kwy}.$$

In the experiments, the shrinkage factor is selected equal to $H = 0.005$ based on empirical evidence.

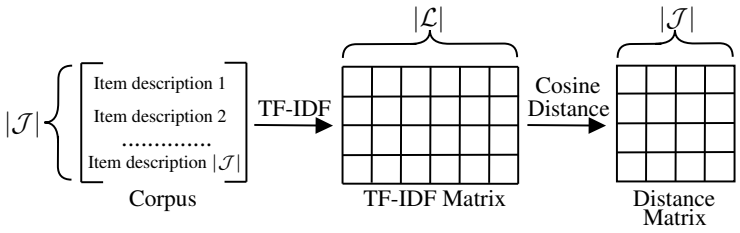


Figure 6: Overall scheme of the TF-IDF algorithm.

A.2 Similarity Estimation

In our application, each product $j \in \mathcal{J}$ has a textual description ρ_j, which contains information regarding the product, like its brand, color, and material.[7] The corpus of strings $\Pi = \{\rho_j\}_{j \in \mathcal{J}}$[8] is represented by the descriptions of all the available products.

With $|\rho_j|$ we denote the dimension of the string computed as the number of words it contains. TF-IDF encoding balances the importance tf_{ij} of a word i in a string ρ_j and the importance idf_i of the word i across the whole textual data set. Formally:

$$tf_{ij} = \frac{\upsilon_{ij}}{|\rho_j|},$$

$$idf_i = \log_{10} \frac{|\Pi|}{|\{\rho \in \Pi \text{ s.t. } i \in \rho\}|},$$

where υ_{ij} is the number of occurrences of the word i in description ρ_j of product j, $|\Pi|$ is the number of string present in the corpus, and $|\{\rho \in \Pi \text{ s.t. } i \in \rho\}|$ is the number of textual descriptions ρ in which the word i is present among the one of the entire catalog Π. The TF-IDF score for the word i in the description ρ_j of product j is computed as follows:

$$tfidf_{ij} = tf_{ij} \cdot idf_i.$$

The result is a vector $\eta_j \in [0,1]^{|\mathcal{L}|}$, where $\mathcal{L}$ is the set of distinct words (obtained after a stop-word removal procedure) in all the texts. For each product j, $[\eta_j]_i = tfidf_{ij}$ is the TF-IDF score of word $i \in \mathcal{L}$ for the text defined by ρ_j. The distance d_{jl} between two products j and l is computed using a transformation of the cosine similarity, formally:

$$d_{jl} = 1 - \frac{\eta_j \cdot \eta_l}{\|\eta_j\| \cdot \|\eta_l\|}.$$

where $\cdot$ represents the scalar product between vectors and $||\eta||$ represents the 2-norm of η. Figure 6 represents the whole process that goes from textual data to the computation of a pairwise distance matrix between the products.[9]

A.3 Simulation Details

Noisy Environment Simulation The volumes for the single product pricing experiments in Section 6.1 are generated from the volume function:[10]

$$v_1(x) = 2e^{-(x+1.2)^{\frac{5}{2}}} + \epsilon,$$

where prices $x \in [0.32, 1]$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian zero-mean noise with variance σ^2. The product had a unitary cost $c = 0.3$. A graphic representation of the volumes curve corresponding to this product is provided in Figure 7.

7. For the sake of presentation we focus on the textual description, but one might also concatenate additional textual information, like the product category or its name.
8. Note that $|\Pi| = |\mathcal{J}|$.
9. Other ways of vectorization such as embedded-based ones are also viable options in the case the textual descriptions are succinct.
10. Notice that the chosen demand function satisfies the motononicity assumption.

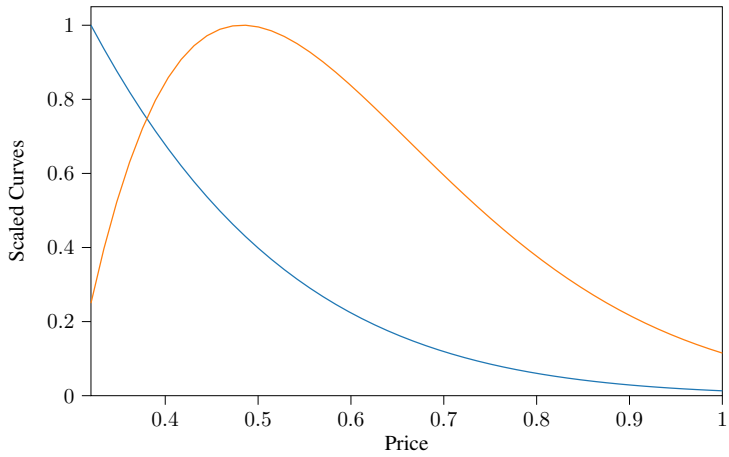


Figure 7: Demand curve used in the noisy experiment and corresponding reward function obtained maximizing profit.

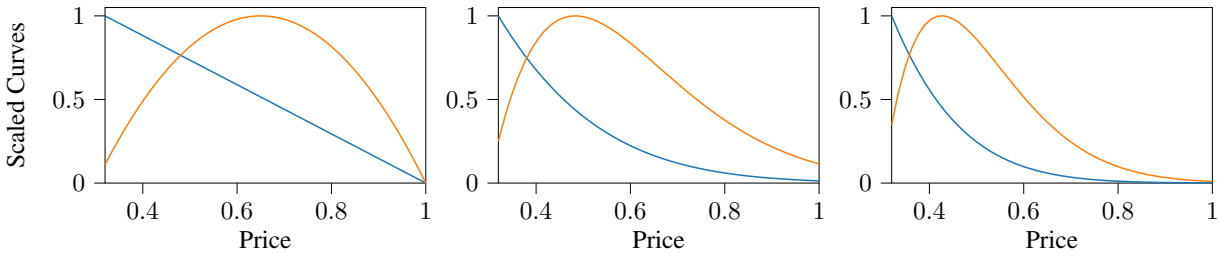


Figure 8: Demand curves used in the non-stationary experiment and corresponding reward functions.

In this, we modified noise's standard deviation σ and introduced some outliers in the data generation process. More specifically, the outliers generation is obtained through the probability $o \in (0, 1)$ that a sample drawn from a demand curve has noise ϵ' s.t. its standard deviation is 10 times the one of ϵ.

Non-stationary Environment Simulation In the second experiment, three different volume functions have been used during the different phases of the non-stationary process. The volume functions were:

$$v_1(x) = \frac{3}{10}(1 - x),$$
$$v_2(x) = 2e^{-(x+1.2)^{\frac{5}{2}}},$$
$$v_3(x) = 7e^{-(x+1.2)^3}.$$

Their corresponding volumes curves are provided in Figure 8. The first abrupt change substituted the volume function $v_1(x)$ with $v_2(x)$, the second substituted $v_2(x)$ with $v_3(x)$, and the third one $v_3(x)$ with $v_1(x)$. In this set of experiments the noise's standard deviation is $\sigma = 0.001$, and the outliers' percentage is $o = 0\%$.

Algorithm Settings In the first scenario, the demand curve have been estimated using Bernstein's Polynomial with $Z = 75$. The priors for the Lognormal and Gaussian distribution of the BRL model have been set with $\sigma_h = 0.75$ and $\sigma_h = 0.5$, respectively. The values for the hyper-parameters have been chosen basing on an independent data set. The sampling procedure described in Section 4 have been applied to the set of margins $\mathcal{M}$ of evenly spaced values over the domain $[0.05, 1.5]$, where $|\mathcal{M}| = 50$.

In the second scenario, we use the same configurations for the Bernstein's Polynomial and the sampling procedure. Conversely, the prior parameters for the Lognormal and Gaussian priors were set to $\sigma_h = 0.75$ and $\sigma_h = 2$, respectively.

Notice that the clairvoyant solution to the problem of maximizing the profits is non-trivial even knowing the real volume functions, due to the fact that the introduction of noise and outliers do not allow to compute it in a closed form solution. We estimated the optimal solution using Monte Carlo approach, *i.e.*, we simulated $10,000$ samples from each one of the margins used in the experiments and averaged the values of the profit gained with such a margin. Then we took the maximum over the computed profits as the optimal solution for the problem. Thanks to this approach, the empirical regret is computed as the difference between this value and the one obtained using the analyzed algorithms.

A.4 Algorithm Running Time

The algorithm running time can be analyzed by dividing the process into two phases: first, the distance estimation and the tree structure generation, then, the proper optimal price estimation.

Similarity and Tree Structure This phase is required to be performed only when there are changes in the catalog of the products. The running time for the distance estimation algorithm is $\mathcal{O}(|\mathcal{J}|^2)$ for what concerns the operations required to construct the distance matrix. Building the agglomerative clustering tree structure requires a running time $\mathcal{O}(|\mathcal{J}|^2 \log |\mathcal{J}|)$ when using single linkage, and $\mathcal{O}(|\mathcal{J}|^3)$ in the general case. It is worth noting that adding a new product to the catalog corresponds to an incremental update of the distance matrix, *i.e.*, adding a new row and column to the matrix consisting of the distance of the new products w.r.t. the previous ones.

Optimal Pricing The proper estimate of the optimal price must be performed at every time t, as well as the association of a product j with the related meta-product $\mathcal{K}$. This is because the cluster stopping condition is defined over transactions data, which changes over time. Given $|\mathcal{J}|$ products, we must estimate at most (worst-case scenario) $|\mathcal{J}|$ BLR models.