Learning Algorithms for Dynamic Pricing: A Comparative Study

Chaitanya Amballa[1], **Narendhar Gugulothu**[1], **Manu K. Gupta**[2] **and Sanjay P. Bhat**[1]

[1] *TCS Research and Innovation, India*

[2] *Department of Management Studies, Indian Institute of Technology, Roorkee, India.*

Abstract

We consider the problem of dynamic pricing, or time-based pricing in which businesses set flexible prices for products or services based on current market demands. Various learning algorithms have been explored in literature for dynamic pricing with the goal of achieving the minimal regret. We present a comparative experimental study of the relative performance of these learning algorithms as well as two practical improvements in the context of dynamic pricing, and list the resulting insights. Performance is measured in terms of expected cumulative regret, which is the expected regret of not suggesting the best price in hindsight.

Keywords: Thompson sampling, Dynamic pricing, Exploration strategies, Regret.

1. Introduction

One of the most important problem any retailer has to solve is, how to price items correctly without accurate prior knowledge of the true demand. One method to learn the demand function is price experimentation, where the retailer modifies prices adaptively to learn the hidden demand function and use that to estimate the revenue-maximizing price. Nevertheless, proper price experimentation is a challenging problem as the potential revenue loss during the learning horizon can be substantially large. In fact optimally balancing the trade-off between randomly selecting prices to expedite the *learning* versus selecting prices that maximize the expected earning has been the subject of recent research in dynamic pricing literature (see Keskin and Zeevi (2014), den Boer and Zwart (2014)). While the primary research trend in this field has been to study the performance of a given learning algorithm, this paper focuses on the comparison of various learning frameworks.

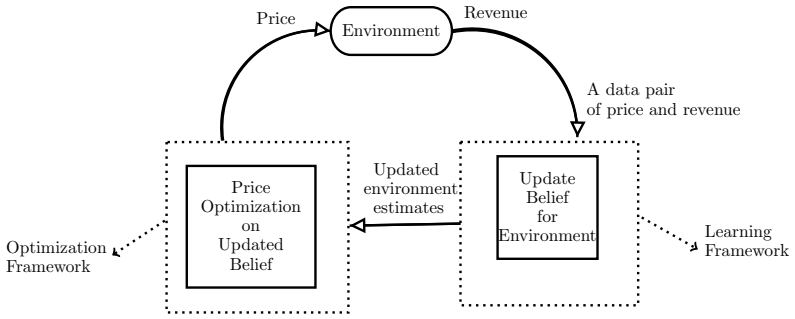

Figure 1: Model flow for dynamic pricing

Figure 1 shows a general framework for the dynamic pricing problem where the environment generates revenue based on the unknown demand curve and some price as an input. The learning framework updates its belief (that is, learns the demand curve) based on the price and revenue pair (input data). The optimization framework suggests the best (optimal) price based on its current belief (estimate) of the environment. In a dynamic pricing setting, the objective is to learn the optimal price as quickly as possible.

2. Model description

Suppose a firm sells a product over a time horizon of T periods. In each period $t = 1, 2, \dots T$, the seller must choose a price p_t from a given feasible set $[p_{\min}, p_{\max}] \in \mathbb{R}$, where $0 \leq p_{\min} < p_{\max} < \infty$. The seller observes the demand d_t according to the following linear demand model $d_t = \alpha - \beta p_t + \xi_t$ for $t = 1, 2, \dots, T$, where $\alpha, \beta > 0$ represent the parameters of the unknown demand model, and $\xi_t \sim \mathcal{N}(0, \sigma^2)$ represents unobserved demand perturbations. The seller's single period revenue r_t in period t equals $r_t = d_t p_t$. This leads to a quadratic dependence of r_t on p_t. More generally, one can consider demand models that lead to a higher degree polynomial dependence of revenue on the price. Hence, we consider a general polynomial for the revenue function $r_t = g(p_t) + \xi_t$, where $g(p_t) = \tilde{\mu}_0 + \tilde{\mu}_1 p_t + \tilde{\mu}_2 p_t^2 + \cdots + \tilde{\mu}_n p_t^n$. The firm's goal is to learn the unknown parameters $\tilde{\mu}_0, \tilde{\mu}_1, \cdots, \tilde{\mu}_n$ from noisy observations of price and revenue pairs $\{(p_t, r_t)\}_{t=1}^T$ well enough to reduce the T-period expected regret, defined as

$$R(T) = \sum_{t=1}^{T} \left[r^* - \mathbb{E}(r_t) \right],$$

where $r^* = \max_{p \in [p_{\min}, p_{\max}]} g(p)$ is the optimal expected single-period revenue. For each positive integer n, define $f_n : \mathbb{R} \to \mathbb{R}^{n+1}$ by $f_n(p) = [1, p, p^2, \dots, p^n]^{\mathrm{T}}$. On denoting $x_t = f_n(p_t)$, we see that $\mathbb{E}(g(p_t)|x_t) = \tilde{\mu}^{\mathrm{T}} x_t$ with $\tilde{\mu} = [\tilde{\mu}_0, \tilde{\mu}_1, \cdots \tilde{\mu}_n]$. Thus, the dynamic pricing problem as posed above can be cast as stochastic linear optimization problem with bandit feedback. In subsequent sections, we discuss several algorithms to learn the unknown parameters of the polynomial $g(\cdot)$ for the above bandit optimization problem.

3. Different algorithms for dynamic pricing

In this section, we discuss various learning algorithms for dynamic pricing. First, we present some simple baseline methods which have known theoretical guarantees in terms of regret bounds (see Keskin and Zeevi (2014)).

3.1 Baseline algorithms

The two methods for dynamic pricing that we consider as baselines are: 1) Iterated least square (ILS) and 2) Constrained iterated least square (CILS). ILS estimates the revenue curve by applying least squares to the set of prior prices and realized demands, and then selects the price for the next period greedily with respect to the estimated revenue curve. ILS has been shown to be sub-optimal, while CILS has been shown to achieve asymptotically optimal regret (see Keskin and Zeevi (2014)). CILS achieves asymptotically optimal regret by integrating forced price-dispersion with ILS. More precisely, CILS with threshold

parameter k suggests at time t the price

$$p_t = \begin{cases} \bar{p}_{t-1} + \text{sgn}(\delta_t) k t^{-1/4} & \text{if } |\delta_t| < k t^{-1/4}, \\ \text{ILS price} & \text{otherwise.} \end{cases}$$

where $\delta_t = p_{t-1} - \bar{p}_{t-1}$, with $\bar{p}_{t-1}$ as the average of the prices suggested over $t-1$ periods.

3.2 Action Space Exploration

Action Space Exploration (ASE) (see Watkins (1989); Vemula et al. (2019)), in principle, combines ILS with the ϵ-greedy exploration strategy, which is one of the simplest and most widely used strategies for exploration. More precisely, at each decision instant, action space exploration updates the estimates of the parameters of the revenue curve by using gradient descent on the mean-square error (MSE) loss between the estimated and true revenue curve computed on the price-revenue pairs observed till then. It then selects a random price with probability ϵ, and selects the price that is greedy with respect to the updated estimate of the revenue curve with complementary probability $1 - \epsilon$. For the experimental results, we have used ϵ-greedy strategy with exponentially decreasing ϵ (see Vermorel and Mohri (2005)). Due to space constraints, we refrain from giving a detailed algorithm here, and instead refer the reader to the papers cited above.

3.3 Parameter Space Exploration

Parameter Space Exploration (PSE) has been studied in Fortunato et al. (2017); Plappert et al. (2017); Miyamae et al. (2010); Wang et al. (2018); Rückstiess et al. (2010); Neelakantan et al. (2015) and used in reinforcement learning, control and ranking tasks. Like Thompson sampling, parameter space exploration also, in essence, maintains a posterior distribution over the parameters of the revenue curve, and selects a price that is greedy with respect to a parameter vector w sampled from the posterior. The sampling is achieved by perturbing the current estimated parameter vector $\hat{\mu}$ with a zero-mean, uncorrelated Gaussian noise vector $\hat{\sigma} \circ \hat{\epsilon}$, where $\hat{\epsilon}$ is a vector of independent samples of a standard normal random variable. The parameter estimate $\hat{\mu}$ and standard deviations $\hat{\sigma}$ of the Gaussian noise variables are updated as in Plappert et al. (2017) by using gradient descent on the mean-square error (MSE) loss between the sampled and true revenue curves on the price-revenue pairs observed so far. The detailed algorithm is presented as Algorithm 1 in Appendix A.

3.4 Thompson Sampling

We use Bayesian linear regression with Thompson sampling (TS) for learning the unknown parameters. At time t, TS maintains a multivariate Gaussian posterior distribution with mean μ_t and covariance $A_t \in \mathbb{R}^{n \times n}$ over the unknown parameters. At time $t+1$, TS chooses the price p_{t+1} greedily according to a parameter vector w_t sampled from the prior at time t. This results in a new observation r_{t+1} of the revenue. The new observation is used to update the posterior distribution according to the following update equations (see Bagnell (2005) for details).

$$A_{t+1}^{-1} = A_t^{-1} + \sigma^{-2} x_{t+1} x_{t+1}^T, \text{ and } A_{t+1}^{-1} \mu_{t+1} = A_t^{-1} \mu_t + \sigma^{-2} r_{t+1} x_{t+1}, \tag{1}$$

where $x_t = f_n(p_t)$. Usually, σ is the standard deviation of the noise in the observed revenue. However, we treat σ as a parameter in the update equations which we may leverage to improve regret performance. The detailed TS algorithm is described in Algorithm 2.

Thompson sampling suffers from a drawback, namely, that it continues sampling even after the true revenue curve is learnt sufficiently well, potentially leading to unnecessarily large regret. We propose the following two methods for controlling the exploration, and show later on that they lead to smaller regret compared to plain Thompson Sampling.

1. **Controlled sampling with stopping criterion:** We introduce a stopping criterion to restrict the growth of regret due to prolonged sampling. The stopping criterion compares the optimal price for the latest estimated parameters with the average of optimal prices for estimated parameters from the previous five iterations. If these two prices are close enough, the algorithm stops sampling and starts suggesting the greedy price according to the latest parameter estimate.

2. **Controlled sampling by varying σ:** Another alternative for controlling unnecessary sampling is to change σ in the update equations (1). A smaller value for σ causes the covariance matrix of the posterior distribution to converge to zero faster. This helps in restricting unnecessary exploration, and potentially gives smaller regret.

4. Regret comparison with baseline methods

We implemented all the learning algorithms as described in Section 3 along with the baseline methods (ILS and CILS) for dynamic pricing. We call these the standard form implementation, and the resulting regret plots are displayed on the left side in figures 2, 3 and 4 (the abbreviation std is used for the standard form implementation in the legends).

We also consider the effect of modifying each algorithm either by restricting exploration through the use of the stopping criterion or, in the case of TS, tuning σ, or by doing an initial least squares fit on revenues observed at prices chosen from a barycentric spanner. Note that barycentric spanners are used for efficient exploration (see Awerbuch and Kleinberg (2004), Hazan and Karnin (2016)), and can be computed using the algorithm in Awerbuch and Kleinberg (2004). The regret plots resulting from these modifications are displayed on the right side in figures 2, 3 and 4. (The abbreviation "imp" in the legends indicates the modified version of the algorithm. In case of TS, "stop" and "0.1" indicate the controlled exploration via stopping criterion and tuning with $\sigma = 0.1$ respectively.) All plots for expected cumulative regret are obtained by averaging over 10 runs.

Figure 2 shows the cumulative regret for the case where the true revenue curve is the second degree polynomial arising from the linear demand function considered in Keskin and Zeevi (2014): $r = 1.1p - 0.5p^2 + \xi$, where ξ is a zero-mean Gaussian noise sample with $\sigma = 0.1$. Apart from regret plots, we also include results for a few robustness checks as described below.

1. **Different polynomial degrees:** We compared the algorithms for various degrees of the true polynomial revenue function. In Figure 2, we plot the regret for the case when the true revenue function is a fourth degree polynomial. Specifically, the price p generates the revenue $r = -p^4 + 22p^3 - 165p^2 + 480p - 150 + \xi$, where ξ is a zero-mean Gaussian noise sample with $\sigma = 10$.

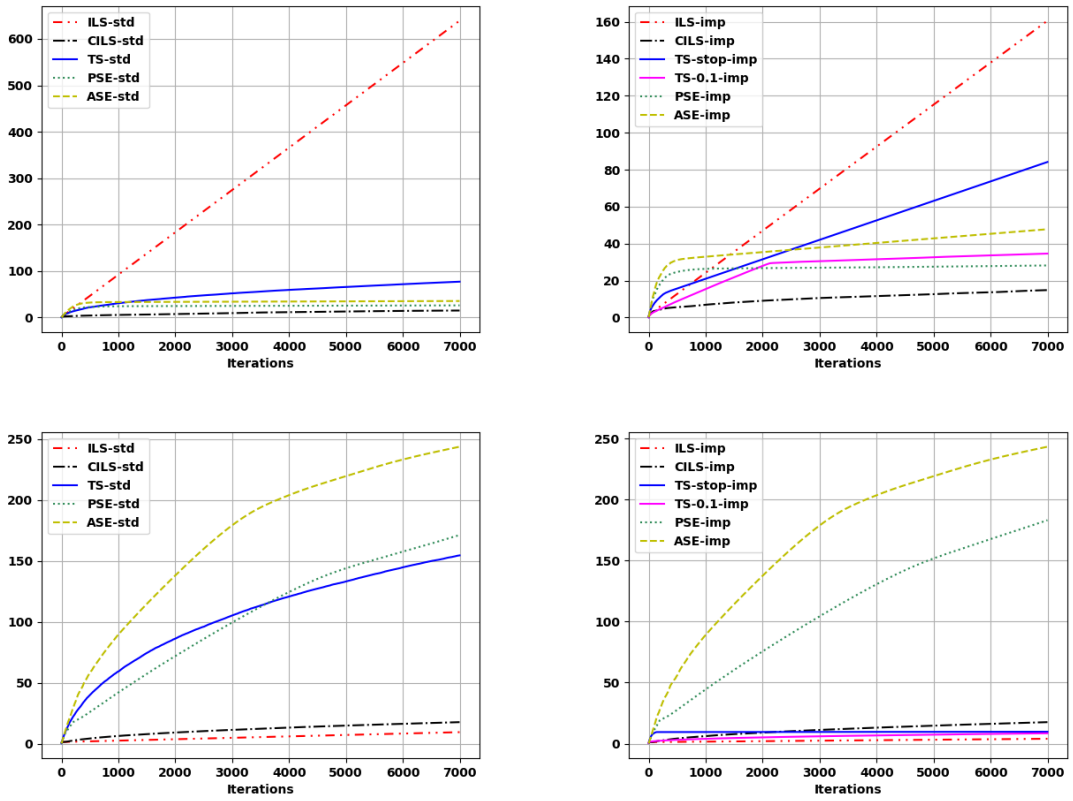

Figure 2: Expected cumulative regret with 2nd (top) and 4th (bottom) degree polynomial.

2. **Different degree for the polynomial vs model:** We also did a wide range of experiments when the degree of the polynomial that needs to be learnt is different from that of the assumed model. Figure 3 presents experimental regret plots for the case where the second and fourth degree polynomials given above are learnt using fourth and second degree models, respectively.

3. **Non polynomial models:** Figure 4 shows the regret comparison for the case where the assumed model for the revenue curve is a polynomial, but the true revenue function is the radial basis function: $100 \cdot e^{-(p-5)^2/20}$ with zero mean Gussian noise ($\sigma = 3$).

5. Concluding remarks

An initial least squares fit using barycentric points decreases the regret of ILS (see Figure 2, 3 and 4). Figure 2 also shows that when the degree of the polynomial to be learnt is smaller, then all the methods perform equally well, but as the degree increases, the ASE and PSE deviate from the other methods. Also, in the case when the degree of the true revenue curve is lower than that of the model (see Figure 3), most of the methods are learning the true optimal. However, TS-stop-imp seems to stop at a sub-optimal point resulting in higher regret. On the other hand, in the case where the degree of the true revenue curve is higher than the model degree, only TS-0.1-imp seems to learn the true optimal as seen from the

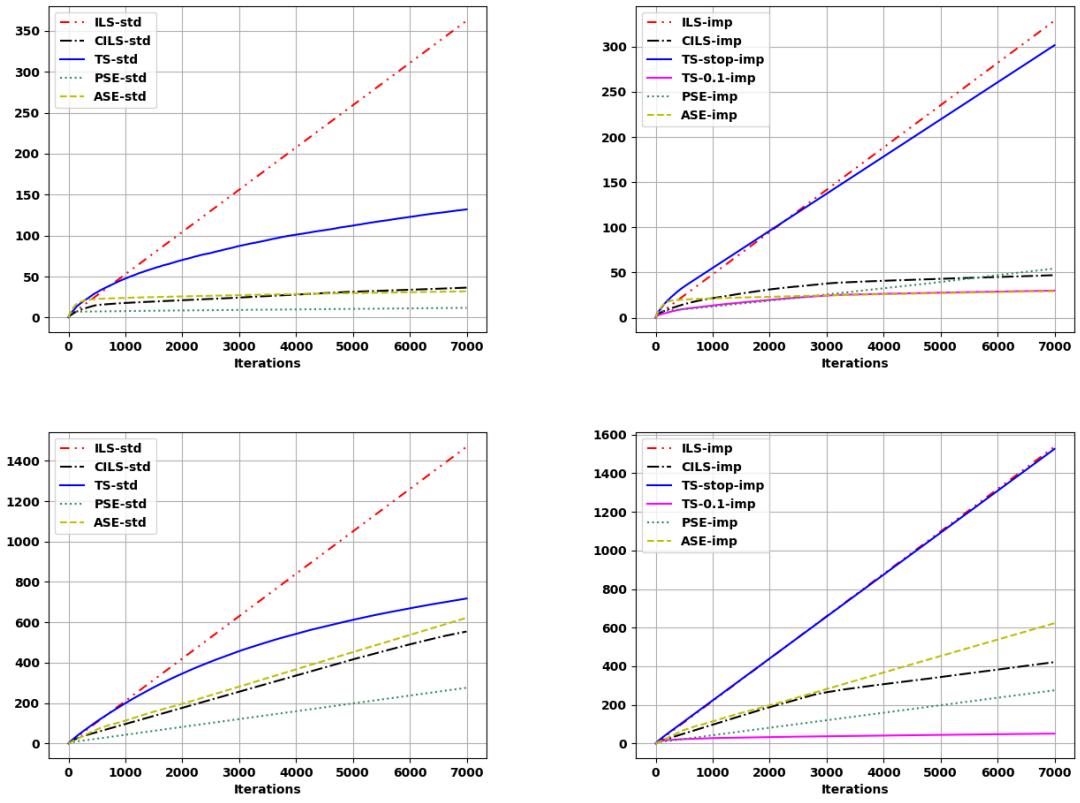

Figure 3: Expected cumulative regret when 2nd degree polynomial is learnt by a 4th degree model (top) and vice versa (bottom).

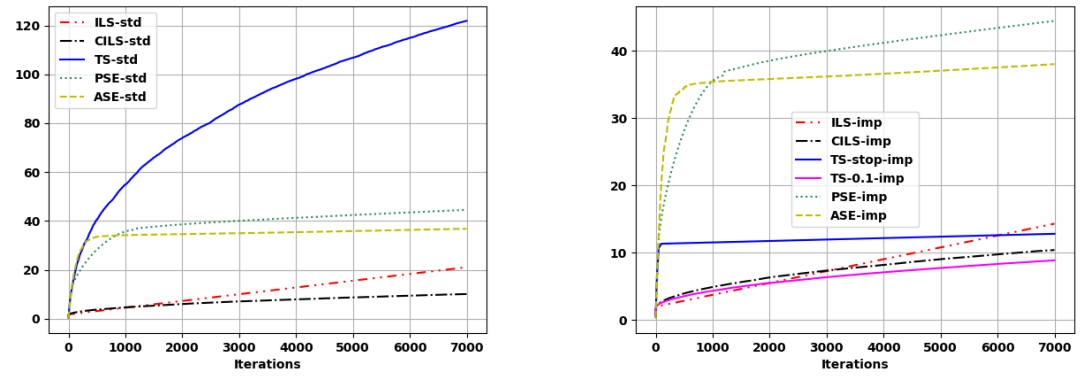

Figure 4: Expected cumulative regret for radial basis function learnt using a 4th degree polynomial.

flat regret. In contrast, the linearly increasing regret for the other methods indicates that all other methods fail to learn the optimal value (see bottom part of Figure 3). We conclude that, on average standard CILS seems to be performing well among standard methods, while modified versions of TS and CILS perform better than modifications of the other methods.

References

Baruch Awerbuch and Robert D Kleinberg. Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 45–53, 2004.

Drew Bagnell. Bayesian linear regression description. `http://www.cs.cmu.edu/~16831-f14/notes/F14/16831_lecture20_jhua_dkambam.pdf`, 2005. Accessed: 2020-06-08.

Arnoud V den Boer and Bert Zwart. Simultaneously learning and optimizing using controlled variance pricing. *Management science*, 60(3):770–783, 2014.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Elad Hazan and Zohar Karnin. Volumetric spanners: an efficient exploration basis for learning. *The Journal of Machine Learning Research*, 17(1):4062–4095, 2016.

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

Atsushi Miyamae, Yuichi Nagata, Isao Ono, and Shigenobu Kobayashi. Natural policy gradient methods with parameter-based exploration for control tasks. In *Advances in neural information processing systems*, pages 1660–1668, 2010.

Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Thomas Rückstiess, Frank Sehnke, Tom Schaul, Daan Wierstra, Yi Sun, and Jürgen Schmidhuber. Exploring parameter space in reinforcement learning. *Paladyn, Journal of Behavioral Robotics*, 1(1):14–24, 2010.

Anirudh Vemula, Wen Sun, and J Andrew Bagnell. Contrasting exploration in parameter and action space: A zeroth-order optimization perspective. *arXiv preprint arXiv:1901.11503*, 2019.

Joannes Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *European conference on machine learning*, pages 437–448. Springer, 2005.

Huazheng Wang, Ramsey Langley, Sonwoo Kim, Eric McCord-Snook, and Hongning Wang. Efficient exploration of gradient space for online learning to rank. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 145–154, 2018.

Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

Appendix A. Pseudo-code for parameter space exploration

Algorithm 1: Parameter space exploration for dynamic pricing

Input: True weight vector $\tilde{\mu} = [\tilde{\mu}_0, \tilde{\mu}_1, \tilde{\mu}_2, \cdots, \tilde{\mu}_n]$, total_iterations, noise σ^2 and learning rate α.
Initialization: Set $D = \{p := (1, p, p^2, \cdots, p^n), p \in [p_{\min}, p_{\max}]\}$, initialize $\hat{\mu}_0$ and $\hat{\sigma}_0$.
Step 1. Initial weight vector $w_0 = \hat{\mu}_0 + \hat{\sigma}_0 \circ \hat{\epsilon}$, where $\hat{\epsilon} \sim \mathcal{N}(0, 1)$.
Step 2. Set $g_0(p) = w_0^T p$ and find $p_0^* = \arg \max_{p_{\min} \le p \le p_{\max}} g_0(p)$.
Step 3. Set $p_0 = [1, p_0^*, (p_0^*)^2, \cdots, (p_0^*)^n]$ and $t = 0$.
while $t \le$ total_iterations **do**
Environment: $r_t \leftarrow$ Environment(p_t^*); this implies $\tilde{\mu}^T.p + \xi$, where $\xi \sim \mathcal{N}(0, \sigma^2)$.
Loss function: $\mathcal{L}(\hat{\mu}, \hat{\sigma}) = \frac{1}{t} \sum_{i=1}^{t} (w_t^T p_i - r_i)^2$
Learning: $\hat{\mu}_{t+1} = \hat{\mu}_t - \alpha \cdot \nabla \mathcal{L}_{\hat{\mu}}$ and $\hat{\sigma}_{t+1} = \hat{\sigma}_t - \alpha \cdot \nabla \mathcal{L}_{\hat{\sigma}}$.
Sampled weight vector: $w_{t+1} = \hat{\mu}_{t+1} + \hat{\sigma}_{t+1} \circ \hat{\epsilon}$, where $\hat{\epsilon} \sim \mathcal{N}(0, 1)$.
Optimization: Set $g_{t+1}(p) = w_{t+1}^T p$ and find $p_{t+1}^* = \arg \max_{p_{\min} \le p \le p_{\max}} g_{t+1}(p)$.
Set $t \leftarrow t + 1$.
end while

Appendix B. Pseudo-code for Thompson sampling

Algorithm 2: Thompson sampling for dynamic pricing

Input: True weight vector $\tilde{\mu} = [\tilde{\mu}_0, \tilde{\mu}_1, \tilde{\mu}_2, \cdots, \tilde{\mu}_n]$, total_iterations, noise σ^2.
Initialization: Set $D = \{p := (1, p, p^2, \cdots, p^n), p \in [0, 1]\}$.
Step 1. Find barycentric spanner $b_1, b_2, \cdots b_{n+1}$ for D.
Step 2. Set $A_0^{-1} = \sum_{i=1}^{n+1} b_i b_i^T$ and sample $w_0 \sim \mathcal{N}(0, A_0)$.
Step 3. Set $g_0(p) = w_0^T p$ and find $p_0^* = \arg \max_{p_{\min} \le p \le p_{\max}} f_0(p)$.
Step 4. Set $p_0 = [1, p_0^*, (p_0^*)^2, \cdots, (p_0^*)^n]$ and $t = 0$.
while $t \le$ total_iterations **do**
Environment: $r_t \leftarrow$ Environment(p_t^*); this implies $\tilde{\mu}^T.p + \xi$ with $\xi \sim \mathcal{N}(0, \sigma^2)$.
Learning: $A_{t+1}^{-1} = A_t^{-1} + \frac{p_t {p_t}^T}{\sigma^2}$, $A_{t+1}^{-1} \mu_{t+1} = A_t^{-1} \mu_t + \frac{r_t p_t}{\sigma^2}$.
Sampling: $w_{t+1} \sim \mathcal{N}(\mu_{t+1}, A_{t+1})$ and set $g_{t+1}(p) = w_{t+1}^T p$.
Optimization: Find $p_{t+1}^* = \arg \max_{p_{\min} \le p \le p_{\max}} g_{t+1}(p)$.
Set $t \leftarrow t + 1$.
end while