Single-Agent Dynamic Pricing with Reinforcement Learning

Div Dasani

Advised by Jeremy Watt
Adjunct Assistant Professor, Northwestern University

Abstract

Dynamic pricing is a highly researched problem in computational economics with several real-world implications. This paper presents the advantages of dynamic pricing and describes a modelling implementation. A dynamic pricing algorithm is proposed via a reinforcement learning framework. The algorithm is trained on generated sample data, analyzed against a baseline economic model, and scrutinized under situations in which the underlying data possesses high levels of noise.

1 Introduction

Dynamic pricing is the practice of varying the price for a product or service based on consumer or market attributes. At its core, the problem of dynamic pricing reflects a tradeoff between profit per unit, which increases with price, and demand for the good, which decreases with price. As a result, the problem of setting the optimal pricing policy, particularly in an automatic and adaptable manner, to maximize the overall utility of a supplier is an ongoing research topic in the fields of machine learning and computational economics.

As a motivating example, consider the differences in the revenue models between television network providers, such as NBC or CBS, and online video streaming platforms, such as YouTube or Vimeo. Both of these groups sell the product of consumer attention to advertisers, wherein these companies are paid in exchange for displaying advertisements to N viewers. Let $U(k)$ denote the utility attained by an advertiser to advertise to k consumers, and $P(k)$ denote the price paid by an advertiser to advertise to k consumers. The former group serves all of its viewers the same advertisement, regardless of the viewers' interest in the advertised product. Therefore, an actor attempting to advertise on a television platform will pay a cost $P(N)$, but will only gain a utility of $U(n^a < N)$, since they will likely only be able to favorably effect the purchasing behavior in a small subset of consumers given the diverse characteristics of the broader population. In contrast, a video streaming platform couples the employment of consumer characteristic data with the serving of content on an individualized basis to serve targeted ads. This gives advertisers the ability to display their advertisements only to the subset of the population that shares characteristics with their target demographic. As a result, actors are better off advertising on video streaming platforms, since they are awarded a utility of $U(n^b)$ and pay only a cost $P(n^b)$, where $\frac{U(n^b)}{P(n^b)} > \frac{U(n^a)}{P(N)}$. Thus, a video streaming platform can dynamically price its advertising space by charging different actors different costs to advertise depending on how closely each consumer aligns with desired characteristics for each advertiser.

This is only one example for the use of dynamic pricing, but this method of pricing goods can be similarly utilized in a variety of fields, such as virtual gaming or salary negotiation. This paper sheds light on dynamic pricing approaches for economic agents in single-agent settings by exploring the methods and advantages of dynamic pricing using a powerful automatic control scheme powered by reinforcement learning, a type of machine learning technique[1].

2 Related Work

Both dynamic pricing and reinforcement learning are highly researched fields, and there are

1. Code repository publicly available at https://github.com/divdasani/Dynamic-Pricing

many authors working on related problems in either or both of these spaces.

Narahari et al. [1] survey different models of dynamic pricing, focusing on models based on inventory or auctions. The authors motivate the practice of dynamic pricing and elaborate on the conditions in a market necessary for dynamic pricing to succeed. Their paper concludes with a detailed case study that illustrates the implications of dynamic pricing.

Employing the motivation of the economic software agent and the information economy, Kephart et al. [2] survey research conducted internally by IBM to understand the potential impact of pricing agents on prices in digital economies such as e-commerce platforms. The authors note several potential harmful effects that could result from the widespread usage of such agents, and propose solutions to remedy these behaviors.

Busoniu et al. [3] provide a broad overview of reinforcement learning, describing multi-agent systems and their advantages over more traditional agents in detail. The authors delve into the usage of these systems in broad fields ranging from telecommunications to robotics. The paper is concluded by identifying open issues and potential directions for future research.

All of these works are tangentially related to the problem of dynamic pricing in the context of this paper. However, this research covers the advantages of a reinforcement learning approach to the dynamic pricing problem while developing a comprehensive and robust model designed to be applicable across a wide range of fields, which has not yet been explored in this field.

3 Sample Generation

In this research, a sample intended to represent a hypothetical consumer base is generated and a dynamic pricing approach is contrasted with the static pricing approach dictated by standard microeconomic theory. Each consumer $i \in [1, N]$ is represented as a vector $(\boldsymbol{X_i}, p_i)$, where $\boldsymbol{X_i} = (x_{i,1}, \dots, x_{i,J})$ represents the characteristic vector of consumer i, and p_i represents consumer i's value of the good. Let $\boldsymbol{X} \in \mathbb{R}^{N \times J}$ then denote the sample matrix and $\boldsymbol{P} \in \mathbb{R}^N$ denote the price vector, where each row corresponds to the characteristic information and price of one consumer, respectively. To generate data in this format, the characteristic vector is first randomly generated as a categorical vector. A weight vector $\boldsymbol{w^G} = (w_0, w_1, \dots, w_J)$ is also created, where w_0 is a bias term and w_j for $j \in [1, J]$ represents the effect that possessing characteristic j has on one's value of the good. For example, if characteristic j denoted whether a given consumer is male, a negative value of w_j would indicate that male consumers on average value the good less than female consumers. To generate p_i for each consumer i, the parameter α_i is first calculated by taking the dot product of the weight vector $\boldsymbol{w^G}$ and $[1; \boldsymbol{X_i}]$. p_i is then randomly sampled from $\Gamma(\alpha_i, \sqrt{\alpha_i})$, where Γ denotes the gamma distribution. This process is repeated for all i to construct $\boldsymbol{X}$ and $\boldsymbol{P}$.

This sample generation method was selected for several reasons. Firstly, randomly generating characteristic data for each consumer allows these characteristics to remain arbitrary, maintaining the generality of the model. Additionally, by multiplying the characteristic information with a weight vector to attain the α parameter, the model can be fine tuned by modifying the significance of the effect of each characteristic on consumer behavior. Furthermore, drawing p_i from a distribution defined by the characteristics of consumer i allows for the existence of a correlation between these characteristics and the generated value, which mimics real consumer behavior, but still allows for stochasticity in the data. Finally, the gamma distribution is utilized because its probability density function plausibly mimics real consumer behavior while maintaining the desired support of $(0, \infty)$.

For illustration purposes, two hypothetical price distributions are depicted in *Figure 1* for $\alpha = 25$ and $\alpha = 50$.

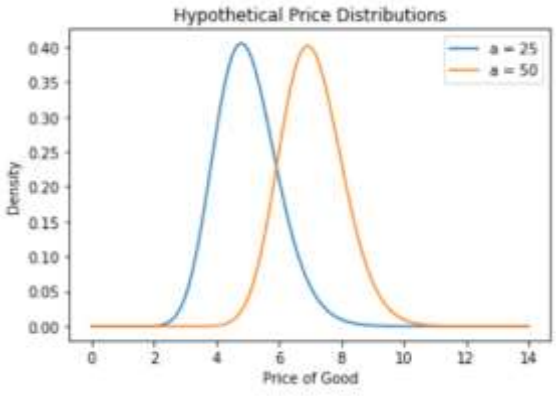

Figure 1: Hypothetical price distributions with parameters $\alpha = 25$ (left) and $\alpha = 50$ (right)

4 Model and Implementation

For the purposes of implementation, in this paper, the sample size of consumers, N, was capped at 2500, and the number of characteristics measured J, was set to 4. In practice, actors typically have much larger consumer bases and measure far more characteristics of these consumers, but this paper employs a limited context to illustrate the ability of dynamic pricing even in restricted settings.

Both models receive 80% of the data to train on, in which they are provided with both characteristic and value information for each consumer, and are then evaluated with the remaining 20% of the data, in which they are only provided with characteristic information and expected to predict value information. These models are evaluated on the average profit per person (APP) metric. This metric is calculated as,

$$APP(\widehat{\boldsymbol{P}}) = \frac{1}{N}\sum_{I=1}^{N} (\max(\hat{P}_i, c) - c) * I(p_i > \max(\hat{P}_i, c))$$

where $\widehat{\boldsymbol{P}}$ is the predicted price vector, c is the unit cost of producing each good, and I is the indicator function whose value is 1 if the condition is met and 0 otherwise. Intuitively, the metric rewards the model for guessing a price $\hat{P}_i$ for consumer i that is as high as possible, so long as $\hat{P}_i < p_i$, since the consumer will not purchase the good if its price is greater than her value for the good. If the model predicts a $\hat{P}_i < c$, the metric charges the consumer c, because any value less would result in a negative profit value for consumer i.

4.1 Static Price Profit Maximization

The static profit maximization method is used as a benchmark to analyze the dynamic price profit maximization method discussed in the next section. The demand for a good at price p is given by,

$$D(p) = \sum_{i=1}^{N} I(p_i > p)$$

The demand curve for the population is shown in *Figure 2*. The static pricing algorithm is trained as governed by microeconomic theory. That is, the selected price, p^* is the one that maximizes the profit equation,

$$\pi(p) = (p - c) * D(p)$$

A plot of profit vs. price is given in *Figure 3*. According to the figure, $p^* = 6.0$, where the APP is metric is maximized at 3.68.

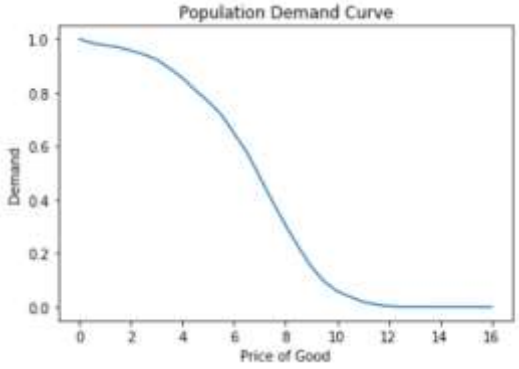

Figure 2: Demand (as % of population) as a function of price for generated sample

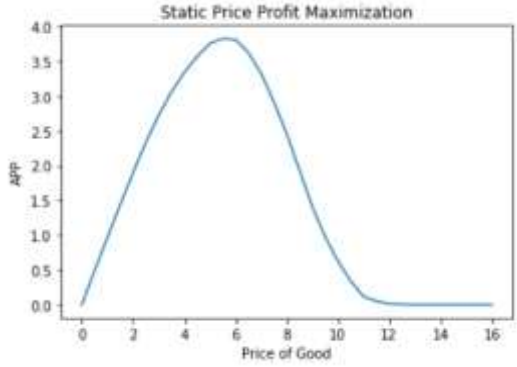

Figure 3: Price of good vs. APP using static price profit maximization method

4.2 Dynamic Price Profit Maximization

To dynamically price the good for different consumers, an automatic controller will be trained to recognize patterns that map values from the characteristic space to the value space. The algorithm will be trained using a loss function, which will penalize it for predicting incorrect values. In this case, the L2 loss function is employed, which is given by,

$$L(\widehat{\boldsymbol{w}}) = \sum_{i=1}^{N} (\boldsymbol{X}\widehat{\boldsymbol{w}} - \boldsymbol{P})_i^{\,2}$$

Essentially, the automatic controller is attempting to construct the vector $\widehat{\boldsymbol{w}}$ that minimizes the difference between the predicted price vector $\widehat{\boldsymbol{P}} = \boldsymbol{X}\widehat{\boldsymbol{w}}$ and the actual price vector $\boldsymbol{P}$. The algorithm is

trained iteratively through a commonly used process known as gradient descent, where the penalty or loss assigned to the function at each timestep is given by the sum of squared distances between the actual price and predicted price for each consumer. Note that the algorithm is not given any information regarding $\boldsymbol{w^G}$, the sample-constructed weight vector, nor information regarding the distribution from which prices are generated for consumers.

The automatic controller is trained with loss function L and a learning rate of 0.1 across 500 iterations of gradient descent. The loss over time is depicted in *Figure 4*.

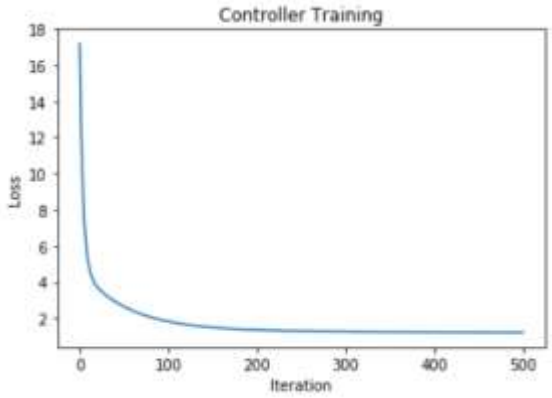

Figure 4: Loss over time for automatic controller

After the algorithm completes its training, it outputs a $\widehat{\boldsymbol{w}}^*$ that minimizes L. The APP metric can then be calculated for this model as $APP(\boldsymbol{X}\widehat{\boldsymbol{w}}^*)$, which returns a value of 3.24.

4.2.1 The Dampening Parameter

Sections 4.1 and 4.2 demonstrate that the dynamic pricing algorithm generated a lower APP value than its static pricing counterpart. Though this may seem counterintuitive at first, it is actually expected. The loss function L works to minimize the absolute difference between $\boldsymbol{X}\widehat{\boldsymbol{w}}$ and $\boldsymbol{P}$, meaning that it would prefer a weight vector $\widehat{\boldsymbol{w}}^{\boldsymbol{a}}$ that produces a difference $\sum_{i=1}^{N}(\boldsymbol{X}\widehat{\boldsymbol{w}}^{\boldsymbol{a}} - \boldsymbol{P})_i = \varepsilon$ over $\widehat{\boldsymbol{w}}^{\boldsymbol{b}}$ that produces $\sum_{i=1}^{N}(\boldsymbol{X}\widehat{\boldsymbol{w}}^{\boldsymbol{b}} - \boldsymbol{P})_i = -2\varepsilon$ for some arbitrary small $\varepsilon > 0$. However, evaluating both of these values with the APP metric reveals that,

$$APP(\widehat{\boldsymbol{w}}^{\boldsymbol{b}}) \geq APP(\widehat{\boldsymbol{w}}^{\boldsymbol{a}}) = 0$$

This is because the use of $\widehat{\boldsymbol{w}}^{\boldsymbol{a}}$ predicts prices that are higher than the value held by consumers for the good, so they will not purchase the good at the predicted price, resulting in an APP of 0. Meanwhile, though the use of $\widehat{\boldsymbol{w}}^{\boldsymbol{b}}$ predicts prices that are slightly farther from the true values than those predicted by $\widehat{\boldsymbol{w}}^{\boldsymbol{a}}$, they are lower than those prices, which allows the APP metric to reap the difference between the predicted price and the cost per unit as profit, since consumers will purchase the good.

Thus, the reason the dynamic pricing algorithm is underperforming is due to the high likelihood for slight overestimation in price prediction. To counteract this effect, a dampening parameter $d \in (0,1]$ is employed to artificially dampen the offered price from $\boldsymbol{X}\widehat{\boldsymbol{w}}$ to $d * \boldsymbol{X}\widehat{\boldsymbol{w}}$. As a result, even if the use of $\widehat{\boldsymbol{w}}$ results in a slight overestimation of $\boldsymbol{P}$, the dampening parameter will maximize the likelihood that $d * \boldsymbol{X}\widehat{\boldsymbol{w}} \leq \boldsymbol{P}$ while minimizing $\boldsymbol{P} - d * \boldsymbol{X}\widehat{\boldsymbol{w}}$. To select the optimal value of d, several values on the interval $[0.7,1]$ are plotted, and the APP metric is recalculated on the training data for each instance. The results are illustrated in *Figure 5*.

As visualized by the figure, the training APP metric is maximized at 4.68, for which $d = 0.80$. By fixing d at this value and computing the APP metric on the test data, a value of 4.65 is returned. Therefore, the increase in performance by the dynamic pricing model from the static pricing model is given by,

$$\frac{APP_D}{APP_S} - 1 = 0.264 = 26.4\%$$

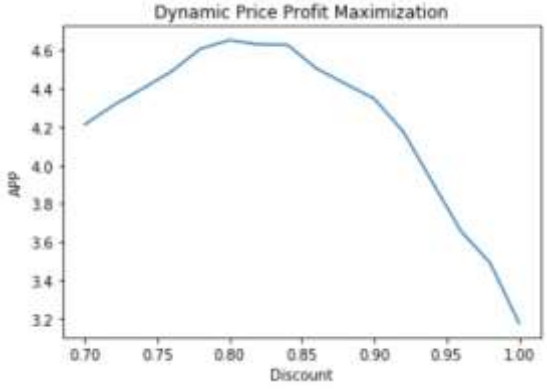

Figure 5: APP as a function of the dampening parameter

4.2.2 Model Performance and Price Distribution Variance

As described in Section 3, the price values for each consumer is generated from the gamma distribution with shape parameter α_i and rate parameter $\sqrt{\alpha_i}$. As a result of this design, the distributions had an expected value and variance given by,

$$E[p_i] = \frac{\alpha_i}{\sqrt{\alpha_i}} = \sqrt{\alpha_i}$$

$$\mathrm{Var}(p_i) = \frac{\alpha_i}{(\sqrt{\alpha_i})^2} = 1$$

Though this construction is convenient in reducing the mathematical complexity of the model, a fixed variance is unlikely to be exhibited in real-world instances of price distributions.

Therefore, the comparative performance of dynamic pricing relative to static pricing was investigated as the underlying variance of the price distribution was allowed to increase, and the results are shown in *Figure 6*.

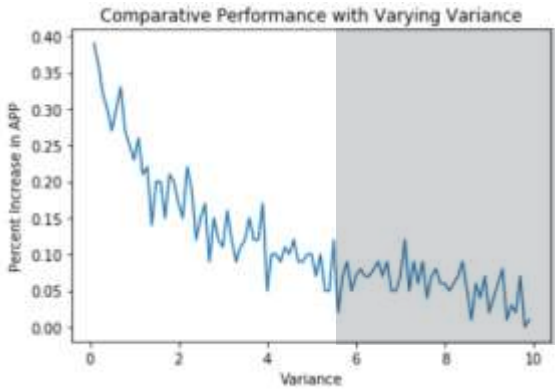

Figure 6: Comparative performance of dynamic pricing algorithm vs. variance of price distribution. Statistically insignificant differences highlighted in gray.

As the variance of the price distribution increases, the deterministically calculated α value has less authority over the stochastically determined price value p, increasing the difficulty of isolating the effect of the characteristics from the noise for the dynamic pricing algorithm. Thus, as suggested by *Figure 6*, the comparative edge of the dynamic pricing algorithm drops exponentially as the variance of the price distribution increases. The area of the graph shaded in gray indicates levels of variance where the comparative edge of the dynamic pricing algorithm is no longer statistically significant.

The dynamic pricing scheme is able to generate a strong comparative advantage with low variance, which is exemplified in Section 4.2.1, but still performs well in high-variance scenarios. According to *Figure 6*, even as the variance is 4.5, which encapsulates nearly 30% of the overall price range, the dynamic pricing algorithm is still able to learn well enough to perform 10% better than its static pricing counterpart. The automatic controller's ability to outperform the baseline model while remaining robust to high levels of noise allows it to perform well in real-world settings.

5 Conclusion

The practice of dynamically pricing goods is a particularly powerful in certain environments. In particular, products and services that are individualized or non-interchangeable lend themselves to be dynamically priced, as demonstrated in Section 1. Additionally, possessing characteristic data on the attributes of consumers allows actors to leverage this data to build highly sophisticated dynamic pricing models. This is best achieved with reinforcement learning, a robust machine learning framework whose attributes include immense predictive power even on relatively small datasets, and the ability to recognize patterns despite high levels of noise in the underlying data.

References

[1] Narahari, Y. et al., "Dynamic pricing models for electronic business" *Sadhana* (2005).

[2] Kephart, Jeffrey et al., "Dynamic Pricing by Software Agents" *Brown University* (2000)

[3] Busoniu, L. et al., "Multi-agent reinforcement learning: An overview" *Innovations in Multi-Agent Systems and Applications* (2010)