DYNAMIC PRICING ON E-COMMERCE PLATFORM WITH DEEP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic pricing problem has been studied for decades and varieties of methodologies were developed under different assumptions. We developed an approach based on deep reinforcement learning (DRL) to address the dynamic pricing problem on an E-commerce platform with few assumptions. This paper first modeled dynamic pricing as a Markov Decision Process, defined various reward functions, with both discrete and continuous pricing action space. And then it introduced the methods to pre-train the model with the historical sales data. Offline evaluations and field experiments were designed and conducted to validate our approach.

1 INTRODUCTION

Dynamic pricing, often referred to revenue management as well, is to adjust prices according inventories left and demand response observed from time to time to maximize revenue. It has draw great attentions during the past decades since the deregulation of the airline industry in the 1970s. A significant amount of research has been conducted to derive varieties of methodologies for optimal pricing policies for airline tickets. Weatherford & Bodily (1992) and Talluri & Van Ryzin (2006) an overview of the research that has been done in the field of perishable-asset revenue management, which is a field that combines the areas of yield management, overbooking, and pricing. Airline operators adjust air-ticket prices dynamically based on number of seats left and demand forecasting.

During the recent development of business, many industries has become more active in revenue management. Riding-share platforms like Uber has implemented dynamic pricing strategy, known as 'surge' pricing and Chen & Sheldon (2016) showed that it has significant impact on motivations for more driving times. Retailers like Zara has implemented systematic dynamic markdown pricing strategy. Caro & Gallien (2012) studied the clearance pricing for Zara to increase the revenue generated by each item while maintaining a large number of items to sell. Kroger is now testing electronic price tag at one store in Kentucky (Nicas (2015)).

Online retailers have a stronger desire for dynamic pricing strategies due to more complex operations. Amazon.com sells 356 million products (562 million now), Taobao.com sells more than 2 billion products and Walmart.com sells 4.2 million products according to a 2017 estimate[1]. Operation specialists have to set prices for these items periodically to remain competitive, which will be mission impossible when the number of items goes this high. As a result, Amazon has implemented automatic pricing systems and it is reported that Amazon.com can change prices every 15 minutes[2]. Chen et al. (2016) studied the pricing strategies for Amazon.com empirically and derived significant factors for pricing.

In this paper, we proposed a reinforcement learning approach to address the dynamic pricing problem for online retailers. The case study we selected is to set price dynamically for items sold on `https://chaoshi.tmall.com` (known as Tmall Supermarket) which is a brand for the E-commerce platform Tmall.com. For online-platform retailers like Tmall.com and Amazon.com, many of the products to sell are not procured by the platform but owned by a third party retailer and as a result, the third party retailers may not be able to utilize data accumulated on the platform and set optimal prices. The platform would like to provide automatic pricing tools to third party retailers to improve their operations efficiency. There are many difficulties for pricing by a platform. First, it

[1] https://www.scrapehero.com/how-many-products-does-walmart-com-sell-vs-amazon-com
[2] https://www.whitehouse.gov/sites/default/files/docs/Big_Data_Report_Nonembargo_v2.pdf

is hard to know the procurement cost for items since third party retailers are not willing to share this information with platforms; it might be hard to know the appropriate pricing space. Second, promotions are highly active in Chinese E-commerce market, which introduces many exceptional cases of sales and price records in the historical data, e.g. the sales volume for a product of fast noodles with daily sales of 10 may go up to 10000 by having a discount of 90% during the promotion day. Third, unlike brick-and-mortar retailers which have more static allocations of shelf space, the display of items may change dramatically from day to day since recommendation systems may adjust customer preference from day to day. That will significantly impact customer traffic of a single item and lead to dramatic sales volumes in different days even of the same price. Forth, unlike recommendation system, it is impossible to do online A/B testing since it is illegal to expose different prices to different customers for the same product. That is a major obstacle to measurement of performances of different pricing strategies.

To overcome these difficulties, we propose a framework of pricing strategy and develop a DRL model to optimize long-term revenue and sales. Within the framework, the platform Tmall Supermarket negotiates with third party retailers, asking them to set up lower bounds for prices of items. We call them red-line prices. Then the system is able to set prices automatically with the restriction of the red-line prices. To capture the relationship between prices and demand response of customers, we defined different reward functions for different E-commerce pricing scenarios. Among these reward functions, the one defined by change of revenue (rather than sales volume itself), showed good potential for a good pricing policy. Very few assumptions are made for the model: we assume the demand for each item is independent of each other; the demand function is unknown in advance and not necessarily stationary overtime, the revenue function is not necessarily convex or concave; and customer behaviors could be strategic.

Our main contributions of this paper are summarized as follows:

- To our knowledge, it is the first time dynamic pricing problem is modelled comprehensively by deep reinforcement learning aiming to maximize long-term revenue.
- A systematic demonstration to train and test models based on historical sales data is developed and presented.
- A methodology based on deep reinforcement learning is designed, and many more features are extracted to construct the network, which is different from many research works on reinforcement learning for dynamic pricing.
- Comparisons among deep reinforcement learning and MAB methods are conducted based on the methods introduced previously and field experiments.
- An offline assessment is conducted for pre-evaluation and an online assessment approach is designed to overcome the difficulty for not being able to do A/B test.
- Different approaches are implemented and validated in the online environment and evaluated by real data, which proves that our approach outperforms other approaches.

2 LITERATURE REVIEW

Much research has been done in revenue management for decades. We refer to den Boer (2015) for a comprehensive review for recent developments in dynamic pricing. It combined two research fields: (1) statistical learning, specifically applied to the problem to estimate demand and (2) price optimization. Most of prior research has focused on the cases where a functional relationship between price and demand is assumed, to be known to decision makers. Cournot (1897) is acknowledged to be the first to mathematically described the price-demand relation of products and solve the mathematical problem to achieve the optimal price. However, it assumed that the relationship is static over time which usually does not hold true in reality. Evans (1924) assumes that the demand is not only a function of price, but also the time-derivative of price, leading to a dynamic demand function of price over time. Kamrad et al. (2005) introduced a stochastic model to capture demand uncertainty while optimizing the prices. Gallego & Van Ryzin (1994) considers constraints like limited inventories and a finite planning horizon.

In practice, decision makers do not know the exact demand function. Much recent research focuses on the dynamic pricing with unknown demand function. We see that later study addresses

the problem with unknown demand function. Researchers first address the problem by parametric approaches. Bertsimas & Perakis (2006) assumes parametric families of demand functions to be learned over time. Farias & Van Roy (2010) proposes an approach to learn from the historical purchase data. Harrison et al. (2012) utilizes Bayesian dynamic pricing policies to address demand uncertainties. However, revenue may depart from the optimal due to mis-specifying the demand family. Therefore, much recent research mainly revolves around non-parametric approaches. Besbes & Zeevi (2009), Besbes & Zeevi (2015), Wang et al. (2014) looks deep inside learning while earning approaches; and Wang et al. (2014) announces to have the smallest gap so far. However, they all assume the revenue function is strictly concave and differentiable, which may not hold true in E-commerce retail industry, since the number of page visitors may fluctuate dramatically and this could lead to non-concavity. To overcome this difficulty, researchers try to use multi-armed bandit to address the problem, e.g. Misra et al. (2018).

With the development of computation, reinforcement learning is introduced to address dynamic problems. Vengerov (2007) and Kim et al. (2016) utilize reinforcement learning to optimize prices in energy market. Raju et al. (2003) aims to solve the dynamic pricing problem in retail market. Motivated by successful stories of deep reinforcement learning, e.g. Silver et al. (2016), we utilize deep reinforcement learning to optimize pricing strategy for the E-commerce platform to overcome difficulties of the non-convexity of the revenue function and unknown demand function.

The paper is organized as follows: Section 3 introduces approaches we designed for dynamic pricing, where the problem is modeled as a Markov Decision Process model. Both discrete pricing action model and continuous pricing action model are proposed. In Section 4, the results from both offline and online experiments are introduced, which validate our approach of the problem. The conclusions and future work directions are summarized in Section 5.

3 METHODOLOGY

We now consider how to build up decision making models for the dynamic pricing problem we discussed above. We first represent the problem as a Markov Decision Process (MDP). The agent periodically changes prices of the products as its action after observing environment state. The new environment state could then be observed and the reward could also be received. Each pricing episode reaches its end if the product is out of stock. Then we introduce an approach to take advantages of historical sales data and the previous operations specialists' pricing actions, to form demonstrations. And the last part of this section introduces the methodology for pre-training and evaluating the model using these demonstrations.

3.1 MARKOV DECISION PROCESS MODEL

For E-commerce dynamic pricing problems, we first define the pricing period d, which can be one day or one week. At the beginning of each period, the retailer will decide to keep or change the price for each product. Then, during each period, the prices of products are settled and keep influencing the market in different ways. For example, the prices will influence the sales and lead the customer traffic. Different kinds of data could be collected to describe the market environment during each period. At the end of each pricing period, (at time step t=1,2,...,T), the market environment could then be described by the collected data during the last period t as the state $s_t \in S$. Then, the price could be changed depending on these information, which could be regarded as an action $a_t \in A$ at time t. As a consequence of the action a_t, the information to describe the new state s_{t+1} as well as the reward $r_t \in R$ could be received from the market in the next period p_{t+1}. A decision process for product i reaches its end when the product is out of stock.

Therefore, the MDP model for E-commerce dynamic pricing problem with five-tuple $< S, A, P_a, R, \gamma >$ can be defined as:

- State Space S: the states of the market environment.
- Action Space A: the prices the retailer can periodically set in any given state.
- Transition Probability $P_a(s, s') = Pr(s_{t+1} = s'|s_t = s, a_t = a)$: the probability that the pricing action a in state s at time t leads the market environment to state s' at time $t + 1$.

- Reward Function $R_a(s, s')$: Immediate reward defined by received revenue or sales when the market environment changes from s to s' after the retailer taking pricing action a.
- Discount factor $\gamma \in [0, 1]$: the factor representing the difference in importance between future revenue and present immediate revenue.

Here in our model, each product i is priced separately, with four different types of data at time step t to describe the state $s_{i,t}$: price features, sales features, customer traffic features and competitiveness features. Price features contain the actual payment for product, the discount rate, the coupons, etc. Sales features contain the sales volume, revenue, etc. Customer traffic features contain the time the page of the product i has been viewed ($pv_{i,t}$), the number of unique user viewed the product i ($uv_{i,t}$), the number of buyers for product i, etc. The comments and the states of the similar products contribute to competitiveness features.

As for the pricing action, we first define the action space for each product i. We use the maximum price $P_{i,max}$ of the product i during a certain number of periods to define the upper bound of pricing action space. It assumes that the pricing framework should not output a price over this maximum price. As for the lower bound, we use the red-line price we introduced above as the minimum price $P_{i,min}$ of product i.

At last, we define the reward for our dynamic pricing models. We tried different ways to define the immediate reward $r_{i,t}$ that the agent could receive after taking pricing action $a_{i,t}$ at state $s_{i,t}$. We first tried to use the revenue of the product $revenue_i$ directly. But as we mentioned above, we may not be able to precisely control the customer traffic, which could obviously influences the revenue of E-commerce products. Therefore, even if we take the previous costumer traffic as features in the state, we could get total different rewards with similar state and action pairs, which could severely mislead our agents. To solve this problem, we use the revenue conversion rate, dividing $revenue_i$ by uv_i, to help define the reward for product i. This definition for reward works fine with our markdown pricing application when 1) the retailer would not raise the price but only reduce or keep it during the markdown season; 2) there is a very clear stock determining the life-cycle of the markdown process; 3) most of the markdown products are low-sales-volume products having low revenue conversion rate. However, for our another application, pricing for fast moving customer goods (FMCG), there is no limit for pricing direction, and we care more about distinguishing different prices, that to decide whether raise, reduce or keep the price facing a specific state. And, the retailer may re-order products at anytime causing no clear end point for pricing process while the FMCG daily sales volumes and revenue conversion rates could be much higher than the low-sales-volume markdown products. Therefore, with a simple reward defined by revenue conversion rate, the convergence of the model could hardly be guaranteed. To satisfy these conditions, we define a different reward function using the change of the revenue conversion rate in Eq.(1).

$$r_{i,t} = \frac{revenue_{i,t}}{uv_{i,t}} - \frac{revenue_{i,t-1}}{uv_{i,t-1}} \tag{1}$$

where $revenue_{i,t}$ and $revenue_{i,t-1}$ represent the revenue for product i during period p_t and p_{t-1} respectively, and $uv_{i,t}$ and $uv_{i,t-1}$ represent the unique user viewed the product i during period p_t and p_{t-1} respectively. In some parts of this work, we also use profit conversion rate, if we happen to have the knowledge of the inventory cost.

3.2 Discrete Pricing Action Models

In this part, we start to introduce the methods we use for solving dynamic pricing MDP we defined above. We first use multi-armed bandit (MAB, Robbins (1985)) and Q-learning (Watkins (1989)) to find the optimal policy. To apply these two methods, we divide the pricing space from $P_{i,min}$ to $P_{i,max}$ into K separated areas, as K arms in MAB or K discrete actions in Q-learning. The price $p_{i,k}$ satisfies the inequalities:

$$P_{i,min} + \frac{(P_{i,max} - P_{i,min})}{K} \cdot (k-1) \leq p_{i,k} < P_{i,min} + \frac{(P_{i,max} - P_{i,min})}{K} \cdot k \tag{2}$$

will be regarded as choosing the kth arm in MAB or kth pricing action in Q-learning for product i, $k \in [1, K]$.

Specifically, we use the LinUCB (Li et al. (2010)) as the MAB algorithm, trying to find the upper bound of the expected reward from the action facing the market environment state, which is regarded as the context. It assumes that the expected reward of a pricing arm a is linear in the d-dimensional environment feature $\mathbf{s}$ with some coefficient vector θ_a:

$$\mathbb{E}[r|s] = s^{\mathsf{T}}\theta_a \tag{3}$$

Q-learning is a value iteration method to compute the optimal policy. It starts with randomly initialed Q value and recursively iterates to get the optimal Q^* as well as the optimal policy

$$Q_{t+1}(s,a) \leftarrow (1-\alpha)\cdot Q_t(s,a) + \alpha \cdot [r + \gamma \cdot \max_{a'} Q_t(s',a')] \tag{4}$$

Here $\alpha \in (0,1]$ is the learning rate. Due to the high dimension of the state space, we use a deep network to map the Q-values from the state space, which follows the idea of deep Q-networks (DQN, Mnih et al. (2015)). To update the action value network, a one-step off-policy evaluation is used to minimize the loss function:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim D}[r + \gamma \cdot \max_{a'} Q(s',a'|\theta') - Q(s,a|\theta)]^2 \tag{5}$$

3.3 Continuous Pricing Action Model

Pricing on discrete action space encounters an obvious conflict setting the number of discrete actions. If the number is too small, different prices will be regarded as the same. At the same time, algorithms with discrete action space could only output a subspace of the pricing action space, which could be unclear if the total action space divided into few subspaces. If the number is too large, a lot of actions will not be explored in the history and the exploration in the future could also be inefficient. Therefore, we apply the actor-critic algorithm (Witten (1977)) to make pricing decision in continuous action space.

Specifically, we apply deep deterministic policy gradient (DDPG, Lillicrap et al. (2015)) as our actor-critic method. The actor takes the environment state as its input and directly output the continuous pricing action $a = \mu_\theta(s)$, while the critic takes both the state and action as input and output the action value function $Q(s,a)$. So, the loss function would be:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim D}[r + \gamma \cdot Q(s', \mu(s'|\theta^{\mu'})|\theta^{Q'}) - Q(s,a|\theta^Q)]^2 \tag{6}$$

Here D is a distribution over transitions $t = (s,a,r,s')$ contained in a replay buffer. And it takes the gradient of Q-value to update the policy network:

$$\nabla_{\theta^\mu}\mu \approx \mathbb{E}_{\mu'}[\nabla_a Q(s,a|\theta^Q)|_{a=\mu(s)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)] \tag{7}$$

3.4 Training with Demonstrations

If we directly apply LinUCB or reinforcement learning algorithms to E-commerce dynamic pricing, they will start with very poor performance and may cause huge loss. In some other areas like robotics Levine et al. (2016) and games Silver et al. (2016), there may be accurate simulators, within which the agent could learn policy. However, there is no such a simulator for dynamic pricing problem. Instead, we have enough data of the environment and the pricing decisions made by some previous controllers. These controllers could be some managers or some rules, and some of their pricing decisions may be reasonable. The records of their decision facing the environment could be regarded as demonstrations, which has been proved to be effective for pre-training the agent in Sendonaris & Dulac-Arnold (2017).

As mentioned before, the pricing actions were taken periodically. Thus, the state of the environment as well as the rewards could be represented by the data collected within these periods between actions. Therefore, we form the demonstration in tuples $< s_t, a_t, r_t, s_{t+1} >$. Here, we also use demonstrations to pre-train our LinUCB as a benchmark for DRL algorithms. Pseudo-code is sketched in Algorithm 1 (in appendix).

For reinforcement learning algorithms, we use the ideas of Deep Q-learning from Demonstration (DQfD) Sendonaris & Dulac-Arnold (2017) to form our DQN method and Deep Deterministic Policy Gradient from Demonstration (DDPGfD) Vecerík et al. (2017) for our DDPG method.

3.5 EVALUATION METHODOLOGY

In this part, we first introduce the methods to evaluate our models using offline data. We need both to evaluate the accuracy of the model during pre-training and compare the performances of the output policies. The way to evaluate the online performance of our models encountering the instability of E-commerce environment will be introduced in the next section with the online experiments result.

To evaluate with demonstrations, we first use the latest T periods of records to from tuples $< s_t, a_t, r_t, s_{t+1} >, t \in [1, T]$. And then, we divide these tuples into two parts: the first D tuples will be used for pre-training, where $t \in [1, D]$. And for $D < t < T$, the tuples will be used for evaluation. As for the MAB algorithm LinUCB, we use Eq.(4) to calculate the expectation of the reward and compare it with the real reward. The error is defined in Eq.(8):

$$e = |\frac{\mathbb{E}[r_t] - r_t}{r_t}| = |\frac{s_t^{\mathsf{T}}\theta_a - r_t}{r_t}| \tag{8}$$

For deep reinforcement learning algorithm with discrete action space, DQN, we could use Eq.(4) to calculate the expected immediate reward between two state s_t and s_{t+1} from the model:

$$\mathbb{E}[r_t] = Q(s_t, a_t) - \gamma \cdot \max_a Q(s_{t+1}, a) \tag{9}$$

and then compare it with the real reward r_t to get the error rate e:

$$e = |\frac{Q(s_t, a_t) - \gamma \cdot \max_a Q(s_{t+1}, a) - r_t}{r_t}| \tag{10}$$

While for actor-critic method, Eq.(9) and Eq.(10) should be changed into Eq.(11) and Eq.(12) respectively:

$$\mathbb{E}[r_t] = Q(s_t, a_t) - \gamma \cdot Q(s_{t+1}, \mu(s_{t+1})) \tag{11}$$

$$e = |\frac{Q(s_t, a_t) - \gamma \cdot Q(s_{t+1}, \mu(s_{t+1})) - r_t}{r_t}| \tag{12}$$

For dynamic pricing problem, we particularly concern the outcome from changing between prices. To have a well knowledge between two prices before and after pricing. The ability to tell whether a price is better than another correctly at a certain state may be more important than mapping a price to its own revenue accurately. Therefore, we evaluate the models with their ability to classify the outcome of the pricing actions, causing the increasing or decreasing of the revenue. If both of them were positive, negative or very close to zero, we take it as a right estimation of the change of the revenue.

After evaluate the models' accuracy, it is also necessary to evaluate the policy. Here we use the same evaluation method introduced by Li et al. (2010). We use the evaluation tuples and sum the rewards only if the action a is close to the output of the policy $a - \epsilon < \pi(s) < a + \epsilon$. The detail of the evaluate algorithm is sketched in Algorithm 2 (in appendix).

After applying the model online interacting with the market environment, we could then directly evaluate the policies with the total revenue from the pricing actions, comparing with benchmark group using the policy from other methods.

4 EXPERIMENTAL RESULTS

In this section we introduce the experiment results of our methods. For all of our experiments, we mainly evaluated three different algorithms for dynamic pricing: LinUCB, Deep Q-learning from Demonstrations and Deep Deterministic Policy Gradient from Demonstrations. We first introduce the offline experiments, using data from Tmall Supermarket. Then we introduce the online experiment results where we changed the prices for Tmall Supermarket on a daily basis as well as in the markdown scenario.

4.1 OFFLINE EXPERIMENTS

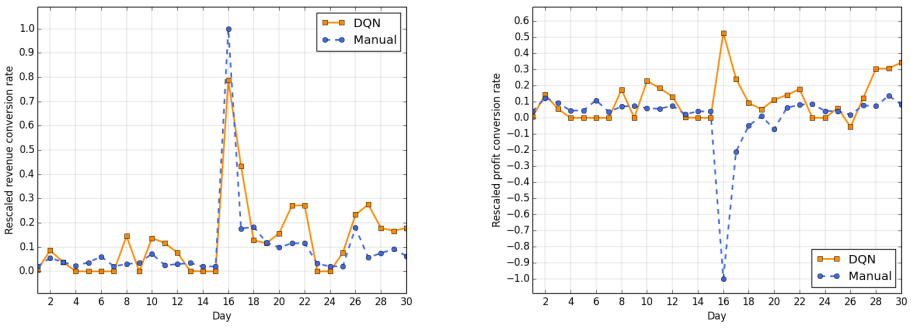
We first present the results for offline policy evaluation using the recorded data from Tmall Supermarket. We chose 40,000 different kinds of fast moving consumer products with 60 days selling records forming about 2,400,000 tuples in total. We use the first 59 days' records as demonstrations for pre-training and the last day's for evaluation. We set $K = 10$, and gradually increase γ from 0.5 to 0.99. For DDPG policy evaluation, we set $\epsilon = 0.05$. The result is shown in Figure 3 (in Appendix).

We noticed that, LinUCB outperformed DRL methods in offline policy evaluation. One of the main reasons is that, MAB methods aim to maximize immediate reward while RL methods maximize long-term reward. Therefore, in the offline policy evaluation, policy from LinUCB prefers the prices maximize immediate revenue and then lead to higher reward in evaluation. However, the policy to maximize immediate revenue may not be the optimal policy for E-commerce platform, which will be proved in our online experiments latter.

Then we did some experiments for testing the accuracy of three different models during the pre-training. The results of these experiments about error rate, rate of right estimation discussed in subsection 3.5 are shown in Figure 4 (in appendix). It shows that, DRL methods have higher accuracy and lower error rate than MAB during pre-training.

4.2 ONLINE EXPERIMENTS

After pre-trained with demonstrations, we put the agent online to interact with the environment and to price the products. We first applied DQN algorithm for pricing the markdown products with the reward function defined with the revenue conversion rate we discussed above and compared the revenue as well as the profit with products also joined the markdown season but pricing manually. The result of the experiment is shown in Figure 1.



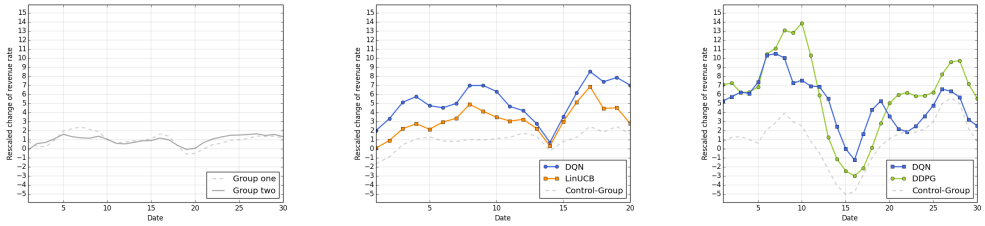
(a) (b)

Figure 1: Sub-figure (a) and (b) are the rescaled revenue conversion rate and rescaled profit conversion rate plots comparing products priced by DQN and products priced manually. From Day 1 to Day 15, products were in their normal sales prices. DQN group and manual group both had a revenue conversion rate of 0.04, and a profit conversion rate of 0.06. Markdown pricing started from Day 15 and ended at Day 30. DQN group achieved a revenue conversion rate of 0.22 on average and manual group's conversion rate is 0.16 for this period (with Day 16 manual group's revenue conversion rate rescaled to 1). In (b), during the markdown season, DQN group obtained a profit conversion rate of 0.16, while manual group's profit conversion rate dropped to -0.04 on average (with Day 16 manual group's profit rate rescaled to -1).

In the first 15 days of this online experiment, these products were in their daily prices. Then the markdown season started at the 16th day and lasted for 15 days. During this online experiment, manual pricing method pulled up the revenue by decreasing the prices lower than the cost, causing negative profit during the markdown season while DQN pricing policy successfully boosted the revenue while keep positive profit.

To investigate the performance of reward function in Eq.(1) for FMCG daily pricing, especially under the instability of the E-commerce environment, we defined some simi-products, the products with the same brand, same category and similar selling behaviors. We have found that, with the same pricing policy, even if two groups of simi-products have different total revenue, the year-on-year change of their revenue could be very close (Figure 2(a)). Therefore, we could evaluate pricing policies by investigating the change of their revenues year on year of the simi-products.

We first divided one of the simi-product group (group two in Figure 2(a)) into two groups, for using LinUCB and DQN algorithm while kept the group one as the control group. The change of the revenue for the products within 20 days in these three groups are shown in Figure 2(b). Then we again divided the group two into two groups for testing DQN and DDPG and the result is shown in Figure 2(c).



(a) (b) (c)

Figure 2: Comparing the change of revenue conversion rate for different groups of similar products. (a) shows two groups of similar products which have similar change of revenue conversion rates within 30 days. The average for group one is rescaled to 1.00 and the average for group two becomes 0.99 after rescaling. (b) shows the changes of the revenue conversion rates for similar product groups priced by DQN, LinUCB and control group respectively for 20 days. The averages of DQN and LinUCB pricing groups are 5.24 and 3.09, with the average of the control group rescaled to 1.00. (c) shows changes of the revenue conversion rates for similar product groups priced by DDPG, DQN, the control group and control group respectively for 30 days. The averages of DDPG and DQN pricing groups are 6.07 and 5.03 respectively with the average of the control group rescaled to 1.00.

5 Conclusions and Discussion

In this work, we proposed a deep reinforcement learning framework for dynamic pricing on E-commerce platform. We defined the pricing process as a Markov Decision Process and then defined the state space, discrete and continuous action space, and different reward function for different pricing applications. We applied our methods to train for pricing policies and applied these policies to online pricing in real time. To address the legal issue of A/B testing for different pricing strategies, we designed a systematic mechanism for online pricing policy evaluation. We showed that pricing policies from DDPG and DQN outperformed other pricing policies significantly. We also tried deep reinforcement learning methods for pricing in different situations, e.g., markdown pricing. The field experiment showed that it outperformed the manual markdown pricing strategy.

This work is the first to use deep reinforcement learning for dynamic pricing problem. In this work, there are a few limiting assumptions which could be removed. First, our pricing framework trains each product separately. As a result, the low-sales-volume products may not have sufficient training data. This could be solved by clustering similar products and using transfer learning to price the products in the same cluster. Meta-learning may also help for this problem. Second, our framework outputs pricing policy for each product separately, however, sometimes we hope to price different products together to form certain marketing strategies. This may be solved by combinatorial action space. Third, in our pricing framework, we take only the features related to the products to describe the environment state. In the future, we would try to take more kinds of features into consideration for pricing under more specific scenarios, e.g., promotion pricing or membership pricing.

REFERENCES

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

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

Omar Besbes and Assaf Zeevi. On the (surprising) sufficiency of linear models for dynamic pricing with demand learning. *Management Science*, 61(4):723–739, 2015.

Felipe Caro and Jérémie Gallien. Clearance pricing optimization for a fast-fashion retailer. *Operations Research*, 60(6):1404–1422, 2012.

Le Chen, Alan Mislove, and Christo Wilson. An empirical analysis of algorithmic pricing on amazon marketplace. In *Proceedings of the 25th International Conference on World Wide Web*, pp. 1339–1349. International World Wide Web Conferences Steering Committee, 2016.

M Keith Chen and Michael Sheldon. Dynamic pricing in a labor market: Surge pricing and flexible work on the uber platform. In *EC*, pp. 455, 2016.

Antoine Augustin Cournot. *Researches into the Mathematical Principles of the Theory of Wealth*. Macmillan, 1897.

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.

G. C. Evans. The dynamics of monopoly. *The American Mathematical Monthly*, 31(2):77–83, 1924. ISSN 00029890, 19300972. URL `http://www.jstor.org/stable/2300113`.

Vivek F Farias and Benjamin Van Roy. Dynamic pricing with a prior on market response. *Operations Research*, 58(1):16–29, 2010.

Guillermo Gallego and Garrett Van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management science*, 40(8):999–1020, 1994.

J Michael Harrison, N Bora Keskin, and Assaf Zeevi. Bayesian dynamic pricing policies: Learning and earning under a binary prior distribution. *Management Science*, 58(3):570–586, 2012.

Bardia Kamrad, Shreevardhan S Lele, Akhtar Siddique, and Robert J Thomas. Innovation diffusion uncertainty, advertising and pricing policies. *European Journal of Operational Research*, 164(3): 829–850, 2005.

Byung-Gook Kim, Yu Zhang, Mihaela Van Der Schaar, and Jang-Won Lee. Dynamic pricing and energy consumption scheduling with reinforcement learning. *IEEE Transactions on Smart Grid*, 7(5):2187–2198, 2016.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670. ACM, 2010.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Kanishka Misra, Eric M Schwartz, and Jacob Abernethy. Dynamic online pricing with incomplete information using multi-armed bandit experiments. 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

JACK Nicas. Now prices can change from minute to minute. *Wall Street Journal*, 2015.

CVL Raju, Y Narahari, and K Ravikumar. Reinforcement learning applications in dynamic pricing of retail markets. In *E-Commerce, 2003. CEC 2003. IEEE International Conference on*, pp. 339–346. IEEE, 2003.

Herbert Robbins. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pp. 169–177. Springer, 1985.

Andrew Sendonaris and COM Gabriel Dulac-Arnold. Learning from demonstrations for real world reinforcement learning. *arXiv preprint arXiv:1704.03732*, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Kalyan T Talluri and Garrett J Van Ryzin. *The theory and practice of revenue management*, volume 68. Springer Science & Business Media, 2006.

Matej Vecerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin A Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *CoRR, abs/1707.08817*, 2017.

David Vengerov. A gradient-based reinforcement learning approach to dynamic pricing in partially-observable environments. 2007.

Zizhuo Wang, Shiming Deng, and Yinyu Ye. Close the gaps: A learning-while-doing algorithm for single-product revenue management problems. *Operations Research*, 62(2):318–331, 2014.

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

Lawrence R Weatherford and Samuel E Bodily. A taxonomy and research overview of perishable-asset revenue management: Yield management, overbooking, and pricing. *Operations research*, 40(5):831–844, 1992.

Ian H Witten. An adaptive optimal controller for discrete-time markov environments. *Information and control*, 34(4):286–295, 1977.

A Additional Details on Algorithms

After forming the demonstration in tuples $< s_t, a_t, r_t, s_{t+1} >$, LinUCB model could then be also pre-trained, which is introduced in Algorithm 1.

To evaluate the policy, we referred the evaluation method introduced by Li et al. (2010). As for our continuous action space method, DDPG, we may not have the exact same actions from both the recorded data and model. Thus we release the condition to $a - \epsilon < \pi(s) < a + \epsilon$. And the full algorithm is introduced in Algorithm 2.

Algorithm 1 LinUCB from Demonstrations.

Require: D^{demo}:demonstration data set; T: the length of demonstration data set; $\alpha \in \mathbb{R}_+$ the rate of exploration;
Ensure: $p_{t,a}$: the probability for choosing arm a at time step t;

for all $a \in A$ **do**
  $\mathbf{A}_a \leftarrow \mathbf{I}_d$ (d-dimensional identity matrix)
  $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ (d-dimensional zero vector)
end for
for step $t \in \{1, 2, ..., T\}$ **do**
  Read demonstration data D^{demo} : $\mathbf{s}_t, a_t, r_t$
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{s}_t\mathbf{s}_t^\mathsf{T}$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{s}_t$
end for
for step $t \in \{1, 2, ...\}$ **do**
  Observe state $\mathbf{s}_t \in \mathbb{R}^d$
  $\boldsymbol{\theta}_a \leftarrow \mathbf{A}_a^{-1} b_a$
  $p_{t,a} \leftarrow \boldsymbol{\theta}_a^{-1}\mathbf{s}_t + \alpha\sqrt{\mathbf{s}_t^{-1}\mathbf{A}_a^{-1}\mathbf{s}_t}$
  Choose arm $a_t = \arg\max_a p_{t,a}$ and receive the reward r_t
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{s}_t\mathbf{s}_t^\mathsf{T}$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{s}_t$
end for

Algorithm 2 Policy Evaluation with Demonstration Tuples.

Require: $T > 0$: number of demonstration tuples for evaluation; π: the policy to evaluate;
Ensure: R_π/T: average reward form policy π;

$R_\pi = 0$
for step $t \in \{1, 2, ..., T\}$ **do**
  repeat
    Get next tuple $< s, a, r, s' >$
  until $a - \epsilon < \pi(s) < a + \epsilon$
  $R_\pi \leftarrow R_\pi + r$
end for

B Additional Experiment Results

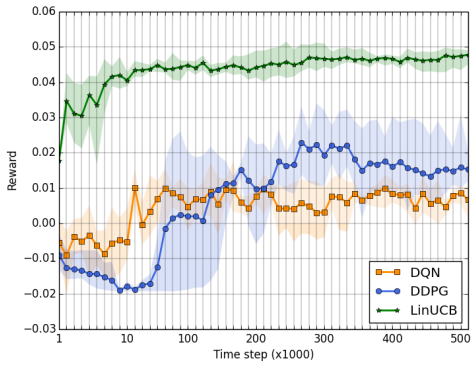


Figure 3: Policy evaluation for DQN, DDPG and LinUCB with Tmall Supermarket offline sales data. Reward here has been rescaled due to data security reason.

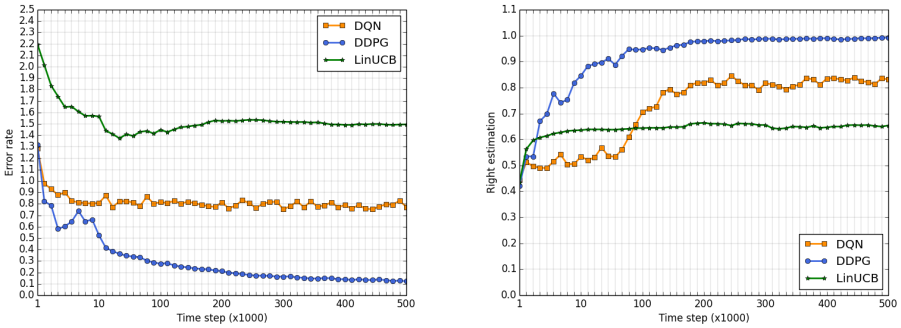


Figure 4: The error rate and the rate of right estimation for the change of the revenue between prices for three different methods during pre-training with Tmall Supermarket offline data.