Batch Reinforcement Learning for Dynamic Pricing in E-Commerce

Andrii Holovko and Taras Firman

Ukrainian Catholic University, Lviv, Ukraine
{holovko, firman}@ucu.edu.ua

Abstract. There are existing reinforcement learning approaches for dynamic pricing that rely on off-policy algorithms, pre-trained on a replay buffer. However, the approximation error problem subjects off-policy algorithms, trained without interaction with the environment, to fail in real-world environments due to overestimated value estimates. In dynamic pricing, this leads to sub-optimal pricing decisions, long-term loss of customers, and revenue drop. This work is dedicated to building and testing a reliable dynamic pricing engine for an E-commerce platform based on an offline reinforcement learning algorithm trained on a fixed batch of data.

Keywords: Reinforcement learning · Dynamic pricing · Price optimization.

1 Introduction

The task of the right price selection for the product is required, but complex though. It directly influences the customers' loyalty and income of the business. The creation of a proper pricing strategy requires a lot of expertise in the domain, while a great number of factors must be taken into account in order to make a proper pricing decision.

In terms of large retail, E-commerce and other online businesses, the problem becomes yet more complex as the number of products and services can be huge. Each product requires a proper pricing policy to manage the companies revenue in an optimal fashion.

Thus, machine learning comes in hand within the dynamic pricing domain. Particularly, deep reinforcement learning (DRL), which is aimed exactly to solve the problem of reward maximization. The latest strides in DRL show that algorithms can outperform human performance in certain fields. Recent works on DRL for dynamic pricing [18,24,26] show the evidence that off-policy DRL agents are able to outperform human performance for this particular field. At the same, the progress in improving DRL algorithms is still on. And the aim of the proposed research is to examine whether the latest advances in DRL allow outperforming older DRL approaches with proven performance in such real-world application as dynamic pricing.

2 Background

In reinforcement learning we assume that our environment describes by a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces respectively, p denotes the transition dynamics matrix, r is a reward function and $\gamma \in [0, 1)$ is the discount factor. At each discrete time step t a reinforcement learning agent takes an action $a \in \mathcal{A}$ in the state $s \in \mathcal{S}$ and, accordingly to the transition dynamics $p(s', r|s, a)$, receives the reward $r(s, a, s')$ and a new state $s' \in \mathcal{S}$. Projecting these onto the dynamic pricing problem, we usually consider s to be the current state of the given product item and a to be the price selected by our agent for this product. The continuous time can be discretized with the discretization period d in order to fit such a model of environment. The agent's goal is the maximization of the sum of discounted rewords denoted as return $R_t = \sum_{i=t+1}^{N(\infty)} \gamma^i r(s_i, a_i, s_{i+1})$. Agent makes its decisions using policy $\pi : \mathcal{S} \to \mathcal{A}$. $\pi(\cdot)$ maps a given state s to a distribution over actions. Value function Q_π of an action a at the state s for a given policy π is an expected return of an agent following policy π: $Q_\pi(s, a) = \mathbb{E}_\pi(R_t|s, a)$. Using a greedy optimization we can find a new policy π' having equal or better performance in terms of value function: $\pi' = \arg\max_a Q_\pi(s, a)$. Greedy selection over the optimal value function $Q^*(s, a) = \max_\pi Q_\pi(s, a)$ gives the optimal policy $\pi^* = \arg\max_a Q^*(s, a)$. Q_π and Q^* are the unique fixed points of the Bellman operator $\mathcal{T}_\pi$ [2] and optimality operator $\mathcal{T}^*$ [3] respectively:

$$\mathcal{T}_\pi(s, a) = \mathbb{E}_{s', r, a' \sim \pi}[r + \gamma Q(s', a')] \tag{1}$$

$$\mathcal{T}^*(s, a) = \mathbb{E}_{s', r}[r + \gamma \max_{a'} Q(s', a')]. \tag{2}$$

In deep reinforcement learning, a neural network Q_θ parameterized with θ is used to approximate the value function. Deep Q-Network algorithm (DQN) [28] updates θ by approximating the optimality operator using Q-learning [34]:

$$\mathcal{L}(\theta) = l_\kappa\big(r + \gamma \max_{a'} Q_{\theta'}(s', a') - Q_\theta(s, a)\big), \tag{3}$$

where l_κ is the Huber loss [14]. A target network $Q_{\theta'}$ with frozen parameters is used to maintain a fixed target over multiple updates, where θ' is updated to θ after a set number of learning steps. The loss (equation 3) is minimized over mini-batches of tuples (s, a, r, s'), which are usually samples from a replay buffer $\mathcal{B}$. For off-policy algorithms $\mathcal{B}$ can be generally obtained by any policy, not only by the current agent's one.

In order to take into account the specific of the dynamic pricing problem and to clarify the search field within the reinforcement learning domain at this point, we should define some properties of the desired algorithm so it can be applied for the given dynamic pricing problem, some of them are contradicting:

- be *off-model* as the model of pricing environment is unknown and must be learned by the agent from the data;

- be *offline* as we can not train and explore using the environment as it requires applying the price, chosen by for sure not yet well-trained agent which can lead to highly sub-optimal pricing decisions;
- be *sample efficient* as we are limited in the data we have for agent's learning as opposed to classical reinforcement learning, where the environment is completely available for exploration and thus the size of the training dataset is essentially unlimited;
- support *continuous action space* as the price is a continuous value.

3 Related Work

3.1 Deep Reinforcement Learning for Dynamic Pricing

A lot of researches address the dynamic pricing with unknown demand function. Some works utilize parametric methods for the problem. [4] considers the usage of parametric families of functions to learn demand functions over time. [8] uses information about historical purchases to learn. [13] propose Bayesian methods to cope with the uncertainties of the demand function. [5,6,33] focus on non-parametric approaches to eliminate problem of sub-optimal result due to too narrow demand function family. They use an assumption that revenue is a strictly concave and differentiable function of price. [24] shows this can not be held in a real E-commerce environment.

The advances in computation and reinforcement learning allowed yet another approach to be used for dynamic problems. In [17], a price bot is developed using Q-learning in order to properly adjust the price in response to changes in the market state. [30] address the dynamic pricing problem by Temporal Difference method. [29] looks at the dynamic pricing problem with a single seller and two sellers setup and solves it with the help of reinforcement learning. [21] utilizes several asynchronous multi-agent RL algorithms for making correct pricing decisions. [18,32] consider the usage of RL for the problem of price optimization for the energy market.

[26] attacks the problem of unfair pricing for different customers group in the case of using DRL agent for dynamic pricing. Authors use Deep Q-Network (DQN [34]) with some adjustments and a specific reward function. However, those works use simplified market setting, using deep neural networks for the approximation of a discrete price space.

[24] is one of the most prominent of the up to date works on dynamic pricing with RL. Authors compare DQN with discrete price space setup and Deep Deterministic Policy Gradients (DDPG [23]) with continuous price space setup. While previous works considered the direct optimization of revenue as a reward function, in this work, another reward function was chosen, which is the difference of the revenue conversion rates between current and previous time steps. Firstly, both DQN and DDPG agents are pre-trained on fixed data, which was obtained under specialists' pricing decisions or some pricing rules. The dataset is also used for offline model evaluation by taking into account only the rewards for state-action pairs, which are close to the available data samples. Then agents are

evaluated using an experiment in the real market environment. Results showed that DDPG outperformed DQN. Also, they both outperformed other pricing policies significantly.

3.2 Offline Reinforcement Learning

Off-Policy Deep Reinforcement Learning without Exploration [11] shows that due to errors introduced by extrapolation, standard off-policy deep RL algorithms, such as DQN and DDPG, are incapable of learning without data correlated to the distribution under the current policy, making them ineffective for the fixed batch of data. This phenomenon is denoted as *extrapolation error*, which is an error in off-policy value learning, introduced by the distribution mismatch between the dataset and state-action pairs, visited under the learned policy. The value estimate $Q(s, a)$ is affected by an extrapolation error during value update where the target policy selects an unfamiliar action a' at the next state s' for value estimate, such that (s', a') is unlikely to be in the dataset. Authors experiment with the DDPG algorithm, demonstrating that the off-policy agent (which uses the trained policy) is hugely outperformed by the behavioural agent with the gaussian noise added to the actions for exploration.

In order to avoid the extrapolation error, the authors propose an idea to have a policy that induces a state-action visitation similar to the dataset. Such a policy is denoted as *batch-constrained* and trained with respect to three objectives:

(1) minimize the distance of selected actions to the data in the batch;
(2) lead to states where familiar data can be observed;
(3) maximize the value function.

The work presents the Batch-Constrained deep Q-learning (BCQ) algorithm, which claimed to be the first continuous control DRL algorithm that can learn effectively from an arbitrary fixed batch data. By this, batch-constrained reinforcement learning was introduced as a novel class of off-policy algorithms. This algorithm can be treated as DDPG [23] with several modifications to maintain the notion of *batch-constrained*. The conditioned by state generative model G_ω is used to sample actions, similar to the distribution of a dataset. A Conditional Variational Auto-encoder (CVAE [15,19]) is used for a generative model. The sampled action a is then added to the output of perturbation model ξ_ϕ to adjust a within the range $[-\Phi, \Phi]$, enabling access to actions in a constrained region. ξ_ϕ is trained to maximize value function through the Deterministic Policy Gradient algorithm [31]. All in all, this lead to the following agent's policy:

$$\pi(s) = \underset{a_i + \xi_\phi(s, a_i, \Phi)}{\arg\max} Q_\theta(s, a_i + \xi_\phi(s, a_i, \Phi)), \{a_i \sim G_\omega(s')\}_{i=1}^n. \tag{4}$$

The work provides the comparison of BCQ against DDPG [23] and DQN [28] with a discretized action space. The results show that BCQ is the only algorithm that succeeds at all tasks, outperforming all other agents in each of several experiments.

Stabilizing Off-Policy Q-Learning via Bootstrapping [20] Another batch reinforcement learning algorithm, Bootstrapping Error Accumulation Reduction Q-Learning (BEAR-QL [20]), is based on the same idea as BCQ. It also uses a generative model of the data distribution in batch, but uses a trained actor π_ϕ instead of perturbation model to sample actions. It also uses an ensemble of K Q-networks for value estimation. The actor π_ϕ is trained with deterministic policy gradient algorithm, minimizing loss through dual gradient descent:

$$\mathcal{L}(\phi) = -\big(\frac{1}{K}\sum_k Q^k_\theta(s,\hat{a}) - \tau \mathrm{var}_k Q^k_\theta(s,\hat{a})\big) \quad \text{s.t. } \mathrm{MMD}(G_\omega(s), \pi_\phi(s)) \leq \varepsilon, \quad (5)$$

where MMD is the maximum mean discrepancy [12].

The policy π_ϕ is used to sample actions during training, while the policy for the inference is denoted as follows:

$$\pi(s) = \underset{\hat{a}\sim\pi_\phi(s)}{\arg\max} \frac{1}{K}\sum_k Q^k_\theta(s,\hat{a}). \quad (6)$$

Authors compare the performance of BEAR-QL, Twin Delayed Deep Deterministic Policy Gradient (TD3 [10]) and BCQ on different datasets obtained with random, mediocre and optimal policies. BEAR-QL outperformed BCQ on random data and hugely outperformed TD3 on optimal data.

Other works on offline reinforcement learning [27] presents a framework that enables offline learning from a large set of diverging and sub-optimal demonstrations by selectively imitating local sequences from the dataset, called Implicit Reinforcement without Interaction at Scale (IRIS). It was designed to match the purpose of offline learning of agent to perform robotic manipulation tasks. The decision-making process is split into a high-level mechanism that sets goal states for a low-level controller to try and reach. At a state s_t, this mechanism selects a goal state s_g that is held for the next T timesteps. CVAE is used to sample a set of proposals similar to the data distribution, and a value function $V(s)$ is used to select the most promising goal proposal. The low-level controller is conditioned on s_g and is given T timesteps to try and reach that state in a closed-loop fashion. The low-level controller is a goal conditioned Recurrent Neural Network (RNN [25]) $\pi_\theta(s|s_g)$ that outputs an action a_t at each timestep, given a current observation s_t and goal s_g. The RNN is trained using Behavioral Cloning loss $\mathcal{L}_\theta(a_{t:t+T}, s_{t:t+T}) = \sum_{k=t}^{t+T-1}(a_k - \pi_\theta(s_k|s_g))^2$.

[9] provides a nice overview of recent at the moment off-policy and batch reinforcement learning algorithms and benchmarks the performance of these algorithms under unified setting on the Atari domain, continuing the work [1] on examining widely-used off-policy algorithms on Atari environments. More specifically, authors compare Quantile Regression DQN (QR-DQN [7]), Random Ensemble Mixture (REM [1]), Batch-Constrained deep Q-learning (BCQ [11]), Bootstrapping Error Accumulation Reduction Q-Learning (BEAR-QL [20]), KL-Control [16] and Safe Policy Improvement with Baseline Bootstrapping DQN

(SPIBB-DQN [22]). Two of these algorithms, BCQ and BEAR-QL, are suited for continuous action-spaces, thus being of special interest to the dynamic pricing problem. Another contribution of the authors in this work is a discrete and simplified variant of BCQ. Experiments on comparison of the mentioned algorithms (except BEAR-QL and SPIBB-DQN) show that a discrete version of BCQ outperforms the competitors, while the competitors are even underperforming the off-policy DQN baseline under the given settings. However, all the tests were made on discrete action space environments.

4 Problem Setting

The main objective of our work is to measure and compare the performance of the novel batch deep reinforcement algorithms, such as DQN and BEAR-QL, in the dynamic pricing domain with the generic off-policy DDPG algorithm, which has already shown good results in this domain, and TD3 algorithm, which seems to be the intermediate algorithm between the DDPG and the batch DRL algorithms. Besides that, the following goals are aimed:

- evaluate the performance in both offline and online (with the simulated environment) setups;
- compare the performance for the different target metrics (e.g., sales quantity, revenue, profit, margin) also used as different options reward functions;
- compare the performance on the product sets with different characteristics like seasonality, offer uniqueness and category;
- find out the best hyperparameters set for particular algorithms as well as the most performant value of the time discretization period d.

5 Approach

5.1 Dataset and Environment

To train the pricing engine based on the reinforcement learning agent, we need a large dataset having a sufficient number of records for each particular product. It should contain information about the price and sales quantity of the products as well as product features, which can be used by the agent to catch the similarities between different products. For that purpose, we plan to use the data provided by the Helium 10[1] service. It provides access to the database, containing 450 million products from Amazon with sales and pricing history, other information about products, and data about competitors.

In order to be able to test agents within the online setup, we also need an environment. For this purpose, a simulated environment must be built. The environment must return a new state s' and reward r defined by the target function given the current state s and price action a taking into account the defined model of dependencies. We state that the environment must implement the following properties of the demand:

[1] https://www.helium10.com/

- seasonality;
- own elasticity;
- cross-elasticity between different products;
- cross-elasticity relative to competitor's price.

5.2 Training

Here we want to train the four previously outlined agents: DDPG, TD3, BCQ and BEAR-QL, using the implementation of the authors of the works [11] [2] for DDPG and BCQ and [20] [3] for TD3 and BEAR-QL. For both the online and offline setup, we train the agents by taking the batch $\mathcal{B}$ of tuples (s, a, r, s') from the environment or from the dataset correspondingly. As long as the learned policy may be completely different in terms of evaluation performance for the batches collected under different policies, we eager to run training on the data from the environment in both online and offline setups. For the online setup, we assume that the data from the environment is collected by the policy of the current agent, while for the offline setup, the data is collected before training under the different policy (e.g., with the help of DDPG agent).

5.3 Evaluation

We need to define performance metrics to be used in order to show the suitability of trained agents for our problem. The chosen main metrics are the following:

(1) mean total reward over all time steps,
(2) difference between value-function estimation and ground truth values.

(1) is considered the most representative as it shows the value of a real target metric we want to optimize with agent's pricing decisions. Obviously, we can not calculate (1) directly for the setup with the batch training setup. In this case, we will use the approach proposed in [24]. We take into account the reward r from an arbitrary tuple (s, a, r, s') from the evaluation batch $\mathcal{B}$ if and only if agent's action $\pi(s)$ satisfies $a - \epsilon < \pi(s) < a + \epsilon$. Thus, tracking the (3) fraction of all rewards accounted for total reward calculation is also desirable.

(2) is important because it indicates the magnitude of value overestimation, which is a root of mismatch between optimal actions and the actions taken by an agent. For the batch training setup, it can be approximated in the same way as (1).

6 Research Plan

✓ Choose a particular topic
✓ Perform literature search and review

[2] https://github.com/sfujim/BCQ
[3] https://github.com/aviralkumar2907/BEAR

- ✓ Write an abstract
- ✓ Formulate the refined research objectives
- ✓ Look for marketplace sales data sources and algorithms implementations
- ✓ Write and submit a paper for the MS-AMLV-2021
- Extract dataset from the data sources
- Implement an artificial environment with a given market demand function
- Create an environment-like interface for batch-constrained model training
- Run the iterative process of agent's training, hyperparameters selection, evaluation and environment settings adjustment
- Perform the final analysis of the research outcomes
- Write the final version of the Master thesis work

7 Conclusion

The deep reinforcement learning domain of machine learning is developing rapidly nowadays. While the one researchers are working on implementing the latest advances for solving particular problems, the others are creating novel theories and yet more promising algorithms. In this work, we described our motivation do to the research on deep reinforcement learning for the dynamic pricing domain and outlined the properties of an algorithm, which are needed for the domain. After literature review, we identified the potential problems of existing methods and assumed that BCQ and BEAR-QL batch reinforcement learning algorithms could be more suitable for dynamic pricing than the previously used approaches and have great potential to outperform those approaches. Finally, we stated our research goals, presented the approach to achieve those goals and described the research plan.

References

1. Agarwal, R., Schuurmans, D., Norouzi, M.: An optimistic perspective on offline reinforcement learning (2020)
2. Bellman, R.E.: Dynamic Programming (1957)
3. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-dynamic programming (1996)
4. Bertsimas, D.J., Perakis, G.: Dynamic pricing: A learning approach pp. 45–79 (2006)
5. Besbes, O., Zeevi, A.: Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. Operations Research **57**(6), 1407–1420 (2009)
6. Besbes, O., Zeevi, A.: On the surprising sufficiency of linear models for dynamic pricing with demand learning. Management Science **61**(4), 723–739 (2015)
7. Dabney, W., Rowland, M., Bellemare, M.G., Munos, R.: Distributional reinforcement learning with quantile regression (2017)
8. Farias, V.F., Roy, B.V.: Dynamic pricing with a prior on market response. Operations Research **58**(1), 16–29 (2010)
9. Fujimoto, S., Conti, E., Ghavamzadeh, M., Pineau, J.: Benchmarking batch deep reinforcement learning algorithms. arXiv preprint arXiv:1910.01708 (2019)

10. Fujimoto, S., van Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods (2018)
11. Fujimoto, S., Meger, D., Precup, D.: Off-policy deep reinforcement learning without exploration (2019)
12. Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.: A kernel two-sample test. J. Mach. Learn. Res. **13**(null), 723–773 (Mar 2012)
13. Harrison, J.M., Keskin, N.B., Zeevi, A.: Bayesian dynamic pricing policies: Learning and earning under a binary prior distribution. Management Science **58**(3), 570–586 (2012)
14. Huber, P.J.: Robust estimation of a location parameter. Annals of Mathematical Statistics **35**(1), 492–518 (1964)
15. Im, D.J., Ahn, S., Memisevic, R., Bengio, Y.: Denoising criterion for variational auto-encoding framework (2016)
16. Jaques, N., Ghandeharioun, A., Shen, J.H., Ferguson, C., Lapedriza, A., Jones, N., Gu, S., Picard, R.: Way off-policy batch deep reinforcement learning of implicit human preferences in dialog (2019)
17. Kephart, J.O., Hanson, J.E., Greenwald, A.R.: Dynamic pricing by software agents. Computer Networks **32**(6), 731–752 (2000)
18. Kim, B.G., Zhang, Y., van der Schaar, M., Lee, J.W.: Dynamic pricing and energy consumption scheduling with reinforcement learning. IEEE Transactions on Smart Grid **7**(5), 2187–2198 (2016)
19. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2014)
20. Kumar, A., Fu, J., Soh, M., Tucker, G., Levine, S.: Stabilizing off-policy q-learning via bootstrapping error reduction. In: Advances in Neural Information Processing Systems. vol. 32, pp. 11784–11794 (2019)
21. Kutschinski, E., Uthmann, T., Polani, D.: Learning competitive pricing strategies by multi-agent reinforcement learning. Journal of Economic Dynamics and Control **27**(11), 2207–2218 (2003)
22. Laroche, R., Trichelair, P., des Combes, R.T.: Safe policy improvement with baseline bootstrapping (2019)
23. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning (2019)
24. Liu, J., Zhang, Y., Wang, X., Deng, Y., Wu, X.: Dynamic pricing on e-commerce platform with deep reinforcement learning (2019)
25. Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., Sermanet, P.: Learning latent plans from play. arXiv: Robotics (2019)
26. Maestre, R., Duque, J.R., Rubio, A., Arévalo, J.: Reinforcement learning for fair dynamic pricing. In: Proceedings of SAI Intelligent Systems Conference. pp. 120–135 (2018)
27. Mandlekar, A., Ramos, F., Boots, B., Savarese, S., Fei-Fei, L., Garg, A., Fox, D.: Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 4414–4420 (2020)
28. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**, 529–33 (02 2015). https://doi.org/10.1038/nature14236
29. Raju, C., Narahari, Y., Ravikumar, K.: Reinforcement learning applications in dynamic pricing of retail markets. In: EEE International Conference on E-Commerce, 2003. CEC 2003. pp. 339–346 (2003)

30. Schwind, M., Wendt, O.: Dynamic pricing of information products based on reinforcement learning: A yield-management approach. In: KI '02 Proceedings of the 25th Annual German Conference on AI: Advances in Artificial Intelligence. pp. 51–66 (2002)
31. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. 31st International Conference on Machine Learning, ICML 2014 **1** (06 2014)
32. Vengerov, D.: A gradient-based reinforcement learning approach to dynamic pricing in partially-observable environments. Future Generation Computer Systems **24**(7), 687–693 (2008)
33. Wang, Z., Deng, S., Ye, Y.: Close the gaps: A learning-while-doing algorithm for single-product revenue management problems. Operations Research **62**(2), 318–331 (2014)
34. Watkins, C.J.C.H.: Learning from delayed rewards. PhD thesis, King's College, University of Cambridge (1989)