Value Penalized Q-Learning for Recommender Systems

Chengqian Gao†
Tsinghua University
Shenzhen, China
gcq19@mails.tsinghua.edu.cn

Ke Xu
Tencent AI Lab
Shenzhen, China
kaylakxu@tencent.com

Kuangqi Zhou
National University of Singapore
Singapore
kzhou@u.nus.edu

Lanqing Li
Tencent AI Lab
Shenzhen, China
lanqingli1993@gmail.com

Xueqian Wang
Tsinghua University
Shenzhen, China
wang.xq@sz.tsinghua.edu.cn

Bo Yuan
Tencent AI Lab
Shenzhen, China
boyuan@ieee.org

Peilin Zhao†
Tencent AI Lab
Shenzhen, China
peilinzhao@hotmail.com

ABSTRACT
Scaling reinforcement learning (RL) to recommender systems (RS) is promising since maximizing the expected cumulative rewards for RL agents meets the objective of RS, i.e., improving customers’ long-term satisfaction. A key approach to this goal is offline RL, which aims to learn policies from logged data rather than expensive online interactions. In this paper, we propose Value Penalized Q-learning (VPQ), a novel uncertainty-based offline RL algorithm that penalizes the unstable Q-values in the regression target using uncertainty-aware weights, achieving the conservative Q-function without the need of estimating the behavior policy, suitable for RS with a large number of items. Experiments on two real-world datasets show the proposed method serves as a gain plug-in for existing RS models.

CCS CONCEPTS
• Theory of computation → Reinforcement learning; • Information systems → Learning to rank; Recommender systems.

KEYWORDS
offline reinforcement learning, sequential recommender systems, long-term satisfaction

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGIR ‘22, July 11–15, 2022, Madrid, Spain
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8732-3/22/07...$15.00
https://doi.org/10.1145/3477495.3531796

1 INTRODUCTION

Practical recommender systems (RS) are usually trained to generate relevant items for users, considering less on their long-term utilities. Reinforcement learning (RL) methods maximize the discounted cumulative rewards over time, meeting the original needs of RS. Driven by this, there has been tremendous progress in developing reinforcement learning-based recommender systems (RLRS) [1].

Directly utilizing RL algorithms to train agents from the offline data often results in poor performance [16, 25, 27], even for off-policy methods, which can leverage data collected by other policies in principle [7, 14]. The main reason for such failures is the overestimation of out-of-distribution (OOD) queries [17], i.e., the learned value function is trained on a small set of action space while is evaluated on all valid actions. Offline RL [7, 17] focuses on the problem of training agents from static datasets, addressing the OOD overestimation problem without interacting with the environment, making it achievable to build RLRS agents from offline datasets.

There are still challenges for offline RL-based RS models due to the different settings between RL and RS tasks. The first is the large action space (the number of candidate items for RS agents is usually above 10k), which exacerbates the OOD overestimation problem and also makes it difficult to estimate the behavior policy that generated the datasets [6, 7, 9, 13, 14, 21, 22]. Another problem is the mismatch between relevant and valuable recommendations. RL approaches produce recommendations with the highest long-term utility while ignoring their relevance to users. Finally, the non-stationary dynamic of RS also differs from RL environments. In general offline RL problems, e.g., D4RL [5], trajectories are collected by policies with an environment whose dynamics are exactly as same as the test environment. While in RS scenarios, the dynamics constantly change as the user preferences change over time.

In this work, we propose Value Penalized Q-learning (VPQ), an uncertainty-based offline RL method, to alleviate the OOD overestimation issue. Then we integrate it into classic RS models, enhancing their capacity to generate relevant and valuable recommendations.
Specifically, we use a sequential RS model to represent the sequence of user-item interactions (mapping the last 10 items into hidden state $s_t$). The value function regresses on the discounted cumulative reward $Q(s_t, a_t) = \sum_{i=0}^{inf} \gamma^i r(s_{t+i}, a_i)$ for each recommendation item $a_i$ with respect to the state $s_t$, immediate reward $r$ (1.0 for purchases and 0.2 for clicks), and a discount factor $\gamma$.

Algorithmically, we take two techniques to address the OOD overestimation problem. Firstly, we stabilize the Q-predictions using an ensemble of Q-functions with different random learning rates and set the variance of their predictions as the uncertainty quantifier. Secondly, we penalize the unstable Q-values in regression target with uncertainty-aware weights, reducing the overestimation for OOD actions. The key component is to reduce the unstable predictions with the proposed $p\text{-mul}$ form penalty. We will detail it in the method section. In order to exploit the learned value function for the non-stationary environment, we employ the critic and long-term satisfaction. To summarize:

1. We propose an uncertainty-based offline RL method, VPQ, to alleviate the OOD overestimation problem during training by penalizing the unstable predictions with uncertainty quantification, without estimating the behavior policy.
2. We empirically show that it is more effective to attain the conservative value function by the proposed $p\text{-mul}$ form, i.e., multiplying the unstable predictions with uncertainty-aware weights, in the recommendation scenarios.
3. Extensive experiments show that the benefit of integrating with VPQ contains better representation and reinforcing actions with long-term rewards.

2 METHODS

We propose Value Penalized Q-learning (VPQ), an uncertainty-based offline RL algorithm, to estimate the long-term rewards for recommendation candidates. We then integrate it into classic RS models to generate relevant and valuable recommendations.

2.1 Value Penalized Q-Learning

2.1.1 Preliminary: Q-Learning Algorithm. The Q-learning framework for recommendation has been adopted in many previous works [24, 28–31]. It usually learns a state-action value function $Q_\theta(s_t, a_t)$ by minimizing $L_{\theta} = \frac{1}{2} \mathbb{E}_{(s_t, a_t, r, s_{t+1}) \sim D} \left[ (y - Q_\theta(s_t, a_t))^2 \right]$ with a dataset $D$, and a target $y = r + \gamma \max_a Q_\theta(s_{t+1}, a)$ with frozen parameters from historical value function $Q_\theta$.

However, when learning from static datasets, the max operator inhibits the performance as it queries all successor Q-values while only a small set of actions has been recorded. Unstable predictions on unseen actions undermine the learned Q-function and thus, in turn, exacerbate instability, resulting in erroneous Q-function when learning from static datasets [7, 14].

2.1.2 Two Different Ways of Using the Uncertainty Metric. An intuitive way to uncertainty-based offline RL [4, 10, 17], denoted as $p\text{-sub}$, is removing the successor uncertainty from each query via:

$$y = r + \gamma \max_a (Q(s_{t+1}, a) - \lambda \text{Unc}(s_{t+1}, a)),$$

where Unc quantifies the amount of uncertainty for each Q query.

Our approach to penalizing the unstable prediction is:

$$y = r + \gamma \max_a \left( Q(s_{t+1}, a) \cdot W(s_{t+1}, a) \right),$$

with an uncertainty-aware weight $W(s_{t+1}, a)$ designed as:

$$W(s_{t+1}, a) = \frac{1}{1 + \lambda \text{Unc}(s_{t+1}, a)},$$

with $\lambda > 0$ controlling the strength of penalty on uncertain queries. Unc$(s_{t+1}, a)$ is defined as the standard deviation across the target Q-function ensemble, i.e., $
\hat{\sigma}(s_{t+1}, a) = \text{SD}(Q_\theta(s_{t+1}, a))_{k=1}^K$. We denote this form as $p\text{-mul}$ for the stable predictions are estimated by multiplying the unstable values by uncertainty-aware weights.

2.1.3 Analysis: Penalizing in a Stable Manner. Assuming that unstable predictions on OOD actions for a given state are i.i.d. random variables follow a normal distribution $x_i = Q(s_t, a_t) \sim N(\mu, \sigma^2)$, then the target value for OOD actions (denoted as $y_{\text{OOD}}$) follows:

$$y_{\text{OOD}} = r + \gamma \max_{1 \leq i \leq N(\mu, \sigma^2)} x_i,$$

where $n$ is close to the number of candidate items in RS, and for brevity, we denote $Q_T$ as the output of the max operation. We can approximate the expectation of $Q_T$ through the form [3, 8, 19]:

$$\mathbb{E} \left[ \max_{1 \leq i \leq N(\mu, \sigma^2)} x_i \right] \approx \mu + \sigma \Phi^{-1} \left( \frac{n - 0.375}{n - 2 \times 0.375 + 1} \right),$$

where $\Phi^{-1}$ is the inverse of the standard normal CDF.

**Algorithm 1** VPQ: Value Penalized Q-Learning

**Input:** $K$ Q-functions with parameters $(\theta_k)_{k=1}^K$, a categorical distribution $P_A$ for Random Ensemble Mixture, a scale factor $\lambda$ for VPQ, and an offline dataset $D$.

**Output:** $(\theta_k)_{k=1}^K$

1. Initialize $(\theta_k)_{k=1}^K$.
2. **while** not done **do**
   3. Sample a mini-batch of transitions $(s_t, a_t, r, s_{t+1})$ randomly from $D$.
   4. Sample mixture weights $(\alpha_k)_{k=1}^K$ from $P_A$.
   5. Compute the sample standard deviation $\hat{\sigma}(s_{t+1}, a)$.
   6. Compute random mixture of target networks:
      $$\hat{\mu}(s_{t+1}, a) = \sum_{k=1}^K \alpha_k Q_{\theta_k}(s_{t+1}, a).$$
   7. Compute the uncertainty-aware weight:
      $$W(s_{t+1}, a) = (1 + \lambda \hat{\sigma}(s_{t+1}, a))^{-1}.$$
   8. Compute the penalized target:
      $$y_t = r + \gamma \max_a \left( \hat{\mu}(s_{t+1}, a) \cdot W(s_{t+1}, a) \right).$$
   9. Update each Q-function with the mixture weight $\alpha_k$:
      $$L_{\theta} = \frac{1}{2} \left[ (y_t - \sum_{k=1}^K \alpha_k \cdot Q_{\theta_k}(s_t, a_t))^2 \right].$$
3. **end while**

By using the properties of the expectation and max operator, we have expectation of penalized $Q_T$ with the $p\text{-sub}$ formulation:

$$\mathbb{E}[Q_T - \lambda \sigma] = \mu + (C_0 - \lambda) \sigma,$$

and the expected value of that with the $p\text{-mul}$ form:

$$\mathbb{E}[Q_T + \lambda \sigma] = \frac{1}{1 + \lambda \sigma} (\mu + \sigma C_0),$$

where $C_0$ is a constant number for a given $n.$
The proposed form of penalization has two advantages. Firstly, the \( p\text{-mul} \) form is more robust. For training RLRS agents, \( C_0 \) increases with the dimension of action space. To reduce unstable predictions, \( p\text{-sub} \) has to increase its \( \lambda \) in Equation (6), increasing the risk of producing negative Q-values, and thus resulting in unstable training. By contrast, the penalty form \( p\text{-mul} \) would always keep the target value greater than zero, even with a large scale factor. Secondly, the proposed \( p\text{-mul} \) can heavily penalize the unstable and large predictions, while the \( p\text{-sub} \) formulation mainly concerns the small and unstable Q-values and thus may fail to obtain a conservative prediction. To illustrate this, we provide a toy experiment in Figure 1.

A keen reader may note that the proposed \( p\text{-mul} \) form fails to penalize Q-values that less than zero. However, we argue that such nulls can be avoided via a linear reward transformation [18].

![Figure 1](https://drive.google.com/file/d/1sEl2TkscyCAXRH28OxNLphE1H9sSm?usp=sharing)

**Figure 1**: Penalize the simulated unstable predictions in different ways. Heavy black lines are dense points sampled from a Gaussian distribution and sorted by their values. Scaling factor \( \lambda \) controls the strength of penalization.

### 2.1.4 Analysis: Uncertainty-Aware Discount Factor

We note the concern that small and unstable Q-values and thus may fail to obtain a conservative prediction. To illustrate this, we provide a toy experiment in Figure 1.

![Figure 1](https://www.kaggle.com/retailrocket/ecommerce-dataset)

![Figure 1](https://recsys.acm.org/recsys15/challenge/)

### 2.1.5 Details about the Proposed Algorithm

Inspired by the powerful penalty form, \( p\text{-mul} \), we developed VPQ. It contains an ensemble of Q-functions with the Random Ensemble Mixture (REM) technique [2] for diversity and assigns the standard deviation of Q-values across the ensemble as an uncertainty indicator.

VPQ eliminates the need to estimate the difficult behavior policy constraint [6, 7, 9, 13, 14, 21, 22]. Besides, as an uncertainty-based method, VPQ enjoys benefits from the progress in uncertainty measurement. Finally, we summarize our method in Algorithm 1.

### 2.2 Relevant vs. Valuable

So far, we have developed VPQ to indicate the long-term rewards for recommendation candidates. However, there exists another concern for RLRS, i.e., how to exploit the learned agent. Generating recommendations via \( a = \arg \max_a Q(s_t, a) \) leads to valuable recommendations without considering their relevance, while classic sequential RS models recall and/or rank relevant items ignoring their long-term utilities. To tackle this, we introduce the critic framework [24], which is inspired by Konda [12].

Specifically, we use the classic sequential RS model to extract hidden states for the input interaction sequences and map them to two types of outputs, Q-values through the Q head and classification logits using a CE head. We optimize the Q head with VPQ algorithm and minimize the reweighted cross-entropy loss for CE head, i.e.,

\[
L_\psi = CE(s_t, a) \cdot \text{no-gradient}(Q(s_t, a))
\]

At test time, we only use the CE head to generate recommendations. Although it is a trade-off between the classic recommendation paradigms and alluring RL methods, the critic framework improves the performance of RS with stability, as shown in our experiments.

### 3 EXPERIMENTS

In order to verify the effectiveness of the proposed VPQ, we conduct experiments on two standard recommender systems datasets, namely Retailrocket [3] and Yoochoose [4], following the data processing details in Xin et al. [24]. We use two metrics: HR (hit ratio) for recall and NDCG (normalized discounted cumulative gain) for rank and compute the performance on clicked and ordered items separately. For example, we define HR for clicks as:

\[
\text{HR(click)} = \frac{\# \text{hits among clicks}}{\# \text{clicks}}
\]

### 3.1 Performance Gains from VPQ

We integrate VPQ with three classic sequential RS methods: Caser [20], NextItNet [26] and SASRec [11]. We also compare its performance with the following online and offline RL-based methods: SAC [24], REM [2], Minus (involved the \( p\text{-sub} \) formulation in Equation (1)), CQL [15], and UWAC [23]. All the above methods are embedded with the classic RS models under the critic framework. We select the best hyper-parameter \( \lambda \) for Minus, \( \alpha \) for CQL, and \( \beta \) for UWAC by sweeping on each dataset. \( \lambda \) for VPQ is set to 20.

Code is available at [https://drive.google.com/file/d/1sEl2TkscyCAXRH28OxNLphE1H9sSm?usp=sharing](https://drive.google.com/file/d/1sEl2TkscyCAXRH28OxNLphE1H9sSm?usp=sharing)

1https://www.kaggle.com/retailrocket/ecommerce-dataset

2https://recsys.acm.org/recsys15/challenge/
Table 1: Overall performance comparison on two recommendation datasets, averaged over 5 runs. NG is short for NDCG.

| Models        | Retailrocket | Yoochoose | Total          |
|---------------|--------------|-----------|----------------|
|               | purchase     | click     | purchase       | click     |               |
|               | HR@20        | NG@20     | HR@20          | NG@20     | HR@20         | NG@20         |               |
| Caser         | .4226        | .3076     | .2813          | .1875     | .6607         | .3911         | .2545         | 2.9665 (0.00%)|
| Caser-SAC     | .4617        | .3336     | .3006          | .1985     | .6889         | .4173         | .4513         | 2.472         | 3.0991 (4.47%)|
| Caser-REM     | .4711        | .3325     | .3019          | .1999     | .6916         | .4216         | .4481         | 2.452         | 3.1184 (5.12%)|
| Caser-Minus   | .4618        | .3325     | .2991          | .1984     | .7029         | .4330         | .4429         | 2.419         | 3.1124 (4.92%)|
| Caser-CQL     | .4699        | .3376     | .3048          | .2009     | .6705         | .4008         | .4598         | 2.527         | 3.0970 (4.40%)|
| Caser-VPQ     | .4775        | .3454     | .3067          | .2036     | .7081         | .4350         | .4459         | 2.439         | 3.1661 (6.73%)|
| Next          | .6564        | .4961     | .3329          | .2130     | .5934         | .3352         | .4937         | 2.715         | 3.3922 (0.00%)|
| Next-SAC      | .6741        | .5354     | .3346          | .2149     | .5899         | .3383         | .4730         | 2.584         | 3.4186 (0.78%)|
| Next-REM      | .6839        | .5387     | .3433          | .2216     | .6025         | .3509         | .4781         | 2.623         | 3.4812 (2.62%)|
| Next-Minus    | .6835        | .5432     | .3446          | .2227     | .6047         | .3511         | .4771         | 2.622         | 3.4890 (2.85%)|
| Next-CQL      | .6845        | .5399     | .3423          | .2215     | .6073         | .3509         | .4802         | 2.635         | 3.4901 (2.89%)|
| Next-UWAC     | .6840        | .5310     | .3452          | .2206     | .6155         | .3538         | .4861         | 2.694         | 3.5102 (3.48%)|
| Next-VPQ      | .6990        | .5714     | .3556          | .2330     | .6226         | .3672         | .4944         | 2.744         | 3.6175 (6.64%)|
| SASRec        | .6387        | .4599     | .3516          | .2190     | .6630         | .3733         | .5044         | 2.761         | 3.4861 (0.00%)|
| SASRec-SAC    | .6981        | .5629     | .3603          | .2347     | .6786         | .3955         | .5015         | 2.760         | 3.7077 (6.35%)|
| SASRec-REM    | .6655        | .5113     | .3650          | .2361     | .6809         | .3986         | .5033         | 2.779         | 3.6387 (4.37%)|
| SASRec-Minus  | .6666        | .5125     | .3634          | .2353     | .6815         | .3965         | .5038         | 2.783         | 3.6378 (4.35%)|
| SASRec-CQL    | .7046        | .5599     | .3749          | .2404     | .6631         | .3805         | .5013         | 2.751         | 3.6997 (6.12%)|
| SASRec-UWAC   | .6657        | .5015     | .3715          | .2367     | .6786         | .3959         | .5126         | 2.850         | 3.6475 (4.63%)|
| SASRec-VPQ    | .7171        | .5914     | .3785          | .2484     | .6841         | .4063         | .5081         | 2.814         | 3.8153 (9.44%)|

Figure 2: Comparison of performance of utilizing the learned Q-function in different ways. Error bars show standard deviations.

The consistent performance improvement shown in Table 1 demonstrates that the integration with the learned Q-function generally outperforms the original RS models. Compared to other baseline algorithms, the proposed VPQ achieves more performance gains, illustrating the effectiveness of our approach.

### 3.2 Ablation Study

We conduct ablation study with four settings: (1) **Q-only**, which generates recommendations with the highest Q-values (the most valuables), (2) **CE**, which generates recommendations with only the original model (the most relevant ones), (3) **Q-aux**, minimizing the TD-error as an auxiliary task without utilizing the Q-values (for representation), (4) **Q-critic**, reinforcing the recommendation actions with Q-values (using the loss function in Equation (9)).

Results are shown in Figure 2. **Q-only** achieves the worst performance on both datasets, illustrating that valuable recommendations (items with high Q-values) tend to lose relevance. **Q-aux** surpasses CE, suggesting that minimizing the TD-error benefits representation learning. The results above suggest that improvement of **Q-critic** comes from better state representation (auxiliary loss) and exploiting accurate Q-values (reinforcing the valuable actions).
3.3 Hyper-parameter Study

The scaling factor $\lambda$ controls the strength of uncertainty penalty for VPQ (Equation 3) and thereby affects the performance. For a small scaling factor, VPQ degrades to REM [2] without explicit penalty on OOD predictions. With a large one, VPQ tends to regress on immediate rewards rather than the long-term rewards. Results in Figure 3 verify this for the Next-VPQ model. The best $\lambda$ for the other two base models and the Retailrocket dataset is also 20.

4 CONCLUSION

Directly scaling RL methods into offline setting often end up with unsatisfactory results. This work proposes an uncertainty-based offline RL method, Value Penalized Q-Learning (VPQ), which captures uncertainty information across an ensemble of Q-functions to tackle the overestimations on OOD actions and thus estimates the long-term rewards for RLRS, without estimating the difficult behavior policy and thus suitable for RS scenarios. Evaluations on two standard RS datasets demonstrate that the proposed methods could serve as a gain plug-in for classic sequential RS models.

REFERENCES

[1] M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2021. Reinforcement learning based rec-ommendation systems: A survey. arXiv:2009.06168v2.
[2] Rushabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. 2020. An Optimistic Perspective on Offline Reinforcement Learning. In ICML 2020, 13–18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119). PMLR, 104–114. http://proceedings.mlr.press/v119/agarwal20c.
[3] Gunnar Blohm. 1958. Statistical Estimates and Transformed Beta Variables. Almquist & Wiksell, John Wiley & Sons, Inc., Sweden.
[4] Jacob Buckman, Carles Gelada, and Marc G. Bellemare. 2020. The Importance of Pessimism in Fixed-Dataset Policy Optimization. arXiv:2009.06799 https://arxiv.org/abs/2009.06799
[5] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. 2020. DARL: Datasets for Deep Data-Driven Reinforcement Learning. arXiv:2004.07219 https://arxiv.org/abs/2004.07219
[6] Scott Fujimoto and Shixiang Shen Gu. 2021. A Minimalist Approach to Offline Reinforcement Learning. arXiv:2106.06680 https://arxiv.org/abs/2106.06680
[7] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-Policy Deep Reinforcement Learning without Exploration. In ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97). PMLR, 2052–2062. http://proceedings.mlr.press/v97/fujimoto19a.html
[8] H. Leon Harker. 1961. Expected values of normal order statistics. Statistica Sinica 48, 1 and 2 (1961), 151–165.
[9] Nadasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Ágata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind W. Picard. 2019. Way-Off-Policy Batch Deep Reinforcement Learning of Implicit Human Preferences in Dialogue. arXiv:1907.00456 http://arxiv.org/abs/1907.00456
[10] Ying Jin, Zhurong Yang, and Zhaorong Wang. 2020. Is Pessimism Provably Efficient for Offline RL? arXiv:2012.15085 https://arxiv.org/abs/2012.15085
[11] Xing Xie, and Zhenhui Li. 2018. DRN: A Deep Reinforcement Learning Framework. In SIGIR 2018, Melbourne, VIC, Australia, Australia, November 11-15, 2018. ACM, 582–590. https://doi.org/10.1145/3299670.3299770
[12] Yifan Wu, George Tucker, and Ofir Nachum. 2019. Behavior Regularized Offline Reinforcement Learning. arXiv:1911.11361 http://arxiv.org/abs/1911.11361
[13] Yue Wu, Shaunflei Zhai, Nitisri Srivastava, Joshua M. Suskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. 2021. Uncertainty Weighted Actor-Critic for Offline Reinforcement Learning. In ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 193). PMLR, 11319–11328. http://proceedings.mlr.press/v193/wu21h.html
