Off-Policy Actor-critic for Recommender Systems

Minmin Chen
Google Research
Mountain View, California, USA
minminc@google.com

Can Xu
Google Inc
Mountain View, California, USA
canxu@google.com

Vince Gatto
Google Inc
Mountain View, California, USA
vgatto@google.com

Devanshu Jain
Google Inc
Mountain View, California, USA
devjain@google.com

Aviral Kumar
Google Research
Mountain View, California, USA
aviralkumar@google.com

Ed Chi
Google Research
Mountain View, California, USA
edchi@google.com

ABSTRACT

Industrial recommendation platforms are increasingly concerned with how to make recommendations that cause users to enjoy their long term experience on the platform. Reinforcement learning emerged naturally as an appealing approach for its promise in 1) combating feedback loop effect resulted from myopic system behaviors; and 2) sequential planning to optimize long term outcome. Scaling RL algorithms to production recommender systems serving billions of users and contents, however remain challenging. Sample inefficiency and instability of online RL hinder its widespread adoption in production. Offline RL enables usage of off-policy data and batch learning. It on the other hand faces significant challenges in learning due to the distribution shift.

A REINFORCE agent [3] was successfully tested for YouTube recommendation, significantly outperforming a sophisticated supervised learning production system. Off-policy correction was employed to learn from logged data. The algorithm partially mitigates the distribution shift by employing a one-step importance weighting. We resort to the off-policy actor critic algorithms to addresses the distribution shift to a better extent. Here we share the key designs in setting up an off-policy actor-critic agent for production recommender systems. It extends [3] with a critic network that estimates the value of any state-action pairs under the target learned policy through temporal difference learning. We demonstrate in offline and live experiments that the new framework out-performs baseline and improves long term user experience.

An interesting discovery along our investigation is that recommendation agents that employ a softmax policy parameterization, can end up being too pessimistic about out-of-distribution (OOD) actions. Finding the right balance between pessimism and optimism on OOD actions is critical to the success of offline RL for recommender systems.

CCS CONCEPTS

• **Theory of computation → Reinforcement learning**; • **Information systems → Personalization**.

RecSys '22, September 18–23, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9278-5/22/09.
https://doi.org/10.1145/3523227.3546758

KEYWORDS

reinforcement learning, batch RL, off-policy actor-critic, pessimism, recommender systems

ACM Reference Format:
Minmin Chen, Can Xu, Vince Gatto, Devanshu Jain, Aviral Kumar, and Ed Chi. 2022. Off-Policy Actor-critic for Recommender Systems. In *Sixteenth ACM Conference on Recommender Systems (RecSys '22), September 18–23, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3523227.3546758

1 INTRODUCTION

Recommender systems are heavily relied upon by users to navigate the web. These systems serve users' different information needs by anticipating what they would like to consume next and surfacing relevant contents. Recommendation algorithms powering these systems have evolved over the years, starting from simple collaborative filtering [29, 44] toward deep supervised learning approaches. These algorithms extensively personalize the experience for individual users by predicting their immediate response such as clicks toward recommendations [8, 70]. While they have achieved enormous success in engaging users, their limitations in 1) favoring recommendations that attract short-term engagement; 2) subjecting to strong feedback loop further enhancing such myopic behaviors, are becoming evident in an era where the recommendation platforms are increasingly concerned with keeping users happy in the long run.

Over the years, bandits and reinforcement learning techniques emerged naturally as appealing alternatives [3, 20, 68, 71, 72]. They address the limitations of supervised learning approaches mentioned before: 1) RL provides a mathematical formulation to optimize user-defined long term outcome; With properly defined reward function, these algorithms naturally shift towards learning recommendation policies that optimize long-term user experience as the planning horizon extends. 2) bandits and RL offers tools such as exploration and off-policy learning to combat the feedback loop created by sub-optimal systems. Recognizing observed user behavior is confounded by selection bias of historical system behavior (in favoring certain recommendations over others), these algorithms provide mechanisms to debias and deviate from potentially myopic historical system behavior.

Standing up RL algorithms for production recommender systems however has been proven challenging. RL was designed fundamentally as an online learning paradigm in which the agent can interact with the environment in realtime to collect experience

and improve learning. This setup stands in drastic contrast to the common batch learning setup for production recommender systems. Even if one can adapt the infrastructure to support online learning, instability and sample inefficiency of online RL remain concerns as production recommender systems often need to serve very *dynamic* user and content bases *in the order of billions*. Realizing the interactive nature of online RL has been hindering its wider adoption, researchers seek to bring these techniques offline, which is commonly referred to as batch or offline reinforcement learning [2, 5, 12, 14, 25, 30, 32, 34, 57, 59]. While these techniques enable utilizing off-policy data and batch learning, distribution shift poses significant challenges for learning.

Chen et al. [3] tested a REINFORCE agent for recommendations in YouTube and achieved significant improvement over a sequential recommender trained to predict the next item the user is interested in consuming. The agent learns on historical interaction data collected on the platform, and employs a one-step importance weighting to mitigate the distribution shift. Compared with the full trajectory importance weighting, the one-step approximation produces a slightly biased gradient estimate with much lower variance. We here scale an actor-critic algorithm [16, 28, 55] to production use case to address the distribution shift to a better extent. Together, we make the following contributions:

- **An Off-Policy Actor-critic Algorithm**: We present an off-policy actor-critic method [10, 16, 34, 55] for recommendation that addresses the distribution shift resulted from off-policy learning. It adds a critic network that estimates the value of any state-action pair under the target learned policy through temporal difference learning;
- **Practical Lessons and Ablation Study**: We share lessons on designing the critic network and conduct thorough ablation study to tease apart critical components that improve the learning of the critic network, e.g., temporal difference learning setup, importance of features, target networks, and architecture choices.
- **Offline Analysis Tool**: We offer an offline analysis tool that compares the estimated Q-value from the critic network and the monte-carlo return to understand the preference of the target policy vs the behavior policy;
- **Live Experiments**: We test the proposed method live on a commercial recommendation platform serving billions of users and items, and showcase the benefit of the new approach in improving long-term user experience and its scalability. We corroborate findings in live experiments with the offline analyses and share thoughts on future directions to improve offline RL for recommendations.

2 RELATED WORK

Offline Reinforcement Learning. Classic reinforcement learning approaches [55] can be roughly grouped into model-based and model-free variants, with the latter one further divided into value-based approaches such as Q-learning [63] and policy-based ones such as REINFORCE [64]. Modern RL approaches combine these techniques with high-capacity function approximators, i.e., deep neural networks, achieving immense success in various domains [17, 35, 52]. Online RL however is notoriously sample inefficient, making them less applicable to many real world tasks. Inspired by the data-driven success of supervised deep learning, offline RL [34] where one can utilize off-policy data often available in abundance, is seen as the solution to succeed RL in real life. Offline RL however poses major challenge in correctly estimating values of state-action pairs that have deficient support in the off-policy data. Prior works have attempted to solve this problem by constraining the learned policy to be "close" to the behavior policy used for offline data collection [41, 46, 47, 50, 65]. Another line of work avoids the need to estimate the behavior policy by implicitly penalizing actions not seen in the off-policy data [26, 31]. A lot of recent work in offline RL have been devoted to learning only the action-value functions and determine a policy exclusively from the estimated values, for example through max operations. We instead focus on actor–critic methods [55] with an explicit policy parameterization for two reasons: 1) it allows for explicitly learning a stochastic policy for introducing exploration [3]; 2) it implicitly induces pessimism, which avoids over-estimation issues commonly seen in value-based approaches. While the actor-critic methods have long been developed with different variants [16, 55], our focus is on piecing together key components needed to make it successful for production recommender systems.

Reinforcement Learning for Recommender Systems. The promise of deep RL sparked a lot of interest in utilizing it for recommender systems. Shani et al. [49] formulated recommendation problem as a Markov decision process (MDP) and proposed a model-based approach. Zheng et al. [74] explored DQN for news recommendation. Dulac-Arnold et al. [11] studied RL for application with large discrete action space. Liu et al. [36] studied an actor-critic approach for recommendation. Model-based RL approaches [7, 48, 75] have also been studied for recommendation problems but face significant challenges in accurately capturing the complex dynamics and user response models in recommendations. Among these, [36] is most relevant to ours. However, Liu et al. [36] studies actor-critic methods in online setting while we are interested setting it up in offline settings that scales to production usage. Xin et al. [68] introduced an algorithm named Self-supervised Actor Critic for recommender systems which greatly resembles an actor critic method but replaces the actor with a weighted cross-entropy loss ignoring the off-policy effect. Another related line of work [4, 24, 27, 66] extends policy based approaches with models to predict the immediate reward. Several recent works further extend deep RL to set recommendation [3, 20, 73] and ranking [67] problems. Despite exciting research advances, success of RL in real-world recommendation applications is still rare. Chen et al. [3] scaled a REINFORCE agent to handle extremely large state and action spaces by learning on historical user interaction data and applying off-policy correction to address the distribution shift. Hu et al. [19] framed learning to rank as an MDP and learned a ranking policy through deterministic policy gradient (DPG) and tested the approach on a commercial platform.

3 BACKGROUND

Following [3], we translate sequential recommendation into a Markov Decision Process [1] $(\mathcal{S}, \mathcal{A}, \mathbf{P}, R, \rho_0, \gamma)$ where

[1]The user state is not fully observed, making the problem a Partially Observed Markov Decision Process (POMDP). For simplicity, we leave the notations as is.

- $\mathcal{S}$: a continuous state space describing the user states; Here $s_t = f(o_{1\cdots t-1})$ infers the latent user state at timestamp t based on the user's historical interactions $o_{1\cdots t-1}$;
- $\mathcal{A}$: a discrete action space, containing items available for recommendation;
- $\mathbf{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the state transition probability;
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, where $r(s, a)$ captures the immediate reward obtained by performing action a at user state s; One can imagine $r(s, a)$ to capture different aspects of user feedback toward the recommendation;
- ρ_0 is the initial state distribution;
- γ is the discount factor for future rewards.

The goal of the recommender agent is to learn a policy $\pi_\theta(\cdot|s)$ to maximize the cumulative discounted reward,

$$\max_\theta \quad \mathcal{J}_\pi(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\right], \text{ where } R(\tau) = \sum_{t=0}^{|\tau|} \gamma^t r(s_t, a_t) \quad (1)$$

Note the expectation is taken over trajectories $\tau = (s_0, a_0, s_1, \cdots)$ obtained by sampling according to the policy: $s_0 \sim \rho_0, a_t \sim \pi_\theta(\cdot|s_t)$, $s_{t+1} \sim \mathbf{P}(\cdot|s_t, a_t)$. In other words,

$$\begin{aligned} &\mathcal{J}_\pi(\theta) \\ &= \textstyle\sum_{t=0}^{|\tau|} \mathbb{E}_{s_0\sim\rho_0, a_t\sim\pi_\theta(\cdot|s_t), s_{t+1}\sim\mathbf{P}(\cdot|s_t,a_t)} \left[\sum_{t=0}^{|\tau|} \gamma^t r(s_t, a_t)\right] \\ &= \textstyle\sum_{t=0}^{|\tau|} \mathbb{E}_{s_t\sim d_t^{\pi_\theta}(\cdot), a_t\sim\pi_\theta(\cdot|s_t)} \left[\gamma^t \sum_{t'=t}^{|\tau|} \gamma^{t'-t} r(s_{t'}, a_{t'})\right] \end{aligned} \quad (2)$$

where $d_t^{\pi_\theta}(\cdot)$ denotes the (discounted) state visitation frequency at time t under the policy π_θ.

3.1 REINFORCE with Off-Policy Correction

Different families of methods are available to solve such an RL problem, e.g., value-based [51], policy-based [33, 46, 64], model-based [45] and black box optimization [18]. We focus on a policy-based approach, i.e., REINFORCE [64]. Taking gradient of expected return in eq. (2) w.r.t. the policy parameters θ yields the following REINFORCE gradient

$$\nabla_\theta \mathcal{J}_\pi(\theta) = \sum_{t=0}^{|\tau|} \mathbb{E}_{s_t\sim d_t^{\pi_\theta}(\cdot), a_t\sim\pi_\theta(\cdot|s_t)} \left[R_t(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)\right] \quad (3)$$

where $R_t(s_t, a_t) = \left(\sum_{t'=t}^{|\tau|} \gamma^{t'-t} r(s_{t'}, a_{t'})\right)$. Note the γ^t term in eq.(2) is dropped in practice [56]. This however assumes the agent can interact with the environment to collect trajectories based on the latest learned policy $\tau \sim \pi_\theta$, and then improves its policy using the collected trajectories. Unfortunately in many real-world applications of RL such as recommendations, such an on-policy learning setup is not feasible. Instead the agent is required to learn and improve its policy based on offline trajectories generated from a different behavior policy $\tau \sim \beta$, a setup commonly referred to as batch reinforcement learning [2, 5, 12, 14, 25, 30, 32, 34, 57, 59].

Naively, one can approximate the REINFORCE gradient as defined in eq. (3) using trajectories generated from β with the classic importance sampling correction,

$$\begin{aligned} \nabla_\theta \mathcal{J}_\pi(\theta) = \sum_{t=0}^{|\tau|} \mathbb{E}_{s_t\sim d_t^\beta(\cdot), a_t\sim\beta(\cdot|s_t)} \Bigg[&\frac{d_t^{\pi_\theta}(s_t)}{d_t^\beta(s_t)} \frac{\pi_\theta(a_t|s_t)}{\beta(a_t|s_t)} \times \\ &\left(\prod_{t'=t+1}^{|\tau|} \frac{\pi_\theta(a_{t'}|s_{t'})}{\beta(a_{t'}|s_{t'})}\right) R_t^\beta(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \Bigg] \end{aligned} \quad (4)$$

where $R_t^\beta(s_t, a_t) = \sum_{t'=t, a_{t'}\sim\beta(\cdot|s_{t'})}^{|\tau|} \gamma^{t'-t} r(s_{t'}, a_{t'})$ is the discounted sum of rewards observed on the trajectories generated by the behavior policy. Importance weighting the entire trajectory however leads exponentially growing variance in the gradient estimate, known as "the curse of horizon" [37]. Chen et al. [3] reduce the variance by taking a first-order approximation [1]:

$$\begin{aligned} &\nabla_\theta \mathcal{J}_\pi(\theta) = \textstyle\sum_{t=0}^{|\tau|} \mathbb{E}_{s_t\sim d_t^\beta(\cdot), a_t\sim\beta(\cdot|s_t)} \\ &\left[\frac{\pi_\theta(a_t|s_t)}{\beta(a_t|s_t)} R_t^\beta(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)\right] \end{aligned} \quad (5)$$

This importance weight is further adapted to set recommendation settings. We refer interested readers to [3] for more details. The first-order approximation resulted in gradient estimates of lower variance but biased compared with full-trajectory importance weighting. Specifically, the cumulative future reward $R_t^\beta = \sum_{t'=t}^{|\tau|} \gamma^{t'-t} r(s_{t'}, a_{t'})$ computed using monte carlo estimate on the trajectories from β does not reflect the value of the state-action pair (s_t, a_t) under the target policy π_θ.

4 METHOD

To fully correct the distribution shift under off-policy learning while controlling the variance, we adopted the actor-critic method [16, 55] which introduces an additional critic network to estimate the value of the state-action pair at the target learned policy under consideration. Here we discuss the practical lessons learned from building such a critic network.

4.1 Off-Policy Actor-Critic

In addition to the parameterized policy (actor) network π_θ, an actor critic method introduces another parameterized network called a critic to estimate the expected return of any state-action pair under the target learned policy $Q_\phi^{\pi_\theta}(s_t, a_t)$. One can then replace the monte carlo return $R_t^\beta(s_t, a_t)$ used in eq. (3) and eq. (5) with the estimated $Q_\phi^{\pi_\theta}$ value, arriving at the following update for the policy network,

$$\begin{aligned} &\nabla_\theta \mathcal{J}_\pi(\theta) = \textstyle\sum_{t=0}^{|\tau|} \mathbb{E}_{s_t\sim d_t^\beta(\cdot), a_t\sim\beta(\cdot|s_t)} \\ &\left[\frac{\pi_\theta(a_t|s_t)}{\beta(a_t|s_t)} Q_\phi^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)\right] \end{aligned} \quad (6)$$

As the critic network estimates the value of the state-action pair under the target policy π_θ, the update removes the bias resulted from using the cumulative future return on the behavior trajectory $R_t^\beta(s_t, a_t)$ while ignoring the importance weighting on the

future trajectories $\prod_{t'=t+1}^{|\tau|} \frac{\pi_\theta(a_{t'}|s_{t'})}{\beta(a_{t'}|s_{t'})}$. Note the first-order importance weighting $\frac{\pi_\theta(a_t|s_t)}{\beta(a_t|s_t)}$ remains to address the selection bias that $a_t \sim \beta(\cdot|s_t)$ rather than the learned policy.[2]

By the MDP definition, the Q function of the target learned policy satisfies

$$\begin{aligned} Q_\phi^{\pi_\theta}(s_t, a_t) &= r(s_t, a_t) \\ &+\gamma \mathbb{E}_{s_{t+1}\sim \mathbf{P}(\cdot|s_t,a_t), a_{t+1}\sim\pi_\theta(\cdot|s_{t+1})}[Q_\phi^{\pi_\theta}(s_{t+1}, a_{t+1})] \end{aligned} \tag{7}$$

We learn the parameters of the critic network by minimizing the temporal difference (TD) loss,

$$\min_\phi \mathcal{J}_Q(\phi) = \sum_{t=0}^{|\tau|} \mathbb{E}_{s_t\sim d_t^\beta(\cdot), a_t\sim\beta(\cdot|s_t)} \left[\left(\hat{Q}_\phi^{\pi_\theta}(s_t, a_t) - Q_\phi^{\pi_\theta}(s_t, a_t) \right)^2 \right] \tag{8}$$

where the target

$$\begin{aligned} \hat{Q}_\phi^{\pi_\theta}(s_t, a_t) &= r(s_t, a_t) \\ &+\gamma \mathbb{E}_{s_{t+1}\sim \mathbf{P}(\cdot|s_t,a_t), a_{t+1}\sim\pi_\theta(\cdot|s_{t+1})}[Q_\phi^{\pi_\theta}(s_{t+1}, a_{t+1})] \end{aligned} \tag{9}$$

To deal with outliers in the TD loss, we replaced the squared difference with a Huber loss of $\delta = 10$.

4.2 Estimating target $\hat{Q}_\phi^{\pi_\theta}(s_t, a_t)$

Assuming access to logged trajectories $\mathcal{D} = \left\{\tau : \{(s_t, a_t, r_t)\}_{t=0,\cdots,|\tau|}\right\}$ from the behavior policy: $s_0 \sim \rho_0, a_t \sim \beta(\cdot|s_t), s_{t+1} \sim \mathbf{P}(\cdot|s_t, a_t)$, we investigate two ways to estimate the target $\hat{Q}_\phi^{\pi_\theta}(s_t, a_t)$.

Importance Sampling from Behavior policy. Here we take the next state-action pair observed on the trajectory from the behavior policy: $(s_t, a_t, s_{t+1}, a_{t+1}) \sim \beta$, and estimate the target through importance sampling again

$$\hat{Q}_{IS,\phi}^{\pi_\theta}(s_t, a_t) = r(s_t, a_t) + \gamma \frac{\pi_\theta(a_{t+1}|s_{t+1})}{\beta(a_{t+1}|s_{t+1})} Q_{IS,\phi}^{\pi_\theta}(s_{t+1}, a_{t+1}), \tag{10}$$

where $a_{t+1} \sim \beta(\cdot|s_{t+1})$. The advantages of such an approach are:

- The estimate is unbiased if we have full support in β, meaning $\beta(a_{t+1}|s_{t+1}) = 0 \rightarrow \pi_\theta(a_{t+1}|s_{t+1}) = 0$;
- Q estimates of the next state-action pairs (s_{t+1}, a_{t+1}) that have support in the behavior policy are in general more accurate, resulting in less bootstrapping error;
- As both the current and next state-action pairs (s_t, a_t) and (s_{t+1}, a_{t+1}) chosen by the behavior policy have been served to the user and the user has provided feedback on, the critic network can include user feedback as inputs, resulting in more accurate estimates;

with the following shortcomings:

- When the target policy has deficient support meaning the learned policy can choose actions that are not covered by the behavior policy, we under-estimate the value [43];
- Importance sampling introduces variance in the estimate especially when the learned policy deviates significantly from the behavior policy.

[2]The bias caused by ignoring the state visitation frequency difference between the target policy and the behavior policy $\frac{d_t^{\pi_\theta}(s_t)}{d_t^\beta(s_t)}$ remains.

Sampling from Learned policy. As the policy (actor) network parameterizes an explicit policy, the conventional approach is to directly sample the next action from the target learned policy and estimate the target as

$$\hat{Q}_{S,\phi}^{\pi_\theta}(s_t, a_t) = r(s_t, a_t) + \gamma Q_{S,\phi}^{\pi_\theta}(s_{t+1}, a'_{t+1}) \tag{11}$$

where $a'_{t+1} \sim \pi_\theta(\cdot|s_{t+1})$. The advantages of this approach are

- It address the support deficiency concern when estimating using behavior data with importance sampling;
- One can easily reduce the variance in the estimate by sampling multiple a'_{t+1} and average the resulted Q estimates;

with the following shortcomings

- When sampling an action that has little to no support in the behavior policy, we expect the estimated Q value to be less accurate since the action embedding might not be well learned;
- As we sample an action that has not being recommended to the user at the state s_{t+1}, we were not able to observe user response toward the item, as a result, we cannot include any feedback signal in building the critic network.

We will compare these two approaches in detail in Section 5 and 6.

Sampled Softmax. Since the action space of industrial recommenders is huge, in the orders of millions or billions, sampling from the full softmax policy $\pi_\theta(\cdot|s_{t+1})$ is prohibitively expensive. We instead follow the sampled softmax idea introduced in [22] during training. First, we sample a set $C \subset \mathcal{A}$ following a global categorical distribution μ, where $\mathcal{A}$ contains the entire recommendable corpus and $\mu(a), \forall a \in \mathcal{A}$ is proportional to the global popularity of item a. We then sample an action a'_{t+1} from C according to $\pi_\theta(a|s_{t+1}, C) \sim \exp\left(\log \pi_\theta(a|s_{t+1}) - \log \mu(a)\right), \forall a \in C$ [3]. Note that the same sampled softmax approximation is used to compute $\pi_\theta(\cdot|s_{t+1})$ and $\beta(\cdot|s_{t+1})$ in the importance sampling approach.

4.3 Target Network

As explained in section 4.1, we estimate the value of $Q_\phi^{\pi_\theta}(s_t, a_t)$ from the estimate of a subsequent state-action pair in temporal difference (TD) learning. With function approximation, one can always expect a non-zero residual TD-error in each update. The TD errors accumulate as a result of bootstrapping from subsequent estimates [13], especially with a large discount factor γ. As the estimated $Q_\phi^{\pi_\theta}$ value is used to update the policy as in eq.(6), the estimation error can easily derail the learning.

Target network [13, 60] is a common trick employed in deep RL to stabilize the TD learning. As deep function approximators require multiple gradient updates to converge, a target network provides a stable target for the critic network to learn. In other words, we estimate the TD target $\hat{Q}_{\phi'}^{\pi_\theta}(s_t, a_t)$ using a separate set of parameters ϕ' which has a update frequency that is much slower than the parameters ϕ used for the critic network. We follow the Polyak averaging approach, setting the target network parameters $\phi' = \phi$ at initialization, and updating the parameters as $\phi' \leftarrow$

[3]Please refer to [22] and https://www.tensorflow.org/extras/candidate_sampling.pdf for derivation. Note the estimate is biased by a factor $K(C, s_{t+1})$, which equals 1 if $\pi_\theta(|s_{t+1})$ has the same chance of sampling the set C as the global distribution μ. This can be used to design the sampling distribution μ.

$(1-\alpha)\phi' + \alpha\phi$, where ϕ are the parameters of the critic network, and $\alpha \in [0, 1)$ is the update factor. While smaller α reduces oscillation in the target and let the critic network converges, too small of a α value also causes the critic to regress toward a stale target computed using out-of-date network parameters. In our implementation, we used $\alpha = 0.1$.

4.4 Overall Architecture

Figure 1 depicts the overall architecture of the off-policy actor-critic agent. The left part shows the components used to form the policy network π_θ as well as the ones used to estimate the behavior policy $\beta_{\theta'}$. The right part shows the components used to build the critic and target network. As shown in the figure, there are a lot of component sharing between different parts of the agent to improve learning efficiency. The dotted lines in the figure indicate stop gradient operations where the loss of the latter component does not change the parameters on the former component.

The policy network (as well as the behavior policy network) on the left have two main parts to generate the user state representation $\mathbf{u}_{s_t}$ and item representation $\mathbf{v}_{a_t}$ ($\mathbf{w}_{a_t}$) respectively. It captures the user interests through encoding historical user item sequence $o_1, o_2, \cdots, o_{t-1}$ using RNNs. The resulted RNN hidden state, concatenated with context features forms the overall state representation on which the target policy as well as the behavior policy will condition

$$\pi_\theta(a_t|s_t) = \frac{\exp(\frac{\mathbf{u}_{s_t}^\top \mathbf{v}_{a_t}}{T})}{\sum_{a' \in \mathcal{A}} \exp(\frac{\mathbf{u}_{s_t}^\top \mathbf{v}_{a'}}{T})} \quad \beta_{\theta'}(a_t|s_t) = \frac{\exp(\frac{\mathbf{u}_{s_t}^\top \mathbf{w}_{a_t}}{T})}{\sum_{a' \in \mathcal{A}} \exp(\frac{\mathbf{u}_{s_t}^\top \mathbf{w}_{a'}}{T})} \tag{12}$$

The temperature T controls the sharpness of the policies. The behavior policy network shares most of the parameters with the policy network except the action embeddings. The two however are trained on different objectives. The target policy network is updated using the gradient as in eq. (6) $\nabla_\theta \mathcal{J}_\pi(\theta)$ with an additional entropy regularization $\mathcal{R}_\pi(\theta)$ [42] to increase entropy of the learned policy [4]. Entropy regularization alone has been shown to capture most of the gain coming from the maximum entropy RL framework such as the Soft Actor Critic algorithm [16, 69]. The behavior policy network on the other hand is learned using a behavior cloning loss (cross-entropy) $\mathcal{J}_\beta(\theta')$ [54] to recover distribution of actions conditioning on the state in the off-policy data. The estimated $\beta_{\theta'}(a_t|s_t)$ is used to compute the importance weight in both the policy update as in eq. (6) and the target $\hat{Q}_\phi^{\pi_\theta}$ estimate as in eq. (10). Both policies take the softmax form over the entire action space. The same sampled softmax trick as mentioned in Section 4.2 is used to approximate the full softmax during training to deal with the large action space. Efficient nearest neighbor search [38] between the state representation $\mathbf{u}_{s_t}$ and the action embeddings $\mathbf{v}_a, \forall a \in \mathcal{A}$, combined with Boltzmann exploration [9] is used at serving time to retrieve recommendations.

The critic network (as well as the target network) on the right, take the state representation which includes both user interests and context information, and representation of the action (and possibly the feedback on the state-action pair) as inputs. The concatenation of these features is sent through a multi-layer relu network and finally through a linear projection, producing a scalar estimate of the expected return of the state-action pair under the learned policy $Q_\phi^{\pi_\theta}(s_t, a_t)$. The estimated value is then fed to the policy network for learning. The parameters of the critic network is learned through the temporal difference loss $\mathcal{J}_Q(\phi)$ as in eq. (8). The target network is a copy of the critic network with parameters ϕ' synced to ϕ at a lower pace with the Polyak averaging as explained in Section 4.3.

5 OFFLINE EXPERIMENTS

Recommender systems operate under the partial feedback setup, in which we only observe feedback on items that have been recommended by the deployed policy, but not others. As a result, the off-policy effect has to be accounted for in evaluation in order to estimate the performance of a new policy when deployed. Offline (off-policy) evaluation is a challenging problem by itself [34]. One might argue it is even harder than off-policy learning as former requires accurate estimate of the absolute value of an arbitrary policy while latter only requires finding a policy improvement direction. Existing works mostly leverage inverse-propensity scores, with various variance control techniques such as capping or normalization in order to reduce the variance in the off-policy estimate [15, 37, 40, 57–59]. Even with these variance control techniques, off-policy evaluation often results in an estimate that is noisier than desired for discerning performance of different policies. We thus rely on live experiments to measure the true benefits of the algorithm. Here we focus on using offline experiments to shed light on the design of the critic networks.

5.1 Dataset and Setup

We extract hundreds of millions of user trajectories from a commercial recommendation platform. Each trajectory contains a list of user interactions up to the point of training, $\tau = \{(s_t, A_t, a_t, r_t) : t = 0, \ldots, T\}$. Here A_t denotes all the items that have been recommended to the user at time t, with a_t denotes the item the user interacted with, and r_t being the feedback provided by the user (immediate reward). r_t is a scalar reflecting different aspects of the user feedback/experience, such as engagement, responsibility, satisfaction and exploration [5]. Note that a_t can be empty indicating user did not interact with any item from A_t, and $r_t = 0$ in this case. $s_t = f(o_{1\cdots t-1}) = f(A_{1\cdots t-1}, a_{1\cdots t-1}, r_{1\cdots t-1})$ captures the latent user states based on past observations. The lengths of trajectories between users can vary. Among the collected trajectories, we hold out 1% for evaluation. For our experiments, we set the action space (item corpus) to include the most popular 10 million items. Our goal is to build a recommender agent that chooses among the 10 million corpus the next set of items for users to consume so as to maximize the cumulative long-term reward.

The parameters in the policy, behavior policy as well as the critic networks (θ, θ', ϕ) are learned jointly to minimize the loss function $\mathcal{J}_\pi(\theta) + c\mathcal{R}_\pi(\theta) + \mathcal{J}_\beta(\theta') + \mathcal{J}_Q(\phi)$ as depicted in Figure 1. At each gradient update, we sample 1024 user trajectories and take any state-action-reward-state-action tuple $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ within the most recent 6 hour window to estimate the gradient. We focus

[4] Note that we included the entropy regularization in the off-policy corrected REINFORCE baseline across offline and live experiments for fair comparison.

[5] We refer interested readers to multi-objective RL [39, 53, 61, 62] for more sophisticated approaches to handle different objectives in recommendation.

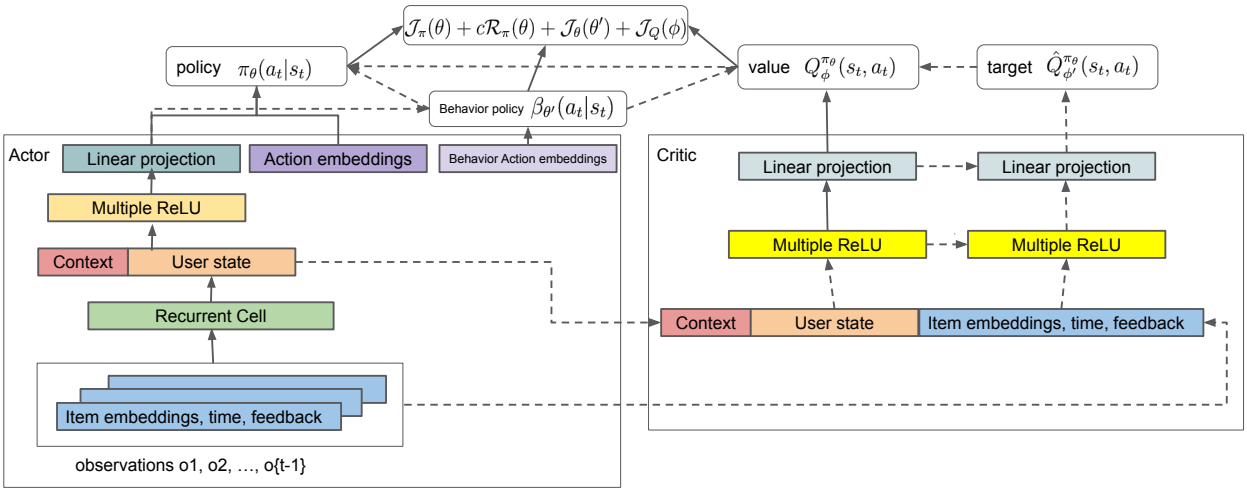


Figure 1: Off-policy actor-critic architecture.

our discussion here on the critic loss $\mathcal{J}_Q(\phi)$. For these experiments, we estimate the target $\hat{Q}_\phi^{\pi_\theta}(s_t, a_t)$ using the importance sampling from behavior policy approach explained in Section 4.2.

5.2 Importance of Features

In the first set of experiments, we study the importance of different features of the state-action pair in learning the critic network. As shown in Figure 1, the critic network is a feed-forward neural network that takes representation of the user state, context, the item, as well as features connecting the user state and the item as inputs. We kept the user state and context unchanged, and partition the remaining features into four groups: 1) item embeddings; 2) time delta indicating the time interval between the state-action pair (s_t, a_t) and the next state-action pair (s_{t+1}, a_{t+1}); 3) novelty/exploration related feature indicating if the item is novel to the user compared to historical items the user has consumed; 4) $\vec{r}(s_t, a_t)$ reward signals capturing different aspects of the immediate user feedback on the recommended item a_t.

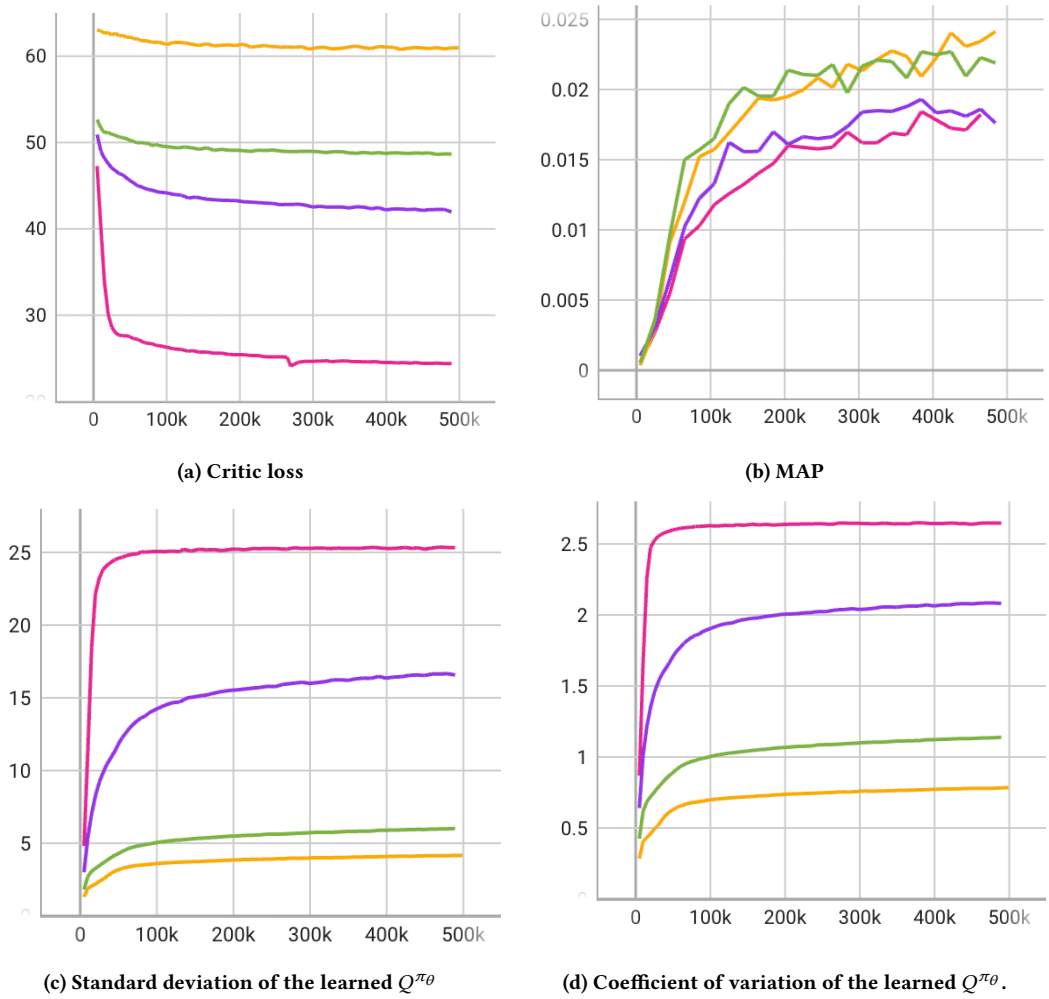
Figure 2a shows the critic loss when different feature groups are used in building the critic network. The yellow curve includes user state and context features along with the 1) item embeddings; the green curve further adds the 2) time delta feature; the purple curve adds the 3) novelty features; and finally the magenta curve includes 4) the immediate reward signals. As one can see in the figure, adding each group of features reduces the critic loss. The reward signals are the most effective in reducing the critic loss, which is not surprising as user's immediate feedback on the recommendations is expected to correlate well with their long term enjoyment. The observation motivates our design to use the next state-action pair appearing in the logged data, on which we have observed user feedback, to estimate the target $\hat{Q}_{IS,\phi}^{\pi_\theta}(s_t, a_t)$ for TD learning, rather than the conventional approach of sampling the next state-action pair from the learned policy $\hat{Q}_{S,\phi}^{\pi_\theta}(s_t, a_t)$. We will compare their performance in live experiments as well.

Figure 2b shows the Mean Average Precision (MAP@1) of the learned policy. In other words, whether the item the user chose to interact with next is the top-1 item returned by the agent. Perhaps counter-intuitively, one observes that the critic network with the highest loss (using only item embeddings) leads to a policy of the highest MAP metrics. This however is due to the partial feedback setup where we only observe feedback on items recommended by the behavior policy. As a result, a policy closest to the behavior policy will have the highest MAP in offline evaluation but will perform sub-optimally in live. The critic network with the least expressive features produces a $Q^{\pi_\theta}(s_t, a_t)$ estimate of the lowest variance as shown in figure 2c and 2d. If $Q^{\pi_\theta}(s_t, a_t) \equiv const$, ignoring the off-policy correction term, the gradient update in eq. (6) will be equivalent to that of behavior cloning. As a result, the policy network with the least expressive critic outputs a policy that resembles the behavior policy the most, leading to higher MAP. Similar concern was raised in [21].

5.3 Difference between $Q^{\pi_\theta}(s, a)$ and $R^\beta(s, a)$

One of the key challenges in applying RL for recommender systems is how to design the reward function to achieve desired behavior. Comparing eq. (5) with (6), the only difference between updates for the policy network in the baseline REINFORCE with off-policy correction and the off-policy actor-critic method is the former uses the monte carlo return on the behavior trajectory $R_t^\beta(s, a)$ and the latter uses the critic estimated $Q_\phi^{\pi_\theta}(s, a)$ of the learned policy as the target. Here we look at the ratio $\frac{Q_\phi^{\pi_\theta}(s_t, a_t)}{R_t^\beta(s_t, a_t)}$ among the state-action pairs $(s, a) \sim d_t^\beta(s_t)\beta(a_t|s_t)$, and how the ratio correlates with different attributes of the recommended item as a way to study how the critic network (and as a result the learned policy) is favoring certain items over others, comparing with the baseline.

As shown in figure 3a, we saw strong negative correlation between the ratio and item popularity (by number of interactions the item accumulated during the training window), suggesting the learned policy under the actor-critic methods favors less-popular



(a) Critic loss

(b) MAP

(c) Standard deviation of the learned Q^{π_θ}

(d) Coefficient of variation of the learned Q^{π_θ}.

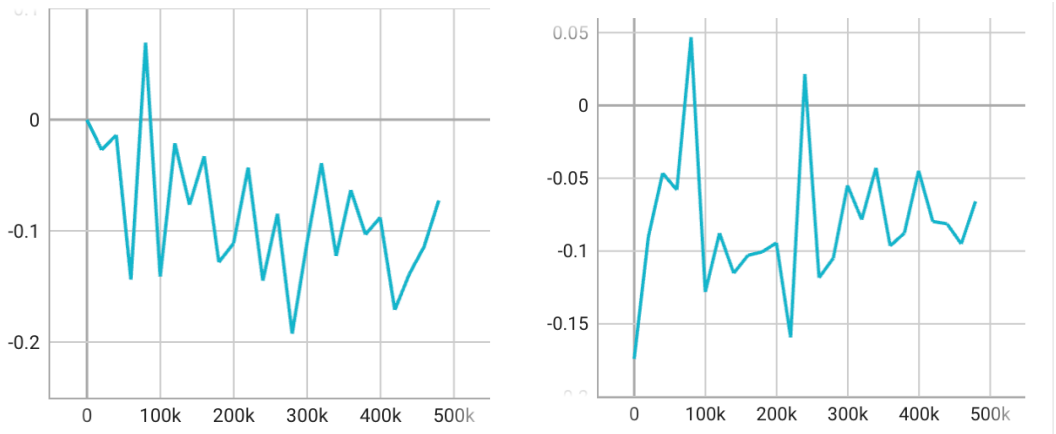
Figure 2: Importance of features in learning the critic network. X-axis are the training steps.

items than the baseline. Similarly, we observed a negative correlation between the ratio and the immediate engagement on the item itself (figure 3b), suggesting the new policy is optimizing more for the future rather than the immediate engagement of the item itself. We also observed a negative correlation of the ratio w.r.t. the time delta between a_t and the next state-action pair with user feedback (figure 3c). Intuitively, one would expect weaker causal relation between the recommendation a_t to the next user feedback that happens one day later than the one that happens one minute later. Figure 3d plots the correlation of the ratio to a binary indicator indicating if the item is novel to the user by comparing the item to a user's historical consumption. An item is novel if its topic cluster [6] is different from any of the user's historical interactions. The correlation coefficient is noisier than others.

6 LIVE EXPERIMENTS

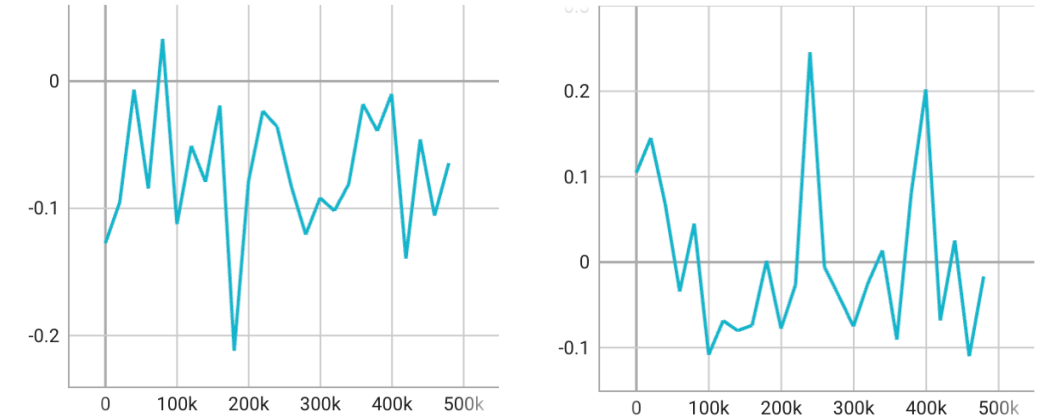
We conduct a series of A/B experiments in a live system serving billions of users to measure the performance of the off-policy actor-critic agent. The agent is built to retrieve hundreds of candidates from a corpus of 10 million items upon each user request. The retrieved candidates, along with those returned by other sources, are scored and ranked by a separate ranking system before showing the top results to the user. The control runs a REINFORCE agent with one-step off-policy correction as explained in Section 3. The experiment arm runs the off-policy actor critic agent as explained in Section 4.1 with the target for critic estimated as $\hat{Q}^{\pi_\theta}_{IS,\phi}$ as explained in Section 4.2. Experiments were run for four weeks, during which both the control and experiment models are updated continuously with new interaction and feedback being used as training data.

Figure 4 summarizes the live experiment results. We can see that off-policy actor-critic improves over the baseline REINFORCE agent on the topline metric which captures overall user enjoyment on the platform, leading to a +0.07% improvement with a 95% confidence interval of [+0.02%, +0.12%], as shown in Figure 4a. The remaining sub-figures examine how the new policy changes its recommendation based on various item attributes as we examined in figure 3 in offline experiments. As shown in figure 4b, we saw



(a) Item popularity

(b) Immediate engagement on the item itself



(c) Time delta from a_t to next user feedback

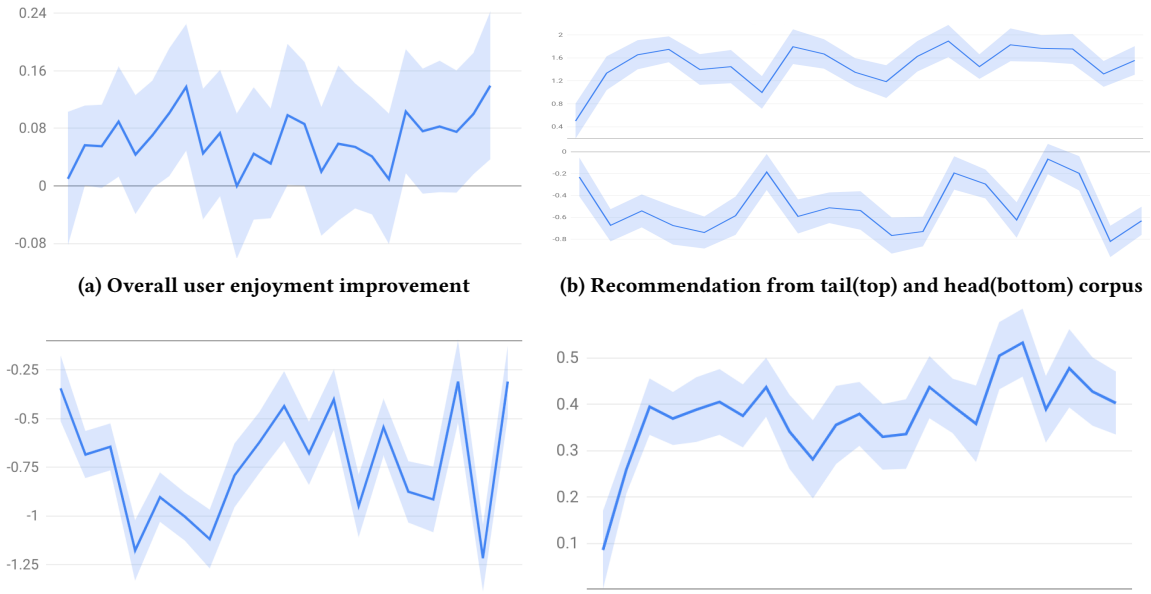
(d) Whether or not a_t is novel to the user

Figure 3: Correlation of the ratio between the Q-value and the behavior monte carlo return $\frac{Q_\phi^{\pi_\theta}(s_t,a_t)}{R_t^\beta(s_t,a_t)}$ with different item attributes. The x-axis plots the training steps, and the y-axis plots the correlation coefficients across training step.

+1.5% more recommendations coming from tail corpus (with correspondingly +0.97% user enjoyment improvement from this slice of corpus), −0.51% reduction in recommendations from top corpus (the user enjoyment metric from this slice of corpus did not change). This aligns with the observation in figure 3a that the new policy prefers less popular items. Similarly, corroborating the result we saw in figure 3b that the $\frac{Q_\phi^{\pi_\theta}(s_t,a_t)}{R_t^\beta(s_t,a_t)}$ ratio negatively correlates with immediate feedback on the recommended item, the new policy recommends items of shorter length in live experiments as shown in figure 4c, while improving the overall long term user enjoyment. As shown in figure 4d, the new policy causes the users to interact with more and more new topics over time compared with the control arm, which has been shown to lead to long term user experience improvement [6]. These findings suggest the algorithm favors recommendations that lead to better long term outcomes rather than those with more short-term rewards.

In a series of follow up experiments running for two weeks, we tease apart the importance of different components used in building the critic. For this set of experiment, we set the control arm to be the off-policy actor-critic method we just introduced. The experiment arms add or modify different components in the algorithm to understand the effect of the individual component. Figure 5a tested the effectiveness of adding a target network, which has been commonly employed in deep Q-learning [60] and actor-critic [13] to stabilize learning. As shown, the change does bring neutral positive effect to the system, leading to +0.02%[−0.02%, +0.06%] improvement on the topline metric. Its impact is however marginal. We hypothesize this is due to: 1) our offline training uses large batch size; In each gradient update, roughly 7000 state-action-reward-state-action tuples are used for gradient estimate, reducing noise in the gradient; 2) The γ discounting factor we used is relatively small, preventing the accumulation of future residual errors.

Figure 2 plots the importance of different inputs in building the critic network in offline experiments. Figure 5b shows the impact of



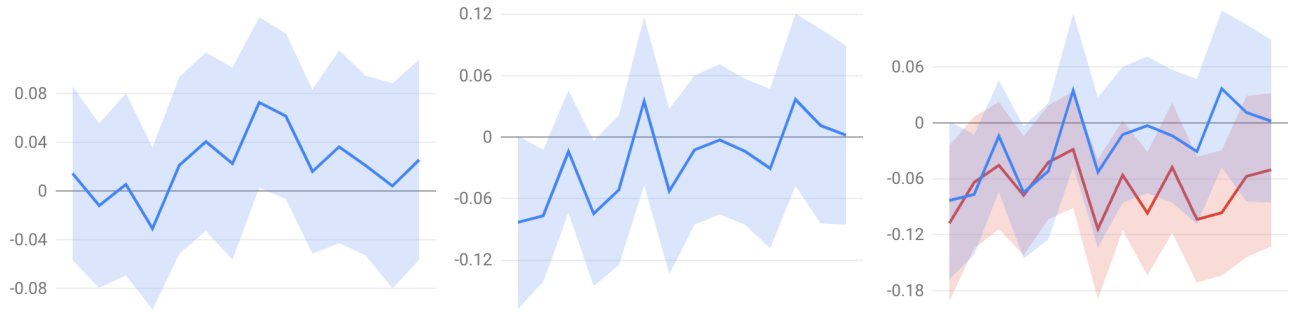
(a) Overall user enjoyment improvement

(b) Recommendation from tail(top) and head(bottom) corpus

(c) Average length of recommended items

(d) Average number of unique topics per user

Figure 4: Comparison of the off-policy actor-critic method vs the baseline REINFORCE with off-policy correction in a live system.

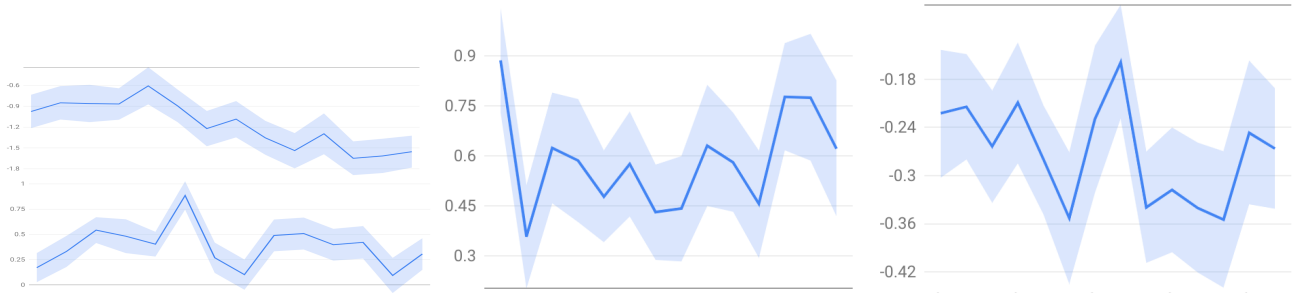


(a) Introducing target network

(b) Ablating user response input

(c) Different strategies for estimating $\hat{Q}_{\phi}^{\pi_\theta}$

Figure 5: Follow up experiments to compare different design choices in building the critic network.



(a) Tail(top) and head(bottom) corpus

(b) Avg. length of recommended items

(c) Avg. number of unique topics per user

Figure 6: Preference difference from the policy learned with a critic using sampling from learned policy w.r.t. the one learned with a critic using importance sampling with behavior policy on different item attributes.

ablating the immediate user response (group 4) in live experiments. Removing the immediate user response feature does significantly slow down the learning of the critic and as a result the convergence of the policy. It takes more than 2 weeks for a policy learned with a

critic ablating these features to catch up with the one learned with a critic using these features. The critic is eventually able to extract this information from TD learning as the immediate response is included in the target $\hat{Q}_{\phi}^{\pi_\theta}$. This ablation study suggests more comprehensive feature set improves learning of the critic network, resulting in more accurate value estimates and faster policy convergence.

In figure 5c, we compare the two approaches to estimate the target $\hat{Q}_{\phi}^{\pi_\theta}$: one uses importance sampling $\hat{Q}_{IS,\phi}^{\pi_\theta}$ with the next state-action pair observed on the trajectories collected from the behavior policy (blue); and the other one samples directly from the learned policy $\hat{Q}_{S,\phi}^{\pi_\theta}$ (red). As mentioned Section 4.2, one of the main advantage of using $\hat{Q}_{IS,\phi}^{\pi_\theta}$ is that we can include as input to the critic network user immediate response on the recommended item to improve accuracy of the critic estimate, which we have confirmed its importance in both offline and live experiment. To isolate the effect of features from the sampling strategies, we here compare the two strategies with the same set of features (both ablating the immediate user response feature). One can see that sampling from learned policy (red) led to negative $-0.07\%[-0.11\%, -0.03\%]$ impact on the topline metric, which is worse than the $-0.02\%[-0.07\%, 0.03\%]$ for the blue curve (ablating the user response feature but still using the $\hat{Q}_{IS,\phi}^{\pi_\theta}$ target).

We diagnose the cause of the performance gap here by comparing the two resulted policies in their preferences on the various item attributes similar to what we have shown in figure 4. A critic learned using sampling from learned policy led to a policy that is more conservative than the one learned using importance weighting from behavior policy. It recommends more popular contents (figure 6a), more longer contents (figure 6b), and less novel contents (figure 6c).

Softmax Policy Induces Pessimism. The result was somewhat surprising to us. One of the main concerns in offline RL is that the agent over-estimates OOD actions. Most of the recent RL literature has been focusing on how to control the over-estimation by introducing pessimism [23, 26, 31]. We as a result were expecting the sampling from learned policy approach will lead to a policy that is more exploratory as the more "optimistic" estimate of the sampled action can lead to a policy that deviates more from the behavior policy. We however observed the contrary as shown in figure 6. We believe the phenomenon is a result of the softmax policy parameterization $\pi_\theta(a_t|s_t) = \frac{\exp(f(s_t,a_t))}{\sum_{a' in \mathcal{A}} \exp(f(s_t,a'))}$. Ignoring the expected return and the importance weighting terms, the actor loss $\mathcal{J}_\pi(\theta)$ as in eq. (6) reduces to

$$\min_\theta \mathcal{J}'_\pi(\theta) = \\ -\mathbb{E}_{s_t\sim d_t^\beta(\cdot), a_t\sim\beta(\cdot|s_t)}[f(s_t,a_t)] + \mathbb{E}_{s_t\sim d_t^\beta(\cdot), a'\in\mathcal{A}}[f(s_t,a')]$$

Minimizing the loss will increase the value $f(s_t, a_t)$ for the action $a_t \sim \beta(\cdot|s_t)$ while reducing the value of any other actions not chosen by the behavior policy. The loss form, combined with sampled softmax, aligns exactly with the penalty term introduced in the recent conservative Q-learning work [31]. Sampling from the learned policy estimates the target $\hat{Q}_{S,\phi}^{\pi_\theta}(s_t, a_t) = r(s_t, a_t) + \gamma Q_{S,\phi}^{\pi_\theta}(s_{t+1}, a'_{t+1})$ as in eq. (11). The overly pessimistic estimate of $Q_{\phi}^{\pi_\theta}(s_{t+1}, a'_{t+1})$ on a'_{t+1} sampled from the learned policy π_θ and unseen from the behavior data causes the target $\hat{Q}_{S,\phi}^{\pi_\theta}(s_t, a_t)$, and the critic estimate $Q_{S,\phi}^{\pi_\theta}(s_t, a_t)$, to be dominated mostly by the immediate reward $r(a_t, a_t)$, leading to more myopic policy. While approaches such as entropy regularization [42] and maximum entropy RL [16], which we already included in our algorithms, encourages learning policies of higher entropy, finding the right balance remains a challenge.

7 CONCLUSION AND DISCUSSION

We here share the experience of scaling an off-policy actor-critic algorithm for industrial recommendation use case, where the agent has to serve billions of users and millions of contents. We aim at bringing light on the critical components needed to build the critic network by providing both offline analysis tools as well as detailed readings from live experiments. While we are delighted to confirm the success of the well-established offline RL method in real life recommendation applications, challenges in learning caused by distribution shift in offline RL for recommendations remain. Over-estimation errors (optimism) of OOD actions are prevalent in valued-based offline RL approaches. On the contrary, policy-based (especially those employing softmax policy parameterisation) under-estimates OOD actions. The pessimism produces policies anchoring too close to the behavior policy, and as a result preventing a real breakthrough. Finding the right balance is critical to realize the full potential of offline RL for recommendations.

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. *arXiv preprint arXiv:1705.10528* (2017).

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. 2019. Striving for Simplicity in Off-policy Deep Reinforcement Learning. *arXiv preprint arXiv:1907.04543* (2019).

[3] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 456–464.

[4] Minmin Chen, Bo Chang, Can Xu, and Ed H Chi. 2021. User response models to improve a reinforce recommender system. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 121–129.

[5] Minmin Chen, Ramki Gummadi, Chris Harris, and Dale Schuurmans. 2019. Surrogate Objectives for Batch Policy Optimization in One-step Decision Making. In *Advances in Neural Information Processing Systems*. 8825–8835.

[6] Minmin Chen, Yuyan Wang, Can Xu, Ya Le, Mohit Sharma, Lee Richardson, Su-Lin Wu, and Ed Chi. 2021. Values of User Exploration in Recommender Systems. In *Fifteenth ACM Conference on Recommender Systems*. 85–95.

[7] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. 2019. Generative adversarial user model for reinforcement learning based recommendation system. In *International Conference on Machine Learning*. PMLR, 1052–1061.

[8] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

[9] Nathaniel D Daw, John P O'doherty, Peter Dayan, Ben Seymour, and Raymond J Dolan. 2006. Cortical substrates for exploratory decisions in humans. *Nature* 441, 7095 (2006), 876.

[10] Thomas Degris, Martha White, and Richard S Sutton. 2012. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839* (2012).

[11] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679* (2015).

[12] Damien Ernst, Pierre Geurts, and Louis Wehenkel. 2005. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research* 6, Apr (2005).

[13] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.

[14] Scott Fujimoto, David Meger, and Doina Precup. 2018. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900* (2018).

[15] Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. 2018. Offline A/B testing for Recommender Systems. In *WSDM*. ACM, 198–206.

[16] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[17] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. 2019. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*. 2555–2565.

[18] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation* 9, 2 (2001), 159–195.

[19] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 368–377.

[20] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. 2019. SlateQ: A tractable decomposition for reinforcement learning with recommendation sets. (2019).

[21] Amir H Jadidinejad, Craig Macdonald, and Iadh Ounis. 2021. The Simpson's Paradox in the Offline Evaluation of Recommendation Systems. *ACM Transactions on Information Systems (TOIS)* 40, 1 (2021), 1–22.

[22] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* (2014).

[23] Olivier Jeunen and Bart Goethals. 2021. Pessimistic reward models for off-policy learning in recommendation. In *Fifteenth ACM Conference on Recommender Systems*. 63–74.

[24] Olivier Jeunen, David Rohde, Flavian Vasile, and Martin Bompaire. 2020. Joint policy-value learning for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1223–1233.

[25] Shivaram Kalyanakrishnan and Peter Stone. 2007. Batch reinforcement learning in a complex domain. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 94.

[26] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. 2020. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems* 33 (2020), 21810–21823.

[27] Haruka Kiyohara, Yuta Saito, Tatsuya Matsuhiro, Yusuke Narita, Nobuyuki Shimizu, and Yasuo Yamamoto. 2022. Doubly robust off-policy evaluation for ranking policies under the cascade behavior model. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 487–497.

[28] Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems* 12 (1999).

[29] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[30] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. 2019. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. *arXiv preprint arXiv:1906.00949* (2019).

[31] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 1179–1191.

[32] Sascha Lange, Thomas Gabel, and Martin Riedmiller. 2012. Batch reinforcement learning. In *Reinforcement learning*. Springer, 45–73.

[33] Sergey Levine and Vladlen Koltun. 2013. Guided policy search. In *International Conference on Machine Learning*. 1–9.

[34] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[35] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. 2018. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research* 37, 4-5 (2018), 421–436.

[36] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, and Yuzhou Zhang. 2018. Deep reinforcement learning based recommendation with explicit user-item interactions modeling. *arXiv preprint arXiv:1810.12027* (2018).

[37] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. 2018. Breaking the curse of horizon: Infinite-horizon off-policy estimation. *Advances in Neural Information Processing Systems* 31 (2018).

[38] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2011. Hashing with graphs. In *Icml*.

[39] Rishabh Mehrotra, Niannan Xue, and Mounia Lalmas. 2020. Bandit based optimization of multiple objectives on a music streaming platform. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 3224–3233.

[40] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. 2019. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *Advances in Neural Information Processing Systems* 32 (2019).

[41] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. 2019. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177* (2019).

[42] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548* (2017).

[43] Noveen Sachdeva, Yi Su, and Thorsten Joachims. 2020. Off-policy bandits with deficient support. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 965–975.

[44] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[45] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature* 588, 7839 (2020), 604–609.

[46] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*. 1889–1897.

[47] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[48] Wenjie Shang, Yang Yu, Qingyang Li, Zhiwei Qin, Yiping Meng, and Jieping Ye. 2019. Environment reconstruction with hidden confounders for reinforcement learning based recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 566–576.

[49] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, Sep (2005).

[50] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. 2020. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396* (2020).

[51] 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. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.

[52] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature* 550, 7676 (2017), 354–359.

[53] Dusan Stamenkovic, Alexandros Karatzoglou, Ioannis Arapakis, Xin Xin, and Kleomenis Katevas. 2022. Choosing the Best of Both Worlds: Diverse and Novel Recommendations through Multi-Objective Reinforcement Learning. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 957–965.

[54] Alex Strehl, John Langford, Lihong Li, and Sham M Kakade. 2010. Learning from logged implicit exploration data. In *Advances in Neural Information Processing Systems*. 2217–2225.

[55] Richard S Sutton, Andrew G Barto, et al. 1998. *Reinforcement learning: An introduction*. MIT press.

[56] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).

[57] Adith Swaminathan and Thorsten Joachims. 2015. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research* 16, 1 (2015), 1731–1755.

[58] Adith Swaminathan and Thorsten Joachims. 2015. The self-normalized estimator for counterfactual learning. In *Advances in Neural Information Processing Systems*. 3231–3239.

[59] Philip Thomas and Emma Brunskill. 2016. Data-efficient off-policy policy evaluation for reinforcement learning. In *ICML*. 2139–2148.

[60] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[61] Kristof Van Moffaert, Madalina M Drugan, and Ann Nowé. 2013. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 191–199.

[62] Kristof Van Moffaert and Ann Nowé. 2014. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research* 15, 1 (2014), 3483–3512.

[63] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.

[64] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[65] Yifan Wu, George Tucker, and Ofir Nachum. 2019. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361* (2019).

[66] Teng Xiao and Donglin Wang. 2021. A general offline reinforcement learning framework for interactive recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4512–4520.

[67] Teng Xiao and Suhang Wang. 2022. Towards off-policy learning for ranking policies with logged feedback. In *The Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI*.

[68] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2020. Self-supervised reinforcement learning for recommender systems. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 931–940.

[69] Haonan Yu, Haichao Zhang, and Wei Xu. 2022. Do You Need the Entropy Reward (in Practice)? *arXiv preprint arXiv:2201.12434* (2022).

[70] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. (2019), 38 pages.

[71] Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin. 2019. " Deep reinforcement learning for search, recommendation, and online advertising: a survey" by Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin with Martin Vesely as coordinator. *ACM SIGWEB Newsletter* Spring (2019), 1–15.

[72] Xiangyu Zhao, Long Xia, Dawei Yin, and Jiliang Tang. 2019. Model-based reinforcement learning for whole-chain recommendations. *arXiv preprint arXiv:1902.03987* (2019).

[73] Xiangyu Zhao, Liang Zhang, Long Xia, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2017. Deep reinforcement learning for list-wise recommendations. *arXiv preprint arXiv:1801.00209* (2017).

[74] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A deep reinforcement learning framework for news recommendation. (2018), 167–176.

[75] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement learning to optimize long-term user engagement in recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2810–2818.