Towards Off-Policy Learning for Ranking Policies with Logged Feedback

Teng Xiao, Suhang Wang

The Pennsylvania State University
{tengxiao, szw494}@psu.edu

Abstract

Probabilistic learning to rank (LTR) has been the dominating approach for optimizing the ranking metric, but cannot maximize long-term rewards. Reinforcement learning models have been proposed to maximize user long-term rewards by formulating the recommendation as a sequential decision-making problem, but could only achieve inferior accuracy compared to LTR counterparts, primarily due to the lack of online interactions and the characteristics of ranking. In this paper, we propose a new off-policy value ranking (VR) algorithm that can simultaneously maximize user long-term rewards and optimize the ranking metric offline for improved sample efficiency in a unified Expectation-Maximization (EM) framework. We theoretically and empirically show that the EM process guides the leaned policy to enjoy the benefit of integration of the future reward and ranking metric, and learn without any online interactions. Extensive offline and online experiments demonstrate the effectiveness of our methods.

1 Introduction

With the advances of deep learning, various deep learning based recommender systems (RS) have been proposed to capture the dynamics of user preferences. However, they are mainly based on the probability ranking principle (Robertson 1977) that the optimal ranking should rank items in terms of probability of relevance to a user. Typically, these methods are trained based on *Maximum Likelihood Estimation* (MLE) on the past logged feedbacks with supervised pointwise (Hu, Koren, and Volinsky 2008), pairwise (Rendle et al. 2012; Burges et al. 2005), or listwise (Cao et al. 2007) loss functions. Probability learning to rank (LTR) is one of the guiding technical principles behind the optimization of ranking models in information retrieval. Recently, reinforcement learning (RL) is gaining a lot of attraction in recommender system (Chen et al. 2019; Zhao et al. 2018b; Zou et al. 2020; Xiao and Wang 2021; Xin et al. 2020). Different from probability LTR which focuses on picking a model under the immediate feedback distribution, RL in general focuses on learning policy that takes actions in a dynamic environment so as to maximize the long-term reward.

Despite its attraction, optimizing the RL objective under the real recommendation scenario has the following challenges. **(1)** Missing online interactions. In contrast to probability ranking methods, training an optimal RL algorithm requires a large number of online interactions with the environments (real users) (Fujimoto, Meger, and Precup 2019), which is impractical as it could hurt user experiences when the RL agent is not well-trained, especially at the beginning of training. Without online interactions, RL algorithms such as the Q-learning (Haarnoja et al. 2017) and Actor-Critic (Haarnoja et al. 2018; Lillicrap et al. 2015) suffer from the overestimation issue (Fujimoto, Meger, and Precup 2019; Kumar et al. 2020), a phenomenon that unseen state-action pairs are erroneously estimated to have unrealistic values if there is no online interaction with real users, which is also verified by our preliminary analysis in § 2.2. **(2)** Lack of characteristics of ranking. Off-policy RL methods, such as one-step Q-learning (Watkins and Dayan 1992) and variants of deep Q networks (DQN) (Schaul et al. 2016), enjoy the advantage of learning from any samples from the same environment (i.e., off-policy learning). However, as shown in our analysis § 2.2, these methods can be seen as a regression problem that focuses on directly estimating optimal state-action value. However, in recommendation, we are more interested in the relative ranking of the actions (items) given user states and optimizing the ranking metric instead of the absolute values of actions, which is the main difference between recommendation and the robotics domain where only one optimal action is needed at each time step. **(3)** Partial and sparse feedback. In RS, click data, which is called implicit data, are easy to collect because they represent the behavior logs of the users. As the implicit data is not the explicit feedback of the users' preferences, one cannot know whether unclicked feedback is negative feedback or unlabeled positive feedback. Directly considering the reward of unclicked feedback as negative would lead to a sub-optimal ranking algorithm. As shown in (Chen et al. 2019), standard off-policy learning methods do not take into account unobserved feedbacks. Motivated by the discussions above, in this paper, we investigate the problem of learning a ranking algorithm that can simultaneously maximize the future reward and optimize the ranking metric based on logged feedbacks. It is a challenging problem as: (i) There is a huge optimization gap between MLE and RL since they are totally different learning paradigms for ranking as we discussed above; (ii) In practice, there is no access to the

reward for every state-action pair. In other words, we only observe *partial* and *sparse* reward on the logged feedbacks; and (iii) RL is prone to be overestimated (Levine et al. 2020), i.e., they will overestimate the values of the actions, resulting in bad performance. The overestimation is much more severe without online interactions (Kumar et al. 2020).

To address these challenges, we propose a novel learning framework to conduct ranking with rewards based on the Expectation-Maximization (EM) principle. Our framework can unify the MLE and RL, where both the RL teacher and the MLE student help each other in a closed-loop and gives extra power for knowledge distillation between reward and feedbacks. To solve the problem of partial and sparse rewards, we propose reward extrapolation and ranking regularization to improve our EM framework. We extend the EM framework to sequential settings and propose a novel algorithm named off-policy value ranking (VR), for maximizing long-term rewards. VR optimizes the ranking metric by learning the relative ranking of value function instead of estimating the absolute values. The main contributions are:
(1) We propose an EM framework that can learn from the sparse and partial reward signal through unifying the probability and reinforcement ranking principles.
(2) We extend our framework to sequential settings and propose an off-policy ranking algorithm that maximizes long-term rewards without online interactions and achieves better ranking performance both empirically and theoretically.
(3) Experiments on two datasets with three state-of-the-art backbones and the simulated online experiments on the RecSim environment show the effectiveness of our framework.

2 Preliminaries

2.1 Notations and Problem Definition

Assume we have a set of users $u \in \mathcal{U}$, a set of items $i \in \mathcal{I}$. Following previous work (Chen et al. 2019; Zhou et al. 2020), we can translate the task of recommendation into a markov decision process (MDP): $(\mathcal{S}, \mathcal{A}, P, R, \rho, \gamma)$ where
•$\mathcal{S}$: a space describing user states: $s_t = (i_1, i_2, \cdots, i_t)$, where i_t denotes the item interacted (clicked) by users;
•$\mathcal{A}$: a discrete action space which are recommending items;
•$P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the state transition probability.
•R: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the reward, where $r(s_t, a_t) = r_t$ is the received reward by performing action a_t at user state s_t;
•$\rho(s_1)$ is the initial distribution and γ is the discount factor;
Formally, we seek a policy $p_\theta(\cdot|s_t)$ which translates the user state $s \in \mathcal{S}$ into a distribution of the action space, so as to maximize expected cumulative rewards as follows:

$$\max_\theta \mathcal{J}(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)], R(\tau) = \sum\nolimits_{t=1}^{T} \gamma^t r\left(s_t, a_t\right), \quad (1)$$

where the expectation is taken over the trajectory $\tau = (s_1, a_1, \cdots s_T)$ obtained by acting policy step by step: $s_1 \sim \rho(s_1), a_t \sim p_\theta\left(a_t|s_t\right)$ and $s_{t+1} \sim p\left(s_{t+1}|s_t, a_t\right)$. Generally, the agent cannot interact with the user as we are only provided with a logged dataset, where the action a_t is chosen by a historical logging (behavior) policy $p_\mathrm{b} = p_\mathrm{b}(a_t|s_t)$. The past *implicit feedbacks* (only positive feedbacks) can be represented as $\mathcal{D} = \{\{s_t^i, a_t^i, r_t^i, s_{t+1}^i\}_t\}_{i=1}^N$, where N is the total number of data. For brevity, we omit superscript i in what follows. With the definition above, our studied problem is formally defined as: *Given past interactions $\mathcal{D}$, our goal is to learn the recommendation policy $p_\theta(a|s)$ which can maximize cumulative future reward, i.e., Eq. (1) for RS.*

2.2 Explorations on MLE- & RL-based RS

MLE: Typically, current sequential recommendation models (Tang and Wang 2018; Kang and McAuley 2018; Xiao, Liang, and Meng 2019) rely on the probability ranking principle and are trained to approximate the ranking metric based on MLE. Specifically, MLE maximizes the log-likelihood of logged feedbacks as follows:

$$\mathcal{L}_{\mathrm{MLE}}(\psi) = \mathbb{E}_{p_\mathrm{b}(a_t|s_t)}\left[\log p_\psi(a_t|s_t)\right]. \quad (2)$$

Note that the actions are clicked or purchased items since we only observe positive feedbacks. Although we do not know $p_\mathrm{b}\left(a_t|s_t\right)$, we can get its samples from logged feedbacks. For listwise models (Cao et al. 2007), $p_\psi(a_t|s_t)$ is parameterized neural networks (Kang and McAuley 2018) with the last softmax layer. Besides the listwise loss, pairwise loss such as Bayesian personalized ranking (Rendle et al. 2012) also have shown to be effective in ranking. Although MLE can naturally approximate the ranking metric (Xia et al. 2008), it cannot maximize the long-term reward of users.
RL: Various RL-based recommendation algorithms have been proposed (Zheng et al. 2018; Zou et al. 2020; Zhao et al. 2018a). Generally, these algorithms try to minimize the following temporal difference (TD) error as:

$$\mathcal{L}_{TD}(\phi) = \left(Q_\phi\left(s_t, a_t\right) - r_t - \gamma Q_{\bar{\phi}}\left(s_{t+1}, \pi\left(s_{t+1}\right)\right)\right)^2, \quad (3)$$

where $Q_\phi\left(s_t, a_t\right)$ is a learning state-action value function and $\pi\left(s_{t+1}\right) = \arg\max_a Q_{\bar{\phi}}\left(s_{t+1}, a\right)$. The $\bar{\phi}$ corresponds to the frozen weights in the target network, and is updated at fixed intervals (Lillicrap et al. 2015). While Q-learning is an effective off-policy algorithm in robotics, we argue that it is not suitable for the ranking problem in RS as: *Q-learning can essentially be viewed as a supervised regression model for predicting the true value; while the relative ranking of value is more important in recommendation*. Suppose the discount factor $\gamma = 0$, then the TD error in Eq. (3) becomes:

$$\hat{\mathcal{L}}_{TD}(\phi) = \mathbb{E}_{\left(s_t, a_t, r_t, s_{t+1}\right) \sim \mathcal{D}}\left(Q_\phi\left(s_t, a_t\right) - r_t\right)^2, \quad (4)$$

which is exactly the mean squared error (MSE) that estimates reward value. However, as shown by previous works (Liang et al. 2018), MSE is not a good proxy for the top-N ranking metric. Besides, when $\gamma \neq 0$, since the TD loss in Eq. (3) selects next actions via $\max_a Q_{\bar{\phi}}\left(s_{t+1}, a\right)$, it suffers from the overestimation (Levine et al. 2020): $\max_a Q_{\bar{\phi}}\left(s_{t+1}, a\right)$ of out-of-distribution actions is too big to be correct.
Empirical analysis. To verify our discussion above, we conduct preliminary analysis to understand the ranking, partial feedbacks and overestimation issues of Q-learning in RS. We compare the Q-learning (DQN) and supervised MLE (list-wise) with our proposed VR on YooChoose (see § 5.1 for settings). We also consider two variants of DQN, i.e., DQN-NS and DQN-ONE. DQN-NS uniformly samples unseen items to provide negative (-1) rewards. DQN-ONE, i.e., Eq. (4) is the one-step DQN which sets the discount factor γ in DQN to zero. We also compute the overestimation

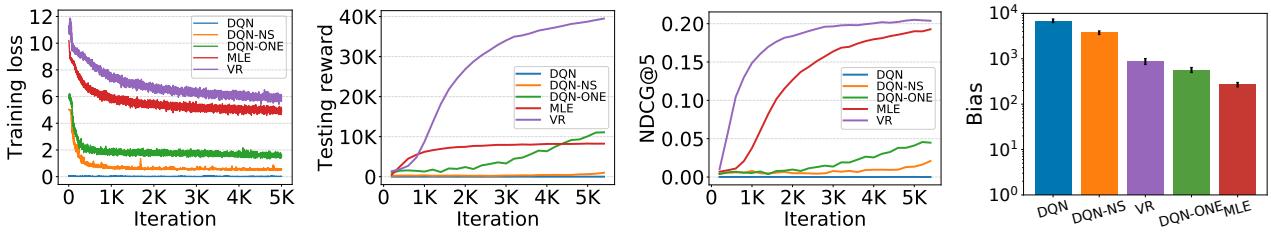

Figure 1: Training curves and overestimation bias of MLE, our VR, DQN, and two variants of DQN.

bias over the test set $\mathcal{D}_t$ as: $\frac{1}{|\mathcal{D}_t|}\sum_{(s,a)\in\mathcal{D}_t}(\max(Q_\phi(s,a)-V(s),0))^2$, where $V(s)$ denotes the discounted sum of rewards. From Fig. 1, we have some important findings. (1) We can find the ranking performance of DQN and DQN-NS is extremely bad, which empirically verifies our discussion in § 2.2 that the TD-based DQN is not suitable for learning ranking policy offline due to the large *overestimation bias* and lack of *ranking characteristics*. (2) DQN-ONE outperforms DQN and DQN-NS, which shows that DQN and DQN-NS can not benefit from future rewards at all due to the overestimation bias. (3) DQN-NS reduces this bias to some extent but still suffer it severely, which demonstrates that incorporating negative evidence is important when training the RL algorithm for the *partial feedbacks* in RS, and this is exactly one of our motivations for adding ranking regularization in our VR. Besides the Q-learning based RL, (Chen et al. 2019) applies off-policy policy gradient to learn from logged feedbacks for RS. Specifically, they utilize the importance sampling technique (Munos et al. 2016) to conduct off-policy estimation. While this method can learn from logged feedback as an off-policy way, it is still biased and has too large variance to be effective (Levine et al. 2020).

3 Learning to Rank with Rewards

Based on the analysis above, we summarize the limitations of current RS algorithms: (1) The probability ranking methods based on MLE cannot directly optimize the future reward nor conduct multi-objective optimization. (2) Q-learning has low variance and is off-policy learning, but it is inherently not suitable for ranking and suffers from the overestimation issue. To address these issues, in this section, we propose a novel off-policy value ranking algorithm.

3.1 The One-step EM framework

In this section, we first present the EM framework which can unify the MLE and RL models to learn from both the reward and ranking signal on the single-step RL environments (bandit setting). This bandit setting provides the reader with a simplified version of our main contribution in this paper, i.e., the off-policy value ranking algorithm.

To this end, we adopt the principle of probabilistic generative model and introduce a binary variable R which is related to the reward by $p\left(R_t=1|s_t,a_t\right)=\exp(\frac{r(s_t,a_t)}{\alpha})$ where α is the temperature parameter, to denote whether the action a_t taken at state s_t is optimal. A larger reward means that conducting action a_t at state s_t is more likely to be optimal. We use R_t to represent $R_t=1$ for brevity. Without loss of generality, we assume the rewards are nonpositive as the rewards can always be scaled and centered to be no greater than 0, which guarantees that $\exp(\frac{r(s_t,a_t)}{\alpha})\leq 1$. To connect MLE with RL optimization, we further consider the model $p_\theta(a_t|s_t)$ as a policy that takes action a_t given the state s_t. Given the definitions above, we consider the following probabilistic generative process of rewards and feedbacks:

$$a_t\sim p_\theta(a_t|s_t),\quad R_t\sim p\left(R_t|s_t,a_t\right). \tag{5}$$

For a given state s_t, the recommendation system selects an action a_t based on the optimizing policy and the optimal variable $\mathcal{R}_t$ is generated by the probability $p\left(R_t|s_t,a_t\right)$ based on the received reward $r(s_t,a_t)$. This generative model can be represented as a joint probability:

$$p(a_t,R_t|s_t)=p_\theta(a_t|s_t)p\left(R_t|s_t,a_t\right). \tag{6}$$

Note that we let the probability conditioned on s_t since $p(s_t)$ is the empirical state distribution which is known given logged feedbacks. To learn this generative model, we are interested in estimating θ and we can directly maximize the log-marginal likelihood $\log p_\theta(a_t|s_t)$ to learn the θ based on the observed feedbacks (s_t,a_t), resulting in the probability supervised ranking method with MLE. However, instead of conducting MLE, we want to learn the optimal policy directly from user rewards. Thus, we consider maximizing the log-marginal likelihood $\log p_\theta(R_t|s_t)$ of observed optimal variable R_t and treat the action a_t as the latent variable:

$$\begin{aligned}&\log p_\theta(R_t|s_t)=\log\int p_\theta\left(a_t|s_t,R_t\right)p_\theta(R_t|s_t)da_t\\&=\int p_\theta\left(a_t|s_t,R_t\right)\log\frac{p_\theta\left(a_t,R_t|s_t\right)}{p_\theta\left(a_t|s_t,R_t\right)}da_t\\&=\mathbb{E}_{p_\theta(a_t|s_t,R_t)}[\frac{r(a_t,s_t)}{\alpha}+\log p_\theta(a_t|s_t)-\log p_\theta\left(a_t|s_t,R_t\right)].\end{aligned} \tag{7}$$

Eq. (7) shows that maximizing the log-marginal likelihood $\log p_\theta(R_t|s_t)$ is equivalent to maximizing the reward at time t with the posterior policy $p_\theta\left(a_t|s_t,R_t\right)$, while minimizing the Kullback-Leibler (KL) divergence between $p_\theta\left(a_t|s_t,R_t\right)$ and $p_\theta\left(a_t|s_t\right)$. α serves as a trade-off parameter here. When $\alpha\to 0$, this objective is equivalent to only maximizing the reward at time t. The posterior $p_\theta\left(a_t|s_t,R_t\right)$ is closely related to the notion of the optimal policy and this can be intuitively understood as: "What is the probability of action given my current state if we want to be optimal?"

To answer this question, for each state, we need to learn the posterior distribution $p(a_t|s_t,R_t)$. We resort to the EM algorithm (Neal and Hinton 1998) that iterates two coordinate ascent optimization steps. We assume a variational distribution $q(a_t|s_t)$. Given this, we could derive the surrogate function to lower-bound the log-marginal likelihood

$\log p_\theta(R_t|s_t)$ of observed optimal variable R_t as follows:

$$\log p_\theta(R_t|s_t) = \log \int p_\theta(a_t, R_t|s_t) da_t =$$

$$\int q(a_t|s_t) \log \frac{p_\theta(a_t, R_t|s_t)}{q(a_t|s_t)} da_t + KL\left(q(a_t|s_t) || p_\theta(a_t|s_t, R_t)\right)$$

$$\geq \int q(a_t|s_t) \log \frac{p_\theta(a_t, R_t|s_t)}{q(a_t|s_t)} da_t \triangleq \mathcal{L}(q, \theta), \tag{8}$$

where the inequality holds since the KL divergence is always non-negative. $\mathcal{L}(q, \theta)$ is known as Evidence Lower BOund (ELBO) (Blei, Kucukelbir, and McAuliffe 2017; Xiao et al. 2021). Instead of directly maximizing the log-likelihood, the EM algorithm maximizes ELBO $\mathcal{L}(q, \theta)$ via an alternative procedure:

E-step. At n-th iteration, given the current θ_n, the E-step that maximizes $\mathcal{L}(q, \theta)$ w.r.t q has a closed-form solution:

$$q_{n+1}(a_t|s_t) = p_{\theta_n}(a_t|s_t, R_t) \propto p_{\theta_n}(a_t|s_t) \exp(\frac{r(s_t, a_t)}{\alpha}). \tag{9}$$

M-step. Given $q_{n+1}(a_t|s_t)$ at last E-step, we maximize $\mathcal{L}(q, \theta)$ w.r.t θ, resulting in the following objective:

$$\theta^{n+1} = \text{argmax}_\theta \, \mathbb{E}_{q_{n+1}(a_t|s_t)} \left[\log p_\theta(a_t|s_t)\right]. \tag{10}$$

Since there is no analytical solution for θ which is typically a neural network, we update θ via stochastic gradient descent with samples from $q_{n+1}(a_t|s_t)$ at the M-step in general.

In E-step, $q(a_t|s_t)$ is constructed by projecting $p(a_t|s_t)$ into a subspace constrained by the rewards with trade-off α in Eq. (9), and thus has desirable properties in learning rewards. In M-step, $q(a_t|s_t)$ serves as a teacher to transfer this reward knowledge into the student policy $p(a_t|s_t)$. This formulation can be understood as a type of supervised knowledge distillation (Hinton, Vinyals, and Dean 2015) and unsupervised posterior regularization (Ganchev et al. 2010).

However, *fundamentally different from supervised and unsupervised learning, it is non-trivial to conduct the EM algorithm in the recommendation setting.* (1) Partial and sparse rewards. There is no way to directly access the reward for every state-action pair. In other words, we only observe reward r_t from logged feedbacks $\mathcal{D}$, but not the others. This limits us to conduct the E-step since it requires the reward $r(a_t, s_t)$ for every state-action pairs. In addition, the reward is relatively sparse and lacks a negative signal. (2) Future rewards. Although this formulation can maximize the reward at the time t, it can not maximize long-term rewards. Before we discuss how to maximize long-term rewards, we first propose reward extrapolation and ranking regularization to solve the challenge of partial and sparse rewards.

3.2 Reward Extrapolation and Ranking Regularization

Extrapolation for partial rewards in the E-step. At each iteration n, we propose to extrapolate the reward from the logged feedbacks. Instead of extrapolating with worst-case rewards, i.e., setting the reward to zero for every unseen state-action pair, we can use a reward regressor to reduce the estimation bias. In particular, we extrapolate reward using the following regression estimator over mini-batches of tuples $(s_t, a_t, r_t, s_{t+1}) \in \mathcal{D}$ with the MSE loss:

$$\mathcal{L}_{\text{Q}}(\phi) = \left(Q_\phi(s_t, a_t) - r_t\right)^2, \tag{11}$$

where the reward estimator $Q_\phi(s_t, a_t)$ can be arbitrary models such as neural networks. Generally, it can extrapolate well since the error is relatively low due to the expressivity of the function approximator. However, if the logging policy is poor and differs from the optimizing policy $p_\theta(a_t|s_t)$, minimizing Eq.(11) would make the regression model fits well for items that are far from optimal, resulting in biased model. To alleviate this, we adopt importance sampling to shift the regression model to the optimizing policy:

$$\mathcal{L}_{\text{Q}}(\phi) = \frac{q_n(a_t|s_t)}{p_\psi(a_t|s_t)} \left(Q_\phi(s_t, a_t) - r_t\right)^2, \tag{12}$$

where $p_\psi(a_t|s_t)$ is the estimated logging policy from logged feedbacks via simple MLE estimation in Eq. (2). This objective corrects the selection bias induced by the logging policy so that it becomes in expectation equivalent to training the estimator with on-policy data from $q_n(a_t|s_t)$. Although the huge action space could lead to very small importance weights for some items, and consequently greater variance (Dudík, Langford, and Li 2011), in § 4, we theoretically show that our EM framework can naturally reduce the variance of this estimator. Given $Q_\phi(s_t, a_t)$ and Eq. (9), the posterior in the E-step can be derived as follows:

$$q_{n+1}(a_t|s_t) \propto p_{\theta_n}(a_t|s_t) \exp\left(\frac{Q_\phi(s_t, a_t)}{\alpha}\right). \tag{13}$$

Regularization for sparse rewards in the M-step. The M-step, i.e., Eq. (10), is for optimizing the policy parameter θ. Although the teacher $q_{n+1}(a_t|s_t)$ can transfer the reward knowledge into $p_\theta(a_t|s_t)$, the reward is relative sparse and lacks negative signal. To make the policy $p_\theta(a_t|s_t)$ be able to take into account negative evidence, we modify the M-step in Eq. (10) by adding a ranking regularization term as:

$$\mathcal{L}_{\text{P}}(\theta) = \underbrace{\beta \mathbb{E}_q[\log p_\theta(a_t|s_t)]}_{\text{Reward}} + (1-\beta) \underbrace{\mathbb{E}_{p_\psi}[\log p_\theta(a_t|s_t)]}_{\text{Ranking}}, \tag{14}$$

where $q \triangleq q_{n+1}(a_t|s_t)$ and β is the trade-off parameter calibrating the relative importance of the two objectives. $p_\psi \triangleq p_\psi(a_t|s_t)$ is the estimated ranking policy from logged feedbacks via simple MLE estimation (list-wise) and serves as regularization for RL policy $p_\theta(a_t|s_t)$. The key difference between reward regression q and ranking regularization p_ψ is that ranking regularization must produce a normalized distribution over actions which does constrain the probability of non-clicked recommendations (unobserved actions). In contrast, reward regression needs to estimate the future rewards of each state-action pair which can not be normalized. By alternately conducting the modified E-step in Eq. (12) and M-step in Eq. (14), α and β provide trading-off between the reward and signal of the ranking metric in the optimizing process of the teacher $q(a_t|s_t)$ and student $p_\theta(a_t|s_t)$.

3.3 The Sequential EM Framework

In the last section, we have introduced how to learn a ranking algorithm from both rewards and ranking signals. However, the bandit setting does not incorporate any information about the dynamics, so it learns to greedily assign high values to actions and can not maximize future rewards, without considering the state transitions that occur as a consequence of actions. In other words, in the full RL setting,

the state distribution $p(s_t)$ is not independent and identically distributed (i.i.d) now but depends on the action caused by the policy in the last step, i.e., the transition probability. We extended the one-step EM framework for the bandit setting in § 3.1 into the sequential setting. Similar to what we did in § 3.1, we introduce a binary variable R_t: $p\left(R_t=1|s_t,a_t\right)=\exp(\frac{r(s_t,a_t)}{\alpha})$ to denote whether the action a_t taken at state s_t is optimal. With the definition of MDP, we have the following generative model with the trajectory $\tau=\{s_1,a_1,\cdots,s_T\}$ and optimal variables $R_{1:T}$:

$$p_\theta(\tau,R_{1:T})=\rho(s_1)\prod_{t=1}^{T}p(s_{t+1}|s_t,a_t)p(R_t|s_t,a_t)p_\theta(a_t|s_t) \tag{15}$$

Compared to the bandit setting in Eq. (6), the distribution of state is non-i.i.d now but depends on the last step action. Given this generative model, we can also conduct the MLE estimation over the logged trajectory, resulting in Eq. (2). However, MLE can not learn from rewards. In addition, the MLE separates the logged feedbacks in the trajectory which makes it unable to maximize the long-term reward. Thus, like the bandit setting in Eq. (6), we are interested in maximizing the log-marginal likelihood $\log p_\theta(R_{1:T}|\tau)$ the inferring the posterior $p(\tau|R_{1:T})$. In particular, the posterior $p(\tau|R_{1:T})$ can be factorized as follows due to the conditional dependency underlaying the generative model in Eq. (15).

$$p(\tau|R_{1:T})=\prod_{t=1}^{T}p(s_t|s_{t-1},a_{t-1},R_{t:T})p(a_t|s_t,R_{t:T}), \tag{16}$$

where $p(s_1|s_0,a_0,R_{1:T}) = p(s_1|R_{1:T})$. Following the derivation of Eq. (7), one can easily verify that maximizing the log-marginal likelihood $\log p_\theta(R_{1:T}|\tau)$ is equivalent to maximizing our RL goal Eq. (1) with optimal policy $p(a_t|s_t,R_{t:T})$. Thus, the probability of action given optimality of the current until end of the episode and current state, i.e., the posterior of action $p(a_t|s_t,R_{t:T})$ is of high interest. Unfortunately, It is difficult to get the posterior $p(a_t|s_t,R_{t:T})$ since the distribution of state is non-i.i.d.

3.4 Off-Policy Value Ranking

In this section, we propose an EM-style algorithm to approximate the posterior, which results in our off-policy value ranking algorithm. Recall that our goal is to infer the posterior $p(a_t|s_t,R_{t:T})$. We consider inferring the posterior through the message passing algorithm (Heskes and Zoeter 2002). Thus, we first define the backward messages as:

$$m\left(s_t,a_t\right)\triangleq p\left(R_{t:T}|s_t,a_t\right) \qquad m\left(s_t\right)\triangleq p\left(R_{t:T}|s_t\right). \tag{17}$$

Then our interested posterior $p(a_t|s_t,R_{t:T})$ is:

$$p(a_t|s_t,R_{t:T})=\frac{p(s_t,a_t|R_{t:T})}{p(s_t|R_{t:T})}=\frac{p(R_{t:T}|s_t,a_t)p_\theta(a_t|s_t)p(s_t)}{p(R_{t:T}|s_t)p(s_t)}$$
$$=\frac{m(s_t,a_t)}{m(s_t)}p_\theta(a_t|s_t)\triangleq q(a_t|s_t). \tag{18}$$

With Eq. (15), reward extrapolation, and ranking regularization proposed in § 3.2, we can obtain the following E and M-steps (see appendix in SM for detailed derivations):

E-step. At the n-th iteration, we minimize the loss w.r.t ϕ:

$$\mathcal{L}_{\mathrm{Q}}(\phi)=\frac{q_n(a_t|s_t)}{p_\psi(a_t|s_t)}\big(Q_\phi\big(s_t,a_t\big)-r_t-\gamma\big(Q_{\bar{\phi}}\left(s_{t+1},a_{t+1}\right)$$
$$+\alpha\log p_{\theta_n}(a_{t+1}|s_{t+1})-\alpha\log q_n\left(a_{t+1}|s_{t+1}\right)\big)\big)^2, \tag{19}$$

where the $\bar{\phi}$ indicates parameters of the target network the same as vanilla Q-learning in Eq. (3) and a_{t+1} is sampled from the policy $q_n\left(a_{t+1}|s_{t+1}\right)$. We add the importance sampling as same as we did in the bandit setting in Eq. (12). We can find the reward extrapolation Eq. (12) in the batch setting becomes the Q value estimation for estimating future rewards. Given Eq. (18), we can get the followings non-parametric optimal policy $p(a_t|s_t,R_{t:T})$:

$$q_{n+1}(a_t|s_t)\propto p_{\theta_n}(a_t|s_t)\exp\left(\frac{Q_\phi\left(s_t,a_t\right)}{\alpha}\right). \tag{20}$$

Comparing with the E-step (Eq. (11)), the reward estimator $Q_\phi(s_t,a_t)$ becomes the value function approximating future rewards. When discount factor $\gamma=0$, the objective Eq. (19) is analogous to the Eq. (11) in the bandit setting.

M-step with ranking regularization. The M-step is analogous to Eq. (14) in the bandit setting, and we also consider the following loss with the ranking regularization term:

$$\mathcal{L}_{\mathrm{P}}(\theta)=\beta\mathbb{E}_q[\log p_\theta(a_t|s_t)]+(1-\beta)\mathbb{E}_{p_\psi}[\log p_\theta(a_t|s_t)], \tag{21}$$

where $q\triangleq q_{n+1}(a_t|s_t)$. By alternately conducting the E-step in Eq. (20) and M-step in Eq. (21), we get our full VR algorithm which can simultaneously maximize the future long-term rewards and learn from the ranking signal.

4 Theoretical Analysis

Overestimation Bias: In this section, we show that our VR can effectively reduce the overestimation bias compared with Q-learning. Recall that the overestimation bias occurs due to the max operator $\max_a Q_{\bar{\phi}}\left(s_{t+1},a\right)$ in the TD loss of Q-learning. Specifically, the overestimation bias at state s can be represented as: $\max_a Q_{\bar{\phi}}(s,a)-\max_a Q(s,a) = \max_a(Q_{\bar{\phi}}(s,a)-V(s)) = \max_a\left(\epsilon_a\right)$, where $V(s)$ and $Q(s,a)$ are the true state and state-action value functions, respectively. Since our TD loss in Eq. (19) selects the action from the policy $q(a|s)$ in the E-step, the overestimation bias can be proved to be no greater than that in Q-learning:

$$\sum_a q(a|s)Q_{\bar{\phi}}(s,a)-V(s)=\sum_a\frac{p_\theta(a'|s)\exp(\frac{Q_{\bar{\phi}}(s,a')}{\alpha})}{\sum_{a'}p_\theta(a'|s)\exp(\frac{Q_{\bar{\phi}}(s,a')}{\alpha})}Q_{\bar{\phi}}(s,a)$$
$$-V(s)=\sum_a\frac{p_\theta(a'|s)\exp(\frac{V(s)+\epsilon_{a'}}{\alpha})}{\sum_{a'}p_\theta(a'|s)\exp(\frac{V(s)+\epsilon_{a'}}{\alpha})}(V(s)+\epsilon_a)-V(s)$$
$$=\sum_a\frac{p_\theta(a'|s)\exp\left[\frac{\epsilon_{a'}}{\alpha}\right]}{\sum_{a'}p_\theta(a'|s)\exp\left[\frac{\epsilon_{a'}}{\alpha}\right]}\epsilon_a\le\max_a\left(\epsilon_a\right). \tag{22}$$

Estimation Variance: We theoretically show that our M-step can bound the variance induced by the importance sampling in the unbiased reward extrapolation. For notation brevity, we denote unbiased reward estimators (Eqs. (12) and (19)) as $w(a_t)\mathcal{L}_{\mathrm{Q}}\triangleq\mathcal{L}_{\mathrm{Q}}^w$ in which $w(a_t)=\frac{q_n(a_t|s_t)}{p_\psi(a_t|s_t)}\triangleq\frac{q}{p}$. we can conclude that the variance of estimation error can be bounded as follows (a proof is provided in the appendix):

$$\mathrm{Var}_{a_t\sim p}[\mathcal{L}_{\mathrm{Q}}^w]=\mathrm{E}_{a_t\sim p}[(\mathcal{L}_{\mathrm{Q}}^w)^2]-(\mathrm{E}_{a_t\sim p}[\mathcal{L}_{\mathrm{Q}}^w])^2\le \tag{23}$$
$$d_{\lambda+1}(q\|p)(\mathrm{E}_{a_t\sim p}\mathcal{L}_{\mathrm{Q}}^w)^{1-\frac{1}{\lambda}}(\sum_{a_t}\mathcal{L}_{\mathrm{Q}})^{1+\frac{1}{\lambda}}-(\mathrm{E}_{a_t\sim p}[\mathcal{L}_{\mathrm{Q}}^w])^2,\ \forall\lambda\ge0$$

where $d_\lambda(q\|p)=2^{D_\lambda(q\|p)}$ and $D_\lambda(q\|p)$ is the Rényi divergence (Rényi et al. 1961). Since $\lim_{\lambda\to0}D_{\lambda+1}(q\|p)=KL(q\|p)$ and our M-step can regularize the policy q towards the logging policy p_ψ, our VR can reduce the variance.

Table 1: Overall performance comparison. The percentage in brackets denote the relative performance improvement over MLE.

Backbones	Methods	YooChoose				RetailRocket			
		HR@5	NDCG@5	HR@20	NDCG@20	HR@5	NDCG@5	HR@20	NDCG@20
SASRec	MLE	0.2811	0.1857	0.4323	0.2389	0.2109	0.1512	0.3136	0.1918
	PG	0.2968(5.6%)	0.1901(2.4%)	0.4417(2.1%)	0.2511(5.1%)	0.2235(6.0%)	0.1638(8.1%)	0.3211(2.4%)	0.2102(9.6%)
	VR	**0.3187**(13.4%)	**0.2033**(9.5%)	**0.4618**(6.8%)	**0.2635**(10.3%)	**0.2326**(10.3%)	**0.1729**(14.4%)	**0.3429**(9.3%)	**0.2336**(21.8%)
GRU4Rec	MLE	0.2614	0.1599	0.4238	0.2117	0.1947	0.1431	0.2987	0.1856
	PG	0.2773(6.1%)	0.1726(7.9%)	0.4391(3.6%)	0.2236(5.6%)	0.2068(6.2%)	0.1543(7.8%)	0.3152(5.5%)	0.1991(7.3%)
	VR	**0.2905**(11.1%)	**0.1887**(18.0%)	**0.4501**(6.2%)	**0.2457**(16.1%)	**0.2188**(12.4%)	**0.1657**(15.8%)	**0.3372**(12.9%)	**0.2208**(19.0%)
Caser	MLE	0.2521	0.1585	0.4171	0.2016	0.1776	0.1255	0.2845	0.1739
	PG	0.2655(5.3%)	0.1677(5.8%)	0.4283(2.7%)	0.2132(5.8%)	0.1901(7.0%)	0.1387(10.5%)	0.2989(5.1%)	0.1822(4.8%)
	VR	**0.2809**(11.4%)	**0.1759**(11.0%)	**0.4426**(6.1%)	**0.2371**(17.6%)	**0.2075**(16.8%)	**0.1451**(15.6%)	**0.3205**(12.7%)	**0.2119**(21.9%)

5 Empirical Evaluation

In this section, we empirically evaluate the effectiveness of the proposed VR on both offline and online settings.

5.1 Offline Ranking Experiment

Datasets. We use two large-scale datasets. (1) *YooChoose*[1]: It contains sequences of user purchases and clicks. We remove the sessions whose length is smaller than 3. (2) *RetailRocket*[2]: This dataset also contains sequences of user purchases and clicks. We remove items which are interacted less than 3 times. We randomly sample 80% sequences as the training set, 10% as validation, and 10% as the test set.

Settings. Our VR is a generic learning algorithm and can be utilized to improve any recommendation backbones. To evaluate the effectiveness of VR, we consider three representative recommendation backbones, i.e., GRU4Rec (Hidasi et al. 2016), Caser (Tang and Wang 2018) and SASRec (Kang and McAuley 2018). Following (Chen et al. 2019), we reuse the user state representation generated from the backbone models, and model SL policy p_ψ, Q function Q_ϕ, and RL policy p_θ with different heads. We compare with two learning baselines: the supervised learning with MLE, and the off-policy policy gradient (PG) with the weight capping (Chen et al. 2019). Since we have compared with Q-learning in § 2.2, we omit it in this section due to space limitations. We use top-k Hit Ratio (HR@k) and Normalized Discounted Cumulative Gain (NDCG@k) to evaluate the ranking performance of the click feedback.

Results. Table 1 summarizes the ranking performance of different learning algorithms with three backbones. From Table 1, we have the following findings: (1) Our VR consistently and significantly outperforms the PG. This is because our framework has the advantage of simultaneously maximizing future rewards and leveraging the signal of ranking metric from logged feedbacks. This validates the effectiveness of VR and our motivation that feedback information is important for better ranking. (2) VR outperforms MLE in all settings, which shows that our VR as an off-policy RL algorithm can outperform supervised MLE via maximizing the future reward although there is no online interaction. (3) VR significantly outperforms the MLE and PG with all backbones on two datasets, which demonstrates the effectiveness of VR and also shows that our VR is robust to different backbones and datasets. (4) We observe that our VR achieves better performance on sizes 5 and 20 of ranking list k, which suggests that our VR is agnostic to the size of top-k list.

[1]https://recsys.acm.org/recsys15/challenge/.

[2]https://www.kaggle.com/retailrocket/ecommerce-dataset

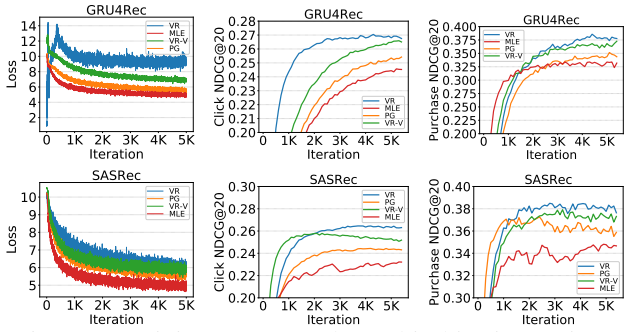

Figure 2: Training curves on the multi-objective setting.

Table 2: Training time (ms) each iteration.

Backbone	MLE	PG	VR-V	VR
GRU4Rec	28.2	31.4	37.9	38.8
Caser	9.8	12.3	15.6	16.5
SASRec	23.1	25.2	31.6	32.5

5.2 Multi-objective Optimization

One advantage of directly optimizing the reward via RL is that it allows designing different rewards for better optimizing the multi-objectives. In this section, we study how the proposed VR performs in the multi-objective setting.

Settings. We follow the same setting in RQ1. However, besides the click, we also consider another feedback, the purchase here. YooChoose contains 43,946 purchases of users. RetailRocket contains 57,269 purchase feedbacks. For VR and off-policy PG, we define the rewards of purchase and click as five and one, respectively. To better show the advantage of VR in the multi-objective setting, we add a variant of our VR called VR-V, which treats the purchase and click the same and assign the same rewards (one) to them.

Results. Fig. 2 shows the curves of the policy training loss and testing performance of purchases and clicks on YooChoose. The results on Caser are similar, so we remove it to Appendix to save space. According to Fig. 2, we find that: (1) Our VR and VR-V outperform MLE and off-policy PG in all settings, which shows the effectiveness of our method in the multi-objective setting. (2) The phenomenon that VR outperforms VR-V shows that we can achieve better performance by assigning different rewards to different feedbacks. (3) Our VR is training-stable and converges faster than MLE which needs more epochs to achieve stable performance. One possible reason could be that the ranking regularization can reduce the instability in the TD updates. (4) We find that although VR has a bigger training loss, it achieves better performance on the test metric which indicates our VR

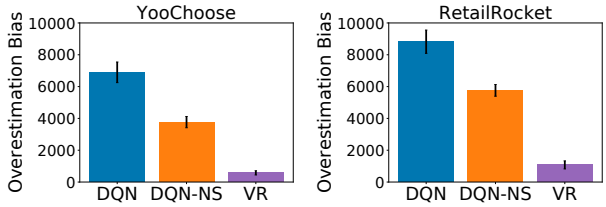

Figure 3: The overestimation bias on two datasets.

Table 3: Ablation study results (% NDCG@5) varying β.

Ablation	β=0.2	β=0.4	β=0.6	β=0.8
VR-NW	19.21±0.6	19.52±0.7	18.02±0.5	17.67±0.6
VR	**19.92**± 0.9	**20.43**±1.1	**19.56**±1.5	**18.92**±1.8

has a much stronger generalization ability. Table 2 shows the results of training time on YooChoose. With Table 2 and Fig. 2, we can find VR is efficient and can achieve superior performance without much additional computational cost.

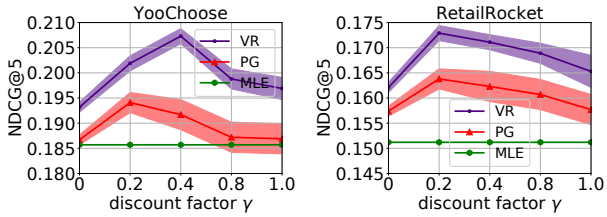

Figure 4: Performance with various discount factor γ.

5.3 Further Analysis

We take an examination on VR to understand how each component affects its performance. We follow the same setting in RQ1 and use SASRec as the backbone.

Future rewards. We investigate whether maximizing future rewards improves the ranking metric. In VR, the discount factor γ controls the contribution of future rewards. For example, if $\gamma = 0$, VR is the bandit setting proposed in 3.1 and does not consider future rewards. Fig. 4 shows the performance varying γ. We can find that (1) appropriate values of γ (maximizing future rewards) can boost the performance but large values hurt the performance, and (2) VR consistently performs better than baselines with every different γ.

Knowledge transfer. One of the most important properties of VR is transferring the knowledge between teacher and student. This transferring knowledge is controlled by two key parameters α and β. To understand their effects, we train VR with various α and β. From Fig. 5, we can find smaller β generally leads better performance. This is because neglecting the ranking regularization term will result in the overestimation issue and lack of ranking signal. This is also consistent with our motivation. The performance can be boosted by choosing appropriate values for α and β.

Overestimation bias and variance. As shown in theoretical analysis in § 4, the estimation variance of the weighted TD loss in Eq. (19) via the importance sampling (IS) can be reduced since our M-step Eq. (21) regularizes RL policy to the logged data via β. Table 3 shows the results of VR trained via TD loss and non-weighted TD loss. As shown in Table 3, although IS introduces some variance, VR can achieve a better bias-variance trade-off via adjusting β. As

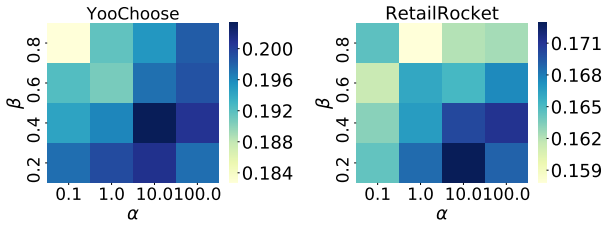

Figure 5: Performance (NDCG@5) with various α and β.

Table 4: Online evaluation results of CTR and coverage@3. Note that coverage just indicates higher diversity and we list behavior policy just for reference, not for comparison.

	Random		Maximum	
	CTR	Coverage@3	CTR	Coverage@3
Pop	39.5	30.0	41.8	30.0
MLE	63.5 ± 1.7	75.0 ± 1.8	72.1 ± 2.3	74.8 ± 2.1
DQN	50.7 ± 4.0	66.2 ± 2.3	54.3 ± 2.1	68.4 ± 3.7
PG	72.5 ± 4.6	76.0 ± 2.9	75.2 ± 3.8	76.5 ± 3.6
VR-NW	80.2 ± 1.2	78.5 ± 1.9	78.9 ± 1.7	79.7 ± 0.9
VR	**84.3 ± 2.3**	**82.2 ± 2.7**	**84.7 ± 3.3**	**85.6 ± 2.1**
Behavior	63.1	97.0	75.2	74.5

shown in § 2.2, our VR can effectively reduce the overestimation bias. Fig. 3 shows the overestimation bias on YooChoose and RetailRocket. From Figs. 1 and 3, we can find our VR achieves the best performance and can significantly reduce the overestimation bias compared with Q-learning, which empirically verifies our theoretic analysis in § 4.

5.4 Online Performance Comparison

In this section, we conduct the online evaluation. We adopt RecSim (Ie et al. 2019) which is a simulation environment for testing RL for RS. We consider the default settings of RecSim with the number of categories as 20. We also consider logged data obtained from two behavioral policies: random (exploration) policy and maximum reward policy which recommends the top k items with the highest ground-truth reward. From Table 4, we can find: (1) DQN can not beat the behavior policy even the supervised MLE methods, which is consistent with our observations in the offline test. This is because that DQN suffers from overestimation bias. (2) More importantly, our proposed VR significantly improves upon the behavior policy in CTR, which confirms that our methods can also perform well on online test in which there is an explicit distribution shift. (3) Our VR significantly outperforms VR-NW, which shows the effectiveness of the importance sampling in the online testing.

6 Conclusion

In this paper, we investigated offline learning of ranking policies for the sequential recommendation. We have presented a general EM framework, which can effectively learn from both rewards and ranking signals. To maximize long-term rewards, we extended the EM framework into sequential settings and proposed an off-policy VR algorithm. We provide theoretical guarantees of reducing overestimation bias and estimated variance. We show empirically that for the task of sequential recommendation, backbones learned by our VR outperform the same models that are optimized by other criteria such as MLE, off-policy PG, and DQN.

Acknowledgments

This research is supported by, or in part by, the National Science Foundation (NSF) under grant IIS-1909702 and IIS-1955851, and Army Research Office (ARO) under grant W911NF21-1-0198.

References

Blei, D. M.; Kucukelbir, A.; and McAuliffe, J. D. 2017. Variational inference: A review for statisticians. *JASA*.

Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; and Hullender, G. 2005. Learning to rank using gradient descent. In *ICML*, 89–96.

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; and Li, H. 2007. Learning to rank: from pairwise approach to listwise approach. In *ICML*.

Chen, M.; Beutel, A.; Covington, P.; Jain, S.; Belletti, F.; and Chi, E. H. 2019. Top-K Off-Policy Correction for a REINFORCE Recommender System. In *WSDM*, 456–464.

Dudík, M.; Langford, J.; and Li, L. 2011. Doubly Robust Policy Evaluation and Learning. In *ICML*.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-policy deep reinforcement learning without exploration. In *ICML*.

Ganchev, K.; Graça, J.; Gillenwater, J.; and Taskar, B. 2010. Posterior regularization for structured latent variable models. *JMLR*.

Haarnoja, T.; Tang, H.; Abbeel, P.; and Levine, S. 2017. Reinforcement learning with deep energy-based policies. In *ICML*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*.

Heskes, T.; and Zoeter, O. 2002. Expectation propagation for approximate inference in. In *UAI*.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. *stat*, 1050: 9.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*.

Ie, E.; Hsu, C.-w.; Mladenov, M.; Jain, V.; Narvekar, S.; Wang, J.; Wu, R.; and Boutilier, C. 2019. Recsim: A configurable simulation platform for recommender systems. *arXiv*.

Kang, W.-C.; and McAuley, J. 2018. Self-attentive sequential recommendation. In *ICDM*, 197–206. IEEE.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*.

Levine, S.; Kumar, A.; Tucker, G.; and Fu, J. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Liang, D.; Krishnan, R. G.; Hoffman, M. D.; and Jebara, T. 2018. Variational autoencoders for collaborative filtering. In *WWW*, 689–698.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv*.

Munos, R.; Stepleton, T.; Harutyunyan, A.; and Bellemare, M. G. 2016. Safe and efficient off-policy reinforcement learning. In *NIPS*, 1054–1062.

Neal, R. M.; and Hinton, G. E. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, 355–368. Springer.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.

Rényi, A.; et al. 1961. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*.

Robertson, S. E. 1977. The probability ranking principle in IR. *Journal of documentation*.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2016. Prioritized Experience Replay. In *ICLR*.

Tang, J.; and Wang, K. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, 565–573.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning*.

Xia, F.; Liu, T.-Y.; Wang, J.; Zhang, W.; and Li, H. 2008. Listwise approach to learning to rank: theory and algorithm. In *ICML*.

Xiao, T.; Chen, Z.; Wang, D.; and Wang, S. 2021. Learning How to Propagate Messages in Graph Neural Networks. In *KDD*, 1894–1903.

Xiao, T.; Liang, S.; and Meng, Z. 2019. Hierarchical neural variational model for personalized sequential recommendation. In *WWW*, 3377–3383.

Xiao, T.; and Wang, D. 2021. A General Offline Reinforcement Learning Framework for Interactive Recommendation. In *AAAI*, 4512–4520.

Xin, X.; Karatzoglou, A.; Arapakis, I.; and Jose, J. M. 2020. Self-supervised reinforcement learning for recommender systems. In *SIGIR*, 931–940.

Zhao, X.; Xia, L.; Zhang, L.; Ding, Z.; Yin, D.; and Tang, J. 2018a. Deep reinforcement learning for page-wise recommendations. In *RecSys*, 95–103.

Zhao, X.; Zhang, L.; Ding, Z.; Xia, L.; Tang, J.; and Yin, D. 2018b. Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning. In *KDD*, 1040–1048.

Zheng, G.; Zhang, F.; Zheng, Z.; Xiang, Y.; Yuan, N. J.; Xie, X.; and Li, Z. 2018. DRN: A Deep Reinforcement Learning Framework for News Recommendation. In *WWW*, 167–176.

Zhou, S.; Dai, X.; Chen, H.; Zhang, W.; Ren, K.; Tang, R.; He, X.; and Yu, Y. 2020. Interactive Recommender System via Knowledge Graph-enhanced Reinforcement Learning. In *SIGIR*, 179–188.

Zou, L.; Xia, L.; Du, P.; Zhang, Z.; Bai, T.; Liu, W.; Nie, J.; and Yin, D. 2020. Pseudo Dyna-Q: A Reinforcement Learning Framework for Interactive Recommendation. In *WSDM*, 816–824.