Neurocomputing 417 (2020) 255–269

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Top-aware reinforcement learning based recommendation

Feng Liu [a], Ruiming Tang [b], Huifeng Guo [b], Xutao Li [a], Yunming Ye [a,*], Xiuqiang He [b]

[a] Shenzhen Key Laboratory of Internet Information Collaboration, Harbin Institute of Technology, Shenzhen 518055, China
[b] Noah's Ark Lab, Huawei, China

ARTICLE INFO

Article history:
Received 5 November 2019
Revised 24 June 2020
Accepted 16 July 2020
Available online 6 August 2020
Communicated by Weike Pan

Keywords:
Recommendation
Top-aware
Reinforcement learning

ABSTRACT

Reinforcement learning (RL) techniques have recently been introduced to recommender systems. Most existing research works focus on designing policy and learning algorithms of the recommender agent but seldom care about the *top-aware issue*, i.e., the performance on the top positions is not satisfying, which is crucial for real applications. To address the drawback, we propose a Supervised deep Reinforcement learning Recommendation framework named as SRR. Within this framework, we utilize a supervised learning (SL) model to partially guide the learning of recommendation policy, where the supervision signal and RL signal are jointly employed and updated in a complementary fashion. We empirically find that suitable weak supervision helps to balance the immediate reward and the long-term reward, which nicely addresses the top-aware issue in RL based recommendation. Moreover, we perform a further investigation on how different supervision signals impact on recommendation policy. Extensive experiments are carried out on two real-world datasets under both the offline and simulated online evaluation settings, and the results demonstrate that the proposed methods indeed resolve the top-aware issue without much performance sacrifice in the long-run, compared with the state-of-the-art methods.

1. Introduction

Modeling user dynamics is an important issue in designing interactive recommender systems (IRS). Recently, Reinforcement Learning (RL) techniques have been introduced to IRS to capture the dynamic patterns of user behavior during the interaction with recommender systems and perform planning to optimize long-term performance [1–7].

However, existing works focus more on long-term rewards and may suffer from *top-aware issue*, i.e., the performance on the top positions is not satisfying in RL based recommendations. For example, in [1–5], the authors only evaluate their models by long-term reward [3], or by Precision@k and NDCG@k with a very large k value (e.g. $k = 40$ in [1], $k = 32$ in [2]). None of them evaluates the performance of top positions, which is important in real-world recommender systems. Fig. 1 shows a typical observation in line with such founding, on MovieLens 1M and Yahoo! Music datasets. The settings of all the algorithms follow [1,4]. We find that the matrix factorization based method, SVD++, achieves much better Precision@1 performance than the RL based methods. In other words, RL based methods tend to deliver unsatisfactory top position performance, because they are prone to recommending the items with larger long-term value.

To understand the impact of the top-aware issue, we analyze a mainstream App Store. We find that top positions are very crucial as they attract the majority of users and bring the majority of app downloads. As shown on the left side of Fig. 2, only about 2.5% of users will continue searching on the second page and about 94% of the population stop searching on the first page. From the right side of Fig. 2, we can see that the first position brings about 16% of total downloads and nearly 80% of total downloads comes from the top 15 positions, which is displayed in the first page. Moreover, similar phenomenons also exist, such as in streaming recommendation services, like Pandora Radio, NetEase Cloud Music, TikTok video, and in the E-commerce, search engine, etc. The 'risky' recommendations made by the RL recommender agent in top positions may severely affect the user experiences. Such examples suggest that the recommendations for the first a few rounds are of remarkable significance and need to be carefully designed.

In this paper, we aim to solve the top-aware issue that exists in RL based recommendation approaches. To achieve this goal, we propose a **S**upervised deep **R**einforcement learning **R**ecommendation framework named as SRR. Within this framework, we utilize a

* Corresponding author.
E-mail addresses: fengliu@stu.hit.edu.cn (F. Liu), tangruiming@huawei.com (R. Tang), huifeng.guo@huawei.com (H. Guo), lixutao@hit.edu.cn (X. Li), yeyunming@hit.edu.cn (Y. Ye), hexiuqiang1@huawei.com (X. He).

https://doi.org/10.1016/j.neucom.2020.07.057
0925-2312/

supervised learning (SL) model to partially guide the learning of recommendation policy to generate recommendations, where the supervision signal and RL signal are jointly employed and updated in a complementary fashion. The SL model plays two important roles. On one hand, it provides the policy component with a supervision signal to learn a combined policy, which enables the agent to focus more on immediate reward and promote the performance on top positions. On the other hand, the SL model is updated with the RL model, which avoids the long-term reward to sacrifice too much. Moreover, to investigate how different supervision signals impact the recommendation policy, we develop two models. One is the classification based model, SRR-L and the other is the ranking based model, SRR-R. The main contributions of this paper can be summarized as follows:

- We find that top-aware issue generally exists in RL based recommendation methods. To address this problem, we propose a supervised deep reinforcement learning recommendation framework named as SRR.
- Extensive experiments are carried out on two real-world datasets under both the offline and simulated online evaluation settings, and the results demonstrate that the proposed methods indeed resolve the top-aware issue by achieving better accuracy in the top positions without much performance sacrifice in the long-run when compared with state-of-the-art methods.

2. Related work

Our work mostly relates to the recommendation techniques, based on the characteristics of whether they can perform dynamic adaptation and long-term planning, we divide them into two branches, i.e., conventional recommendation techniques and RL based recommendation techniques.

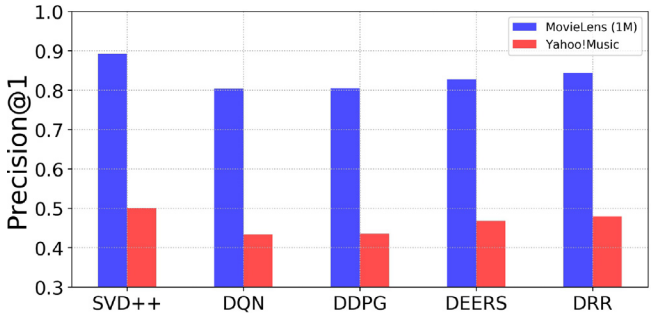

Fig. 1. Precision@1 performance for SVD++ [8] and several RL based recommendation models (DQN [6], DDPG [9], DEERS [1], and DRR [4]) on two public datasets.

2.1. Conventional recommendation techniques

Conventional recommendation methods develop from the static content-based filtering [10], matrix factorization based methods [11–13,8], logistic regression [14], factorization machines and its variants [14–16], and until recently deep learning models [17–20], to multi-armed bandit methods [21–25].

At the beginning of this century, content-based filtering [10] is proposed to recommend items by considering the content similarity between items. Later, collaborative filtering (CF) is put forward and extensively studied. The rationale behind CF is that users with similar behaviors tend to prefer the same items, and the items consumed by similar users tend to have the same rating. However, the conventional CF-based methods suffer from the data scarcity because the similarity calculated from sparse data might be unreliable. So the Matrix factorization (MF) is proposed as an advanced CF technique, plays an important role in recommender systems. MF models [11–13,8] characterize both items and users by vectors in the same space, which are inferred from the observed user-item interactions. Moreover, regarding the recommendation as a binary classification problem, logistic regression and its variants [14] are also utilized in recommender systems. Nevertheless, logistic regression based models are hard to generalize to the feature interactions that never or rarely appear in the training data. Therefore, factorization machines [15] and its variants [16] are proposed to model pairwise feature interactions as an inner product of latent vectors between features and show promising results. As an extension to FM, Field-aware FM (FFM [16]) enables each feature to have multiple latent vectors to interact with different fields.

Recently, deep learning (DL) models [19,17,18,20] improve the performance of recommender systems by using deep neural networks to model the complicated feature interactions, which enhance the model capability greatly. The authors in [17] propose a product based deep neural network to capture high-order feature interactions. Cheng et al. [19] propose an interesting hybrid network structure (Wide & Deep) that combines a linear (Wide) model and a deep model for recommendation. Moreover, the authors in [18] impose a factorization machine as a 'wide' module in Wide & Deep [19] with no need for feature engineering. Zhou et al. [20] propose a novel model, Deep Interest Network (DIN), where a local activation unit is designed to adaptively learn the representation of user interests from historical behaviors with respect to a certain ad.

As a distinguished direction, contextual multi-armed bandits (MAB) are also utilized to model the interactive nature of recommender systems [21–25]. Li et al. apply Thompson Sampling (TS) and Upper Confident Bound (UCB) to balance the trade-off between exploration and exploitation in [23,21], respectively. The authors in [22] further learn hidden features for each arm to model the

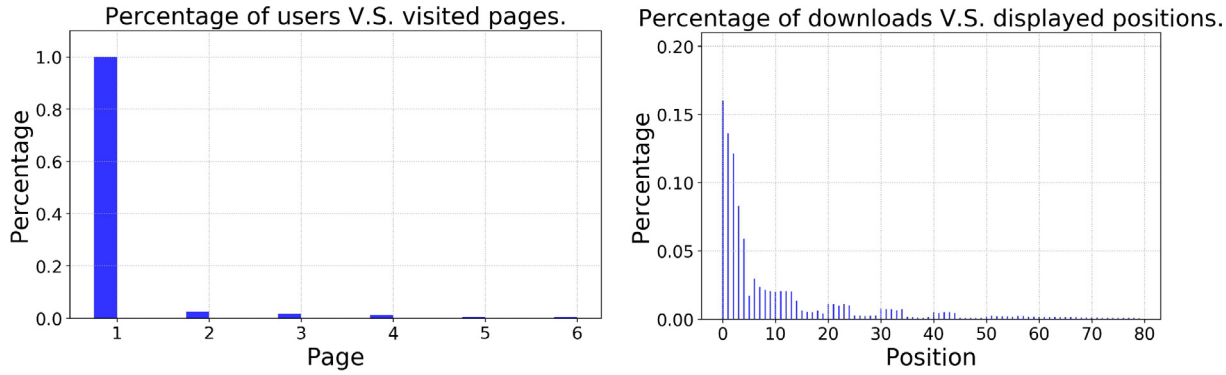

Fig. 2. Analysis on percentage of visiting users per page and percentage of app downloads per position in a mainstream App Store.

potential reward based on [21]. Moreover, Zeng et al. [25] propose a context drift model to address the time-varying problem. To integrate the latent vectors of items and users with some exploration, the authors of [24] combine matrix factorization with multi-armed bandits.

In another line of studies, researchers work on the sequential recommendation (SR) algorithms [26–33]. The authors in [26] considers both user-item similarities and first-order item-item transitions for sequential recommendation. Moreover, in [27], the authors utilize Gated Recurrent Units (GRU) to model the sequential dynamics for session-based recommendation, and use session-parallel mini-batches technique to train the model. What's more, an improved version is proposed in [28], where a novel ranking loss function and an efficient sampling strategy are proposed. In addition to the RNN-based methods, Convolutional Neural Network (CNN) is also adopted for sequential recommendation [30,31]. In [30], the researchers embeds the recent engaged items into an "image" in the latent space, then employ different convolutional kernels to extract sequential patterns. However, such RNN- and CNN-based methods always encodes the user interactions into hidden states or latent factors without considering the different impacts of the items consumed at different time steps on current decision. Therefore, the attention based models, which exhibit promising performance in sequence learning, are also utilized in sequential recommendation [32]. In addition, the authors in [33] propose to utilize gated network for sequential recommendation, where a feature gating layer and an instance gating layer are employed to select what item features can be passed to the downstream layers from the feature and instance levels, respectively.

However, the above mentioned methods either consider the recommendation procedure as a static process, i.e., they assume the user underlying preference keeps unchanged (CF, DL, MAB), or do not explicitly model the long-term rewards that the recommendations can make (SR).

2.2. RL based recommendation techniques

Recently, the research of RL based recommendation techniques has become a hot topic, and several approaches have been proposed [34–38,1,39,3,2,6,40,5]. Such RL methods can be divide into two categories: *model-based methods* [34–38] and *model-free methods* [39,3,2,6,40,5].

For the model-based methods, they always utilize a model of the environment to predict rewards for unseen state-action pairs. The MDP-Based CF model in [34] utilizes a finite sliding window of past history to define the current state to approximating the partial observable MDP, and they incorporate three strategies (value function approximation, policy optimization, stochastic sampling) to reduce the computational complexity. Chen et al. [38] propose a model-based reinforcement learning framework by developing a generative adversarial network to imitate user behavior dynamics and learn her reward function, where a combinatorial recommendation policy can be learned by cascading DQNs. Zhao et al. [36] propose a multi-agent reinforcement learning based approach for recommendations with multiple scenarios, which can capture the sequential correlation among different scenarios and jointly optimize multiple recommendation strategies. Moreover, the authors in [37] adopt generative adversarial network to model user-agent interactions for offline recommender policy learning.

For the model-free methods, we can divided them into branches: *policy-based methods* [1,39,3,2] and *value-based methods* [1,6]. Specifically, policy-based approaches [3,9,7,4,2,41] aim to generate a policy, of which the input is a state, and the output is an action. Firstly, one type of those works applies deterministic policies [3,7,4] based on the deterministic policy gradient algorithms [42,43], which generates an action directly. Dulac-Arnold et al. [3] resolve the large action space problem by modeling the state in a continuous item embedding space and selecting the items via a neighborhood method. However, as the underlying algorithm is essentially a continuous-action algorithm, its performance may be cursed by the gap between the continuous and discrete action spaces. Zhao et al. [7] propose to employ a Deep Deterministic Policy Gradient framework (DDPG) [43] with a page-display approach for page-wise recommendation. And the authors in [4] comprehensively study the state representation issue in RL based recommendation framework by explicitly modeling the user-item interactions, as the state representation is crucial for reinforcement learning [44]. Secondly, another type of studies employ the stochastic policies [2,41] according to the policy gradient theorem [45], which output the distribution of the actions, and the action is sampled from such distribution. The authors in [2] propose to utilize a balanced hierarchical clustering tree to tackle the large discrete action space issue with REINFORCE algorithm [45], where picking an item is formulated as seeking a path from the root to a certain leaf of the tree. Chen et al. [41] present an off-policy correction framework to address the data bias issue in a top-K recommender system at YouTube, where such data bias is caused by only observing historical feedbacks on previous recommendations. For value-based approaches [1,6], the action with maximum Q-value over all the possible actions is selected as the best action. Zhao et al. [1] take both user's positive feedback and negative feedback into consideration when modeling user state. Dueling Q-network is utilized in [6], to model Q-value of a state-action pair. Moreover, a minor update with exploration by dueling bandit gradient descent is proposed. However, such value-based approaches need to evaluate the Q-values of all the actions under a specific state, which is very inefficient when the action space is large.

Although the RL-based methods attract considerable attention in the recommendation community, the top-aware issue hinders their usability in real-world applications.

3. Methodology

In this paper, we formulate the recommendation task as a Markov Decision Process (MDP), which can be solved by reinforcement learning. Fig. 3 illustrates the user-recommender interactions in MDP formulation. At each time step, the recommender (agent) takes an action $a \in \mathcal{A}$ (recommend an item in the candidate set) according to the user (environment) state $s \in \mathcal{S}$ (user current preference over the items in the candidate set), and receives a reward $R(s,a)$ (user feedback of the recommended item). Then, the user state is updated to s' with transition probability $p(s'|s,a)$. Formally, the elements of this MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ are defined as follows:

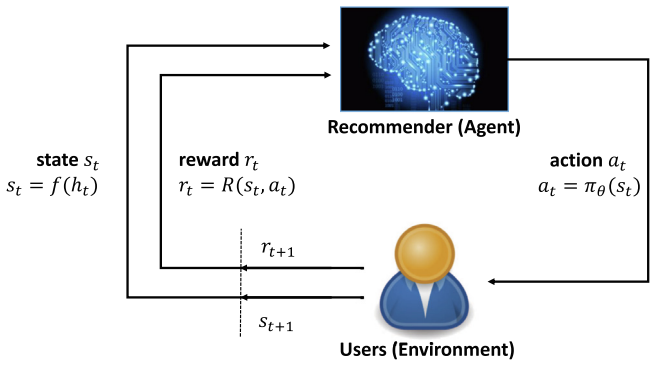

Fig. 3. An illustration of user-recommender interaction in MDP.

- $\mathcal{S}$: $\mathcal{S}$ denotes the user state space, which is the representation of the user's positive interaction history with the recommender, as well as her demographic information.
- $\mathcal{A}$: $\mathcal{A}$ is the action space, containing the items available for recommendation. In this paper, we assume that the agent only recommends one item to the user each time.
- $\mathcal{P}$: $\mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition probability, which is deterministic in our model once the user feedbacks are collected. Since the state is modeled by the user's positive feedback, once the user's feedback is collected, the state transition $p(s'|s,a)$ is determined.
- $\mathcal{R}$: $\mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is the reward. Given the recommendation based on the action a and the user state s, the user will provide her feedback, i.e., click, not click, or rating, etc. The recommender receives immediate reward $R(s,a)$ according to the user's feedback.
- $\gamma \in [0,1]$ is the discount factor measuring the present value of long-term rewards.

The target of the recommender agent is to find an optimal policy π_θ ($\mathcal{S} \times \mathcal{A} \mapsto [0,1]$), which maximizes the expected cumulative rewards for the recommender system.

3.1. Overview of SRR framework

As shown in Fig. 4, the SRR framework consists of four components, namely *embedding component* (EC), *state representation component* (SRC), *policy component* (PC) and *supervised learning component* (SLC).

EC: As a bottom layer, EC maps the items together with the user demographic information from a high dimensional sparse vector to a low dimensional dense one $\mathbf{e_t} = [\mathbf{p_u}, \mathbf{h_t}]$ by concatenating $\mathbf{p_u}$ and $\mathbf{h_t}$, where $\mathbf{p_u}$ is the latent representation of the user demographics and $\mathbf{h_t} = \{\mathbf{q_1}, \ldots, \mathbf{q_n}\}$ denotes the embeddings of n latest positive interacted items at time t. The interaction history $\mathbf{h}$ works in a sliding window manner. When the recommender agent recommends an item $\mathbf{q_t}$, we have $\mathbf{h_{t+1}} = \{\mathbf{q_2}, \ldots, \mathbf{q_n}, \mathbf{q_t}\}$ if the user provides positive feedback, otherwise $\mathbf{h_{t+1}} = \mathbf{h_t}$.

SRC: In the middle of SRR framework, a carefully designed SRC, which can be a fully-connected neural network [6], a recurrent neural network [1] or a carefully designed neural network [4], is to model the user states. For simplicity, we term SRC as a function $f(\cdot)$, such that the state is represented as $\mathbf{s_t} = f(\mathbf{e_t})$.

PC: PC is appended to SRC, whose input is state $\mathbf{s_t}$ and output is Q-value for value-based models or policy for policy-based models. Upon the output, value-based models decide the action by choosing the one with highest Q-value; on the other hand, policy-based models stochastically or deterministically decide the action from the output policy. Then, the reward $R(s,a)$ is obtained based on the user feedback. We take a popular value-based model, i.e., DQN [46] and a policy-based model, i.e., DDPG, as examples to illustrate the learning procedure of PC. For the value-based DQN, the parameters are updated according to the temporal-difference (TD) learning approach [45], i.e., minimizing the mean squared error as $L_{RL} = \frac{1}{N}\sum_i (y_i - Q_\theta(\mathbf{s_i}, a_i))^2$, where $y_i = r_i + \gamma Q_{\theta'}(\mathbf{s_{i+1}}, \pi_{\theta'}(\mathbf{s_{i+1}}))$ and N is the batch size. The target network technique [46] is always adopted, where θ' is the set of parameters of the target Deep Q-network. For the policy-based DDPG, PC stands for the Actor part (the update of parameters in critic part is the same as the one in above DQN, which is updated by the TD learning process). PC outputs a continuous ranking vector [4]. The items are recommended according to the ranking scores by the inner product operation between the generated ranking vector and the item embeddings. And the RL signal is computed according to the sampled Policy Gradient [43]: $\nabla_\theta J(\pi_\theta) \approx \frac{1}{N}\sum_t \nabla_a Q_\omega(s,a)|_{s=s_t, a=\pi_\theta(s_t)} \nabla_\theta \pi_\theta(s)|_{s=s_t}$, where θ denotes the parameters of PC, ω indicates the parameters of Critic network.

SLC: SLC acts as an indicator to show the difference between the current recommendation policy and users' historical preference, which can be a classification model or ranking model. To stabilize the training procedure, before feeding into SLC, the output of PC is re-scale into the range $[0,1]$ by min–max normalization (i.e., SL Ops in Fig. 3), which represents the probabilities that the user prefers the corresponding item. When updating the model, the gradient of SLC (i.e., SL signal) is back-propagated to PC. That is to say, the RL signal and SL signal are jointly employed and updated in a complementary fashion. Next, we focus on SLC and explain how it addresses the top-aware issue.

For easy to follow, the major symbols are listed in Table 1, following common symbolic notation, upper case bold letters denote matrices, lower case bold letters denote column vectors without any specification, and non-bold letters represent scalars.

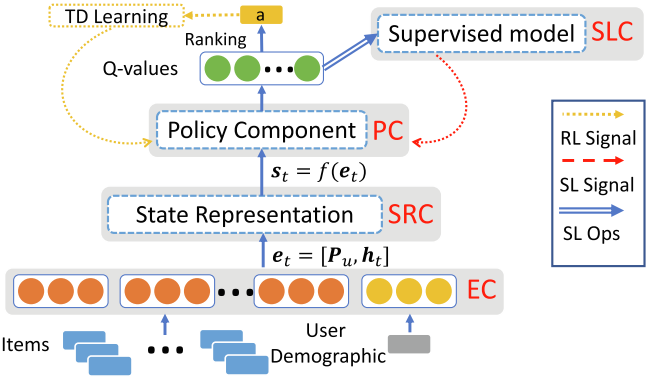

Fig. 4. SRR Framework for value-based RL models. For simplicity, we omit the SRR Framework for policy-based models.

3.2. How SRR resolves the top-aware issue

In the early stage of the interaction, the agent tries to perform comprehensive explorations to grasp the users' preference for better long-term performance, i.e., the recommendation policy is aggressive. Such explorations probably generate 'risky' items, which lead to poor performance in the top positions. To resolve such an issue, we propose to utilize an SL model (i.e., SLC in the previous section) to supervise the RL model, where the recommender agent learns combined policy π_θ to generate moderate items in the direction of partially matching SL signal. Consequently, such a learning scheme delivers better performance in the top positions. Employing an SL signal to amend the objective function of the reinforcement learning task provides the agent a guide to performing a moderate exploration (i.e., not too far away from the users' historical preference).

By considering the SL signals and RL signals jointly, we update the recommendation policy by maximizing the following objective function:

$$J(\theta) = \alpha J_{RL}(\theta) - (1-\alpha) J_{SL}(\theta), \tag{1}$$

where $J_{RL}(\theta)$ is the RL objective which maximizes the expected cumulative rewards, i.e., satisfy the user demands in the long-run. And $J_{SL}(\theta)$ is the SL objective, which ensures the recommendation quality in the first several rounds. Here, $\alpha \in (0,1)$ is a trade-off factor to balance the RL and SL. Hence, the goal we optimize $J(\theta)$ is to resolve the top-aware issue without much performance sacrifice in the long-run. Next, we elaborate on the RL signals and SL signals, respectively.

Table 1
Notations.

Notation	Description
$\mathcal{S}$	state space
$\mathcal{A}$	action space
$\mathcal{U}, \mathcal{V}$	user and item set
u, v	user u and item v
$\mathbf{p}_u, \mathbf{q}_v$	embeddings of user u and item v
h	a set of latest positive interaction history
d	dimension of embedding vectors
s	current user state s
a	current action a
n	length of h
N	batch size
θ	parameters in SRR

For the RL signals, we consider both the policy-based and value-based models. For policy-based methods, we take the on-policy Policy Gradient (PG) [45] as an example, PG is always utilized to handle the high dimensional and continuous actions. The objective function is:

$$J_{RL}(\theta) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[Q^\pi(s,a)] = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi_\theta(s,a) Q^\pi(s,a) da ds, \quad (2)$$

where ρ^π is the state distribution, π is the recommendation policy, s is the user current state, a is the recommended item based on the recommendation policy π and the user state s. Maximize $J_{RL}(\theta)$ can be treated as we learn an optimal policy π which maximizes the expectation of the return ($Q^\pi(s,a)$). And the gradient of PG can be calculated by the *policy gradient theorem* [45], where we term $\nabla^p_\theta J_{RL}$ as the gradient of PG:

$$\begin{aligned}\nabla^p_\theta J_{RL}(\theta) &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(s,a) Q^\pi(s,a) da ds \\ &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi_\theta(s,a) \frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} Q^\pi(s,a) da ds \\ &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi_\theta(s,a) \nabla_\theta \log \pi_\theta(s,a) Q^\pi(s,a) da ds \\ &= \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^\pi(s,a)]. \end{aligned} \quad (3)$$

From Eq. (3), we can conclude that if the return $Q^\pi(s,a)$ is positive (i.e., the long-term user feedback is positive), which indicates that the decision of recommending item a according to the user state s is positive, hence we update θ to the direction of increasing the probability $\pi_\theta(s,a)$; If $Q^\pi(s,a)$ is negative, we update θ to the direction of decreasing the probability $\pi_\theta(s,a)$.

For value-based methods, we take the off-policy Deep Q-network (DQN) as an example. DQN employs deep neural networks to estimate the Q function $Q_\theta(s,a)$,[1] and the Q value reflects the quality of the recommended item a under user state s by the recommender agent. The temporal-difference (TD) learning approach [45] is utilized to learn the Q function, i.e., minimize the mean square error:

$$J_{RL}(\theta) = \mathbb{E}_{s,a}[(y - Q_\theta(s,a))^2] y = \mathbb{E}_{r,s',a'}[r + \gamma \max_{a'} Q_{\theta'}(s',a')], \quad (4)$$

where the target network technique [46] together with the replay buffer technique [47] that to ensure independent and identical distribution of samples for training is adopted, and θ' is the parameter of the target Q network. And we term $\nabla^v_\theta J_{RL}$ as the gradient of the parameter of the Q network:

$$\begin{aligned}\nabla^v_\theta J_{RL}(\theta) &= \mathbb{E}_{s,a}[\nabla_\theta (y - Q_\theta(s,a))^2] \\ &= \mathbb{E}_{s,a,r,s',a'}[(r + \gamma \max_{a'} Q_{\theta'}(s',a') - Q_\theta(s,a)) \nabla_\theta Q(s,a)]. \end{aligned} \quad (5)$$

It is often computational efficient to sample a mini-batch of samples to optimize θ, instead of computing the full expectation of the above gradient.

For the supervision signals, we consider two manners. One is the classification based model and the other is the ranking based model. For classification based SL signal, we employ the cross entropy loss:

$$J^L_{SL}(\theta) = \sum_{k=1}^{K} -[y_k \log(score_k) + (1 - y_k) \log(1 - score_k)], \quad (6)$$

where y_k is a binary variable indicating whether the k-th item is consumed, K denotes the number of total items, $score_k$ is the Q value of k-th item. We term this method as **SRR-L**.

For the ranking based SL signal, we adopt pairwise ranking based loss [48]:

$$J^R_{SL}(\theta) = \sum_{v_j, v_k \in I} [\hat{P}_{j,k} \log(P_{j,k}) + (1 - P_{j,k}) \log(1 - \hat{P}_{j,k})], \quad (7)$$

where $P_{j,k} = \frac{1}{1+e^{-\sigma(score_j - score_k)}}$ is the predicted probability distribution that the user prefers item v_j than v_k, $\hat{P}_{j,k} = \frac{1}{2}(1 + S_{j,k})$ is the real probability distribution. $S_{j,k} \in \{0, \pm 1\}$, if the user prefers item v_j than v_k, the value of $S_{j,k}$ is 1, otherwise -1. If the user has the same preference on item v_j and v_k, $S_{j,k}$ is 0. We name this method as **SRR-R**.

With the above RL objective and SL objective functions, the parameter θ of the learned policy is updated as:

$$\theta = \theta + \eta[\alpha \nabla J_{RL}(\theta) - (1 - \alpha) \nabla J_{SL}(\theta)], \quad (8)$$

where η is the learning rate, and $\nabla J_{RL}(\theta)$ and $\nabla J_{SL}(\theta)$ are calculated according to the RL and SL objective, respectively.

In sum, SLC plays two important roles. On one hand, it provides PC with a supervision signal to learn a combined policy, which enables the agent to focus more on immediate reward and promote the performance of top positions. On the other hand, SLC is updated with the RL model, which ensures the long-term performance not to sacrifice too much.

Comparison to the work in [5]. In this work, we focus on how to fix the top-aware issue in RL-based recommendation methods. However, in [5], the authors find that EC cannot be nicely trained with the other two components simultaneously. Previous studies bypass the obstacle through a pre-training and fixing strategy, which is unable to model evolving preference of users and item correlations in the dynamic environment. Hence, they focus on addressing the training compatibility between the three components in RL based recommendations, where the RL-based models can be nicely trained in an end-to-end fashion. Therefore, the research problems of the two studies are definitely different, although they are in the same research background, i.e., RL-based recommendation.

4. Training and evaluation procedure

4.1. Training procedure

In this work, the training algorithm of the proposed SRR framework is detailed in Algorithm 1 in terms of value-based DQN. Specifically, in time step t, the training procedure mainly includes two phases, i.e., transition generation (lines 6–11) and model updating (lines 12–16). For the first stage, the recommender observes the current state $\mathbf{s}_t$ that is calculated by the SRC, then generates Q values on all the candidate items through the PC.

[1] The Q function is always parameterized with ω, to be consistent with Eq. (1), in this paper we utilize θ to parameterize the Q function.

Table 2
Statistic information of the datasets.

	ML (100k)	BC	ML (1M)	Jester
# user	943	1,152	6,040	63,978
# item	1,682	5,547	3,952	150
# ratings	100,000	50,239	1,000,209	1,761,439

And the action a_t (recommended item v_t) can be decided by the highest Q value with ε-greedy exploration. Subsequently, the reward r_t can be calculated based on the feedback of the user to the recommended item v_t from the offline log,[2] and the user state is updated (lines 9–10). Finally, the recommender agent stores the transition $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$ into the replay buffer D (line 11).

In the second stage, the recommender samples a minibatch of N transitions with widely used *prioritized experience replay* [47] technique (line 12). Then, the recommender updates the parameters θ according to Eq. (8) (lines 13–15). Note that we also employ the widely-used *target networks* [46] with *soft replace* technique to smooth the learning and avoid the divergence of parameters.

4.2. Reward shaping

The reward $R(\mathbf{s}, a)$ is considered as an evaluation of the quality of the recommended item. Specifically, we define the reward function as

$$R(\mathbf{s}, a) = R_0(\mathbf{s}, a) + \alpha\phi(\mathbf{s}, a), \tag{9}$$

where $R_0(\mathbf{s}, a)$ is the original reward function, which we leverage a supervised learning model Probabilistic Matrix Factorization (PMF) [13] as supervision to the feedbacks (ratings) of the recommended items that the users never rate before. And all the original rewards are empirically normalized into the range $[-1, 1]$. The $\phi(\mathbf{s}, a)$ is the potential reward function that can be treated as a local objective, such as prior knowledge to optimize the Q values, i.e., to enlarge the distribution of the Q values for better evaluating the actions. More precisely, in this paper, we utilize the **promotion** of Normalized Discounted Cumulative Gain (NDCG) [49] (a ranking based evaluation metric) caused by the ranking action a as the potential ranking reward, i.e., in each time step, when we add recommended item v into the recommended list $\mathcal{L}$, we utilize the change of NDCG of $\mathcal{L}$ as $\phi(\mathbf{s}, a)$ [50]. Since the training algorithm learns the model parameters under the supervision of the rewards, defining the rewards based on a ranking based evaluation measure can guide the training process to achieve a better ranking performance. What's more, α is a hyper-parameter to balance the two reward functions.

4.3. Offline evaluation

In the offline evaluation, we follow the evaluation method in [1,7,4,51]. For a given session, the agent only recommends the items that appear in this session rather than the ones in the whole item space. The reason is that we only have the ground truth feedback for the existing items in the session in the recorded offline log. The offline evaluation procedure can be treated as a **rerank** process on the existing items in the current session by iteratively selecting an item w.r.t. the action generated by the PC in the SRR framework. As we recommend only one item per time step, after the end of the whole recommendation procedure, we can calculate the evaluation metrics based on the recommended list $\mathcal{L}$ and the ground truth order of items in this session. Note that the model parameters are not updated in the offline evaluation. We summarize the offline evaluation in Algorithm 1, specifically, for current test session, the recommender first observes the initial state $\mathbf{s}_0$ and the item list $\mathcal{Y}$. Then in the time step t, the recommender observes the current state $\mathbf{s}_t$ that is calculated by the SRC, then generates Q values on all the candidate items ($\mathcal{Y}$) through the PC. And the action a_t (recommended item v_t is add to the recommended list $\mathcal{L}$) can be decided by the highest Q value (line 4–5). Next, the immediate reward r_t can be calculated based on the feedback of the user to the recommended item v_t from the offline log, then the state can be updated (line 6–7). Moreover, we remove the item v_t from the candidate item list $\mathcal{Y}$. After T time steps, we can get the final recommended item list $\mathcal{L}$, then we can calculate the evaluation result (line 9).

4.4. Simulated online evaluation

As it is risky and costly to directly deploy the RL based models on recommender systems [2,52], we build an interactive recommendation simulator to mimic the online environment, i.e., to simulate the reward $R(\mathbf{s}, a)$ according to the feedback of the corresponding recommendation. The online evaluation procedure follows Algorithm 1, i.e., the parameters continuously update during the online evaluation stage. Its major difference from Algorithm 1 is that the feedback of a recommended item is observed by the environment simulator. Moreover, before each recommendation session starting in the simulated online evaluation, we **reset** the parameters back to θ and ω which is the policy learned in the training stage for a fair comparison.

5. Experiment

In this section, we conduct experiments on four datasets to validate the effectiveness of the proposed SRR, and we aim to answer the following research questions:

- **RQ 1** Can SRR address the top-aware issue for RL based recommendation models?
- **RQ 2** How hyper-parameter settings affect SRR?
- **RQ 3** Can SRR generate 'safe' recommendations to promote the performance of top positions?

5.1. Experimental setup

5.1.1. Dataset and evaluation metrics

We adopt four publicly available datasets to conduct experiments, which are **MovieLens (100 k)**,[3] **BookCrossing**,[4] **MovieLens (1 M)** and **Jester (2)**.[5] The MovieLens and BookCrossing are abbreviated as ML and BC, respectively. And the statistics of the datasets are specified in Table 2.

[2] The reward for items that the user never rates before is 0 during the training procedure.

[3] https://grouplens.org/datasets/movielens/.

[4] http://www2.informatik.uni-freiburg.de/ cziegler/BX/.

[5] http://eigentaste.berkeley.edu/dataset/.

Algorithm 1: Training Algorithm of SRR Framework

Initialize the parameters θ in SRR and parameters in target network θ'
Initialize replay buffer D, soft-replace parameter τ
for *session = 1, M* **do**
  Observe the initial state s_0 according to the offline log
  for *t = 1, T* **do**
    Observe state $\mathbf{s}_t = f(h_t)$, where $h_t = \{\mathbf{q}_1, ..., \mathbf{q}_n\}$
    Calculate Q values $Q_\theta(\mathbf{s}_t, a_t)$ for all the candidate items
    Obtain action a_t according to the highest Q value with ε-greedy, recommend item v_t
    Calculate reward $r_t = R(\mathbf{s}_t, a_t)$ based on the feedback of the user
    Observe new state $\mathbf{s}_{t+1} = f(h_{t+1})$, where $h_{t+1} = \{\mathbf{q}_2, ..., \mathbf{q}_n, \mathbf{q}_t\}$ if r_t is positive, otherwise, unchanged
    Store transition $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$ in D
    Sample a minibatch transitions $(\mathbf{s}_i, a_i, r_i, \mathbf{s}_{i+1})$ in D with prioritized experience replay sampling technique
    Calculate the RL signal in Eq. (5)
    Calculate the SL signal in Eq. (6) or Eq. (7)
    update θ in Eq. (8)
    Update the target networks:
    $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
return θ

Algorithm 2: Offline Evaluation Algorithm of SRR Framework

input : state window size n, reward function R and recommended list $\mathcal{L} = \emptyset$, evaluation metrics $g(\cdot)$
output: Evaluation results $\mathcal{O}$
Observe the initial state $\mathbf{s}_0$ and item list $\mathcal{Y}$ according to the current test session
for *t = 1, T* **do**
  Observe current state $\mathbf{s}_t = f(h_t)$, where $h_t = \{\mathbf{q}_1, ..., \mathbf{q}_n\}$
  Execute action a_t according to the highest Q value $Q_\theta(\mathbf{s}_t, a_t)$, and recommend item v_t
  Add v_t in $\mathcal{L}$
  Get reward $r_t = R(\mathbf{s}_t, a_t)$ from the feedback located in the users' log by Eq. (9)
  Update to a new state $\mathbf{s}_{t+1} = f(h_{t+1})$, where $h_{t+1} = \{\mathbf{q}_2, ..., \mathbf{q}_n, \mathbf{q}_t\}$ if r_t is positive, otherwise, $h_{t+1} = h_t$
  remove v_t from $\mathcal{Y}$
Calculate evaluation results: $\mathcal{O} = g(\mathcal{L}, \mathcal{Y})$
return $\mathcal{O}$

- **MovieLens (100 k)**. A benchmark dataset comprises of 0.1 million ratings from users to the recommended movies on MovieLens website.
- **BookCrossing**. This dataset is Collected by Cai-Nicolas Ziegler in a 4-week crawl (August/ September 2004) from the Book-Crossing community, which contains 278,858 users (anonymized but with demographic information) providing 1,149,780 ratings (explicit/implicit) about 271,379 books. We normalize the ratings to discrete values from 0 to 5. We filter the dataset following the MovieLens (100k) treatment, i.e., remove all users and items who had less than 20 and 10 interactions, receptively.
- **MovieLens (1 M)**. A benchmark dataset includes of 1 million ratings from the MovieLens website.
- **Jester (2)**.[6] This dataset contains over 1.7 million real-value ratings (−10.0 to +10.0) over jokes in an online joke recommender system.

In addition, both offline and simulated online evaluations are conducted following [4,1,5]. For the offline evaluation, we employ the widely-used ranking based **Precision@k**, **NDCG@k** as the evaluation metrics [7]:

$$\begin{aligned} Precision@k &= \frac{\sum_{i\in(1,k)} rel(\mathcal{L}_i)}{k}, DCG@k \\ &= \sum_{i\in(1,k)} \frac{2^{rel(\mathcal{L}_i)}-1}{\log_2(i+1)}, IDCG@k \\ &= \sum_{i\in(1,k)} \frac{2^{rel(\hat{\mathcal{L}}_i)}-1}{\log_2(i+1)}, NDCG@k = \frac{DCG@k}{IDCG@k}, \end{aligned} \tag{10}$$

where $rel(\mathcal{L}_i)$ returns 1 if the rating of $\mathcal{L}_i$ is larger than the threshold, and 0 otherwise. And $\hat{\mathcal{L}}$ is the ideal ranking list of $\mathcal{L}$. Moreover, for the simulated online evaluation, the average reward is leveraged.

5.1.2. Compared methods

For the offline evaluation, we employ conventional representative methods including **Popularity** [53], **PMF** [13] and **SVD++** [8]. Moreover, four state-of-the-art RL based methods are utilized: **DQN** [6] and **DEERS** [1] are value-based models, while **DDPG** [9], **DRR** [4], and **EDRR** [5] are policy-based models. For online evaluation, in addition to the four RL based methods, we also consider two bandit algorithms, **LinUCB** [21] and **HLinUCB** [22].

- **Popularity** recommends the most popular item, i.e., the item with the highest average rating, and we also remove a few top most popular items following [53].
- **PMF** makes a matrix decomposition as SVD, while it only takes into account the non zero elements.
- **SVD++** mixes the strengths of the latent model as well as the neighborhood model.
- **LinUCB** selects a contextual arm (item) according to the estimated upper confidence bound of the potential reward.
- **HLinUCB** further learns hidden features for each arm to model the potential reward.
- **DQN** utilizes a Deep Q-network to generate Q-values to evaluate all the possible actions for the current state.
- **DDPG** learns a ranking vector to pick the highest ranking score for recommendation.
- **DEERS** represents the state with both the positive and negative feedbacks by Recurrent Neural Network (RNN) under Deep Q-Network (DQN) framework.
- **DRR** explicitly models the user-item interactions to represent the state to better study recommendation policy under the Actor-Critic framework.
- **EDRR** incorporates a supervised learning component to learn more powerful embeddings for RL-based recommendation in an end-to-end manner.

To verify the effectiveness of the proposed **SRR** framework (both **SRR-L** and **SRR-R**) on both value-based and policy-based RL models, we adapt the above five RL-based models into **SRR** framework, which are **DQN-L(R)**, **DEERS-L(R)**, **DDPG-L(R)**, **DRR-L(R)**, and **EDRR-L(R)**. For value-based models **DQN-L(R)**, **DEERS-L(R)**, and **EDRR-L(R)**, they choose the action with the largest Q-value under the given state, as described in Section 2.1. For policy-based methods **DDPG-L(R)** and **DEERS-L(R)**, PC outputs a ranking vector [4]. The items are recommended according to the ranking score by the inner product operation between the generated ranking vector and the item embeddings. As already stated before, the ranking scores are re-scaled into the range $[0, 1]$ when feeding into SLC.

5.1.3. Settings

Following [4,1,5], For each dataset, we choose 80% of the interactions in each user session as the training set and leave the rest as the test set. Moreover, for ML (100k), BC and ML (1M), the positive ratings are 4 and 5, while for Jester, the positive ones are those higher than 0, the ratings are utilized for generating rewards. For RL-based methods, the number of latest positively rated items n is empirically set to 5. We perform PMF to train the 100-dimensional embeddings of the users and items. Moreover, in each episode, we do not recommend repeated items. The discount rate γ is 0.9, and the batch size is 64. For all the RL methods, the Adam optimizer is adopted with L_2-norm regularization to prevent overfitting. For the other methods, the sizes of hidden vector in **PMF** and **SVD++** both are set to 20, the upper confidence bound factor α (not the α in Eq. (1)) and L2 regularization in LinUCB are set to 0.3 and 0.1, respectively. For HLinUCB, the upper bound factor α_u and α_a are set to 0.3 and 0.1, and the L2 regularization factors λ_1 and λ_2 are both set to 0.1.

5.2. Performance comparison and analysis (RQ1)

5.2.1. Offline evaluation results and analysis

The offline evaluation results are summarized in Tables 3–6. We have the following observations:

(1) Compared with conventional methods such as SVD++, the RL models DQN, DDPG, DEERS, DRR, and EDRR deliver poor performance at top-1 and top-3 positions. This is in accordance with our motivation that RL methods suffer from the top-aware issue.

(2) The performance of the four RL baseline models is improved by applying **SRR-L** and **SRR-R**. More specifically, better accuracy at top-1 and top-3 is yield and comparable performance at top-20 and the long-term rewards are achieved. Such observations suggest the proposed SRR framework addresses the top-aware issue without much long-term performance sacrifice, on both value-based and policy-based RL models. This is attributed to the introduced supervision signals, which enable RL models to better balance the top-aware accuracy and long-term rewards.

(3) Comparing **SRR-L** and **SRR-R**, we find that the RL models with **SRR-R** outperform that with **SRR-L**. Because the pairwise ranking based supervision signal can learn the pairwise preference over the items, and it is more consistent with the action in SRR, which is indeed chosen from the top-ranked item.

[6] http://eigentaste.berkeley.edu/dataset/.

[7] As we recommend only one item per time step, we evaluate the metrics after the end of the whole recommendation procedure.

Table 3
Ranking performance on **ML (100 k)** dataset. (The best results are marked in bold type.)

Model	Precision@1	Precision@3	Precision@20	NDCG@3	NDCG@20
Popularity	0.8842	0.8025	0.5685	0.9328	0.8720
PMF	0.8921	0.8261	0.5845	0.9335	0.8849
SVD++	0.9052	0.8220	0.5876	0.9364	0.8866
DQN	0.8014	0.7446	0.6076	0.9118	0.8815
DQN-L	0.8287	0.7629	0.5942	0.9135	0.8792
DQN-R	0.8335	0.7668	0.6017	0.9142	0.8804
DDPG	0.7978	0.7441	0.6052	0.9106	0.8870
DDPG-L	0.8255	0.7592	0.5912	0.9133	0.8826
DDPG-R	0.8327	0.7665	0.5958	0.9150	0.8841
DEERS	0.8517	0.8031	0.6481	0.9255	0.8933
DEERS-L	0.8754	0.8275	0.6414	0.9342	0.8902
DEERS-R	0.8793	0.8289	0.6433	0.9360	0.8916
DRR	0.8744	0.8136	0.6564	0.9259	0.8982
DRR-L	0.9172	0.8388	0.6504	0.9392	0.8926
DRR-R	0.9277	0.8405	0.6529	0.9418	0.8945
EDDR	0.8875	0.8287	**0.6925**	0.9293	**0.9062**
EDRR-L	0.9252	0.8416	0.6877	0.9408	0.8984
EDDR-R	**0.9337**	**0.8464**	0.6885	**0.9446**	0.9015

Table 4
Ranking performance on **BC** dataset. (The best results are marked in bold type.)

Model	Precision@1	Precision@3	Precision@20	NDCG@3	NDCG@20
Popularity	0.7128	0.6775	0.6302	0.9217	0.8898
PMF	0.7407	0.7045	0.6527	0.9345	0.9037
SVD++	0.7471	0.7067	0.6539	0.9359	0.9054
DQN	0.6937	0.6772	0.6557	0.9126	0.9062
DQN-L	0.7183	0.6987	0.6712	0.9144	0.9025
DQN-R	0.7194	0.6995	0.6738	0.9165	0.9037
DDPG	0.6924	0.6754	0.6548	0.9134	0.9070
DDPG-L	0.7139	0.6968	0.6703	0.9152	0.9028
DDPG-R	0.7172	0.6982	0.6726	0.9170	0.9041
DEERS	0.7075	0.6860	0.6685	0.9273	0.9118
DEERS-L	0.7462	0.7092	0.6619	0.9287	0.9087
DEERS-R	0.7524	0.7117	0.6631	0.9306	0.9104
DRR	0.7146	0.6956	0.6734	0.9289	0.9135
DRR-L	0.7695	0.7278	0.6683	0.9364	0.9106
DRR-R	0.7738	0.7346	0.6706	0.9385	0.9118
EDDR	0.7275	0.7044	**0.6887**	0.9338	**0.9189**
EDRR-L	0.7784	0.7362	0.6812	0.9394	0.9146
EDDR-R	**0.7820**	**0.7387**	0.6825	**0.9415**	0.9155

Table 5
Ranking performance on **ML (1 M)** dataset. (The best results are marked in bold type.)

Model	Precision@1	Precision@3	Precision@20	NDCG@3	NDCG@20
Popularity	0.8798	0.7483	0.5094	0.9265	0.8727
PMF	0.8918	0.7644	0.5213	0.9404	0.8734
SVD++	0.8923	0.7722	0.5183	0.9457	0.8785
DQN	0.8036	0.7329	0.5177	0.9084	0.8759
DQN-L	0.8384	0.7602	0.5080	0.9091	0.8715
DQN-R	0.8403	0.7635	0.5114	0.9097	0.8732
DDPG	0.8045	0.7307	0.5185	0.9097	0.8776
DDPG-L	0.8415	0.7733	0.5124	0.9102	0.8737
DDPG-R	0.8430	0.7784	0.5134	0.9125	0.8746
DEERS	0.8279	0.7850	0.5952	0.9249	0.8874
DEERS-L	0.8425	0.7996	0.5730	0.9264	0.8775
DEERS-R	0.8530	0.8107	0.5814	0.9272	0.8789
DRR	0.8441	0.7990	0.6227	0.9280	0.8912
DRR-L	0.9059	0.8235	0.6075	0.9432	0.8907
DRR-R	0.9072	0.8306	0.6183	0.9446	0.8910
EDDR	0.8628	0.8185	**0.6488**	0.9360	**0.8985**
EDRR-L	0.9155	0.8447	0.6335	0.9464	0.8937
EDDR-R	**0.9178**	**0.8483**	0.6362	**0.9485**	0.8952

Table 6
Ranking performance on **Jester** dataset. (The best results are marked in bold type.)

Model	Precision@1	Precision@3	Precision@20	NDCG@3	NDCH@20
Popularity	0.6718	0.6380	0.5608	0.9064	0.8415
PMF	0.6851	0.6499	0.5876	0.9125	0.8428
SVD++	0.6831	0.6441	0.5908	0.9206	0.8515
DQN	0.6238	0.6130	0.5952	0.8936	0.8548
DQN-L	0.6477	0.6401	0.5894	0.8965	0.8507
DQN-R	0.6513	0.6422	0.5907	0.8988	0.8515
DDPG	0.6267	0.6135	0.5947	0.8931	0.8550
DDPG-L	0.6550	0.6418	0.5904	0.9022	0.8513
DDPG-R	0.6584	0.6472	0.5926	0.9075	0.8537
DEERS	0.6447	0.6308	0.6018	0.9095	0.8748
DEERS-L	0.6642	0.6377	0.5831	0.9125	0.8652
DEERS-R	0.6764	0.6385	0.5852	0.9162	0.8675
DRR	0.6584	0.6394	0.6075	0.9175	0.8834
DRR-L	0.7192	0.6421	0.5975	0.9272	0.8795
DRR-R	0.7216	0.6470	0.6034	0.9306	0.8819
EDDR	0.6778	0.6465	**0.6168**	0.9246	**0.8898**
EDRR-L	0.7342	0.6584	0.6077	0.9315	0.8842
EDDR-R	**0.7370**	**0.6597**	0.6105	**0.9334**	0.8856

5.2.2. Simulated online evaluation results and analysis

The results of the simulated online evaluation are summarized in Tables 7 and 8, where $*$ denotes the corresponding RL method. We observe that proposed models achieve slightly worse performance in term of long-term cumulative reward compared with the base model DRR, but still outperform the other baselines. The fact suggests that the proposed methods do not sacrifice the long-term reward too much. Moreover, we observe that SRR-R models still perform better than SRR-L models.

5.3. Hyper-parameter study (RQ2)

In this subsection, we empirically study the influence of the embedding size d, length of h, batch size N and trade-off factor α on the proposed SRR framework. For simplicity, we take DRR-R as an example to investigate the influence of the parameters.

5.3.1. Embedding size d

To exploit the influence of the embedding size d, we tune it as $d \in \{4, 8, 16, 32, 64, 100, 200\}$, and report the results in Fig. 5. From the results, we observe that the performance of DRR-R is increased at first and then decreased when we enlarge the embedding size d. Both the Precision@1 and average reward achieve the best performance on the four datasets when $d = 100$. As the model capacity (i.e., the ability of modeling features) increases as the embedding size enlarges, which results in better performance. However, after reaching the peak performance, the model capacity will not increase even we enlarge the embedding size d, as the model capacity is limited by the scale of the training data.

5.3.2. Length of h

In proposed SRR, we take a set of latest positive interaction history (h) to represent the user state, here we investigate the influence of the length of h (n represent the length of h). We very $n \in \{3, 5, 7, 10, 15\}$, and Fig. 6 reports the results. From the results,

Table 7
The average rewards on **ML (100 k)** and **BC**. ($*$ denotes the corresponding RL method, and the best results are marked in bold type.)

Model	ML (100k)			BC		
	$*$	$*$-L	$*$-R	$*$	$*$-L	$*$-R
LinUCB	0.4266	–	–	0.3126	–	–
HLinUCB	0.3214	–	–	0.3225	–	–
DQN	0.5806	0.5714	0.5762	0.4237	0.4183	0.4208
DDPG	0.5783	0.5705	0.5748	0.4228	0.4175	0.4192
DEERS	0.7035	0.6842	0.6907	0.4465	0.4324	0.4357
DRR	0.7105	0.7025	0.7054	0.4517	0.4461	0.4493
EDRR	**0.7552**	0.7446	0.7471	**0.4623**	0.4570	0.4587

Table 8
The average rewards on **ML (1 M)** and **Jester**. ($*$ denotes the corresponding RL method, and the best results are marked in bold type.)

Model	ML (1 M)			Jester		
	$*$	$*$-L	$*$-R	$*$	$*$-L	$*$-R
LinUCB	0.4996	–	–	0.2391	–	–
HLinUCB	0.5428	–	–	0.2488	–	–
DQN	0.5944	0.5825	0.5887	0.2791	0.2674	0.2742
DDPG	0.5937	0.5892	0.5915	0.2805	0.2783	0.2792
DEERS	0.6635	0.6378	0.6493	0.3274	0.3086	0.3165
DRR	0.6746	0.6680	0.6717	0.3315	0.3268	0.3291
EDRR	**0.6924**	0.6852	0.6875	**0.3471**	0.3425	0.3437

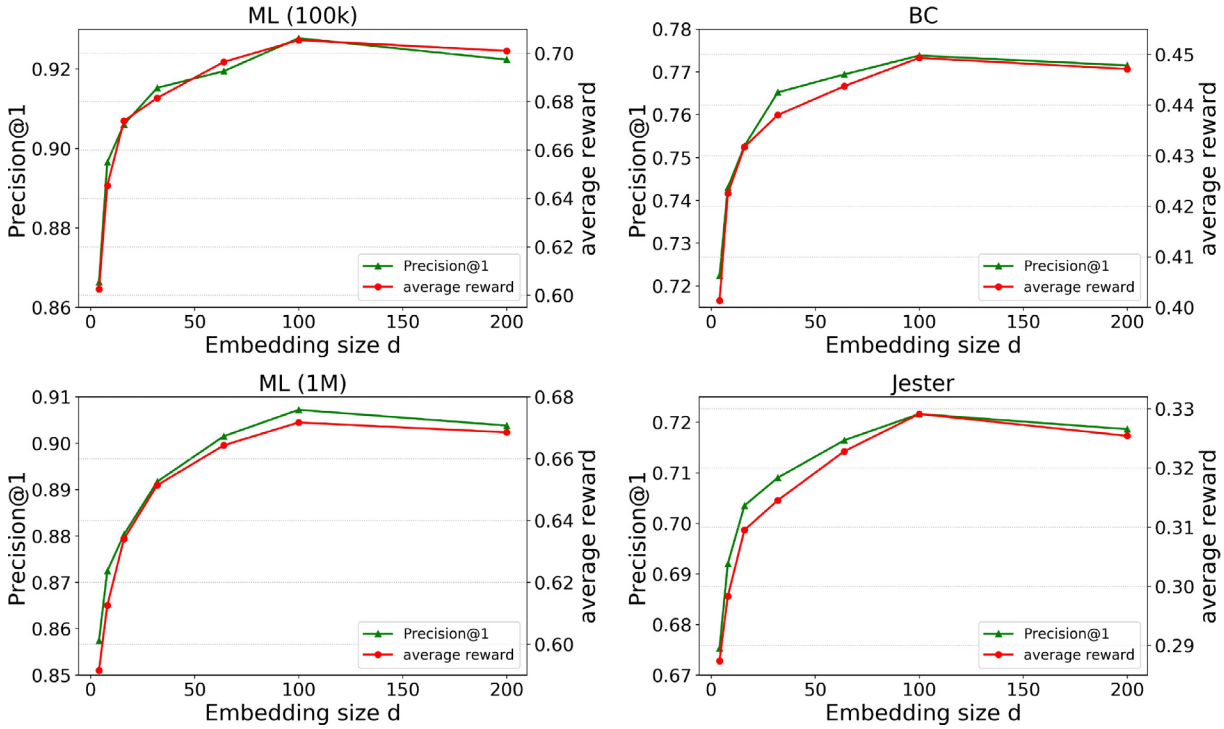

Fig. 5. Parameter study on embedding size *d*.

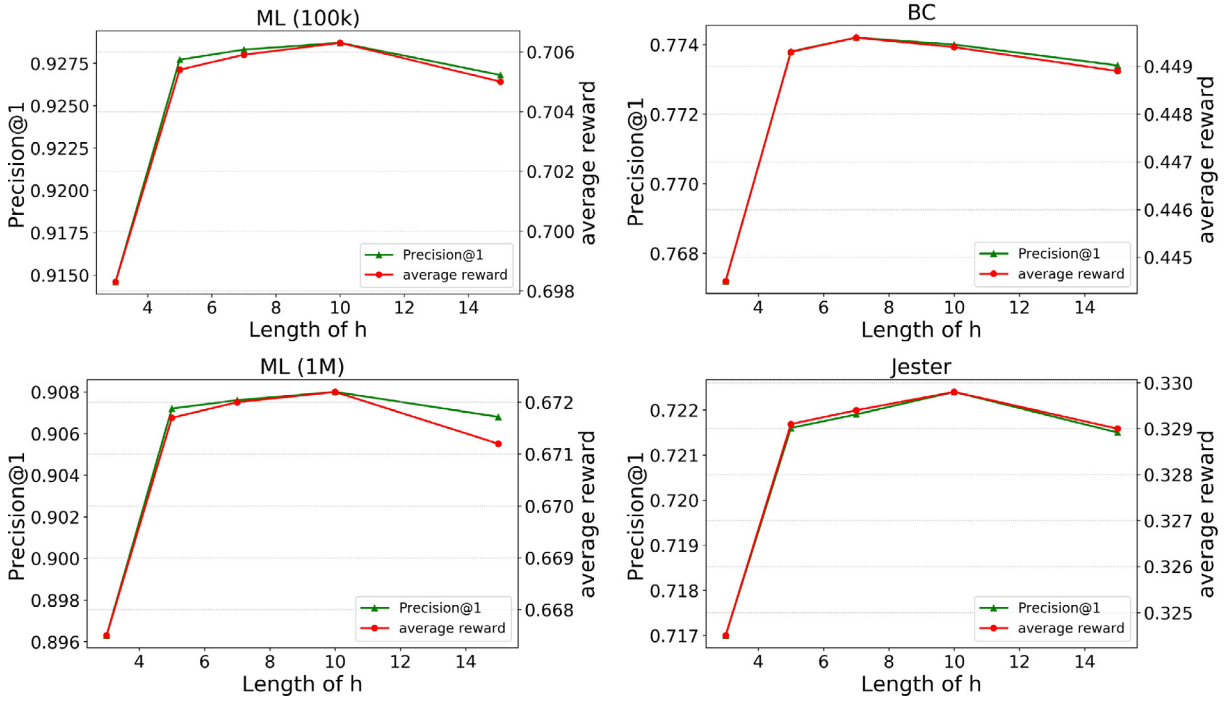

Fig. 6. Parameter study on length of *h*.

we observe that the performance is first increases and then decreases as we enlarge n, and the summit appears at $n = 7$ on BC dataset and $n = 10$ on ML (100k), ML (1M) and Jester datasets. The reasons are as follows: (1) As the length of h is too small, the state of user cannot be fully represented, which leads to lower performance; (2) An appropriate size of h can better represent the user state; (3) When n is too large (n = 15), some out-of-date items in h cannot represent the current state of the user, which also leads to lower performance. In SRR, we empirically choose $n = 5$, which appropriately trade-off the performance and the computational complexity.

5.3.3. Batch size N

In this subsection, we empirically study the influence of the batch size N on DRR-R. We tune the batch size $N \in \{64, 128, 256, 512, 1024\}$, and Fig. 7 reports the results. From

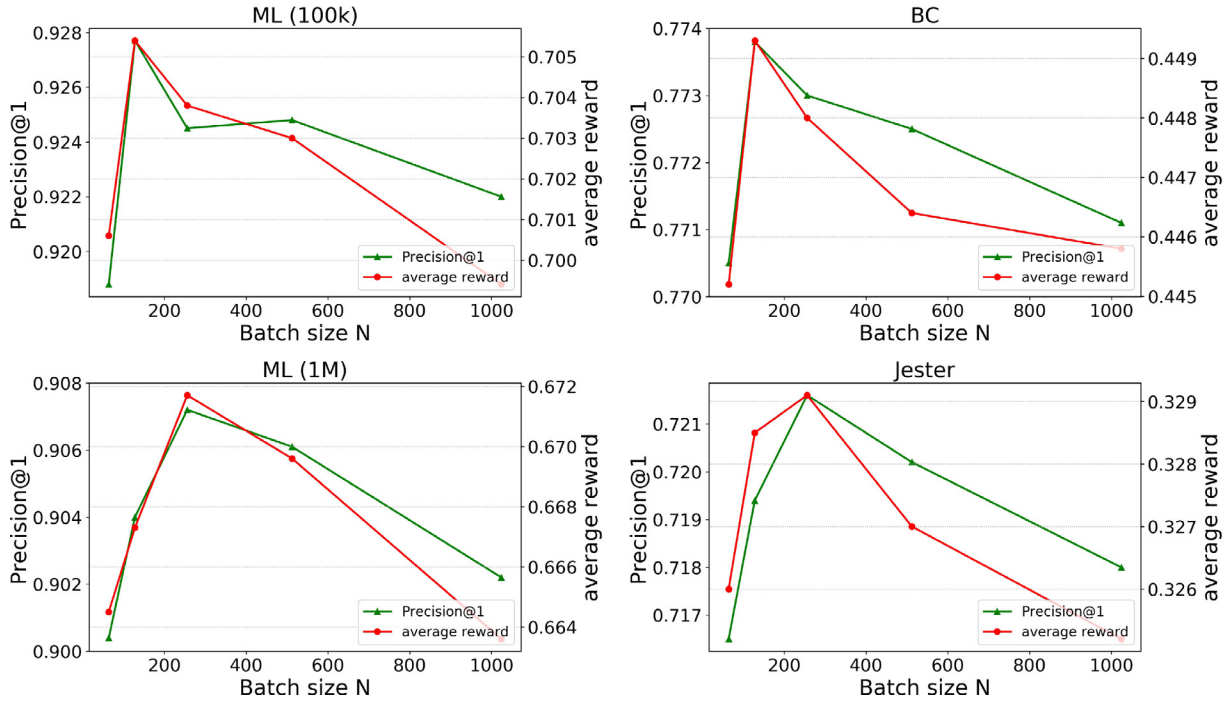

Fig. 7. Parameter study on batch size *N*.

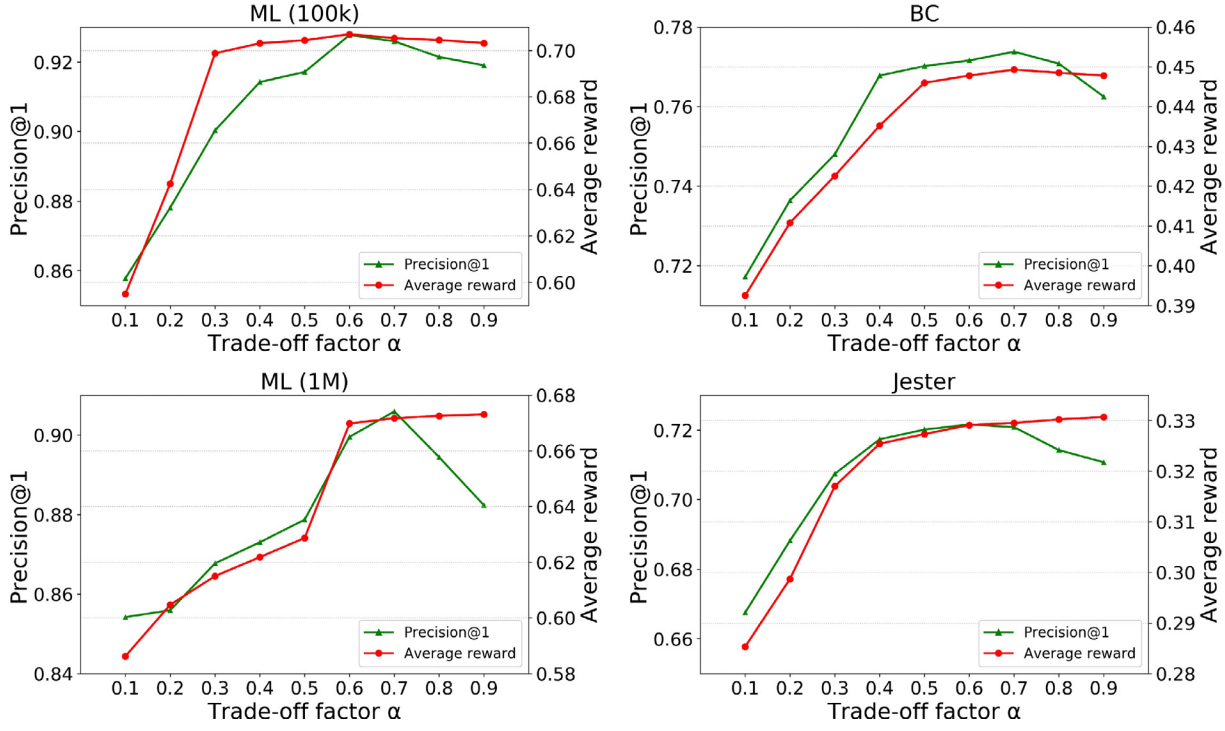

Fig. 8. Parameter study on trade-off factor α.

the results, we observe that the performances of Precision@1 and Average reward both are first increase and then decline when we enlarge batch size *N*, and the summit appears at $N = 128$ on ML (100k) and BC datasets, and $N = 256$ on ML (1M) and Jester datasets. We can conclude that an appropriate small batch size achieves higher performance, nevertheless too large batch size harms the generalization ability that leads to lower performance [54].

5.3.4. Trade-off factor α

The trade-off factor α is to balance the RL and SL signals. Fig. 8 presents the results of DRR-R. We observe that the Precision@1

Table 9
Different recommendation manner between DRR-R and DRR on ML-1M dataset. (The value in (·) denotes the corresponding rating.)

time step	DRR-R	DRR
1	Pinocchio (4)	Road to Wellville (2)
2	Absolute Power (3)	Jaws (4)
3	Vacation (5)	Baby Geniuses(1)
4	Almost Famous (4)	Star Wars: Episode I (5)
5	Forever Young (4)	For a Few Dollars More (3)
6	Titan A.E. (4)	Wayne's World (4)
7	Three Kings (3)	Peter Pan (5)
8	American Graffiti (5)	Aliens (3)
9	Patch Adams (3)	Lawrence of Arabia (4)
10	Yellow Submarine(4)	Top Gun(4)

first increases and then declines when we enlarge α, and the summit appears at $\alpha = 0.7$ for BC and ML (1M) datasets and $\alpha = 0.6$ for ML (100k) and Jester datasets. The average reward persistently increases and achieves comparable performance with DRR, which indicates that our model promotes the performance in top positions without much sacrifice of long-term reward. We can conclude: (1) when α is small, the weight of SL signals is strong, so that the exploration is restricted severely and the performance is not good; (2) a large α indicates weak SL supervision, in this case, the RL and SL signals can be appropriately balanced. Therefore, suitable weak supervision helps to trade off the immediate reward and the long-term reward, and the top-aware issue can be properly addressed.

5.4. Case study (RQ3)

Apart from being comparable in accuracy performance, the key advantage of SRR over other RL based recommendation methods is that it can address the top-aware issue. Towards this end, we show examples drawn from DRR-R and DRR on ML (1M) dataset to demonstrate different recommendation manners in top positions.

Here, we give a case study to show the superiority of DRR-R to DRR in addressing the top-aware issue. Specifically, we randomly pick up a user (the chosen user is with ID 48) from ML (1M) dataset and compare the recommendation procedure of DRR-R and DRR. The results are shown in Table 9. We observe that DRR-R and DRR perform differently in top 5 positions, and DRR-R performs better. DRR explores the user's preference more aggressively but loses accuracy. More specifically, after recommending 'Jaws' with rating 4, then DRR explores to recommend an adventurous movie 'Baby Geniuses' with rating 1, and a safe recommendation 'Star Wars: Episode I – The Phantom Menace' with rating 5, which is in the same genre with 'Jaws'. On the other hand, DRR-R keeps recommending high rating movies in the top positions. The observation demonstrates DRR-R takes moderate exploration to achieve reliable performance in top positions while DRR performs aggressive exploration with accuracy sacrifice in top positions.

6. Conclusion

We empirically find that the RL-based recommendation methods generally suffer from the top-aware issue. To address the drawback, we propose a supervised reinforcement learning recommendation framework SRR. In the framework, a supervised learning model is introduced to guide the training of the RL model. We find that weak supervision is helpful to address the top-aware issue. Extensive experiments on two real-world datasets have been carried out and the results demonstrate the proposed SRR framework indeed resolve the top-aware issue without much performance sacrifice in the long-run, compared with the state-of-the-art methods.

CRediT authorship contribution statement

Feng Liu: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing - original draft. **Ruiming Tang:** Methodology, Investigation, Formal analysis, Writing - review & editing. **Huifeng Guo:** Methodology, Investigation, Formal analysis, Writing - review & editing. **Xutao Li:** Methodology, Investigation, Formal analysis, Supervision, Writing - review & editing. **Yunming Ye:** Supervision, Project administration, Funding acquisition. **Xiuqiang He:** Supervision, Project administration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, D. Yin, Recommendations with negative feedback via pairwise deep reinforcement learning, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19–23, 2018, 2018, pp. 1040–1048. doi:10.1145/3219819.3219886..
[2] H. Chen, X. Dai, H. Cai, W. Zhang, X. Wang, R. Tang, Y. Zhang, Y. Yu, Large-scale interactive recommendation with tree-structured policy gradient, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 3312–3320..
[3] G. Dulac-Arnold, R. Evans, P. Sunehag, B. Coppin, Reinforcement learning in large discrete action spaces, CoRR abs/1512.07679..
[4] F. Liu, R. Tang, X. Li, Y. Ye, H. Chen, H. Guo, Y. Zhang, Deep reinforcement learning based recommendation with explicit user-item interactions modeling, CoRR abs/1810.12027. arXiv:1810.12027..
[5] F. Liu, H. Guo, X. Li, R. Tang, Y. Ye, X. He, End-to-end deep reinforcement learning based recommendation with supervised embedding, in: WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3–7, 2020, 2020, pp. 384–392. doi:10.1145/3336191.3371858..
[6] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N.J. Yuan, X. Xie, Z. Li, DRN: A deep reinforcement learning framework for news recommendation, in: Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23–27, 2018, 2018, pp. 167–176. doi:10.1145/3178876.3185994..
[7] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, J. Tang, Deep reinforcement learning for page-wise recommendations, in: Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2–7, 2018, 2018, pp. 95–103. doi:10.1145/3240323.3240374..
[8] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24–27, 2008, 2008, pp. 426–434. doi:10.1145/1401890.1401944..
[9] X. Zhao, L. Zhang, Z. Ding, D. Yin, Y. Zhao, J. Tang, Deep reinforcement learning for list-wise recommendations, arXiv preprint arXiv:1801.00209..
[10] R.J. Mooney, L. Roy, Content-based book recommending using learning for text categorization, in: ACM DL, 2000, pp. 195–204..
[11] Y. Koren, R.M. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, IEEE Computer 42 (8) (2009) 30–37.
[12] J. Wang, A.P. de Vries, M.J.T. Reinders, Unifying user-based and item-based collaborative filtering approaches by similarity fusion, in: SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6–11, 2006, 2006, pp. 501–508. doi:10.1145/1148170.1148257..
[13] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3–6, 2007, 2007, pp. 1257–1264..
[14] H.B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A.M. Hrafnkelsson, T. Boulos, J. Kubica, Ad click prediction: a view from the trenches, in: The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11–14, 2013, 2013, pp. 1222–1230. doi:10.1145/2487575.2488200..
[15] S. Rendle, Factorization machines, in: ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14–17 December 2010, 2010, pp. 995–1000. doi:10.1109/ICDM.2010.127..
[16] Y. Juan, Y. Zhuang, W. Chin, C. Lin, Field-aware factorization machines for CTR prediction, in: Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15–19, 2016, 2016, pp. 43–50. doi:10.1145/2959100.2959134..

[17] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, J. Wang, Product-based neural networks for user response prediction, in: IEEE 16th International Conference on Data Mining, ICDM 2016, December 12–15, 2016, Barcelona, Spain, 2016, pp. 1149–1154. doi:10.1109/ICDM.2016.0151..
[18] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, Deepfm: A factorization-machine based neural network for CTR prediction, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19–25, 2017, 2017, pp. 1725–1731. doi:10.24963/ijcai.2017/239..
[19] H. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, H. Shah, Wide & deep learning for recommender systems, CoRR abs/1606.07792..
[20] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, K. Gai, Deep interest network for click-through rate prediction, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19–23, 2018, 2018, pp. 1059–1068. doi:10.1145/3219819.3219823..
[21] L. Li, W. Chu, J. Langford, R.E. Schapire, A contextual-bandit approach to personalized news article recommendation, in: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26–30, 2010, 2010, pp. 661–670. doi:10.1145/1772690.1772758..
[22] H. Wang, Q. Wu, H. Wang, Learning hidden features for contextual bandits, in: Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24–28, 2016, 2016, pp. 1633–1642. doi:10.1145/2983323.2983847..
[23] O. Chapelle, L. Li, An empirical evaluation of thompson sampling, in: Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12–14 December 2011, Granada, Spain, 2011, pp. 2249–2257..
[24] H. Wang, Q. Wu, H. Wang, Factorization bandits for interactive recommendation, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4–9, 2017, San Francisco, California, USA, 2017, pp. 2695–2702..
[25] C. Zeng, Q. Wang, S. Mokhtari, T. Li, Online context-aware recommendation with time varying multi-armed bandit, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13–17, 2016, 2016, pp. 2025–2034. doi:10.1145/2939672.2939878..
[26] S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Factorizing personalized markov chains for next-basket recommendation, in: WWW 2010, Raleigh, North Carolina, USA, April 26–30, 2010, 2010, pp. 811–820. doi:10.1145/1772690.1772773. URL:https://doi.org/10.1145/1772690.1772773..
[27] B. Hidasi, A. Karatzoglou, L. Baltrunas, D. Tikk, Session-based recommendations with recurrent neural networks, in: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings, 2016. URL:http://arxiv.org/abs/1511.06939..
[28] B. Hidasi, A. Karatzoglou, Recurrent neural networks with top-k gains for session-based recommendations, in: CIKM, ACM, 2018, pp. 843–852..
[29] H. Guo, R. Tang, Y. Ye, F. Liu, Y. Zhang, A novel KNN approach for session-based recommendation, in: PAKDD, 2019, pp. 381–393.
[30] J. Tang, K. Wang, Personalized top-n sequential recommendation via convolutional sequence embedding, WSDM '18, ACM, New York, NY, USA, 2018, pp. 565–573. doi:10.1145/3159652.3159656. URL:http://doi.acm.org/10.1145/3159652.3159656.
[31] F. Yuan, A. Karatzoglou, I. Arapakis, J.M. Jose, X. He, A simple convolutional generative network for next item recommendation, in: WSDM 2019, Melbourne, VIC, Australia, February 11–15, 2019, 2019, pp. 582–590. doi:10.1145/3289600.3290975..
[32] S. Zhang, Y. Tay, L. Yao, A. Sun, J. An, Next item recommendation with self-attentive metric learning, AAAI 2019, vol. 9, 2019.
[33] C. Ma, P. Kang, X. Liu, Hierarchical gating networks for sequential recommendation, in: KDD 2019, Anchorage, AK, USA, August 4–8, 2019, 2019, pp. 825–833. doi:10.1145/3292500.3330984..
[34] G. Shani, D. Heckerman, R.I. Brafman, An mdp-based recommender system, Journal of Machine Learning Research 6 (Sep) (2005) 1265–1295.
[35] N. Taghipour, A. Kardan, A hybrid web recommender system based on q-learning, in: Proceedings of the 2008 ACM Symposium on Applied Computing, 2008, pp. 1164–1168.
[36] X. Zhao, L. Xia, D. Yin, J. Tang, Model-based reinforcement learning for whole-chain recommendations, arXiv preprint arXiv:1902.03987..
[37] X. Bai, J. Guan, H. Wang, A model-based reinforcement learning with adversarial training for online recommendation, Advances in Neural Information Processing Systems (2019) 10735–10746.
[38] X. Chen, S. Li, H. Li, S. Jiang, Y. Qi, L. Song, Generative adversarial user model for reinforcement learning based recommendation system, arXiv preprint arXiv:1812.10613..
[39] Y. Hu, Q. Da, A. Zeng, Y. Yu, Y. Xu, 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, KDD 2018, London, UK, August 19–23, 2018, 2018, pp. 368–377. doi:10.1145/3219819.3219846..
[40] L. Zou, L. Xia, Z. Ding, J. Song, W. Liu, D. Yin, 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, 2019, pp. 2810–2818.
[41] M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, E.H. Chi, Top-k off-policy correction for a REINFORCE recommender system, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11–15, 2019, 2019, pp. 456–464. doi:10.1145/3289600.3290999..
[42] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M.A. Riedmiller, Deterministic policy gradient algorithms, in: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21–26 June 2014, 2014, pp. 387–395..
[43] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, in: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings, 2016..
[44] O. Maillard, R. Munos, D. Ryabko, Selecting the state-representation in reinforcement learning, in: Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12–14 December 2011, Granada, Spain, 2011, pp. 2627–2635..
[45] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, vol. 1, MIT press Cambridge, 1998.
[46] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.
[47] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, in: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings, 2016..
[48] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to rank using gradient descent, in: Proceedings of the 22nd International Conference on Machine Learning, ICML '05, ACM, New York, NY, USA, 2005, pp. 89–96.
[49] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Transactions on Information Systems (TOIS) 20 (4) (2002) 422–446.
[50] L. Xia, J. Xu, Y. Lan, J. Guo, W. Zeng, X. Cheng, Adapting markov decision process for search result diversification, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2017, pp. 535–544.
[51] X. Zhao, L. Xia, Y. Zhao, D. Yin, j. Tang, Model-based reinforcement learning for whole-chain recommendations, arXiv preprint arXiv:1902.03987..
[52] W. Zhang, U. Paquet, K. Hofmann, Collective noise contrastive estimation for policy transfer learning, in: AAAI, AAAI Press, 2016, pp. 1408–1414.
[53] R. Cañamares, P. Castells, Should I follow the crowd?: A probabilistic analysis of the effectiveness of popularity in recommender systems, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08–12, 2018, 2018, pp. 415–424. doi:10.1145/3209978.3210014..
[54] N.S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P.T.P. Tang, On large-batch training for deep learning: Generalization gap and sharp minima, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings, 2017..

Feng Liu received B.S. and M.S. degrees both in Computer Science from Harbin Institute of Technology in China in 2014 and 2016, respectively. Since Sep, 2016, he has been at the School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, P.R. China, where he is currently a fourth-year PhD student. His research interests include recommender system, reinforcement learning, deep learning, data mining.

Ruiming Tang is currently a senior researcher in Huawei Noah's Ark Lab. He received the B.S. degree from the Department of Computer Science at Northeastern University China in 2009, and the Ph.D. degree from the Department of Computer Science at National University of Singapore, in 2014. He joined Huawei Noah's Ark Lab since 2014. His research interests include machine learning and artificial intelligence, particularly, deep learning and recommender systems.

Huifeng Guo is currently a researcher in Huawei Noah's Ark Lab. He received the Ph.D. degree from the Department of Computer Science and technology from the Shenzhen graduate school, Harbin Institute of Technology in 2018. His research interests include machine learning and artificial intelligence, particularly, deep learning and recommender systems.

Xutao Li received the bachelor's degree from the Lanzhou University, Lanzhou, China, in 2007, and the master's and Ph.D. degrees in computer science and technology from the Harbin Institute of Technology, Harbin, China, in 2009 and 2013, respectively. He was a Post-Doctoral Research Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore, in 2014 and 2015. He is currently an Associate Professor with the Shenzhen Graduate School, Harbin Institute of Technology. His current research interests include data mining, machine learning, graph mining, and social network analysis, especially tensor-based learning and mining algorithms.

Yunming Ye received the PhD degree in Computer Science from Shanghai Jiao Tong University. He is now a professor in the Shenzhen Graduate School, Harbin Institute of Technology. His research interests include data mining, text mining, and ensemble learning algorithms.

Xiuqiang He received the PhD degree in Computer Science from The Hong Kong University of Science and Technology. He is now an expert researcher in Huawei Noah's Ark Lab. His research interests include machine learning and artificial intelligence, particularly, deep learning and recommender systems.