Knowledge-Based Systems 264 (2023) 110335

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Deep reinforcement learning in recommender systems: A survey and new perspectives

Xiaocong Chen [a,*], Lina Yao [a,d,**], Julian McAuley [b], Guanglin Zhou [a], Xianzhi Wang [c]

[a] University of New South Wales, NSW, Australia
[b] University of California, San Diego, CA, USA
[c] University of Technology Sydney, NSW, Australia
[d] Data 61, CSIRO, Australia

ARTICLE INFO

Article history:
Received 17 November 2022
Received in revised form 3 January 2023
Accepted 24 January 2023
Available online 26 January 2023

Keywords:
Deep reinforcement learning
Deep learning
Recommender systems

ABSTRACT

In light of the emergence of deep reinforcement learning (DRL) in recommender systems research and several fruitful results in recent years, this survey aims to provide a timely and comprehensive overview of recent trends of deep reinforcement learning in recommender systems. We start by motivating the application of DRL in recommender systems, followed by a taxonomy of current DRL-based recommender systems and a summary of existing methods. We discuss emerging topics, open issues, and provide our perspective on advancing the domain. The survey serves as introductory material for readers from academia and industry to the topic and identifies notable opportunities for further research.

1. Introduction

Recent years have seen significant progress in recommendation techniques from traditional recommendation techniques (e.g., collaborative filtering, content-based recommendation and matrix factorization [1]) to deep learning based techniques. Deep learning is increasingly used in recommender systems, due to its capability to capture non-linear user–item relationships and deal with various types of data sources such as images and text. It has shown strong advantages in solving complex tasks and dealing with complex data. However, deep learning-based recommender systems have limitations in capturing interest dynamics [2,3] due to distribution shifts, i.e., the training phase is based on an existing dataset which may not reflect real user preferences that undergo rapid change. In contrast, deep reinforcement learning (DRL) aims to train an agent that can learn from interaction trajectories provided by the environment by combining the power of deep learning and reinforcement learning. Since an agent can actively learn from users' real-time feedback to infer dynamic user preferences in DRL, DRL is especially suitable for learning from interactions, such as human–robot collaboration. It has driven significant advances in interactive applications ranging from video games, Alpha Go to autonomous driving [4]. While, before the DRL is widely used in RS, contextual bandit based methods are the main stream to handle such interactive process [5]. However, the contextual bandit cannot deal with the dynamic environment [2]. Hence, DRL would be a better choice for RS. In light of the significance and recent progresses in DRL for recommender systems, we aim to timely summarize and review DRL-based recommendation systems in this survey.

1.1. Differences with existing surveys

There is a recent published survey [6] reviews reinforcement learning in recommender systems but does not investigate the growing area of deep reinforcement learning comprehensively. Moreover, it just provides an overview of the existing works without providing a systemic taxonomy, deep discussion for emerging topics, open questions and, future directions of DRL RS. There is another survey about deep learning based RS did cover some aspects about DRL RS [3]. While the works that mentioned in [3] is out-of-date and did not provide a in deep analysis about the DRL RS. While, it is notable that, there is another work available on arXiv [7]. However, it is later than ours but with very similar structures and taxonomy and being considered as a follow up work with this survey.

1.2. Our contributions

Our survey distinguishes from the past published survey in providing a systematic overview of existing methods, along with

* Corresponding author.
** Corresponding author at: University of New South Wales, NSW, Australia.
E-mail addresses: xiaocong.chen@unsw.edu.au (X. Chen), lina.yao@unsw.edu.au (L. Yao).

https://doi.org/10.1016/j.knosys.2023.110335

a discussion of emerging topics, open issues, and future directions. This survey introduces researchers, practitioners and educators to this topic and fosters an understanding of the key techniques in the area. The main contributions of this survey are as follows:

- We provide an up-to-date comprehensive review of deep reinforcement learning in recommender systems, with state-of-the-art techniques and pointers to key references. To the best of our knowledge, this the first comprehensive survey in deep reinforcement learning-based recommender systems.
- We present a taxonomy of the literature of deep reinforcement learning in recommender systems. Along with the outlined taxonomy and literature overview, we discuss their benefits and drawbacks.
- We shed light on emerging topics and open issues for DRL-based recommender systems. We also point out future directions for advancing DRL-based recommender systems.

The remainder of this survey is organized as follows: Section 2 provides an overview of recommender systems, DRL and their integration. Section 3 provides a taxonomy and classification mechanism. Section 4 reviews the emerging topics. Section 5 points out open questions, and finally, Section 6 gives a few promising future directions.

2. Background

In this section, we introduce the key concepts related to dynamic recommender systems (RS) and deep reinforcement learning (DRL), and motivate the introduction of DRL to dynamic recommender systems.

2.1. Why deep reinforcement learning for recommendation?

Recommender systems require coping with *dynamic* environments by estimating rapidly changing users' preferences and proactively recommending items to users. Let $\mathcal{U}$ be a set of users of cardinality $|\mathcal{U}|$ and $\mathcal{I}$ be a set of items of cardinality $|\mathcal{I}|$. For each user $u \in \mathcal{U}$, we observe a sequence of user actions $\mathbb{X}^u = [x_1^u, x_2^u, \ldots, x_{T_u}^u]$ with item $x_t^u \in \mathcal{I}$, i.e., each event in a user sequence comes from the item set. We refer to a user making a decision as an interaction with an item. Let $\mathcal{F}$ by users' feedback (e.g., ratings or clicking behavior), a dynamic recommender system maintains a recommendation policy π_t^u, which will be updated based on the feedback $f_i^u \in \mathcal{F}$ received during the interaction for item $i \in \mathcal{I}$ at the timestamp t.

The marriage of deep learning and reinforcement learning has fueled breakthroughs in recommender systems. DRL-based RS consists of a pipeline of three building blocks: environment construction, state representation and recommendation policy learning. Environment construction aims to build an environment based on a set of users' historical behaviors. State representation is provided by the environment containing certain user information, such as historical behavior and demographic data. Recommendation policy learning is the key component to understand and predict users' future behavior. DL-based RS learns about users' interests and update the recommender based on user feedback, while DRL-based RS receives the reward provided by the environment to update the policy. The reward provided by the environment is a pre-defined function containing several factors. The detailed processes of DL-based RS and DRL-based RS mapping can be found in Fig. 3.

2.2. Preliminaries of deep reinforcement learning

DRL has the characteristic of leveraging deep learning to approximate reinforcement learning's value function and solve high-dimensional Markov Decision Processes (MDPs) [4]. A MDP can be represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. The agent chooses an action $a_t \in \mathcal{A}$ according to the policy $\pi_t(s_t)$ at state $s_t \in \mathcal{S}$. The environment receives the action and produces a reward $r_{t+1} \in \mathcal{R}$; then, it produces the next state s_{t+1} according to the transition probability $P(s_{t+1}|s_t, a_t) \in \mathcal{P}$. The transition probability $\mathcal{P}$ is unknown beforehand in DRL.

Such a process continues until the agent reaches the terminal state or exceeds a pre-defined maximum time step. The overall objective is to maximize the expected discounted cumulative reward:

$$\mathbb{E}_\pi[r_t] = \mathbb{E}_\pi\Big[\sum_0^\infty \gamma^k r_{t+k}\Big] \tag{1}$$

where $\gamma \in [0, 1]$ is the discount factor that balances the future reward and the immediate reward.

Deep reinforcement learning can be either *model-based* and *model-free* (a detailed taxonomy can be found in Fig. 2). Their major difference is whether the agent can learn a model of the environment: model-based methods aim to estimate the transition function and reward function, while model-free methods estimates the value function or policy from experience. In particular, in model-based methods, the agent accesses the environment and plans ahead; in contrast, model-free methods gain sample efficiency from using models that require extensive development and testing than model-based methods [4].

Deep reinforcement learning approaches are divided into three streams: *value-based*, *policy-based* and *hybrid* methods. In value-based methods, the agent updates the value function to learn a policy; policy-based methods learn the policy directly; hybrid methods or *actor–critic* methods, which combine value-based and policy-based methods by using two different networks, where an actor network uses a policy-based method and the critic uses a value-based method to evaluate the policy learned by the agent. The notations used in this manuscript can be found on Table 1.

Deep reinforcement learning includes *on-policy* and *off-policy* methods. Off-policy methods, use the behavior policy π_b for exploration and the target policy π for decision-making. For on-policy methods, the behavior policy is the same as the target policy.

Q-learning [8] is an off-policy value-based learning scheme for finding a greedy target policy:

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q_\pi(s, a) \tag{2}$$

where $Q_u(s, a)$ denotes the Q-value and is used in a small discrete action space. For a deterministic policy, the Q value can be calculated as follows:

$$Q(s_t, a_t) = \mathbb{E}_{\tau\sim\pi}[r(s_t, a_t) + \gamma Q(s_t', a_t')]. \tag{3}$$

Deep Q learning (DQN) [9] uses deep learning to approximate a non-linear Q function parameterized by θ_q: $Q_{\theta_q}(s, a)$. DQN designs a network Q_{θ_q} that is asynchronously updated by minimizing the MSE:

$$\mathcal{L}(\theta_q) = \mathbb{E}_{\tau\sim\pi}\Big[Q_{\theta_q}(s_t, a_t) - (r(s_t, a_t) + \gamma Q_{\theta_q}(s_t', a_t'))\Big]^2 \tag{4}$$

where τ is the sampled trajectory containing $(s, a, s', r(s, a))$. In particular, s_t' and a_t' come from the behavior policy π_b while s, a comes from the target policy π. The value function $V_\pi(s)$ represents the expected return. $V_\pi(s)$ is used to evaluate the state,

Table 1
Notations used in this manuscript.

Notations	Name	Notations	Name	Notes
$Q(\cdot)$	Q-value function	s	State	Users' preference
$V(\cdot)$	Value function	a	Action	Recommended item(s)
γ	Discount factor	π, $\mu(\cdot)$	Policy	Recommendation policy
$\mathbb{E}$	Expected value	$r(\cdot,\cdot)$	Reward	Users' click behavior
θ	Model parameter	α	constant $\in [0, 1]$	–
$p(\cdot)$	Transition probability	τ	Sampled trajectory	A tuple (s_t, a_t, s'_t, r_t)

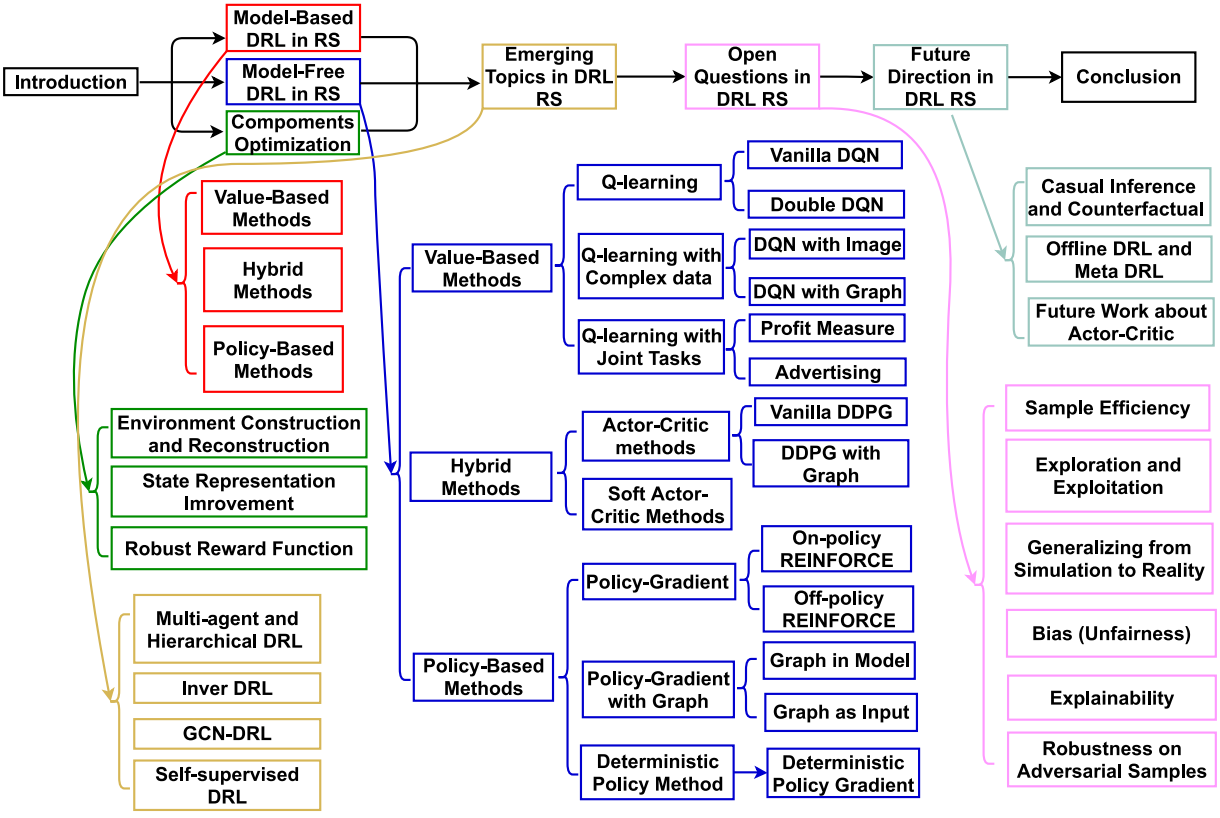


Fig. 1. Taxonomy of deep reinforcement learning-based recommender systems in this survey.

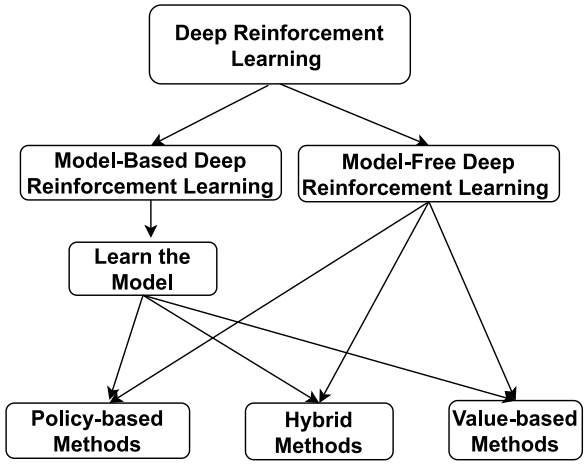


Fig. 2. Taxonomy of deep reinforcement learning in recommender systems.

and $Q_\pi(s_t, a_t)$ is used to evaluate the action. $V_\pi(s)$ can be defined as

$$V_\pi(s) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\gamma^t r(s,a)|s_0 = s\right]. \tag{5}$$

$V_\pi(\cdot)$ and $Q_\pi(\cdot)$ have the following relationship:

$$V_\pi(s) = \mathbb{E}_{a\sim\pi}[Q_\pi(s,a)]. \tag{6}$$

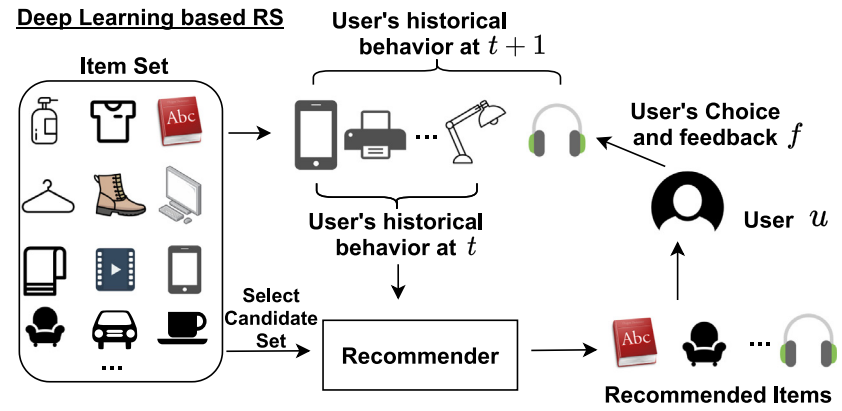
The value function is updated using the following rule with the Temporal Difference (TD) method,

$$V_\pi(s_t) \leftarrow V_\pi(s_t) + \alpha[\underbrace{r(s'_t, a'_t) + \gamma V_\pi(s'_t) - V_\pi(s_t)}_{\text{TD-error}}] \tag{7}$$

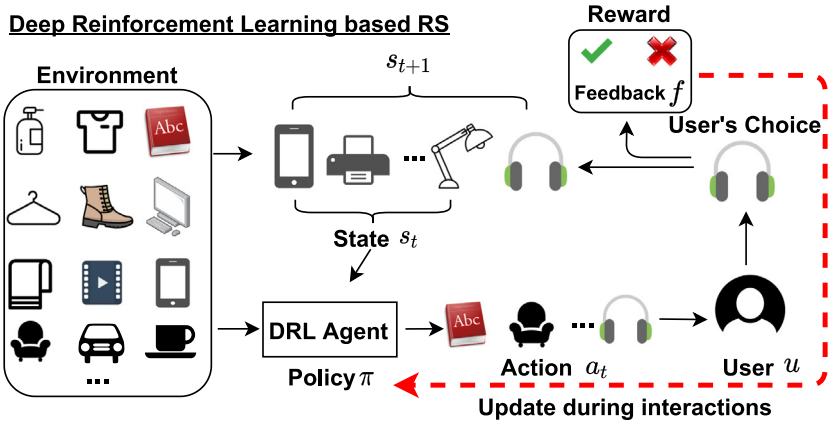
where α is a constant.

Policy gradient [10] is an on-policy method that can handle high-dimensional or continuous actions which cannot be easily handled by Q-learning. Policy gradient aims to find the parameter θ of π_θ to maximize the accumulated reward. To this end, it maximizes the expected return from the start state:

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)] = \int \pi_\theta(\tau) r(\tau) d\tau \tag{8}$$



(a) Deep learning based recommender systems



(b) Deep reinforcement learning-based recommender systems

Fig. 3. Difference between deep learning based RS and DRL-based RS. Deep learning based RSs may only update the recommendation policy during the training stage. They often require re-training, which is computationally inefficient, when users' interests change significantly. DRL-based RS will update the recommendation policy time over time as new rewards are received.

where $\pi_\theta(\tau)$ is the probability of the occurrence of τ. Policy gradient learns the parameter θ by the gradient $\nabla_\theta J(\pi_\theta)$ as defined below:

$$\begin{aligned}\nabla_\theta J(\pi_\theta) &= \int \pi_\theta(\tau) r(\tau) d\tau = \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim d_{\pi_\theta}}[\sum_{t=1}^{T} r(s_t, a_t) \sum_{t=1}^{T} \nabla_\theta \\ &\quad \times \log \pi_\theta(s_t, a_t)]. \end{aligned} \tag{9}$$

The above derivations contain the following substitution:

$$\pi_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(s_t, a_t) p(s_{t+1} | s_t, a_t) \tag{10}$$

where $p(\cdot)$ is independent from the policy parameter θ, which is omitted during the derivation. Monte-Carlo sampling has been used by previous policy gradient algorithm (e.g.,. REINFORCE) for $\tau \sim d_{\pi_\theta}$.

Actor–critic networks combine the advantages from Q-learning and policy gradient. They can be either on-policy [11] or off-policy [12]. An actor–critic network consists of two components: (i) *an actor*, which optimizes the policy π_θ under the guidance of $\nabla_\theta J(\pi_\theta)$; and (ii) *a critic*, which evaluates the learned policy π_θ by using $Q_{\theta_q}(s, a)$. The overall gradient is represented as follows:

$$\mathbb{E}_{s \sim d_{\pi_\theta}}[Q_{\theta_q}(s, a) \nabla_\theta \log \pi_\theta(s, a)]. \tag{11}$$

When dealing with off-policy learning, the value function for $\pi_\theta(a|s)$ can be further determined by deterministic policy gradient (DPG):

$$\mathbb{E}_{s \sim d_{\pi_\theta}}[\nabla_a Q_{\theta_q}(s, a)|_{a=\pi_\theta(s)} \nabla_\theta \pi_\theta(s, a)]. \tag{12}$$

While traditional policy gradient calculates the integral for both the state space $\mathcal{S}$ and the action space $\mathcal{A}$, DPG only requires computing the integral to the state space $\mathcal{S}$. Given a state $s \in \mathcal{S}$, there will be only one corresponding action $a \in \mathcal{A} : \mu_\theta(s) = a$ using DPG. Specifically, deep Deterministic Policy Gradients (DDPG) is an algorithm that combines techniques from DQN and DPG, and the structure can be found on Fig. 5. DDPG contains four different neural networks: Q Network Q, policy network, target Q network Q^{tar}, and target policy network. It uses the target network for both the Q Network Q and policy network μ to ensure stability during training. Let $\theta_q, \theta_\pi, \theta_{q'}$ and $\theta_{\pi'}$ be parameters of the above networks. DDPG soft-updates the parameters for the target network [13]:

$$\text{Actor: } \theta_{\pi'} \leftarrow \alpha\theta_\pi + (1-\alpha)\theta_{\pi'} \text{ Critic: } \theta_{q'} \leftarrow \alpha\theta_q + (1-\alpha)\theta_{q'} \tag{13}$$

2.3. DRL meets RS: Problem formulation

DRL is normally formulated as a Markov Decision Process (MDP). Given a set of users $\mathcal{U} = \{u, u_1, u_2, u_3, \ldots\}$, a set of items $\mathcal{I} = \{i, i_1, i_2, i_3, \ldots\}$, the system first recommends item i to user u and gets feedback f_i^u. The system incorporates the

Table 2
List of publications in model-based DRL-based RS.

Method	Work
Value-based	[14–17]
Policy-based	[18,19]
Hybrid	[20,21]

feedback to improve future recommendations and determines an optimal policy π^* regarding which item to recommend to the user to achieve positive feedback. The MDP modeling treats the user as the environment and the system as the agent. The key components of the MDP in DRL-based RS include the following:

- State $\mathcal{S}$: A state $S_t \in \mathcal{S}$ is determined by both users' information and the recent l items in which the user was interested before time t.
- Action $\mathcal{A}$: An action $a_t \in \mathcal{A}$ represents users' dynamic preference at time t as predicted by the agent. $\mathcal{A}$ represents the whole set of (potentially millions of) candidate items.
- Transition Probability $\mathcal{P}$: The transition probability $p(s_{t+1}|s_t, a_t)$ is the probability of state transition from s_t to s_{t+1} when action a_t is executed by the recommendation agent. In a recommender system, the transition probability refers to users' behavior probability. $\mathcal{P}$ is only used in model-based methods.
- Reward $\mathcal{R}$: Once the agent chooses a suitable action a_t based on the current state S_t at time t, the user will receive the item recommended by the agent. Users' feedback on the recommended item accounts for the reward $r(S_t, a_t)$. The feedback is used to improve the policy π learned by the recommendation agent.
- Discount Factor γ: The discount factor $\gamma \in [0, 1]$ is used to balance between future and immediate rewards. The agent only focuses on the immediate reward when $\gamma = 0$ and takes into account all the (immediate and future) rewards otherwise.

The DRL-based recommendation problem can be defined by MDP as follows. *Given the historical MDP, i.e., $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, the goal is to find a set of recommendation polices $(\{\pi\} : \mathcal{S} \rightarrow \mathcal{A})$ that maximizes the cumulative reward during interaction with users.*

Problem Formulation 1. *Given an environment that contains all items $\mathcal{I}$, when user $u \in \mathcal{U}$ interacts with the system, an initial state s is sampled from the environment which contains a list of candidate items and users' historical data. The DRL agent works out a recommendation policy π based on the state s and produces a list of recommended items a. The user will provide feedback on the list (through 'click' or 'not click'). The DRL agent then utilizes the feedback to improve the recommendation policy and move to the next interaction episode.*

3. Deep reinforcement learning in recommender systems

DRL-based RS has unique challenges like state construction, reward estimation and environment simulation. We categorize the existing work of DRL-based recommendation into model-based and model-free methods (the taxonomy is shown in Fig. 2).

3.1. Model-based deep reinforcement learning-based methods

Model-based methods assume an expected reward or action available for the next step to help the agent update the policy (see Table 2).

Policy-based methods. IRecGAN [18] is a model-based method that adopts generative adversarial training to improve the robustness of policy learning. It can reduce the cost of interaction for RS by using offline data instead of the simulated environment. IRecGAN employs a generative adversarial network [22] to generate user data based on the offline dataset. It trains a recommendation agent using a policy gradient-based DRL method called REINFORCE. The agent aims to learn a policy based on the following gradient,

$$\mathbb{E}_{\tau\sim\{g,data\}}\Big[\sum_{t=0}^{T}\sum_{t'=t}^{T}\gamma^{t'-t}q_D(\tau_{0:t}^n)r_t\nabla_{\theta_a}c_t \in \pi_{\theta_a}(s_t)\Big], \tag{14}$$

$$q_D(\tau_{0:t}^n) = \frac{1}{N}\sum_{n=1}^{N} D(\tau_{0:T}^n),\ \tau_{0:T}^n \in MC^{\mathcal{U}}(N)$$

where $MC^{\mathcal{U}}(N)$ represents the sampled N sequences from the interaction between $\mathcal{U}$ and the agent using the Monte-Carlo tree search algorithm, D is the discriminator, T is the length of τ, g represents the offline data, and *data* represents the generated data.

Hong et al. [19] propose NRSS for personalized music recommendation. NRSS uses wireless sensing data to learn users' current preferences. It considers three different types of feedback: score, option, and wireless sensing data. Since NRSS considers multiple factors are considered as the reward, it designs a reward model consisting of users' preference reward r_p and a novel transition reward r_{trans} which are parameterized by θ_{r_p} and $\theta_{r_{trans}}$. On the above basis, NRSS finds the optimal parameters θ_{r_p} and $\theta_{r_{trans}}$ by using the Monte-Carlo tree search. Wireless sensing feedback lacks generalization ability as it is only available for certain tasks or scenarios, making it hard to determine dynamic user interest.

Value-based methods. Prior to Q-learning, value iteration is a traditional value-based reinforcement learning algorithm that focuses on the iteration of the value function. Gradient Value Iteration (GVI) [14] is proposed to improve the traditional value iteration algorithm by utilizing the transition probability and a multi-agent setting to predict chronological author collaborations. It introduces a new parameter named 'status' to reflect the amount of knowledge that the agent needs to learn from this state. The policy is updated only when the distance between the new status and the old status is lower than a pre-defined threshold. Since value iteration requires the transition probability, which is hard to obtain in most cases, Q-learning and its variants are widely used in DRL-based RS. Cascading DQN (CDQN) with a generative user model [15] aims to deal with environments with unknown reward and environment dynamics. The generative user model adopts GANs to generate a user model based on offline data. Different from previous work, it generates a reward function for each user to explain the users' behavior. As such, the user model can be written as,

$$\underset{\phi\in\triangle^{k-1}}{\operatorname{argmax}}\,\mathbb{E}_{\phi}[r(s_t, a_t)] - R(\phi)/\eta \tag{15}$$

where $\triangle^{k-1}$ is the probability simplex, $R(\phi)$ is the regularization term for exploration, and η is a constant.

Pseudo Dyna-Q (PDQ) [16] points out that Monte-Carlo tree search may lead to an extremely large action space and an unbounded importance weight of training samples. It thereby propose a world model to reduce the instability of convergence and high computation cost for interacting with users by imitating the offline dataset. With the world model, the agent will interact with the learned world model instead of the environment to improve the sample efficiency and convergence stability. The learning

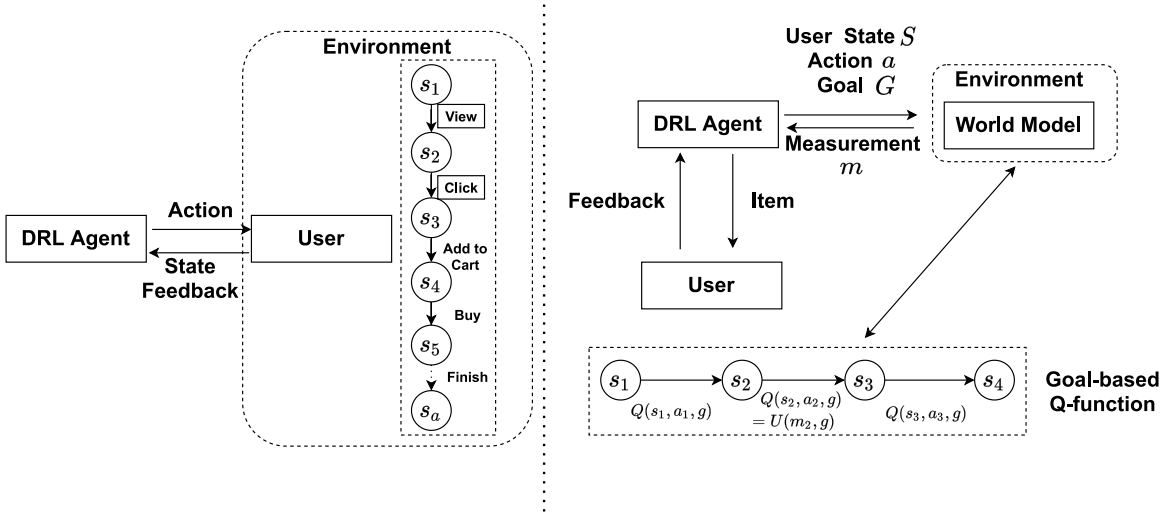


Fig. 4. Left is the general structure of model-free methods. Right is the structure for GoalRec which is a model-based method. A sample trajectory is used to demonstrate the difference between them [17].

process of the world model in PDQ can be described as finding the parameter θ_M:

$$\underset{\theta_M}{\operatorname{argmin}}\, \mathbb{E}_{\xi\in P_\xi^\pi}[\sum_{t}^{T-1}\gamma^t \prod_{j=0}^{t}\frac{\pi(s_j,a_j)}{\pi_b(s_j,a_j)}\Delta_t(\theta_M)] \tag{16}$$

where ξ is generated by the logged policy π_b, $\prod_{j=0}^{t}\frac{\pi(s_j,a_j)}{\pi_b(s_j,a_j)}$ is the ratio used for importance sampling and Δ is the difference between the reward in the world model and real reward.

Furthermore, GoalRec [17] designs a disentangled universal value function to be integrated with the world model to help the agent deal with different recommendation tasks. The universal value function is defined as

$$V_\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} r(s_t,a_t)\prod_{k=0}^{t}\gamma s_k | s_0 = s]. \tag{17}$$

Besides, GoalRec introduces a new variable goal $g \in G$ used to represent users' future trajectory and measurement $m \in M$. m is an indicator of users' response to the given future trajectory based on historical behaviors. Based on that, the optimal action will be selected based on

$$a^* = \max_a U(M(s,a),g) \tag{18}$$

with a customized liner function $U(\cdot)$ (see Fig. 4).

Hybrid methods. Hybrid methods are in the middle of value-based and policy gradient-based methods. DeepChain [20] uses the multi-agent setting to mitigate the sub-optimality problem caused by the *one for all* setting, i.e., optimizing a single policy for all users. Hence, DeepChain designs a multi-agent setting that adopts several agents to learn consecutive scenarios and jointly optimizes multiple recommendation policies. The main training algorithm used is DDPG. To this end, users' actions can be formulated in a model-based form as follows:

$$\sum_{m,d}[p_m^s(s_t,a_t)\gamma Q_\theta(s_t', \pi_m(s_t')) + p_m^c(s_t,a_t)(r_t + \gamma Q_\theta(s_t', \pi_d(s_t')))$$
$$+ p_m^l(s_t,a_t)r_t]1_m \tag{19}$$

where m represents the number of actor networks, c, l, s represent the three different scenarios, 1_m is used to control the activation of two actors and $(m,d) \in \{(1,2),(2,1)\}$.

Montazeralghaem and Allan [21] design a model-based method called RelInCo, which incorporates the sentence-BERT [23] into the policy network to compute the representations of context and world model. RelInCo is based on Actor–critic method but contains two different pairs of actor–critic networks: Arrangement-Actor-Critic and Selector-Actor-Critic. The arrangement network can provide the order of words in the user utterance, while the selector network is designed distinguish the word is relevant to utterance or not.

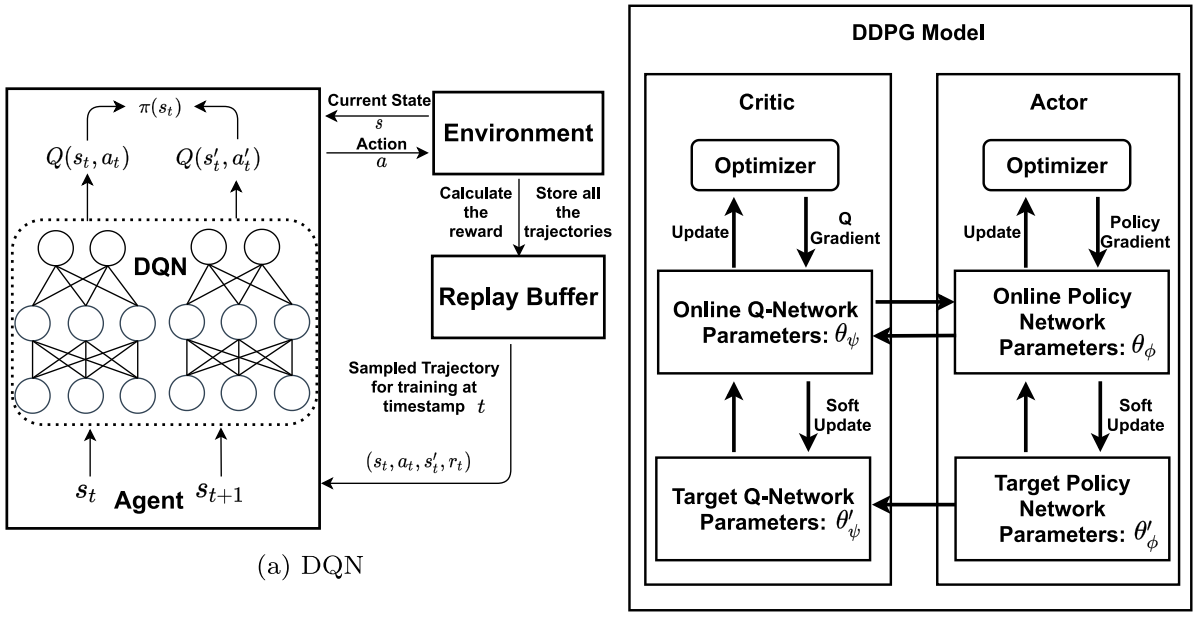
Discussion. Model-based methods aim to learn a model or representation to represent the whole environment so that the agent can plan ahead and receive better sample efficiency. The main drawback of such methods is that the environment might dynamically change and consequently the ground-truth representation of the environment could be unavailable or biased in recommendation scenarios. Moreover, model-based methods use the transition probability function $\mathcal{P}$ to estimate the optimal policy. As mentioned, the transition probability function is equivalent to users' behavior probability and is hard to determine in recommender systems. In view of the above issues, existing work [14–18,20] approximate $\mathcal{P}$ using a neural network or embedding it into the world model; besides, Zhao et al. [20] design a probability network to estimate $\mathcal{P}$, and [15,18] uses a GAN to generate user behavior where $\mathcal{P}$ is embedded in the latent space. Different from the above, [16,17] relies on the world model to predict users' next behavior and feed it into the policy learning process.

In summary, model-based DRL are not widely used in RS due to following reasons:

- $\mathcal{P}$ is hard to determine in real-world recommender systems.
- If approximation is used to estimate $\mathcal{P}$, the overall model complexity will substantially increase as it requires approximating two different functions $\mathcal{P}$ and the recommendation policy π by using a large amount of user behavior data.
- World model-based methods require periodic re-training to ensure the model can reflect user interests in time, which further increases the computation cost.

3.2. Model-free deep reinforcement learning-based methods

Compared with model-based methods, model-free methods are relatively well-studied. Different from model-based methods, Model-free methods assume $\mathcal{P}$ is unknown and enable the agent to learn it from previous experiences. Model-free based DRL in RS can be put into three categories: value-based, policy-based and hybrid methods (see Table 3).



(a) DQN

(b) DDPG

Fig. 5. The typical structure of DQN and DDPG.

Table 3
List of reviewed publications in model-free DRL-based RS.

Tasks	Note	Work
Value-based	Vanilla DQN and its extensions	[24–26]
	DQN with state/action space optimization	[27–30]
	DQN with graph/image input	[31–37]
	DQN for joint learning	[38–40]
Policy-based	Vanilla REINFORCE	[41–50]
	REINFORCE uses graph structure/input	[51–54]
	Non-REINFORCE-based	[55,56]
Hybrid	Vanilla DDPG	[25,57–60]
	With Knowledge Graph	[2,61–64,64–67]

Value based methods. As mentioned, Deep Q-learning and its variants are typical value-based DRL methods widely used in DRL-based RS. DRN [24] is the first work utilizing Deep Q-Networks (DQN) in RS. It adopts Double DQN (DDQN) [68] to build a user profile and designs an activeness score to reflect how frequently a user returns after one recommendation plus users' action (click or not) as the reward. DRN provides a new approach to integrating DRL into RS when dealing with a dynamic environment, with the objective function defined as follows,

$$\mathbb{E}[r_{t+1} + \gamma Q_{\theta'_t}(s_{t+1}, \underset{a'}{\operatorname{argmax}}\, Q_{\theta_t}(s_t, a'))] \tag{20}$$

where a' is the action that gives the maximum future reward according to θ_t, θ_t and θ'_t are different parameters for two different DQNs.

Zhao et al. [25] points out that negative feedback also affect recommendation performance, which is, however, not considered by DRN. Positive feedback is usually sparse due to the large number of candidate items in RS. Hence, only using positive feedback would lead to convergence problems. In this regard, DEERS is proposed to consider both positive and negative feedback simultaneously by using DQN, and Gated Recurrent Units (GRU) are employed to capture users' preferences for both a positive state s^+ and negative state s^-. The final objective function can be computed as follows:

$$\mathbb{E}[r_{t+1} + \gamma \max_{a_{t+1}} Q_{\theta_q}(s^+_{t+1}, s^-_{t+1}, a_{t+1}) | s^+_t, s^-_t, a_t]. \tag{21}$$

Lei et al. [26] introduces attention mechanisms into the DQN to leverage social influence among users. Specifically, a social impact representation U_v is introduced for state representation. Matrix factorization is adopted to determine the similarity among users and hence present the social influence. Social attention is introduced to distill the final state representation. In addition, a few studies focus on user profiling to improve recommendation performance [27–29]. Lei and Li [28] claims that user feedback contains useful information in the previous feedback even when the user does not like the recommended items. Some studies focus on final feedback without considering the influence of the freshness of the feedback. User-specific DQN (UQDN) is proposed to consider multi-step feedback from users. It employs Matrix Factorization to generate user-specific latent state spaces and defines the objective function with the user-specific latent state space as follows:

$$\mathbb{E}[r_{t+1} + \gamma \max_{a_{t+1}} \overline{Q}_{\theta_q}(s_{t+1}, a_{t+1}) + \overline{\mathbf{b}}_u - Q_{\theta_q}(s_{t+1}, a_{t+1})] \tag{22}$$

where $\overline{\mathbf{b}}_u$ is the learned user latent representation.

Zou et al. [29] points out that most studies do not consider users' long-term engagement in the state representation as they focus on the immediate reward. FeedRec is proposed that combines both instant feedback and delayed feedback into the model to represent the long-term reward and optimize the long-term engagement by using DQN. Specifically, time-LSTM is employed to track users' hierarchical behavior over time to

represent the delayed feedback which contains three different operations: $h_{skip}, h_{choose}, h_{order}$. The state space is the concatenation of those operations and users' latent representation. Xiao et al. [27] focuses on the user privacy issue in recommender systems. Deep user profile perturbation (DUPP) is proposed to add perturbation into the user profile by using DQN during the recommendation process. Specifically, DUPP adds a perturbation vector into users' clicked items as well as the state space, which contains users' previous behavior.

Distinct from previous studies, which focus on optimizing user profiles or state spaces, some studies aim to optimize the action space formed by interactions with items. In the situation of basket recommendation, the user is suggested multiple items as a bundle, which is called a recommendation slate. It leads to combinatorially large action spaces making it intractable for DQN-based recommendation models. SlateQ [30] is proposed to decompose slate Q-value to estimate a long-term value for individual items, and it is represented as,

$$Q_{\theta_q}(s_t, a_t) = \sum_{i \in a_t} p(i|s_t, a_t)\overline{Q}_{\theta_q}(s_t, i) \tag{23}$$

where $\overline{Q}_\theta(s, i)$ is the decomposed Q-value for item i. The decomposed Q-value will be updated by the rule similar to traditional DQN:

$$\overline{Q}_{\theta_q}(s_t, i) \leftarrow \alpha\left(r_t + \gamma \sum_{j \in a_{t+1}} p(j|s_{t+1}, a_{t+1})\overline{Q}_{\theta_q}(s_{t+1}, j)\right) + (1-\alpha)\overline{Q}_{\theta_q}(s_t, i). \tag{24}$$

Different from other mode-free methods, Slate-Q assumes that the transition probability $p(i|s_t, a_t)$ is known.

Vanilla DQN methods may not have sufficient knowledge to handle complex data such as images and graphs. Tang and Wang [69] firstly models users' click behavior as an embedding matrix in the latent space to include the skip behaviors of sequence patterns for sequential recommendation. Based on that, Gao et al. [36] propose DRCGR, which adopts CNN and GAN into DQN to help the agent to better understand high-dimensional data, e.g., a matrix. Two different convolution kernels are used to capture users' positive feedback. In the meantime, DRCGR uses GANs to learn a negative feedback representation to improve robustness. Another typical data format is the graph, which is widely used in RS, including knowledge graphs. Lei et al. [31] propose GCQN, which adopts Graph Convolutional Networks (GCN) [70] into the DQN and constructs the state and action space as a graph-aware representation. Differently, GCQN introduces the attention aggregator: $\sum_{w \in \mathcal{N}(i)} \alpha_{iu} e_u$ which demonstrates better performance than the mean-aggregator and pooling-aggregator. For item i, the graph-aware representation can be represented as,

$$\sigma\left(W_{fc}[e_i \oplus \sum_{w \in \mathcal{N}(i)} \alpha_{iu} e_u + b_{fc}]\right) \tag{25}$$

where W_{fc}, b_{fc} are the parameters for the fully-connected layer, e_u is the embedding for user u and $\mathcal{N}(i)$ is the set of one-hot neighbors of item i in graph $G(i)$. Zhou et al. [37] propose KGQR uses a similar strategy to transform the information into a knowledge graph which is fed into the GCN to generate the state representation. Notably, KGQR presents a different state representation generation method. For given node i, the neighborhood representation with a k-hop neighborhood aggregator can be represented as,

$$e_i^k = \sigma\left(W_k \frac{1}{|\mathcal{N}(i)|} \sum_{t \in \mathcal{N}(i)} e_t^{k-1} + B_k e_i^{k-1}\right) \tag{26}$$

where $\mathcal{N}(i)$ is the set of neighboring nodes, W_k, B_k are the parameter of the aggregator. Those neighborhood representations will be fed into a GRU, and the state representation will be generated. Another application domain for using graph data is job recommendation which requires considering multiple factors jointly, such as salary, job description, job location etc. SRDQN [71] constructs a probability graph to represent a candidate's skill set and employs a multiple-task DQN structure to process these different factors concurrently.

Some studies target recommendation and advertising simultaneously in e-commerce environments [38–40]. Pei et al. [38] mentions when deploying RS into real-world platforms such as e-commerce scenarios, the expectation is to improve the system's profit. A new metric, Gross Merchandise Volume (GMV), is proposed to measure the profitability of the RS to provide a new view of evaluating RS in advertising. Different from GMV, Zhao et al. [39] separates recommendation and advertising as two different tasks and proposes the Rec/Ads Mixed display (RAM) framework. RAM designs two agents: a recommendation agent and an advertising agent, where each agent employs a CDQN to conduct the corresponding task. Zhao et al. [40] find that advertising and recommendation may harm each other and formulate a rec/ads trade-off. Their proposed solution, DEARS, contains two RNNs. Two RNNs are employed to capture user preferences toward recommendations and ads separately. Based on that, DQN is employed to take those two outputs as the input to construct the state and output the advertising.

Policy-based methods. Policy-based DRL can be divided into two parts: Policy Optimization (CPO) [72] and policy gradient. Zhang et al. [56] uses CPO to identify the contradiction between text feedback and historical preferences. It provides a solution for using DRL in the situation where users' feedback is entirely different from previous feedback in RS. Policy gradient-based methods are the other stream in policy-based DRL methods for RS. These methods aim to optimize the policy π directly instead of estimating the Q-value like DQN. A well-known and widely used policy gradient method in RS is REINFORCE, which uses the following rule for policy π_{θ_π}:

$$\theta \leftarrow \theta + \alpha \mathbb{E}_{\tau \sim d_{\pi_{\theta_\pi}}}\left[\sum_{t=1}^{T} r(s_t^i, a_t^i) \sum_{t=1}^{T} \nabla_{\theta_\pi} \log \pi_{\theta_\pi}(s_t^i, a_t^i)\right] \tag{27}$$

where i is sampled trajectories from $\pi_\theta(a_t|s_t)$. Pan et al. [41] propose Policy Gradient for Contextual Recommendation (PGCR), which adopts REINFORCE and considers contextual information. PGCR assumes that the policy follows the multinoulli distribution, in which case the transition probability can be estimated easily through sampling from previously seen context. Wang et al. [42] incorporate CNNs and attention mechanisms in REINFORCE for explainable recommendation. Specifically, this work designs a coupled agent structure where one agent generates the explanation and the other makes recommendations based on the generated explanation. Chen et al. [43] increases the scalability of REINFORCE to ensure it can deal with the extremely large action space under recommendation scenarios. Specifically, it introduces a policy correction gradient estimator into REINFORCE to reduce the variance of each gradient by doing importance sampling. The new update rule becomes

$$\theta_\pi \leftarrow \theta_\pi + \alpha \sum_{\tau \sim \beta}\left[\sum_{t=1}^{T} \frac{\pi_{\theta_\pi}(s_t, a_t)}{\pi_\beta(s_t, a_t)} r(s_t^i, a_t^i) \sum_{t=1}^{T} \nabla_{\theta_\pi} \log \pi_{\theta_\pi}(s_t^i, a_t^i)\right] \tag{28}$$

where π_β is the behavior policy trained by state–action pairs without the long-term reward and π_θ is trained based on the

long-term reward only. It is worth mentioning that the vanilla REINFORCE algorithm is on-policy, and importance sampling will make REINFORCE behave like an off-policy method with the following gradient format,

$$\mathbb{E}_{\tau\sim d_{\pi_\theta}}\left[\prod_{t'=1}^{t}\frac{\pi_\theta(s_t,a_t)}{\pi_{\theta_s}(s_{t'},a_{t'})}\sum_{t'=t}^{T}r(s_t,a_t)\sum_{t=1}^{T}\nabla_\theta\log\pi_\theta(s_t,a_t)\right] \tag{29}$$

where π_{θ_s} is the sample policy parameter. Xu et al. [44] finds that the REINFORCE method suffers from a high variance gradient problem and proposes Pairwise Policy Gradient (PPG). Different from policy correction, PPG uses Monte Carlo sampling to sample two different actions a, b and compare the gradient to update θ,

$$\mathbb{E}_{\tau\sim d_{\pi_{\theta_\pi}}}\Bigg(\sum_a\sum_b(r(s,a)-r(s,b))\sum_{t=1}^{T}(\nabla_{\theta_\pi}\log_{\pi_{\theta_\pi}}(s_t,a_t) - \nabla_{\theta_\pi}\log_{\pi_{\theta_\pi}}(s_t,b_t))\Bigg). \tag{30}$$

Ma et al. [45] extends the policy correction gradient estimator into a two-stage setting which are $p(s_t, a^p)$ and $q(s_t, a|a^p)$ and the policy can be written as

$$\sum_{a^p}p(s_t,a^p)q(s_t,a|a^p). \tag{31}$$

In addition, weight capping and self-normalized importance sampling are used to further reduce the variance. Moreover, a large state space and action space will cause the sample inefficiency problems as REINFORCE relies on the current sampled trajectories τ. Chen et al. [46] finds that the auxiliary loss can help improve the sample efficiency [73,74]. Specifically, a linear projection is applied to the state s_t, and the output is combined with action a_t to calculate the auxiliary loss and appended into the final overall objective function for optimization. Moreover, Wang et al. [75] attempt to use different metrics to measure users' long-term goal and design a reward surrogate mechanism to enhance the capability of REINFORCE-based methods.

Another prototype of vanilla policy gradient in DRL-based RS is the policy network. Montazeralghaem et al. [47] designs a policy network to extract features and represent the relevant feedback that can help the agent make a decision. Similar to DQN, this work uses a neural network to approximate the Q-value and the policy directly without theoretical analysis. Ji et al. [48] extend the policy network by introducing spatio-temporal feature fusion to help the agent understand complex features. Specifically, it considers both the current number and the future number of vacant taxis on the route to recommend routes for taxis. Yu et al. [49] introduces multi-modal data as new features to conduct vision-language recommendation by using historical data to train REINFORCE. ResNet and attention are used to encode vision and text information, respectively. Moreover, two rewards are introduced with a customized ratio λ to balance vision and text information. Li et al. [50] find that the REINFORCE algorithm can also be used to determine or generate loss functions for recommender systems. It designs a reward filtering mechanism for customized loss function generation suitable to different recommender systems and datasets. It adopts the REINFORCE algorithm to optimize the parameter in neural architecture search (NAS) algorithms to find the suitable loss function, demonstrating enhancement of generalization capability.

Knowledge Graphs (KG) are widely used in RS to enrich side information, provide explainability and improve recommendation performance. Similar to DQN, vanilla REINFORCE cannot properly handle graph-like data. Wang et al. [51] propose a method named Knowledge-guided Reinforcement Learning (KERL), which integrates knowledge graphs into the REINFORCE algorithm. Specifically, KERL adopts TransE [76] to transfer the knowledge graph into a graph embedding and utilizes a multilayer perceptron (MLP) to predict future knowledge of user preferences. The state representation can be written as follows:

$$h_t \oplus TransE(\mathcal{G}) \oplus MLP(TransE(\mathcal{G})) \tag{32}$$

where h_t is the hidden representation from the GRU for sequential behavior and $\mathcal{G}$ is the knowledge graph.

Different from KERL, Xian et al. [52] propose Policy-Guided Path Reasoning (PGPR), which formulates the whole environment as a knowledge graph. The agent is trained to find the policy to find good items conditioned on the starting user in the KG by using REINFORCE. PGPR uses the tuple (u, e_t, h_t) to represent the state instead of the graph embedding where e_t is the entity the agent has reached at t for user u and h_t is the previous action before t. The action in PGPR is defined as the prediction of all outgoing edges for e_t based on h_t. Wang et al. [53] propose a knowledge graph policy network (KGPolicy) which puts the KG into the policy network and adopts REINFORCE to optimize it. In addition, KGPolicy uses negative sampling instead of stochastic sampling to overcome the false negative issue—sampled items behave differently during training and inference. Similar to GCQN, attention is also employed to establish the representation of its neighbors.

Due to the on-policy nature of REINFORCE, it is difficult to apply it to large-scale RS as the convergence speed will be a key issue. To relieve this, Chen et al. [54] propose TPGR, which designs a tree-structured policy gradient method to handle the large discrete action space hierarchically. TPGR uses balanced hierarchical clustering to construct a clustering tree. Specifically, it splits large-scale data into several levels and maintains multiple policy networks for each level to conduct the recommendation. The results are integrated at the final stage. As mentioned, policy gradient can be further extended to deterministic policy gradient (DPG) [77]. Hu et al. [55] propose Deterministic Policy Gradient with Full Backup Estimation (DPG-FBE) to complete a sub-task of recommendation. DPG-FBE considers a search session MDP (SSMDP) that contains a limited number of samples, where the stochastic policy gradient method like REINFORCE cannot work well.

Hybrid methods. The most common model-free hybrid method used would be the actor–critic algorithm, where the critic network uses the DQN and the actor uses the policy gradient. The common algorithm used to train actor–critic is DDPG with the following objective function,

$$\mathbb{E}[r_t+\gamma Q_{\theta_q'}(s_{t+1},\mu_{\theta_\pi'}(s_{t+1}))-Q_{\theta_q}(s_t,a_t)] \tag{33}$$

where θ_q, θ_q' is the parameter for Q-learning at time $t, t+1$ while θ_π' is the parameter for deterministic policy gradient at time $t+1$. Zhao et al. [57] propose LIRD, which uses the vanilla actor–critic framework to conduct list-wise recommendations. To demonstrate the effectiveness of LIRD, a pre-trained user simulator is used to evaluate the effectiveness of LIRD, where the transition probability is approximated using the cosine similarity for a given state–action pair s_t, a_t. Zhao et al. [25] further extend LIRD into page-wise recommendation and propose DeepPage. Similar to other previous work, GRU is employed to process the sequential pattern. Moreover, similar to DRCGR, DeepPage formulates the state as a page, then CNNs are employed to capture features and fed to the critic network. The final state representation concatenate the sequential pattern and the page features. Besides, some studies focus on different scenarios such as top-aware recommendation [58], treatment recommendation [59], allocating impressions [60] etc. Liu et al. [58] introduces a supervised learning module (SLC) as the indicator to identify the difference between the current policy and historical preferences. SLC will

conduct the ranking process to ensure the recommendation policy will not be affected by positional bias – the item appearing on top receives more clicks. Similarly, Wang et al. [59] also integrates the supervised learning paradigm into DRL but in a different way. An expert action $\hat{a}_t$ is provided when the critic evaluates the policy. The update rule is slightly different than normal DQN,

$$\theta_q \leftarrow \theta_q + \alpha \sum_t [Q_{\theta_q}(s_t, \hat{a}_t) - r_t - \gamma Q_{\theta_{q'}}(s_t, \mu_{\theta_{\pi'}}(s_t))] \nabla_{\theta_q} Q_{\theta_q}(s_t, a_t). \tag{34}$$

However, such a method is not universal as the acquisition of expert action is difficult and depends on the application domain.

Similar to policy gradient and DQN, Knowledge Graphs (KG) are also used in actor-critic-based methods. Chen et al. [2] propose KGRL to incorporate the substantial information of knowledge graphs to help the critic to evaluate the generated policy better. A knowledge graph is embedded into the critic network. Unlike previous studies that use the KG as the environment or state representation, KGRL uses KG as a component in the critic, which can guide the actor to find a better recommendation policy by measuring the proximity from the optimal path. Specifically, a graph convolutional network is used to weight the graph and the Dijkstra's algorithm is employed to find the optimal path for finally identifying the corresponding Q-value. Zhao et al. [61] claim that human's demonstration could improve path searching and propose ADAC. ADAC also searches for the optimal path in the KG but further adopts adversarial imitation learning and uses expert paths to facilitate the search process. Feng et al. [62] propose MA-RDPG, which extends the standard actor–critic algorithm to deal with multiple scenarios by utilizing a multi-actor reinforcement learning setting. Specifically, two different actor networks are initialized while only one critic network will make the final decision. Those two actor networks can communicate with each other to share information and approximate the global state. Zhang et al. [63] find multiple factors can affect the selection of electric charging stations. Hence, it uses a similar idea to recommend the electric vehicle charging station by considering current supply, future supply, and future demand. He et al. [64] figure out that the communication mechanism in MA-RDPG will harm actors as they are dealing with independent modules, and there is no intersection. Hence, He et al. [64] extend MA-RDPG into multi-agent settings which contain multiple pairs of actors and critics and remove the communication mechanism to ensure independence.

Different from [62], He et al. [64] use 'soft' actor–critic (SAC) [65], which introduces a maximum entropy term $\mathcal{H}(\pi(s_t, \phi_t))$ to actor–critic to improve exploration and stability with the stochastic policy $\pi(s_t, \phi_t)$. Similar to the multi-agent idea, Zhao et al. [66] use a hierarchical setting to help the agent learn multiple goals by setting multiple actors and critics. In comparison, hierarchical RL uses multiple actor–critic networks for the same task. It splits a recommendation task into two sub-tasks: discovering long-term behavior and capturing short-term behavior. The final recommendation policy is the combination of the optimal policies for the two sub-tasks. Similarly, Xie et al. [67] use the hierarchical setting for integrated recommendation by using different sourced data. The objective is to determine the sub-polices for each source hierarchically and form the final recommendation policy afterward.

Discussion. In RS, model-free methods are generally more flexible than model-based methods as they do not require knowing the transition probability. We summarize the advantages and disadvantages of the three types of methods described under the model-free category. DQN is the first DRL method used in RS, which is suitable for small discrete action spaces. The problems with DQN in RS are:

- RS normally contains large and high-dimensional action spaces.
- The reward function is hard to determine, leading to inaccurate value function approximation.

Specifically, the high dimensional action space in the context of recommender systems is recognized as a major drawback of DQN [9,78]. The reason lies in the large number of candidate items. Hence, DQN, as one of the most popular schemes, is not the best choice for RS in many situations. Moreover, some unique factors need to be considered when designing the reward function for RS, such as social inference. It introduces extra parameters to the Q-network and hinders convergence. Policy gradient does not require the reward function to estimate the value function. Instead, it estimates the policy directly. However, policy gradient is designed for continuous action spaces. More importantly, it will introduce high variance in the gradient. Actor–critic algorithms combine the advantages of DQN and policy gradient. Nonetheless, actor–critic will map the large discrete action space into a small continuous action space to ensure it is differentiable, which may cause potential information loss. Actor–critic uses DDPG and thus inherits disadvantages from DQN and DPG, including difficulty in determining the reward function and poor exploration ability.

Table 4
List of publications reviewed in this section.

Component	Work
Environment	[79–86]
State	[87–90]
Reward	[91]

3.3. Component optimization in deep reinforcement learning-based RS

There are a few studies that use DRL in RS for goals other than improving recommendation performance or proposing new application domains. We split the literature based on the following components: environment, state representation, and reward function. Existing studies usually focus on optimizing one single component in the DRL setting (as illustrated in Fig. 1) (see Table 4).

3.3.1. Environment simulation and reconstruction

Many environments are available for evaluating deep reinforcement learning. Two popular ones are OpenAI gym-based environment [92] and MuJoCo.[1] Unfortunately, there is no standardized simulation platform or benchmark specific to reinforcement learning-based recommender systems. Existing work on DRL in RS is usually evaluated through offline datasets or deployment in real applications. The drawback of evaluating offline datasets include:

- Existing studies use different environment construction methods, leading to unfair comparison. For instance, some studies use the KG as the environment, while some studies assume the environment is gym-like or design a simulator for specific tasks.
- With offline datasets, users' dynamic interest, and environment dynamics are hard to maintain. Deploying the method into a real application is difficult for academic research as it takes time and costs money. Hence, a standardized simulation environment is a desirable solution (see Fig. 6).

[1] http://mujoco.org/.

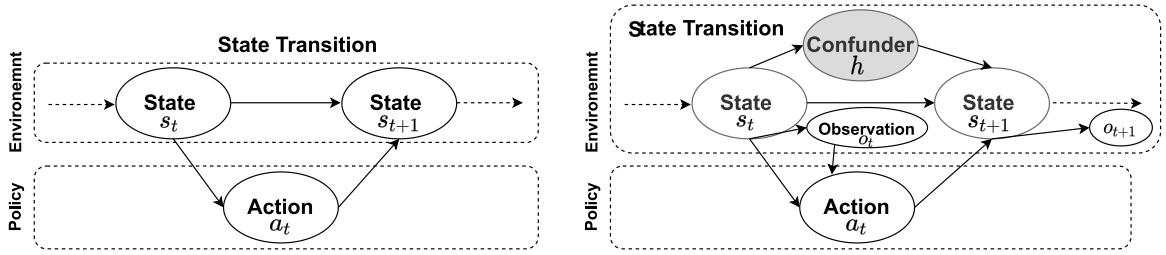


Fig. 6. Left is the traditional MDP transition; Right is the POMDP which considers the environmental confounders such as social influence [83].

Several studies provide standardized gym[2]-based simulation platforms for DRL-based RS research in different scenarios. RecSim [82] is a configurable platform that supports sequential interaction between the system and users. RecSim contains three tasks: interest evolution, interest exploration and long-term satisfaction. RecoGym [80] provides an environment for recommendation and advertising. In addition, RecoGym also provides simulation of online experiments such as A/B-tests. However, RecSim and RecoGym are designed for bandit behavior, which means users' interest does not change over time. VirtualTB [81] is proposed to relieve such problems. VirtualTB employs imitation learning to learn a user model to interact with the agent. GANs are employed to generate users' interests. Similar to VirtualTB, Recsimu [86] uses a GAN to tackle the complex item distribution. In addition, PyRecGym [79] accommodates standard benchmark datasets in a gym-based environment. MARS-Gym [85] provides a benchmark framework for marketplace recommendation. [84] suggests that existing simulation environments are biased because of biased logged data. Two common biases are discussed: popularity bias and positivity bias. To reduce the effect of those biases, SOFA introduces an Intermediate Bias Mitigation Step for debiasing purposes.

One work discusses environment reconstruction by considering confounders [83]. claims that users' interests may be affected by social networks, which may introduce extra bias to the state and affect the decision-making process. A multi-agent setting is introduced to treat the environment as an agent, which can partially relieve the hidden confounder effect. Specifically, a deconfounded environment reconstruction method DEMER is proposed. Different from the aforementioned methods, DEMER assumes the environment is partially observed and models the whole recommendation task as a Partially Observed MDP (POMDP). Different from an MDP, a POMDP contains one more component observation $o \in \mathcal{O}$ and the action a_t is derived based on the observation o_t instead of the state s_t by $a_t = \pi_a(o_t)$(see Fig. 6). DEMER assumes there is a confounder policy π_h for observation o_h which is composed by a_t and o_t: $a_h = \pi_h(a_t, o_t)$. Moreover, another observation o_b is introduced to observe the transition as well. π_b is the corresponding policy and $a_b = \pi_b(o_b) = \pi_b(o_t, a_t, a_h)$. DEMER uses generative adversarial imitation learning (GAIL) to imitate the policy A, B. Given trajectory $\{o_t, o_h, o_b\}$ for different policies A and B, the objective function is defined as

$$(\pi_a, \pi_b, \pi_h) = \underset{(\pi_a,\pi_b,\pi_h)}{\operatorname{argmin}} \mathbb{E}_{s\sim\tau}(L(s, a_t, a_b))$$
$$\text{where } L(s, a_t, a_b) = \mathbb{E}_{(\pi_a,\pi_b,\pi_h)}[\log D(s, a_t, a_b)] - \lambda \sum_{\pi\in\{\pi_a,\pi_b,\pi_h\}} H(\pi) \tag{35}$$

where $L(\cdot)$ is the loss function, $D(\cdot)$ is a discriminator and $H(\pi)$ is introduced in GAIL.

[2] https://gym.openai.com/.

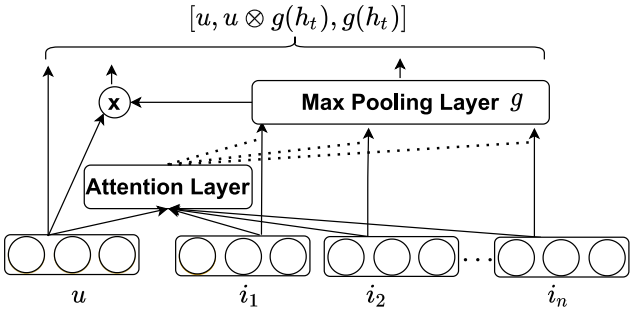


Fig. 7. State representation used in works [88,89]. h_t is the output of an attention layer that takes the representation of users' history at time t as input and $g(\cdot)$ is the pooling operation.

3.3.2. State representation

State representation is another component in DRL-based RS which exists in both model-based and model-free methods. Liu et al. [87] find that the state representation in model-free methods would affect recommendation performance. Existing studies usually directly use the embedding as the state representation. Liu et al. [88,89] propose a supervised learning method to generate a better state representation by utilizing an attention mechanism and a pooling operation as shown in Fig. 7. Such a representation method requires training a representation network when training the main policy network, which increases the model complexity. Huang et al. [90] further study that the attention-based state encoder can boost the performance of those DQN-based methods in RS. It validates that integrating attention mechanism can learn a better state representation when designing the recommendation environment.

3.3.3. Robustness of reward functions

The reward function is crucial for methods involving DQN. A robust reward function can significantly improve training efficiency and performance. Kostrikov et al. [93] find that the DQN may not receive the correct reward value when entering the absorbing state. That is, when the absorbing state is reached, the agent will receive a zero reward and harm policy learning. The reason behind this is that when designing the environment, zero reward is implicitly assigned to the absorbing state as it is hard to determine the reward value in such a state. Chen et al. [91] propose a robust DQN method, which can stabilize the reward value when facing the absorbing state. The new reward formula can improve the robustness, which is defined as follows:

$$r = \begin{cases} r_t & \text{if } s_{t+1} \text{ is an absorbing state} \\ r_t + \gamma Q_{\theta'}(s_{t+1}, a_{t+1}) & \text{otherwise.} \end{cases} \tag{36}$$

The major difference is that r_t is assigned to the absorbing state to ensure the agent can continue learning. One remaining problem in current DRL-based RS is the reward sparsity, i.e., the large state and action spaces deteriorate the reward sparsity problem. One

possible solution would be a better-designed reward by using the reward shaping [94].

4. Emerging topics

While existing studies have established a solid foundation for DRL-based RS research, this section outlines several promising emerging research directions.

4.1. Multi-agent and hierarchical deep reinforcement learning-based RS

Recommender systems are monolithic systems containing tasks such as searching, ranking, recommendation, advertising, personalization, and diverse stakeholders such as users and items. Most existing methods are based on a single agent. Multi-Agent Reinforcement Learning (MARL) is a subfield of reinforcement learning and it is capable of learning multiple policies and strategies. While a single-agent reinforcement learning framework can only handle a single task, many studies consider the multi-task situation in RS and employ multi-agent DRL (MADRL) or hierarchical DRL (HDRL). HDRL [95] is proposed to handle complex tasks by splitting such tasks into several small components. It let the agent determine sub-policies. HDRL belongs to a single-agent reinforcement learning framework such that the agent contains a meta-controller and several controllers. The meta-controller splits the task, and the controllers learn the value and reward functions for designated tasks to get a series of sub-policies. There are a few studies already utilizing HDRL in RS. Xie et al. [67] target integrated recommendation to capture user preferences on both heterogeneous items and recommendation channels. Specifically, the meta-controller is used for item recommendation, and controllers aim to find the personalized channel according to user channel-level preferences. Zhang et al. [96] uses HDRL for course recommendation in MOOCs, which contains two different tasks: profile reviser and recommendation. The meta-controller aims to make course recommendations by using the revised profile pruned by the controllers.

Different from HDRL, MADRL [97] introduces multiple agents to handle the sub-tasks. Gui et al. [32] uses the MADRL for twitter mention recommendation where three agents are initialized. The three agents need to generate different representations for the following tasks: query text, historical text from authors and historical text from candidate users. Once the representations are finalized, the model will conduct the recommendation based on the concatenation of representations. Feng et al. [62] and He et al. [64] provide two different views of the communication mechanism in MADRL and demonstrate that agents could work collaboratively or individually. Zhao et al. [39] designs a MADRL framework for two tasks where two agents are designed to conduct advertising and recommendation respectively. Zhang et al. [14] uses MADRL for collaborative recommendation where each agent is responsible for a single user. MADRL is adopted to help the recommender consider both collaboration and potential competition between users. Zhang et al. [63] designs a charging recommender system for intelligent electric vehicles by using decentralized agents to handle sub-tasks and a centralized critic to make the final decision.

Hierarchical multi-agent RL (HMARL) [98] proves that MARL and HRL can be combined. Recently, Yang et al. [99] introduces HMADRL into the continuous action space, which provides a direction for RS. Zhao et al. [66] uses HMARL for multi-goal recommendation where the meta-controller considers users' long-term preferences and controllers focus on short-term click behavior. While the meta-controller and controllers in HDRL deal with sub-tasks that belong to a single task, HMARL focuses on multi-task or multi-goal learning where the meta-controller and controllers belong to different agents and deal with different tasks or goals. HMADRL would be a suitable solution for future research work in DRL-based RS where HDRL can be used to split a complex task into several sub-tasks such as users' long-term interests and short-term click behavior, and MADRL can jointly consider multiple factors such as advertising Zhao et al. [66].

4.2. Inverse deep reinforcement learning for RS

As mentioned, the reward function plays a critical role in DRL-based recommender systems. In many existing works, reward functions are manually designed. The common method uses users' click behavior to represent the reward and to reflect users' interests. However, such a setting can not represent users' long-term goals [29] as clicking or not only depicts part of the feedback information from users. It requires significant effort to design a reward function, due to the large number of factors that can affect users' decision, such as social engagement or bad product reviews, which may adversely affect recommendation performance. It is difficult to include all potential factors into the reward function because not every factor can be represented properly. A few works [100,101] show that manually designed reward functions can be omitted by employing inverse reinforcement learning (IRL) [102] or generative adversarial imitation learning (GAIL) [103]. Such inverse DRL-based methods require using expert demonstration as the ground truth. However, expert demonstration is often hard to obtain for recommendation scenarios. Those two studies conduct experiments in an offline dataset-based simulation environment that can access expert demonstration. In contrast, Chen et al. [101] use IRL as the main algorithm to train the agent while Gong et al. [100] use both demonstration and reward to train the agent. Zhao et al. [61] also employ GAIL to improve recommendation performance. In this work, GAIL is used to learn the reasoning path inside the KG to provide side information to help the agent learn the policy. Although IRL achieves some progress in RS, the lack of demonstration is a key shortcoming that impedes adoption in RS. One possibility is to use the IRL method in casual reasoning to help improve interpretability [104], thus boosting recommendation performance. Alternately, IRL may be suitable for learning users' long-term and static behavior to support the reward function.

Differently, Hu et al. [105] use the IRL as the recommender to make recommendation in multi-round conversational recommender systems. It mainly relies on a system-ask and user-respond procedure the train the model. Moreover, different from mentioned algorithms, Hu et al. [105] uses several pre-defined rules to supervise the learning process and provide the reward to the agent.

4.3. Graph neural networks for boosting DRL-based RS

Graph data and KG are widely used in RS. Graph modeling enables an RS to leverage interactions between users and the recommender for reasoning or improving interpretability. According to existing studies about deep learning-based RS [3], embedding is a technique used to get the representation for the input data. Graph embedding is a common solution to handle graph-like data. GCN is a type of graph embedding method which are broadly used in RS to process graph data. Wang et al. [106] propose a variant of GCN to learn the embedding for KG. Specifically, they propose knowledge graph convolutional networks (KGCN) to capture the high-order structural proximity among entities in a knowledge graph.

In DRL-based RS, graph data are handled similarly—the whole graph will be transformed into an embedding and fed to the

agent. Wang et al. [51] uses a traditional graph embedding method TransE [76] to generate the state representation for DRL-based RS. There are several studies that use GCN in DRL for recommendations under different settings. Jiang et al. [107] propose a graph convolutional RL (DGN) method which integrates the GCN into the Q-learning framework for general RL problems by replacing the state encoding layer with the GCN layer. Lei et al. [31] extend this method into the deep learning field and apply it to recommender systems. Specifically, multiple GCN layers are employed to process the sub-graphs for a given item i. Chen et al. [2] employs KG inside the actor–critic algorithm to help the agent learn the policy. Specifically, the critic network contains a GCN layer to give weight to the graph and conduct searches in the graph to find an optimal path and hence guide the optimization of policy learning. However, such a method is relatively computationally expensive as it requires jointly training the GCN and the actor–critic network. Gong et al. [100] adopts a Graph Attention Network (GAT) [108] into the actor–critic network to conduct recommendation. In addition, the GAT is used as an encoder to obtain a state representation. A common way of using GCN or its variants in DRL-based RS is the state encoder. The related challenge is the difficulty for the environment to provide a graph-like input to the GCN.

4.4. Self-supervised DRL-based RS

Self-supervised learning (SSL) is a technique in which the model is trained by itself without external label information. SSL-DRL is receiving growing interest in robotics [109,110]. Kahn et al. [109] shows that SSL can be used to learn the policy when doing navigation by providing real-world experience. Zeng et al. [110] demonstrates that SSL-DRL can be used to help the agent learn synergies between two similar policies, thus empowering the agent to conduct two different tasks. Recent advances in SSL RL show that SSL can also provide interpretability for RL, which is promising for interpretable RS research [111]. Shi et al. [111] shows that SSL based RL can highlight the task-relevant information to guide the agent's behavior. Moreover, Xin et al. [112] shows that SSL can be used to provide negative feedback for DRL-based RS to improve recommendation performance. Specifically, a self-supervised loss function is appended into the normal DRL loss function,

$$-\sum_{i=1}^{n} Y_i \log\left(\frac{e^{y_i}}{\sum_{i'=1}^{n} e^{y_{i'}}}\right) + L_{DRL} \tag{37}$$

where Y_i is an indicator function to show users interact with the item i or not. L_{DRL} could vary, if the DQN is adopted, Eq. (4) should be used. If other RL algorithms are adopted, L_{DRL} can be changed to the RL algorithms that are used. SSL demonstrates promising performance in visual representation in recent years, which would be a possible solution to generate the state representation as there are a few DRL-based RS studies that adopt CNNs to process image-like data and transform it into a state [36,58]. Furthermore, as an unsupervised learning approach, SSL would provide a new direction about defining the reward function by learning common patterns between different states as well as multi-task learning.

5. Open questions

In this section, we outline several open questions and challenges that exist in DRL-based RS research. We believe these issues could be critical for the future development of DRL-based RS.

5.1. Sample efficiency

Sample inefficiency is a well-known challenge in model-free DRL methods. Model-free DRL requires a significant number of samples as there is no guarantee that the received state is useful. Normally, after a substantial number of episodes, the agent may start learning as the agent finally receives a useful state and reward signal. A common solution is the experience replay technique, which only works in off-policy methods. Experience replay still suffers the sample inefficiency problem [113] as not every past experience is worth replaying. Isele and Cosgun [114] propose selected experience replay (SER) that only stores valuable experience into the replay buffer and thus improves sample efficiency. While traditional DRL environments only contain several candidate items, in DRL-based RS, the agent must deal with a significantly larger action space as RS may contain lots of candidate items. Existing DRL-based RS studies on traditional experience replay methods often demonstrate slow converge speed. Chen et al. [46] design a user model to improve the sample efficiency through auxiliary learning. Specifically, they apply the auxiliary loss with the state representation, and the model distinguishes low-activity users and asks the agent to update the recommendation policy based on high-activity users more frequently. Wu et al. [115] find that in cross-domain recommendation, those abundant logged interaction data from a source domain can be used to improve the recommendation quality in the target domain. It is achieved by using the reward correlation between two domains. Moreover, Chen et al. [116] design a new experience replay method for DRL based RS to improve the sample efficiency by prioritizing the sampling and storing procedure to those more valuable experience.

On the other hand, model-based methods are more sample efficient. However, they introduce extra complexity as the agent is required to learn the environment model as well as the policy. Due to the extremely large action space and possibly large state space (depending on users' contextual information) in RS, approximating the environment model and policy simultaneously becomes challenging.

5.2. Exploration and exploitation

The exploration and exploitation dilemma is a fundamental and challenging problem in reinforcement learning research and receives lots of attention in DRL. This dilemma describes a trade-off between obtaining new knowledge and the need to use that knowledge to improve performance. Many DQN-based methods focus on exploration before the replay buffer is full and exploitation afterward. Consequently, it requires an extremely large replay buffer to allow all possibilities in recommendation can be stored. DRN employs Dueling Bandit Gradient Descent (DBGD) [117] to encourage exploration while [15,64] introduces a regularization or entropy term into the objective function to do so. [30] uses the sheer size of the action space to ensure sufficient exploration. [51–53] uses a separate KG or elaborated graph exploration operation to conduct exploration. [43] employs Boltzmann exploration to get the benefit of exploratory data without negatively impacting user experience. In addition, ϵ-greedy is the most common technique used to encourage exploration [16,17,26,28,29,31,58,60,67]. Remaining studies rely on a simulator to conduct exploration. However, it may suffer from noise and overfitting [67] because of the gap between simulation and real online application. For most DRL-based methods such as vanilla DQN, policy gradient, or actor-critic-based methods, ϵ-greedy would be a good choice for exploration. In addition, injecting noise into the action space would also be helpful for those actor-critic-based methods [13]. For methods involving KGs, ϵ-greedy may help, but the elaborated graph exploration methods may receive better performance.

5.3. Generalizing from simulation to real-world recommendation

Existing work generally trains DRL algorithms in simulation environments or offline datasets. Deploying DRL algorithms into real applications is challenging due to the gap between simulation and real-world applications. Simulation environments do not contain domain knowledge or social impact. They cannot cover the domain knowledge and task-specific engineering in the real-world recommendation. How to bridge the gap between simulation and real applications is a challenging topic. Sim2real [118] is a transfer learning approach that transfers DRL policies from simulation environments to reality. Sim2real uses domain adaption techniques to help agents transfer learned policy. Specifically, it adopts GANs to conduct adaption by generating different samples. RL-CycleGAN [119] is a sim2real method for vision-based tasks. It uses CycleGAN [120] to conduct pixel-level domain adaption. Specifically, it maintains cycle consistency during GAN training and encourages the adapted image to retain certain attributes of the input image. In DRL-based RS, sim2real would be a possible solution for generalizing the learned policy from simulation environments to reality. However, sim2real is a new technique still under exploration. It shows an adequate capability in simple tasks and requires more effort to handle the complex task such as recommendation. We believe it is a workable solution for generalizing from simulation to reality.

5.4. Bias (Unfairness)

Chen et al. [121] observe that user behavior data are not experimental but observational, which leads to problems of bias and unfairness.

There are two reasons why bias is so common. First, the inherent characteristic of user behavior data is not experimental but observational. In other words, data that are fed into recommender systems are subject to selection bias [122]. For instance, users in a video recommendation system tend to watch, rate, and comment on those movies that they are interested in. Second, a distribution discrepancy exists, which means the distributions of users and items in the recommender system are not even. Recommender systems may suffer from 'popularity bias', where popular items are recommended far more frequently than the others. However, the ignored products in the "long tail" can be equally critical for businesses as they are the ones less likely to be discovered. Friedman and Nissenbaum [123] denote the unfairness as that the system systematically and unfairly discriminates against certain individuals or groups of individuals in favor of others.

A large number of studies explore dynamic recommendation systems by utilizing the agent mechanism in reinforcement learning (RL), considering the information seeking and decision-making as sequential interactions. How to evaluate a policy efficiently is a big challenge for RL-based recommenders. Online A/B tests are not only expensive and time-consuming but also sometimes hurt the user experience. Off-policy evaluation is an alternative strategy that historical user behavior data are used to evaluate the policy. However, user behavior data are biased, as mentioned before, which causes a gap between the policy of RL-based RS and the optimal policy. To eliminate the effects of bias and unfairness, Chen et al. [43] use the inverse of the probability of historical policy to weight the policy gradients. Huang et al. [84] introduce a debiasing step that corrects the biases presented in the logged data before it is used to simulate user behavior. Zou et al. [16] propose to build a customer simulator that is designed to simulate the environment and handle the selection bias of logged data.

5.5. Explainability

Although deep learning-based models can generally improve the performance of recommender systems, they are not easily interpretable. As a result, it becomes an important task to make recommender results explainable, along with providing high-quality recommendations. High explainability in recommender systems not only helps end-users understand the items recommended but also enables system designers to check the internal mechanisms of recommender systems. Zhang and Chen [124] review different information sources and various types of models that can facilitate explainable recommendation. Attention mechanisms and knowledge graph techniques currently play an important role in realizing explainability in RS.

Attention models have great advantages in both enhancing predictive performance and having greater explainability [125]. Wang et al. [42] introduce a reinforcement learning framework incorporated with an attention model for explainable recommendation. Firstly, it achieves model-agnosticism by separating the recommendation model from the explanation generator. Secondly, the agents that are instantiated by attention-based neural networks can generate sentence-level explanations.

Knowledge graphs contain rich information about users and items, which can help to generate intuitive and more tailored explanations for the recommendation system [124]. Recent work has achieved greater explainability by using reinforcement and knowledge graph reasoning. The algorithm from [52] learns to find a path that navigates from users to items of interest by interacting with the knowledge graph environment. Zhao et al. [61] extract imperfect path demonstrations with minimum labeling effort and propose an adversarial actor–critic model for demonstration-guided path-finding. Moreover, it achieves better recommendation accuracy and explainability by reinforcement learning and knowledge graph reasoning.

5.6. Robustness on adversarial samples and attacks

Adversarial samples demonstrate that deep learning-based methods are vulnerable. Hence, robustness becomes an open question for both RS and DRL. Specifically, adversarial attack and defense in RS have received a lot of attention in recent years [126] as security is crucial in RS. Moreover, DRL policies are vulnerable to adversarial perturbations to agent's observations [127]. Gleave et al. [128] provide an adversarial attack method for perturbing the observations, thus affecting the learned policy. Hence, improving the robustness is the common interest for DRL and RS, which would be a critical problem for DRL-based RS. Cao et al. [129] provide an adversarial attack detection method for DRL-based RS which uses the GRU to encode the action space into a low-dimensional space and design decoders to detect the potential attack. However, it only considers Fast Gradient Sign Method (FGSM)-based attacks and strategically-timed attacks [127]. Thus, it lacks the capability to detect other types of attack. Chen et al. [130] design KGAttack, a DRL based knowledge graph-enhanced method to generate fake user profiles and conduct the attack to the recommendaer systems. KGAttack utlizes KG to generate the state representation and designs a Anchor Item Selection Policy to produce the fake profile. If the fake profiles are not rejected by the recommender systems, the attack is recognized as successful.

Based on those existing works, we find that, the current studies are mainly focusing on the attack detection or attack generation while the defence is still an opening question. We believe zero-shot learning techniques would be a good direction for training a universal adversarial attack detector. For defence, it is still an open question for DRL-based RS, though recent advances in adversarial defence in DRL may provide some insights [131–133].

6. Future directions

In this section, we provide a few potential future directions of DRL-based RS. Benefiting from recent advances in DRL research, we believe those topics can boost the progress of DRL-based RS research.

6.1. Causal and counterfactual inference

Causality is a generic relationship between a cause and effect. Moreover, inferring causal effects is a fundamental problem in many applications like computational advertising, search engines, and recommender systems [134]. Recently, some researchers have connected reinforcement learning with learning causality to improve the effects for solving sequential decision-making problems. Besides, Learning agents in reinforcement learning frameworks face a more complicated environment where a large number of heterogeneous data are integrated. From our point of view, causal relationships would be capable of improving the recommendation results by introducing the directionality of cause and the effect. The users' previous choices have impact on the subsequent actions. This can be cast as an interventional data generating the dynamics of recommender systems. By viewing a policy in RL as an intervention, we can detect unobserved confounders in RL and choose a policy on the expected reward to better estimate the causal effect [83]. Some studies improve RL models with causal knowledge as side information. Another line of work uses causal inference methods to achieve unbiased reward prediction [135] or data augmentation [136–138].

Yang et al. [139] propose a Causal Inference Q-network which introduces observational inference into DRL by applying extra noise and uncertain inventions to improve resilience. Specifically, in this work, noise and uncertainty are added into the state space during the training state, and the agent is required to learn a causal inference model by considering the perturbation. Dasgupta et al. [140] give the first demonstration that model-free reinforcement learning can be used for causal reasoning. They explore meta-reinforcement learning to solve the problem of causal reasoning. The agents trained by a recurrent network able to make causal inferences from observational data and output counterfactual predictions. Forney et al. [141] bridge RL and causality by data-fusion for reinforcement learners. Specifically, online agents combine observations, experiments and counterfactual data to learn about the environment, even if unobserved confounders exist. Similarly, Gasse et al. [142] make the model-based RL agents work in a causal way to explore the environment under the Partially-Observable Markov Decision Process (POMDP) setting. They consider interventional data and observational data jointly and interpret model-based reinforcement learning as a causal inference problem. In this way, they bridge the gap between RL and causality by relating common concepts in RL and causality. Wang et al. [143] introduce the counterfactual risk minimization to correct policy learning bias in REINFORCE-based RS which can maximize the long-run rewards. It designs a meta-graph to guide the policy learning direction thus reduce the bias. While the high-variance issue is not identified.

Regarding explainability in RL, Madumal et al. [144] propose to explain the behavior of agents in reinforcement learning with the help of causal science. The authors encode causal relationships and learn a structural causal model in RL, which generates explanations based on counterfactual analysis. With counterfactual exploration, this work can generate two contrastive explanations for 'why' and 'why not' questions. Considering traditional methods rely on local heuristics and predefined score functions, Zhu et al. [145] propose to use reinforcement learning to search DAG for causal discovery. Taking observational data as an input, they use RL agents as a search strategy and output the causal graph generated from an encoder–decoder NN model.

6.2. Offline DRL and meta DRL

Since recommender systems often facejoint recommendation and adverting scenarios, offline DRL and meta DRL provide the promising directions for dealing with multiple scenarios at the same time. Offline DRL is a new paradigm of DRL that can be combined with existing methods such as self-supervised learning and transfer learning to move toward real-world settings. Offline DRL [146] (also known as batch DRL) is designed for tasks which contain huge amounts of data. Given a large dataset that contains past interactions, offline DRL uses the dataset for training across many epochs but does not interact with the environment. Offline DRL provides a solution that can be generalized to new scenarios as it is trained by a large sized dataset. Such generalization ability is critical to RSs for deal with multiple scenarios or multiple customers. While offline DRL provides a new direction for DRL-based RS, it still faces a few problems regarding the distributional shifts between existing datasets and real-world interactions. Meta DRL [147] is defined as meta learning in the field of DRL. Meta DRL is another approach to help agents to generalize to new tasks or environments. Different from offline DRL, meta DRL contains a memory unit formed by the recurrent neural network to memorize the common knowledge for different tasks. Therefore, meta DRL does not require a large amount of data to train.

6.3. Further developments in actor–critic methods

An actor–critic method uses the traditional policy gradient method, which suffers from the high variance problem due to the gap between behavior policy (i.e., the policy that is being used by an agent for action select) and target policy (i.e., the policy that an agent is trying to learn). A method commonly used to relieve the high variance problem is Advantage Actor–critic (A2C). A2C uses an advantage function to replace the Q-function inside the critic network. The advantage function $A(s_t)$ is defined as the expected value of the TD-error. The new objective function for policy gradient is as follows:

$$\mathbb{E}_{\tau\sim d_{\pi_\theta}}[\sum_{t=1}^{T}\underbrace{(Q(s_t,a_t)-V(s_t))}_{A(s_t)}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(s_t,a_t)]. \tag{38}$$

A2C still uses DDPG as the main training algorithm and therefore may suffer function approximation errors when estimating the Q value. Twin-Delayed DDPG (TD3) [148] is designed to improve the function approximation problem in DDPG which uses clipped double Q-learning to update the critic. It defines the gradient update as follows:

$$\mathbb{E}_{\tau\sim d_{\pi_\theta}}[\sum_{t=1}^{T} r(s_t,a_t)+\gamma \min(Q_1(s_t,a_t+\epsilon), Q_2(s_t,a_t+\epsilon))\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(s_t,a_t)]. \tag{39}$$

where $\epsilon \sim clip(\mathcal{N}(0,\sigma,-c,c))$, σ is the standard deviation and c is a constant for clipping.

Other ways to improve actor–critic methods include Trust Region Policy Optimization (TRPO) [149] and Proximal Policy Optimization (PPO) [150], which focus on modification of the advantage function. TRPO aims to limit the step size for each gradient to ensure it will not change too much. The main idea is to add a constraint to the advantage function:

$$\frac{\pi(a|s)}{\pi_{old}(a|s)}A(s), \tag{40}$$

where the KL divergence is used to measure the distance between the current policy and the old policy is small enough. PPO has

the same goal as TRPO and aims to find the biggest possible improvement step on a policy using the current data. PPO is a simplified version of TRPO which introduces the clip operation,

$$\min\left(\frac{\pi(a|s)}{\pi_{old}(a|s)}A(s), \text{clip}\left(\frac{\pi(a|s)}{\pi_{old}(a|s)}, 1-\epsilon, 1+\epsilon\right)A(s)\right). \quad (41)$$

Soft Actor–Critic (SAC) [65] is another promising variant of the actor–critic algorithm and is widely used in DRL research. SAC uses a entropy term to encourage the agent to explore, which could be a promising direction towards solving the exploration and exploitation dilemma. Moreover, SAC assigns an equal probability to actions that are equally attractive to the agent to capture those near-optimal policies. An example of the related work [64] uses SAC to improve the stability of the training process with the multi-task objective in RS. Zhang et al. [151] also employs SAC and multi-task fusion to model the Long-Term User Satisfaction in RS. Zhang et al. [151] shows that SAC can outperform than TD3 in the real industry environment.

7. Conclusion

In this survey, we provide a comprehensive overview the application of deep reinforcement learning in recommender systems. We introduce a classification scheme for existing studies and discuss them by category. We also provide an overview of such existing emerging topics and point out a few promising directions. We hope this survey can provide a systematic understanding of the key concepts in DRL-based RS and valuable insights for future research.

CRediT authorship contribution statement

Xiaocong Chen: Conceptualization, Writing – original draft. **Lina Yao:** Conceptualization, Writing – review & editing, Supervision. **Julian McAuley:** Writing – review & editing. **Guanglin Zhou:** Resources. **Xianzhi Wang:** Writing – review & editing.

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.

Data availability

No data was used for the research described in the article.

References

[1] J. Lu, D. Wu, M. Mao, W. Wang, G. Zhang, Recommender system application developments: a survey, Decis. Support Syst. 74 (2015) 12–32.

[2] X. Chen, C. Huang, L. Yao, X. Wang, W. Zhang, et al., Knowledge-guided deep reinforcement learning for interactive recommendation, in: 2020 International Joint Conference on Neural Networks, IJCNN, IEEE, 2020, pp. 1–8.

[3] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Comput. Surv. 52 (1) (2019) 1–38.

[4] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Process. Mag. 34 (6) (2017) 26–38.

[5] 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, 2010, pp. 661–670.

[6] M.M. Afsar, T. Crump, B. Far, Reinforcement learning based recommender systems: A survey, 2021, arXiv preprint arXiv:2101.06286.

[7] Y. Lin, Y. Liu, F. Lin, P. Wu, W. Zeng, C. Miao, A survey on reinforcement learning for recommender systems, 2021, arXiv preprint arXiv:2109.10665.

[8] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.

[9] 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) 529–533.

[10] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (3–4) (1992) 229–256.

[11] V.R. Konda, J.N. Tsitsiklis, Actor-critic algorithms, in: Advances in Neural Information Processing Systems, Citeseer, 2000, pp. 1008–1014.

[12] T. Degris, M. White, R.S. Sutton, Off-policy actor-critic, 2012, arXiv preprint arXiv:1205.4839.

[13] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.

[14] Y. Zhang, C. Zhang, X. Liu, Dynamic scholarly collaborator recommendation via competitive multi-agent reinforcement learning, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, 2017, pp. 331–335.

[15] X. Chen, S. Li, H. Li, S. Jiang, Y. Qi, L. Song, Generative adversarial user model for reinforcement learning based recommendation system, in: International Conference on Machine Learning, PMLR, 2019, pp. 1052–1061.

[16] L. Zou, L. Xia, P. Du, Z. Zhang, T. Bai, W. Liu, J.-Y. Nie, D. Yin, Pseudo dyna-q: A reinforcement learning framework for interactive recommendation, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 816–824.

[17] K. Wang, Z. Zou, Q. Deng, R. Wu, J. Tao, C. Fan, L. Chen, P. Cui, Reinforcement learning with a disentangled universal value function for item recommendation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 4427–4435.

[18] X. Bai, J. Guan, H. Wang, A model-based reinforcement learning with adversarial training for online recommendation, Adv. Neural Inf. Process. Syst. 32 (2019) 10735–10746.

[19] D. Hong, Y. Li, Q. Dong, Nonintrusive-sensing and reinforcement-learning based adaptive personalized music recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 1721–1724.

[20] X. Zhao, L. Xia, L. Zou, H. Liu, D. Yin, J. Tang, Whole-chain recommendations, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 1883–1891.

[21] A. Montazeralghaem, J. Allan, Extracting relevant information from user's utterances in conversational search and recommendation, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 1275–1283.

[22] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, 2014, arXiv preprint arXiv:1406.2661.

[23] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, 2019, arXiv preprint arXiv:1908.10084.

[24] 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, 2018, pp. 167–176.

[25] 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, 2018, pp. 95–103.

[26] Y. Lei, Z. Wang, W. Li, H. Pei, Social attentive deep Q-network for recommendation, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 1189–1192.

[27] Y. Xiao, L. Xiao, X. Lu, H. Zhang, S. Yu, H.V. Poor, Deep reinforcement learning based user profile perturbation for privacy aware recommendation, IEEE Internet Things J. (2020).

[28] Y. Lei, W. Li, Interactive recommendation with user-specific deep reinforcement learning, ACM Trans. Knowl. Discov. Data (TKDD) 13 (6) (2019) 1–15.

[29] 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.

[30] E. Ie, V. Jain, J. Wang, S. Narvekar, R. Agarwal, R. Wu, H.-T. Cheng, T. Chandra, C. Boutilier, Slateq: A tractable decomposition for reinforcement learning with recommendation sets, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), Macau, China, 2019, pp. 2592–2599.

[31] Y. Lei, H. Pei, H. Yan, W. Li, Reinforcement learning based recommendation with graph convolutional q-network, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 1757–1760.

[32] T. Gui, P. Liu, Q. Zhang, L. Zhu, M. Peng, Y. Zhou, X. Huang, Mention recommendation in Twitter with cooperative multi-agent reinforcement learning, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 535–544.
[33] R.O. Oyeleke, C.-Y. Yu, C.K. Chang, Situ-centric reinforcement learning for recommendation of tasks in activities of daily living in smart homes, in: 2018 IEEE 42nd Annual Computer Software and Applications Conference, Vol. 2, COMPSAC, IEEE, 2018, pp. 317–322.
[34] 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, 2018, pp. 1040–1048.
[35] R. Takanobu, T. Zhuang, M. Huang, J. Feng, H. Tang, B. Zheng, Aggregating e-commerce search results from heterogeneous sources via hierarchical reinforcement learning, in: The World Wide Web Conference, 2019, pp. 1771–1781.
[36] R. Gao, H. Xia, J. Li, D. Liu, S. Chen, G. Chun, DRCGR: Deep reinforcement learning framework incorporating CNN and GAN-based for interactive recommendation, in: 2019 IEEE International Conference on Data Mining, ICDM, IEEE, 2019, pp. 1048–1053.
[37] S. Zhou, X. Dai, H. Chen, W. Zhang, K. Ren, R. Tang, X. He, Y. Yu, Interactive recommender system via knowledge graph-enhanced reinforcement learning, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 179–188.
[38] C. Pei, X. Yang, Q. Cui, X. Lin, F. Sun, P. Jiang, W. Ou, Y. Zhang, Value-aware recommendation based on reinforcement profit maximization, in: The World Wide Web Conference, 2019, pp. 3123–3129.
[39] X. Zhao, X. Zheng, X. Yang, X. Liu, J. Tang, Jointly learning to recommend and advertise, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 3319–3327.
[40] X. Zhao, C. Gu, H. Zhang, X. Yang, X. Liu, H. Liu, J. Tang, DEAR: Deep reinforcement learning for online advertising impression in recommender systems, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 750–758.
[41] F. Pan, Q. Cai, P. Tang, F. Zhuang, Q. He, Policy gradients for contextual recommendations, in: The World Wide Web Conference, 2019, pp. 1421–1431.
[42] X. Wang, Y. Chen, J. Yang, L. Wu, Z. Wu, X. Xie, A reinforcement learning framework for explainable recommendation, in: 2018 IEEE International Conference on Data Mining, ICDM, IEEE, 2018, pp. 587–596.
[43] 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, 2019, pp. 456–464.
[44] J. Xu, Z. Wei, L. Xia, Y. Lan, D. Yin, X. Cheng, J.-R. Wen, Reinforcement learning to rank with pairwise policy gradient, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 509–518.
[45] J. Ma, Z. Zhao, X. Yi, J. Yang, M. Chen, J. Tang, L. Hong, E.H. Chi, Off-policy learning in two-stage recommender systems, in: Proceedings of the Web Conference 2020, 2020, pp. 463–473.
[46] M. Chen, B. Chang, C. Xu, E.H. Chi, User response models to improve a REINFORCE recommender system, in: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021, pp. 121–129.
[47] A. Montazeralghaem, H. Zamani, J. Allan, A reinforcement learning framework for relevance feedback, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 59–68.
[48] S. Ji, Z. Wang, T. Li, Y. Zheng, Spatio-temporal feature fusion for dynamic taxi route recommendation via deep reinforcement learning, Knowl.-Based Syst. 205 (2020) 106302.
[49] T. Yu, Y. Shen, R. Zhang, X. Zeng, H. Jin, Vision-language recommendation via attribute augmented multimodal reinforcement learning, in: Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 39–47.
[50] Z. Li, J. Ji, Y. Ge, Y. Zhang, AutoLossGen: Automatic loss function generation for recommender systems, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 1304–1315.
[51] P. Wang, Y. Fan, L. Xia, W.X. Zhao, S. Niu, J. Huang, KERL: A knowledge-guided reinforcement learning model for sequential recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 209–218.
[52] Y. Xian, Z. Fu, S. Muthukrishnan, G. De Melo, Y. Zhang, Reinforcement knowledge graph reasoning for explainable recommendation, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 285–294.
[53] X. Wang, Y. Xu, X. He, Y. Cao, M. Wang, T.-S. Chua, Reinforced negative sampling over knowledge graph for recommendation, in: Proceedings of the Web Conference 2020, 2020, pp. 99–109.
[54] 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.
[55] 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, 2018, pp. 368–377.
[56] R. Zhang, T. Yu, Y. Shen, H. Jin, C. Chen, Text-based interactive recommendation via constraint-augmented reinforcement learning, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019.
[57] X. Zhao, L. Zhang, L. Xia, Z. Ding, D. Yin, J. Tang, Deep reinforcement learning for list-wise recommendations, in: DRL4KDD '19, 2019.
[58] F. Liu, R. Tang, H. Guo, X. Li, Y. Ye, X. He, Top-aware reinforcement learning based recommendation, Neurocomputing 417 (2020) 255–269.
[59] L. Wang, W. Zhang, X. He, H. Zha, Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2447–2456.
[60] Q. Cai, A. Filos-Ratsikas, P. Tang, Y. Zhang, Reinforcement mechanism design for e-commerce, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 1339–1348.
[61] K. Zhao, X. Wang, Y. Zhang, L. Zhao, Z. Liu, C. Xing, X. Xie, Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 239–248.
[62] J. Feng, H. Li, M. Huang, S. Liu, W. Ou, Z. Wang, X. Zhu, Learning to collaborate: Multi-scenario ranking via multi-agent reinforcement learning, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 1939–1948.
[63] W. Zhang, H. Liu, F. Wang, T. Xu, H. Xin, D. Dou, H. Xiong, Intelligent electric vehicle charging recommendation based on multi-agent reinforcement learning, in: Proceedings of the Web Conference 2021, 2021, pp. 1856–1867.
[64] X. He, B. An, Y. Li, H. Chen, R. Wang, X. Wang, R. Yu, X. Li, Z. Wang, Learning to collaborate in multi-module recommendation via multi-agent reinforcement learning without communication, in: Fourteenth ACM Conference on Recommender Systems, 2020, pp. 210–219.
[65] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.
[66] D. Zhao, L. Zhang, B. Zhang, L. Zheng, Y. Bao, W. Yan, Mahrl: Multi-goals abstraction based deep hierarchical reinforcement learning for recommendations, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 871–880.
[67] R. Xie, S. Zhang, R. Wang, F. Xia, L. Lin, Hierarchical reinforcement learning for integrated recommendation, in: Proceedings of AAAI, 2021.
[68] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30, 2016.
[69] J. Tang, K. Wang, Personalized top-n sequential recommendation via convolutional sequence embedding, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 565–573.
[70] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, ICLR, 2017.
[71] Y. Sun, F. Zhuang, H. Zhu, Q. He, H. Xiong, Cost-effective and interpretable job skill recommendation with deep reinforcement learning, in: Proceedings of the Web Conference 2021, 2021, pp. 3827–3838.
[72] J. Achiam, D. Held, A. Tamar, P. Abbeel, Constrained policy optimization, in: International Conference on Machine Learning, PMLR, 2017, pp. 22–31.
[73] M. Jaderberg, V. Mnih, W.M. Czarnecki, T. Schaul, J.Z. Leibo, D. Silver, K. Kavukcuoglu, Reinforcement learning with unsupervised auxiliary tasks, 2016, arXiv preprint arXiv:1611.05397.
[74] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, G. Brain, Time-contrastive networks: Self-supervised learning from video, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 1134–1141.
[75] Y. Wang, M. Sharma, C. Xu, S. Badam, Q. Sun, L. Richardson, L. Chung, E.H. Chi, M. Chen, Surrogate for long-term user experience in recommender systems, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 4100–4109.

[76] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Neural Information Processing Systems, NIPS, 2013, pp. 1–9.
[77] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International Conference on Machine Learning, PMLR, 2014, pp. 387–395.
[78] A. Tavakoli, F. Pardo, P. Kormushev, Action branching architectures for deep reinforcement learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.
[79] B. Shi, M.G. Ozsoy, N. Hurley, B. Smyth, E.Z. Tragos, J. Geraci, A. Lawlor, PyRecGym: a reinforcement learning gym for recommender systems, in: Proceedings of the 13th ACM Conference on Recommender Systems, 2019, pp. 491–495.
[80] D. Rohde, S. Bonner, T. Dunlop, F. Vasile, A. Karatzoglou, Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising, 2018, arXiv preprint arXiv:1808.00720.
[81] J.-C. Shi, Y. Yu, Q. Da, S.-Y. Chen, A.-X. Zeng, Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 4902–4909.
[82] E. Ie, C.-w. Hsu, M. Mladenov, V. Jain, S. Narvekar, J. Wang, R. Wu, C. Boutilier, Recsim: A configurable simulation platform for recommender systems, 2019, arXiv preprint arXiv:1909.04847.
[83] W. Shang, Y. Yu, Q. Li, Z. Qin, Y. Meng, J. Ye, Environment reconstruction with hidden confounders for reinforcement learning based recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 566–576.
[84] J. Huang, H. Oosterhuis, M. de Rijke, H. van Hoof, Keeping dataset biases out of the simulation: A debiased simulator for reinforcement learning based recommender systems, in: Fourteenth ACM Conference on Recommender Systems, 2020, pp. 190–199.
[85] M.R. Santana, L.C. Melo, F.H. Camargo, B. Brandão, A. Soares, R.M. Oliveira, S. Caetano, MARS-gym: A gym framework to model, train, and evaluate recommender systems for marketplaces, 2020, arXiv preprint arXiv:2010.07035.
[86] X. Zhao, L. Xia, L. Zou, D. Yin, J. Tang, Toward simulating environments in reinforcement learning based recommendations, 2019, arXiv preprint arXiv:1906.11462.
[87] F. Liu, R. Tang, X. Li, W. Zhang, Y. Ye, H. Chen, H. Guo, Y. Zhang, Deep reinforcement learning based recommendation with explicit user-item interactions modeling, 2018, arXiv preprint arXiv:1810.12027.
[88] F. Liu, H. Guo, X. Li, R. Tang, Y. Ye, X. He, End-to-end deep reinforcement learning based recommendation with supervised embedding, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 384–392.
[89] F. Liu, R. Tang, X. Li, W. Zhang, Y. Ye, H. Chen, H. Guo, Y. Zhang, X. He, State representation modeling for deep reinforcement learning based recommendation, Knowl.-Based Syst. 205 (2020) 106170.
[90] J. Huang, H. Oosterhuis, B. Cetinkaya, T. Rood, M. de Rijke, State encoders in reinforcement learning for recommendation: A reproducibility study, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 2738–2748.
[91] S.-Y. Chen, Y. Yu, Q. Da, J. Tan, H.-K. Huang, H.-H. Tang, Stabilizing reinforcement learning in dynamic environment with application to online recommendation, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1187–1196.
[92] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, 2016, arXiv preprint arXiv:1606.01540.
[93] I. Kostrikov, K.K. Agrawal, D. Dwibedi, S. Levine, J. Tompson, Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning, in: International Conference on Learning Representations, 2019, URL: https://openreview.net/forum?id=Hk4fpoA5Km.
[94] A.Y. Ng, D. Harada, S. Russell, Policy invariance under reward transformations: Theory and application to reward shaping, in: Icml, Vol. 99, 1999, pp. 278–287.
[95] T.D. Kulkarni, K. Narasimhan, A. Saeedi, J. Tenenbaum, Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation, Adv. Neural Inf. Process. Syst. 29 (2016) 3675–3683.
[96] J. Zhang, B. Hao, B. Chen, C. Li, H. Chen, J. Sun, Hierarchical reinforcement learning for course recommendation in MOOCs, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 435–442.
[97] M. Egorov, Multi-agent deep reinforcement learning, in: CS231n: Convolutional Neural Networks for Visual Recognition, 2016, pp. 1–8.
[98] R. Makar, S. Mahadevan, M. Ghavamzadeh, Hierarchical multi-agent reinforcement learning, in: Proceedings of the Fifth International Conference on Autonomous Agents, 2001, pp. 246–253.
[99] Z. Yang, K. Merrick, L. Jin, H.A. Abbass, Hierarchical deep reinforcement learning for continuous action control, IEEE Trans. Neural Netw. Learn. Syst. 29 (11) (2018) 5174–5184.
[100] Y. Gong, Y. Zhu, L. Duan, Q. Liu, Z. Guan, F. Sun, W. Ou, K.Q. Zhu, Exact-k recommendation via maximal clique optimization, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 617–626.
[101] X. Chen, L. Yao, A. Sun, X. Wang, X. Xu, L. Zhu, Generative inverse deep reinforcement learning for online recommendation, 2020, arXiv preprint arXiv:2011.02248.
[102] A.Y. Ng, S.J. Russell, et al., Algorithms for inverse reinforcement learning, in: Icml, Vol. 1, 2000, p. 2.
[103] J. Ho, S. Ermon, Generative adversarial imitation learning, Adv. Neural Inf. Process. Syst. 29 (2016) 4565–4573.
[104] I. Bica, D. Jarrett, A. Hüyük, M. van der Schaar, Learning"what-if" explanations for sequential decision-making, in: International Conference on Learning Representations, 2020.
[105] C. Hu, S. Huang, Y. Zhang, Y. Liu, Learning to infer user implicit preference in conversational recommendation, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 256–266.
[106] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems, in: The World Wide Web Conference, 2019, pp. 3307–3313.
[107] J. Jiang, C. Dun, T. Huang, Z. Lu, Graph convolutional reinforcement learning, in: International Conference on Learning Representations, 2020, URL: https://openreview.net/forum?id=HkxdQkSYDB.
[108] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.
[109] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, S. Levine, Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 5129–5136.
[110] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, T. Funkhouser, Learning synergies between pushing and grasping with self-supervised deep reinforcement learning, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2018, pp. 4238–4245.
[111] W. Shi, G. Huang, S. Song, Z. Wang, T. Lin, C. Wu, Self-supervised discovering of interpretable features for reinforcement learning, IEEE Trans. Pattern Anal. Mach. Intell. (2020).
[112] X. Xin, A. Karatzoglou, I. Arapakis, J.M. Jose, Self-supervised reinforcement learning for recommender systems, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 931–940.
[113] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, 2015, arXiv preprint arXiv:1511.05952.
[114] D. Isele, A. Cosgun, Selective experience replay for lifelong learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.
[115] J. Wu, Z. Xie, T. Yu, H. Zhao, R. Zhang, S. Li, Dynamics-aware adaptation for reinforcement learning based cross-domain interactive recommendation, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 290–300.
[116] X. Chen, L. Yao, J. McAuley, W. Guan, X. Chang, X. Wang, Locality-sensitive state-guided experience replay optimization for sparse rewards in online recommendation, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 1316–1325.
[117] Y. Yue, T. Joachims, Interactively optimizing information retrieval systems as a dueling bandits problem, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 1201–1208.
[118] W. Zhao, J.P. Queralta, T. Westerlund, Sim-to-real transfer in deep reinforcement learning for robotics: a survey, in: 2020 IEEE Symposium Series on Computational Intelligence, SSCI, IEEE, 2020, pp. 737–744.
[119] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, M. Khansari, Rl-cyclegan: Reinforcement learning aware simulation-to-real, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11157–11166.
[120] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2223–2232.
[121] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, X. He, Bias and debias in recommender system: A survey and future directions, 2020, arXiv preprint arXiv:2010.03240.
[122] T. Schnabel, A. Swaminathan, A. Singh, N. Chandak, T. Joachims, Recommendations as treatments: Debiasing learning and evaluation, 2016.
[123] B. Friedman, H. Nissenbaum, Bias in computer systems, ACM Trans. Inf. Syst. (TOIS) 14 (3) (1996) 330–347.
[124] Y. Zhang, X. Chen, Explainable recommendation: A survey and new perspectives, Found. Trends® Inf. Retr. 14 (1) (2020) 1–101.

[125] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Comput. Surv. 52 (1) (2019).
[126] Y. Deldjoo, T.D. Noia, F.A. Merra, A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks, ACM Comput. Surv. 54 (2) (2021) 1–38.
[127] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, M. Sun, Tactics of adversarial attack on deep reinforcement learning agents, 2017, arXiv preprint arXiv:1703.06748.
[128] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, S. Russell, Adversarial policies: Attacking deep reinforcement learning, in: International Conference on Learning Representations, 2020, URL: https://openreview.net/forum?id=HJgEMpVFwB.
[129] Y. Cao, X. Chen, L. Yao, X. Wang, W.E. Zhang, Adversarial attacks and detection on reinforcement learning-based interactive recommender systems, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 1669–1672.
[130] J. Chen, W. Fan, G. Zhu, X. Zhao, C. Yuan, Q. Li, Y. Huang, Knowledge-enhanced black-box attacks for recommendations, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 108–117.
[131] B. Lütjens, M. Everett, J.P. How, Certified adversarial robustness for deep reinforcement learning, in: Conference on Robot Learning, PMLR, 2020, pp. 1328–1337.
[132] F. Wang, C. Zhong, M.C. Gursoy, S. Velipasalar, Defense strategies against adversarial jamming attacks via deep reinforcement learning, in: 2020 54th Annual Conference on Information Sciences and Systems, CISS, IEEE, 2020, pp. 1–6.
[133] T. Chen, J. Liu, Y. Xiang, W. Niu, E. Tong, Z. Han, Adversarial attack and defense in reinforcement learning-from AI security view, Cybersecurity 2 (1) (2019) 1–22.
[134] L. Bottou, J. Peters, J. Quiñonero-Candela, D.X. Charles, D.M. Chickering, E. Portugaly, D. Ray, P. Simard, E. Snelson, Counterfactual reasoning and learning systems: The example of computational advertising, J. Mach. Learn. Res. 14 (11) (2013).
[135] R. Guo, L. Cheng, J. Li, P.R. Hahn, H. Liu, A survey of learning causality with data: Problems and methods, ACM Comput. Surv. 53 (4) (2020).
[136] X. Chen, S. Wang, L. Yao, L. Qi, Y. Li, Intrinsically motivated reinforcement learning based recommendation with counterfactual data augmentation, 2022, arXiv preprint arXiv:2209.08228.
[137] S. Wang, X. Chen, L. Yao, J. McAuley, Deep reinforcement learning for dynamic recommendation with model-agnostic counterfactual policy synthesis, 2022, arXiv preprint arXiv:2208.05142.
[138] S. Wang, X. Chen, L. Yao, Model-agnostic counterfactual synthesis policy for interactive recommendation, arXiv preprint arXiv:2204.00308 (2022).
[139] C.-H.H. Yang, I. Hung, T. Danny, Y. Ouyang, P.-Y. Chen, Causal inference Q-network: Toward resilient reinforcement learning, 2021, arXiv preprint arXiv:2102.09677.
[140] I. Dasgupta, J. Wang, S. Chiappa, J. Mitrovic, P. Ortega, D. Raposo, E. Hughes, P. Battaglia, M. Botvinick, Z. Kurth-Nelson, Causal reasoning from meta-reinforcement learning, 2019, arXiv preprint arXiv:1901.08162.
[141] A. Forney, J. Pearl, E. Bareinboim, Counterfactual data-fusion for online reinforcement learners, in: International Conference on Machine Learning, PMLR, 2017, pp. 1156–1164.
[142] M. Gasse, D. Grasset, G. Gaudron, P.-Y. Oudeyer, Causal reinforcement learning using observational and interventional data, 2021, arXiv preprint arXiv:2106.14421.
[143] X. Wang, Q. Li, D. Yu, Z. Wang, H. Chen, G. Xu, Mgpolicy: Meta graph enhanced off-policy learning for recommendations, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 1369–1378.
[144] P. Madumal, T. Miller, L. Sonenberg, F. Vetere, Explainable reinforcement learning through a causal lens, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 2493–2500.
[145] S. Zhu, I. Ng, Z. Chen, Causal discovery with reinforcement learning, 2019, arXiv preprint arXiv:1906.04477.
[146] S. Levine, A. Kumar, G. Tucker, J. Fu, Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020, arXiv preprint arXiv:2005.01643.
[147] J.X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J.Z. Leibo, R. Munos, C. Blundell, D. Kumaran, M. Botvinick, Learning to reinforcement learn, 2016, arXiv preprint arXiv:1611.05763.
[148] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, PMLR, 2018, pp. 1587–1596.
[149] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International Conference on Machine Learning, PMLR, 2015, pp. 1889–1897.
[150] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.
[151] Q. Zhang, J. Liu, Y. Dai, Y. Qi, Y. Yuan, K. Zheng, F. Huang, X. Tan, Multi-task fusion via reinforcement learning for long-term user satisfaction in recommender systems, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 4510–4520.