International Journal of Computational Intelligence Systems (2023) 16:4
https://doi.org/10.1007/s44196-022-00179-1

RESEARCH ARTICLE

A Deep Reinforcement Learning Real-Time Recommendation Model Based on Long and Short-Term Preference

Yan-e Hou[1,2] · Wenbo Gu[1,2] · WeiChuan Dong[3] · Lanxue Dang[1,2]

Received: 15 July 2022 / Accepted: 26 December 2022

Abstract
With the development of Internet technology, the problem of information overload has increasingly attracted attention. Nowadays, the recommendation system with excellent performance in information retrieval and filtering would be widely used in the business field. However, most existing recommendation systems are considered a static process, during which recommendations for internet users are often based on pre-trained models. A major disadvantage of these static models is that they are incapable of simulating the interaction process between users and their systems. Moreover, most of these models only consider users' real-time interests while ignoring their long-term preferences. This paper addresses the above-mentioned issues and proposes a new recommendation model, DRR-Max, based on deep reinforcement learning (DRL). In the proposed framework, this paper adopted a state generation module specially designed to obtain users' long-term and short-term preferences from user profiles and user history score item information. Next, Actor-Critical algorithm is used to simulate the real-time recommendation process.Finally, this paper uses offline and online methods to train the model. In the online mode, the network parameters were dynamically updated to simulate the interaction between the system and users in a real recommendation environment. Experimental results on the two publicly available data sets were used to demonstrate the effectiveness of our proposed model.

Keywords Recommendation system · Deep reinforcement learning · Online recommendation mode · Actor–critic · Long and short-term preference

1 Introduction

With the rapid development of network technology, the amount of information on the network has been growing exponentially. With increasing concerns about information overload on the Internet, it is critical for Internet companies to accurately and efficiently sift and recommend information for users based on their preferences. A promising solution to information overload is to build a high-performance recommender system. Recently, recommender systems are extensively researched and many successful recommender systems have been developed in the business world, including the GroupLens [1], the video streaming company Netflix [2], the online shopping website JD [3, 4] and many others. According to the research of Rostami [5], TOPSIS model is used to search suitable tourist sites for users using ABC algorithm according to user preferences.

Recommendation algorithms are the core part of a recommendation system, aiming to provide users with accurate recommendations. In general, recommendation algorithms are classified into traditional recommendation approaches and deep learning-based approaches. The clustering methods from the traditional recommendation approaches can recommend user's items that are similar to their interests while effectively solve the data sparsity problem [6, 7]. The clustering-based methods has been widely used in recommendation algorithms, where matrix

✉ Lanxue Dang
danglx@vip.henu.edu.cn

Yan-e Hou
houyane@henu.edu.cn

Wenbo Gu
guwenbo@henu.edu.cn

WeiChuan Dong
wdong@kent.edu

[1] Henan Key Laboratory of Big Data Analysis and Processing, Henan University, Kaifeng 475004, Henan, China

[2] College of Computer and Information Engineering, Henan University, Kaifeng 475004, Henan, China

[3] Department of Geography, Kent State University, Kent, Oh 44240, USA

Published online: 21 January 2023

decomposition techniques [8] have advantages in addressing the sparse matrix problem. The advancement in deep learning techniques has provided new design ideas for recommendation algorithms. For example, two popular reinforcement learning (RL) techniques, the value-based [9, 10] and the policy-based [3, 4, 11] have been applied in many recommendation systems.

Since the user interaction with a recommendation system is a continuous behavior, an ideal recommendation system should consider real-time preferences of users. However, most existing recommendation systems are pre-trained models and it is difficult for them to capture users' real-time interests. To address this problem, several RL-based approaches, such as POMDP [12] and Q-learning [13], have been developed to improve the quality of recommendations, and have been used in multiple recommender systems [3, 4, 9–11]. However, these models are limited in accommodating complex recommendation scenarios. For example, the value-based RL recommendation systems [9, 10] are capable of accurately predicting the probability of a user's subsequent actions, yet the efficiency of Q-value computation is decreased due to the large computational space. In contrast, the policy-based RL recommendation systems [3, 4, 11] take all action spaces as a continuous parameter vector to represent all actions, followed by the next recommendation and the update of Q-value. This policy-based approach can avoid large-scale Q-value computation but it cannot capture the interaction process between the user and the item accurately.

Most of the traditional methods are based on collaborative filtering, which has the problems of cold start and excessive computing cost with the increase of data set size. The recommendation algorithm based on deep learning can effectively solve the problem of excessive computing cost by pre training the model, modeling in a nonlinear way, and encoding more complex abstractions as higher level data representations, but at the same time, the pre trained model can not effectively face the real changing recommendation environment. Therefore, this paper proposes to combine reinforcement learning with deep learning, use the excellent decision-making ability of reinforcement learning, and simulate the real real-time recommendation environment.

In view of this, this paper aims to propose a deep reinforcement learning-based recommender system model (donated as DRR-Max), where Max refers to the maximum pooling layer used in this paper for feature extraction. Our DRR-Max model consists of two parts: a state generation module and an Actor–Critic algorithm [14]. The state generation module is well-designed in the DRR-Max model, the user item matrix is decomposed into user specific diagnosis matrix and item specific detection matrix through PMF, which solves the sparse matrix problem faced by traditional algorithms. In the process of interaction, the user's long-term and short-term preferences are also taken into account to generate more accurate user state information. The generated state information of user's is then put into the Actor–Critic algorithm, which is used to simulate the interaction process between the user and the recommender system, solve the problem that previous models are static. The Actor–Critic algorithm is applied to predict the next action according to the user's state and to evaluate the action. At the same time, the network parameters are updated dynamically. Finally, both the online and offline experiments were designed to verify the efficiency of our DRR-Max model using the Book-crossing and the Amazon-b data sets.

The main contributions of this paper are as follows: (1) In order to solve the problem that the static model used by traditional algorithms can not adapt well to the dynamic interest changes of users, this paper adopts Actor–Critic algorithm, and the proposed DRR-Max model regards recommendation as a continuous process, simulating the real gradual recommendation environment. (2) A new state generation module is designed to efficiently simulate the interaction between users and recommendation items. (3) The DRR-Max model is tested using two real-world public data sets, and the results of the comparison with the five traditional models suggested that our proposed model had better performance regarding the effectiveness of recommendation.

The rest of the article is organized as follows: Sect. 2 briefly describes the recent development of recommender systems. The introduction of the deep learning in recommender systems is described in Sect. 3. Section 4 describes the details of our proposed model and the procedures of training the model. Experimental results on two public databases and comparison with existing models are analyzed in Sect. 5. Finally, Sect. 6 draws conclusions of the current study and give directions of future work.

2 Related Works

This section will discuss the traditional recommendation algorithms and deep-learning-based recommendation algorithms.

2.1 Traditional Recommendation Algorithm

Traditional recommendation algorithm consists of content-based methods, collaborative filtering methods, and hybrid recommendation methods. Content-based methods use target-related or user-related information or user's operation behavior on the target to construct a recommendation model, which generally rely on the user's own behaviors without involving other users' behaviors [1, 15]. Collaborative filtering algorithm plays a very important role in the existing recommendation algorithm. They aim to detect a group of users with similar tastes and preferences to the target user, and to recommend items favored by other users in the group

to the current user. The first collaborative filtering recommendation method proposed by Goldberg [16] can filter out a set of items of interest for a specific user. Later, a collaborative filtering recommendation algorithm called Dynamic Decay Collaborative Filtering (DDCF) proposed by Chen et al. [17] combines the memory curve with collaborative filtering and considers the user's long and short-term preferences. Liao and Li [18] adopted a self-constructed clustering algorithm to improve the clustering computation efficiency in collaborative filtering recommendation method without decreasing the quality of algorithm. In additional, fuzzy C-means clustering method [19], Top-N method [20], and other techniques [21] were designed to improve the accuracy of the collaborative filtering recommendation algorithms.

A real-world recommendation environment is often complex, where a single recommendation approach is often not competent in achieving optimal results. Hybrid recommendation algorithms take advantage of multiple recommendation approaches in obtaining better quality of recommendation. Tian [22] et al, a hybrid recommender system was designed to recommend books of most interest to users from a large number of candidates have adopted a user rating matrix and clustering method in addressing the sparse matrix problem [22]. Cai et al. [23] proposed a recommendation system based on multi-objective optimization to solve the problem of different needs of various users. These hybrid recommendation algorithms [22, 23] improve performance of the recommendation to a certain extent.

2.2 Deeping Learning-Based Recommendation Algorithm

Nowadays, deep learning-based recommendation algorithms have become the focus of current research due to the excellent performance of deep learning technique in handling complex tasks. Zhang et al. [24] summarized the latest research of recommendation systems based on deep learning, classify deep learning recommendation systems, including recommendation model and techniques, as well as gives the future research direction. Forouzandeh et al. [25] proposed a new food recommendation system, which divided the recommendation into two stages. In the first stage, graphical clustering was used, and in the second stage, a method based on deep learning was used to cluster users and food to overcome the shortcomings of the previous system, such as cold start and food composition problems.

Among all deep learning-based algorithms, several algorithms adopt traditional collaborative filtering to improve the quality of recommendation [26, 27]. Nassar et al. [27] proposed a deep learning-based multi-criteria collaborative filtering model. In this hybrid model, the features of users and items are first obtained, and then input into a deep neural network to predict standard scores. These standard scores will further be put into the whole ranking deep neural network to get the overall score ranking. Maxim et al. [28] designed a new parallel scheme that enabled efficient computation of fully connected layers in learning-based recommendation model.

For temporal recommendation systems, there are some successful models based on deep learning. Tang and Wang [29] combines convolutional sequence embedding model with Top-N sequential recommendation as a way for temporal recommendation. Li et al. [30] designed a new neural attention recommender network to consider the sequential behavior of user in current session. Zhang et al. [31] introduced attention mechanism into the sequence-sense recommendation model to represent user's temporal interest. Wu et al. [32] constructed a session-based graph neural network recommendation model (SR-GNN). In SR-GNN, session sequences are modeled as image-structured data, and the GNN can capture complex transformation of recommendation items. The recent review of temporal recommendation models and algorithms can be found in [33].

There are also several successful deep learning-based models in temporal recommendation systems. Tang and Wang [29] combined convolutional sequence embedding model with Top-N sequential recommendation as a way for temporal recommendation. Li et al. [30] designed a new neural attention recommender network that considered the sequential behavior of user in current session. Zhang et al. [31] introduced attention mechanism into the sequence-sense recommendation model that represented user's temporal interest. Wu et al. [32] constructed a session-based graph neural network recommendation model (SR-GNN). In SR-GNN, session sequences are modeled as image-structured data, and the graph neural network can capture complex transformation of recommendation items. Fang et al. [33] summarizes recent recommendation models and algorithms based on chronological order.

To avoid the problem of only focusing on short-term session data and ignoring the long-term interests of users, a recent study proposed a recommendation model based on improved recurrent neural network [34]. The authors of this study subsequently implemented a variety of parallel recurrent neural networks to model the session graph [35]. For the vector representation of users and recommendation items, Liu and Chen [36] developed an end-to-end graph neural network with memory units. In this graph neural network, the gated recurrent unit was introduced into to solve the information loss between high-order connected nodes. Then, convolutional neural networks are adopted to merge feature vectors between network output layers to obtain user preferences at different stages.

In recent years, deep RL techniques have promoted the rapid development of commendation systems. Huang et al. [37] treated the recommendation process as a Markov decision process (MDP) and used the recurrent neural network

to simulate the interaction between the recommendation system and the user. They proposed a top-N interactive recommender system to maximize long-term recommendation accuracy. Zheng et al. [9] used a deep Q-learning-based recommendation framework to improve the real-time recommendation of news. Liu et al. [38] proposed a deep RL-based recommendation framework, in which the recommendation process was considered as a sequential decision-making process. Combined with the Actor–Critic algorithm, the framework simulates the interaction between the user and the environment. Zhao et al. [3] designed an online user-agent interaction environment simulator, where the model can be trained and parameters can be evaluated in an offline mode to reduce the train data scale and the running time of the model. In a subsequent study [4], the authors further introduced a deep RL-based page recommendation framework to provide real-time feedback optimization items in the page.

3 Preliminaries

Due to its powerful decision-making capability, RL has been widely used in many fields. RL simulates the interaction between the agent and the environment, where the agent selects the action and the environment provides the response to the agent and changes to the new state, enabling RL to learn from interactions. Through the continually interactions, the agent attempts to obtain the best total rewards. The decision process of RL can be considered as a Markov decision process (MDP). MDP can be defined as (S, A, P, R, γ). Where S is the state space, A is the action space, P is the state transfer function, R is the return function, and γ is the discount factor. The goal of the agent in MDP is to find an optimal strategy to maximize the expected cumulative rewards under any state, or equivalently maximize the cumulative expected reward for an action in any state.

According to the above description, we can regard the recommendation process of the recommendation system as a continuous decision-making process. When interacting with the recommender system, the user can be treated as the environment and the recommender system is considered as the agent, which maximizes the cumulative rewards of the recommender process. Therefore, the whole recommendation process can be regarded as a MDP process which is defined as follows.

- State space S: State $s_t = \{s_t^1, s_t^2, \ldots, s_t^N\} \in S$ is defined as the top N items that the user interacts with between time t. Here, s_t is arranged in chronological order.
- Action space A: Action $a_t = \{a_t^1, a_t^2, \ldots, a_t^K\} \in A$ is the list of selectable actions based on the user's state s_t at time t, and K represents the recommended items to the user each time number of items.
- State transition P: P is a probability function. $P(s_{t+1} \min s_t, a_t)$ represents the probability of transition from state s_t to s_{t+1} in the case of state s_t and action a_t. Suppose $P(s_{t+1} \mid s_t, a_t, \ldots, s_1, a_1) = P(s_{t+1} \mid s_t, a_t)$.
- Reward R: In state s_t, the recommender system will take action a_t to recommend a list of items for the user, and the user will give feedback on the recommendation list. The agent will receive different rewards $r(s_t, a_t)$ for different actions made by the user on the recommendation list.
- Discount factor γ: $\gamma \in [0,1]$ is defined as a strategy that we use to measure rewards. When γ=0, we only consider immediate rewards and ignore long-term rewards. Conversely, when γ=1, we regard immediate and long-term rewards as equally important.

In the recommendation model proposed in this paper, the action represents a continuous vector of parameters instead of a single item or a group of items. The vector is subjected to an inner product operation with the item feature matrix, which in turn yields the ranking of the candidate items and recommends the top N items to the user. The user receives the recommended items from the system and provides feedback to the recommendation system, based on which the user status is updated and the recommendation system receives rewards.

4 Methodology

In this section, we first introduce the model in detail, and elaborated the training process of the model.

4.1 Proposed Model

4.1.1 State Generation Module

The state generation module generates the current state of the user based on user feature information and user's historical interaction item feature matrix. An efficient state generation network not only generates high quality state information, but also helps the Actor–Critic network perform action generation.

Figure 1 gives the structure of state generation module. As shown in Fig. 1, the user feature matrix and the feature matrix of the user's first N interaction items at moment t are the input data and the user's current state is the output of the module. Among them, this paper decomposes the probability matrix of the test data set to obtain the user characteristic matrix and project characteristic matrix with the scale of N, and we express the user characteristic matrix as: $U = \{u_1, u_2, ..., u_n\}$, and the project specific diagnosis matrix as: $I = \{i_1, i_2, ..., i_n\}$. There are three parts in state generation module: the first one is the user feature matrix u on the left; the second one is the matrix information of the feature

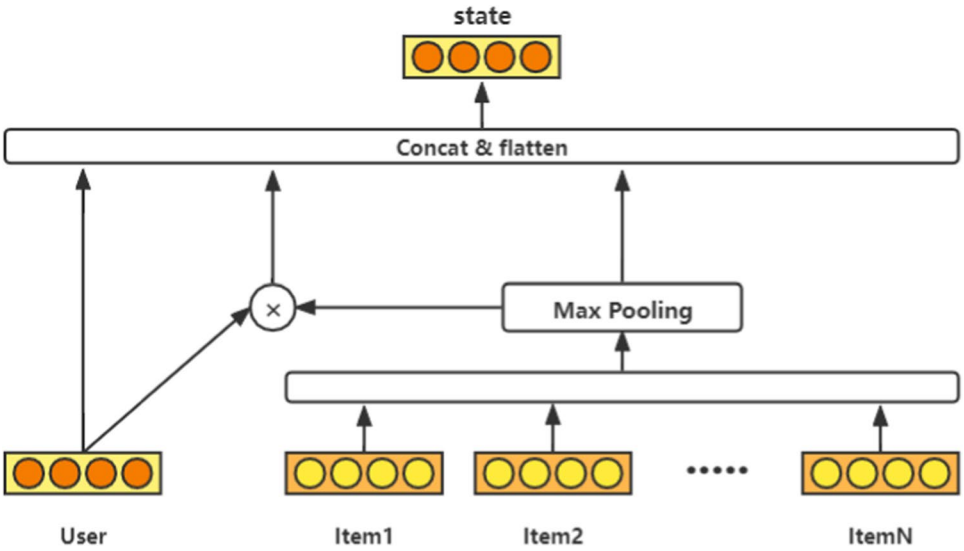


Fig. 1 State generation network

matrix of the N items after the maximum pooling layer on the right; the last one is the matrix multiplication operation of the left and right parts in the middle. Therefore, the model not only obtains information about the user's characteristics, but also extracts information about the user's historical behavior. Meanwhile, the user's information and the behavioral information can be interacted with each other to obtain a more accurate user's current state. The state generation module can be represented in Eqs. (1) and (2):

$$s_t = [u, u \otimes \{g(i_a \mid a = 1, ..., n)\}, \{g(i_a \mid a = 1, ..., n)\}] \quad (1)$$

$$g(i_a) = \max(i_a) \mid a = 1, ..., n \quad (2)$$

where $\otimes$ denotes the product of elements, i_a is the special diagnosis matrix of item a, and $g(\cdot)$ denotes the max-pooling layer. Both user u and item i have dimension k, the dimension of s_t is $3k$.

4.1.2 Actor–Critic Network

The current state of the user is obtained by the state generation module and used as input of the Actor–Critic network, which generates action information and thus a list of item recommendations for the user. Actor–critic algorithm combines the characteristics of strategy-based and value-based RL algorithms. Actor is responsible for generating actions and interacting with the user based on the state, and Critic is responsible for evaluating the action.

The actor network, known as a policy network, is used to generate actions based on the current state of the user. The actor network in this paper is shown in the left part of Fig. 2. Its input is the state s_t, and the output is the user's action a_t at time t. Specifically, the state s_t is transformed into the action $a_t = \pi_\theta(s_t)$ after two *ReLU* layers and a *Tanh* layer. From the previous section, we know that the dimension of s is $3 * k$, and the action $a_t \in R^{1 \times k}$. Through the action, we can perform a matrix product operation with the item space to obtain the ranking of the items, as shown in Eq. (3):

$$ranking_I = i_t a_t^T \quad (3)$$

where i_t represents the recommended project space. For the final list of recommendation, we take the top N from the ranking. In this process, we use the $\epsilon - greedy$ strategy.

The critic network shown in the right part of Fig. 2, which is the key part of the whole network. The critic network is designed as an approximator to learn the action-value function $Q(s_t, a_t)$, which is used to determine whether the action a_t generated by the actor network matches the state s_t that the user is currently in. Then, according to the action value function, the actor–critic network updates the network parameters in the direction of improving the accuracy of the prediction. It will be helpful to adapt to the state that user is in during subsequent action generation.

Many applications in RL use the most action-value function $Q^*(s_t, a_t)$. Its optimal strategy achieves the maximum expected rewards, and the corresponding Bellman equation is described as follows:

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}[r_t + \gamma max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \mid s_t, a_t] \quad (4)$$

In the real environment the whole action space A needs to be calculated to select the optimal action a_{t+1}. It is impractical to get the best action by computing Eq. (4), because the action space of the real environment is usually huge. So, the critic network in our proposed model uses a definite action a, which is provide by the actor network. This way can avoid the calculation cost of the entire action space A in Eq. (4). Equation (5) is the Q-value evaluation function that we are adopted in this paper:

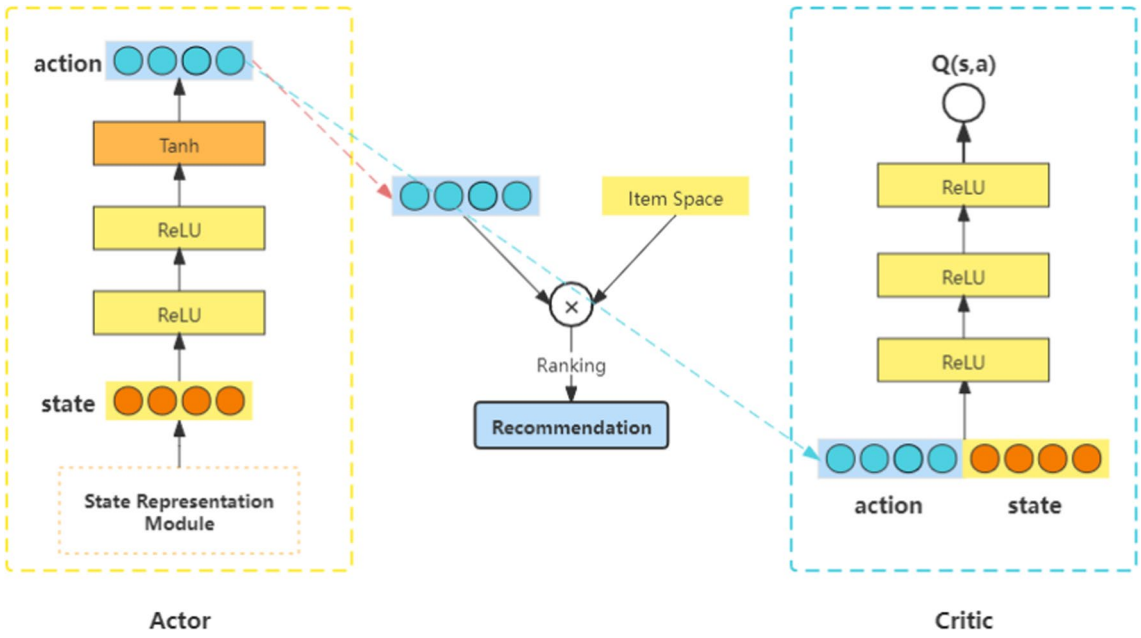


Fig. 2 Actor-critic network

$$Q_\omega(s_t, a_t) = \mathbb{E}_{s_{t+1}}[r_t + \gamma Q_\omega(s_{t+1}, a_{t+1}) \mid s_t, a_t] \tag{5}$$

where $Q_\omega(s_t, a_t)$ represents the evaluation function of the pair of state-action (s_t, a_t), critic network, which also denote the match degree of s_t and a_t. Critic network evaluates the state s generated by the state generation module and the action a generated by the actor network, and then outputs Q-value. According to the Q-value, network parameters can be updated to improving the accuracy of action a, i.e., increasing the value of $Q_\omega(s_t, a_t)$. We update the Actor network parameters by Eq. (6), and we use the sampling policy gradient to execute the update operation.

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{N} \sum_t \nabla_a Q_\omega(s, a) \mid_{s=s_t, a=\nabla_\theta \pi_\theta(s_t)} \nabla_\theta \pi_\theta(s) \mid_{s=s_t} \tag{6}$$

In Eq. (6), $J(\pi_\theta)$ is the expected value of all Q-values that adhere to the π_θ strategy. In the process of updating the critic network, we use a mini-batch strategy and a time difference learning approach. Equation (7) defines the mean squared error, and y_i is shown in Eq. (8).

$$L = \frac{1}{N} \sum_i (y_i - Q_\omega(s_i, a_i))^2 \tag{7}$$

$$y_i = r_i + \gamma Q_{\omega'}(s_{i+1}, \pi_{\theta'}(s_{i+1})) \tag{8}$$

In the above formula, N is the batch size, θ' is a parameter of the Actor network and ω' is a parameter of the Critic network.

4.2 Model Training

In this section, inspired by previous studies [4, 9], we performed the work about the model training in offline and online modes separately. The offline mode is similar to a traditional recommender system, which uses a pre-trained model to make recommendations for the user. The online mode adopts dynamically updated network parameters to simulate the interaction between the system and users in a real recommendation environment.

Algorithm 1 State Generation Module

Input: User embeddings U of size $w \times 1$ and Items embeddings I of size $w \times N$.
Output: User state embeddings s.

1: right = maxpool(I)
2: middle = U*right
3: s = concat(U, middle, right)

4.2.1 Offline Mode

As mentioned above, the offline mode uses a traditional training and testing model. The process of offline training consists of the two steps. First, we use the state generation module in our proposed model to generate the current state of the user. Then, the actor–critic network will be used to get the best result.

Algorithm 1 is the pseudocode of the state generation module. This paper uses PMF [8] to randomly generate initialized user feature matrix U and item feature matrix I as the input of the state generation module. Where $I = \{i_1, i_2, ..., i_n\}, U = \{u_1, u_2, ..., u_n\}$, n is the dimension of the characteristic matrix. Through line 1, we can get the obtain information on the characteristics of I. In line 2, get the user and project interaction information. Merge the user profile information with the above two types of information by line 3. When the Algorithm 1 is finished, we will get the current state of the user.

Algorithm 2 Offine Train Mode

Data: Actor learning rate η_a, critic learning rate η_c, discount factor λ, batch size B, recommend window size K, user space U, item space I.

1: Initialize actor network π_θ and critic network Q_ω with random weights θ add ω ;
2: Initialize the target network $\pi^{'}$ and $Q^{'}$ with weights $\theta^{'} \leftarrow \theta$ and $\omega^{'} \leftarrow \omega$;
3: Initialize replay buffer D;
4: **for** j in U **do**
5: Get user j historical recommended items I_j;
6: Discard users with less than N reviews ;
7: **if** length of $I_j < N$ **then**
8: continue;
9: **end if**
10: Get the current state s_t of j by Algorithm 1;
11: Get recommended actions a_t through the actor network by policy $a_t = \pi_\theta(s_t)$;
12: Calculate recommended item I_k according to equation (3) ;
13: **for** i in I_k **do**
14: **if** User ratings for i **then**
15: Reward r_t=$R(s_t, a_t)$;
16: **else**
17: Reward r_t=$PMF(i, j)$;
18: **end if**
19: **end for**
20: Through the critic network get the $Q_\omega(s_t, a_t)$;
21: Observe new state s_{t+1} by Algorithm 1;
22: Store transition (s_t, a_t, r_t, s_{t+1}) in D;
23: Sample a minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) in D with prioritized experience replay sampling technique;
24: Set y_i by equation (8) ;
25: Update the critic network by minimizing the loss: $L = 1/N \sum_i (y_i - Q_\omega(s_i, a_i))^2$
26: Update the actor network using the sampled policy gradient: $\nabla_\theta(\pi_\theta) \approx 1/N \sum_t \nabla_a Q_\omega(s, a) \mid_{s=s_t, a=\nabla_\theta \pi_\theta(s_t)} \nabla_\theta \pi_\theta(s) \mid_{s=s_t}$;
27: Update the target networks: $\theta^{'} \leftarrow \tau\theta + (1-\tau)\theta^{'} \quad \omega^{'} \leftarrow \tau\omega + (1-\tau)\omega^{'}$ **return** θ, ω ;
28: **end for**

Once we get the current state of the user, we will train the model by the Actor–Critic network in offline mode. The training process of the actor–critic network is described in Algorithm 2. The whole training procedure consists of three phases, which are the action generation, evaluation generation and model updating, respectively. The description of Algorithm 2 is introduced in the following.

The training process starts with random initialization of the network weights and the buffer D (lines 1∼3). The main loop of the Algorithm 2 is shown in lines 4∼28. In the action generation phase (lines 5∼11), we set a minimum number of evaluations N for user selection (lines 6∼9) to obtain enough available information. After obtaining the user feature matrix and the first N historical evaluation feature matrices, the user's state matrix s_t at the current moment can be obtain by Algorithm 1(line 10). The state matrix s_t, is inputted to the actor network to obtain the user's current action a_t. Each loop, we select a user$j$from the user space$U$ to recommend services. Then, the recommended actions a_t, $a_t \in R^{1\times k}$, $k$is the length of item space$I$. is gotten by the actor network applying the policy (line 11).

In the second stage (lines 12∼23), we use the Critic network to calculate the optimal action value function $Q_\omega(s_t, a_t)$. First, we take the top K items, which sorted by the scores obtained through Eq. (3), as the recommendation list (line 12). Next, the reward will be calculated (lines 13∼19). If user rates an item in the recommendation list, the reward function is based on the user's evaluation score. If user does not rate, the PMF predicts the score as a reward. Based on the action a_t at time t, and the state matrix s_t, the optimal action value function $Q_\omega(s_t, a_t)$ is calculated through the Critic network (line 20). The user's next state s_{t+1} is updated based on the reward, and the quaternion (s_t, a_t, r_t, s_{t+1}) is stored in the cache D (lines 21∼23).

In the third stage (lines 24∼27), that is mode updating, the iterative target value is obtained by Eq. (8), and update the Critic and Actor networks according to Eqs. (7) and (6) to

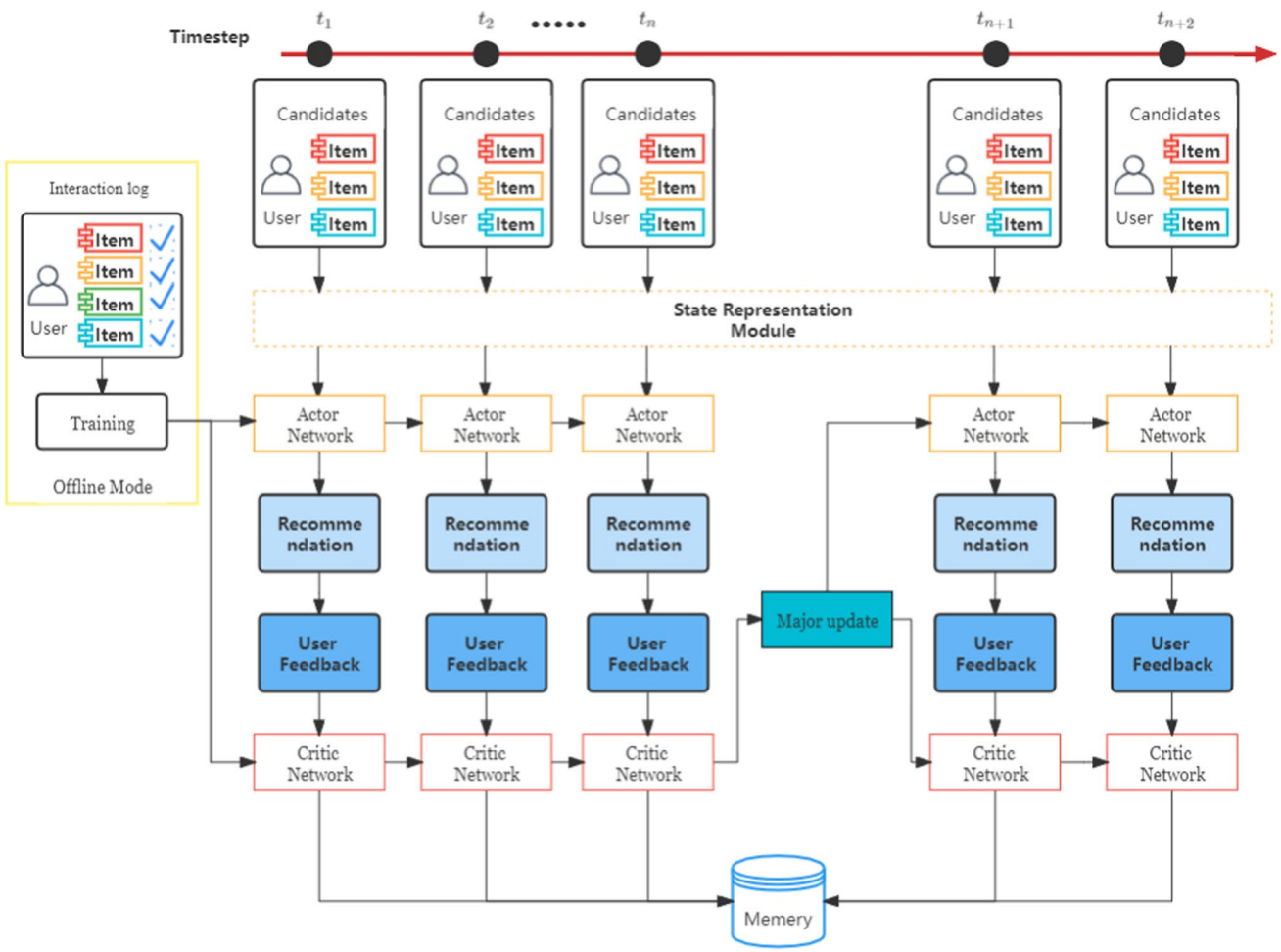


Fig. 3 Online mode

minimize errors (lines 24∼26). Finally, the target networks parameters θ and ω are updated.

4.2.2 Online Mode

Although offline training model is the same as the traditional training, it does not exactly match the real recommendation scenarios in reality and cannot capture the changes in users' interests. Thus, we designs an online recommendation model to update the network synchronously during the recommendation process. The online model employs an update strategy shown in Fig. 3, which uses the users' historical comment data sorted by timestamps, and makes recommendations in chronological order to simulate users' online patterns.

The online training mode is made up of the recommendation stage and the training stage. In the project recommendation stage, the system recommends projects for users according to their interests and preferences. Once the user feedback is obtained, the next recommendation will be made. After finishing several recommendation processes, a certain number of user feedback information will be obtained, and then network parameters are updated through the training step. Therefore, the online training mode is able to simulate the real recommendation environment to a certain extent.

5 Experiments

To determine the validity of DRR-MAX model, we designed two experiments including offline and online recommendation modes. Experimental validation was performed on two publicly available data sets: Book-crossing and Amazon-b. In the offline mode experiment, we adopt three evaluation indicators, including recall rating, precision, and F1 to evaluate the proposed model. Meanwhile, we compared our model with other five models—CDL [39], CMF, PMF [40], DLMR–DAE [41], and HBSADE [42]. For the online mode experiment, we used the reward function to evaluate the effect of recommendation. All algorithms and experiments were designed and implemented using Python 3.8 and Torch 1.9.1 in a computer with an i7-10875 H CPU and RTX3060 GPU.

5.1 Data Set Description

The description of two public real data sets are introduces in the following:

(1) Book-crossing: The Book-crossing data set is a set of rating data sets for books, which contains 1,149,780 ratings for 271,379 books from 278,858 users. The rating of every book ranges from 1 to 10. This data set contains three columns of data, which are user ID, book ISBN and user rating, and each row represents the user's rating for a certain book.

(2) Amazon-b: The Amazon-b data set is a rating data set for books in the Amazon data set, which contains 1,048,576 rating information for 33,122 books by 705,955 users. The value of rating is a number between1 and5. The difference between this data set and the Book-crossing data set is that there is an additional set of timestamp information to record the user's rating time, expect for the user ID, the book ID and the rating.

From the number of users, books and ratings in both data sets, it is easy to see that both data sets are highly sparse. To improve computing efficiency, the rating data has been pre-processed and the rating values have been uniformly normalized to a range between -1 and 1. At the same time, we divided the data set into the training set and test set with an 8:2 ratio.

5.2 Evaluation Metrics

The main purpose of this recommender system is to generate top-N recommended items for users. Therefore, we use precision and recall, which are donated as $Precision@N$ and $Recall@N$, respectively, to evaluate the recommendation quality of the recommender system. In our model, we recommend N items for a user at a given moment in time, and these two metrics are defined in Eqs. (9) and (10).

$$Precision@N = \frac{TP}{TP + FP} \tag{9}$$

$$Recall@N = \frac{TP}{TP + FN} \tag{10}$$

In the above equations, TP represents the number of positive ratings for users in the predicted items, FN is the number of incorrect prediction results, FP denotes the number of negative cases in user evaluation, thus $TP + FP$ denotes the number of user evaluation items and $TP + FN$ denotes the number of recommended items at a time N.

The $F1$ score is also used to assess the balance between $Recall@N$ and $Precision@N$. The formula for calculating $F1$ is described as follows:

$$F1 = 2 * \frac{Precision@N * Recall@N}{Precision@N + Recall@N} \tag{11}$$

While for the online recommendation model simulated, we use the reward function $Reward$ to evaluate the recommendation effect. Due to the limited number of user evaluations, we divide the reward function into two cases whether the

recommended item is rated by users. The two reward functions are defined in Eqs. (12) and (13), respectively. When the recommendation item has a user rating, the reward value is the user rating regularized score, which is assumed a number between 1 and 5. When the recommended item is not rated by the user, we adopt the product of the feature matrices simulated by PMF of the user and the item to obtain the predicted rating of user, and then take the predicted rating as the reward value.

$$Reward\,(r_t) = \frac{1}{2}(rate_{i,j} - 3) \tag{12}$$

$$Reward\,(r_t) = PMF(i,j) \tag{13}$$

5.3 Parameter Settings

For the two data sets, a recommended item is considered as a successful recommendation, when the user rating of it is above 0.5. Before training, a 200-dimensional feature matrix of users and items are first generated randomly using PMF. During the recommendation process, we remove the recommended successful items from the candidate set each time to avoid recommending duplicate items. We set the learning rate of the actor network to 0.0001, the critic network to 0.001, the discount rate γ to 0.9, and the batch size to 64. We also use the *Adam* optimizer to update the network parameters and *L2* paradigm regularization to prevent overfitting.

5.4 Compared Models

We compared our model with several representative methods, including CDL [39], CMF, PMF [40], DLMR–DAE [41] and HBSADE [42], to determine the effectiveness of our model. The details of these five models are as follows.

- CDL: CDL is a tightly coupled hybrid recommendation algorithm, which combines the stack autoencoder SDAE and CTR through a Bayesian graph.
- CMF: CMF is a basic cross-domain recommendation method, which performs cross-domain recommendation by sharing factors among users and decomposing the cross-domain joint scoring matrix.
- PMF: PMF was proposed by [40], which uses the SVD method to decompose the matrix, and the decomposition process to ignore zero values.
- DLMR–DAE: This model obtains information by user reviews, and generates recommendation list by convolution matrix decomposition.
- HBSADE: HABASE is a hybrid Bayesian stacking automatic de-encoder model, which obtains contextual information through the MF method, identifies user's interests, and then makes recommendations based on user interests.

5.5 Results and Analysis

5.5.1 Offline Mode Experiment

From Tables 1, 2, 3, 4, the comparison results on two real public data sets are given. For every model considered in this paper, we can find their precision and recall values. From the

Table 1 Precision for Book-crossing data set

Algorithm	N=5	N=10	N=15	N=20	N=25
PMF	0.107	0.098	0.089	0.085	0.078
CMF	0.185	0.147	0.137	0.124	0.117
CDL	0.190	0.175	0.113	0.123	0.112
DLMR–DAE	0.198	0.195	0.186	0.157	0.158
HBSADE	0.283	0.253	0.256	0.237	0.232
DRR-Max	0.413	0.356	0.321	0.281	0.264

Table 2 Precision for Amazon-b data set

Algorithm	*N*=5	*N*=10	*N*=15	*N*=20	*N*=25
PMF	0.103	0.093	0.082	0.071	0.069
CMF	0.183	0.164	0.132	0.122	0.116
CDL	0.193	0.186	0.124	0.119	0.118
DLMR–DAE	0.295	0.276	0.253	0.234	0.234
HBSADE	0.312	0.297	0.272	0.258	0.258
DRR-Max	0.673	0.548	0.504	0.504	0.518

Table 3 Recall for Book-crossing data set

Algorithm	*N*=5	*N*=10	*N*=15	*N*=20	*N*=25
PMF	0.105	0.114	0.123	0.161	0.134
CMF	0.157	0.168	0.164	0.191	0.197
CDL	0.156	0.177	0.195	0.209	0.217
DLMR–DAE	0.305	0.327	0.344	0.354	0.353
HBSADE	0.327	0.349	0.364	0.382	0.371
DRR-Max	0.257	0.388	0.460	0.481	0.522

Table 4 Recall for Amazon-b data set

Algorithm	*N*=5	*N*=10	*N*=15	*N*=20	*N*=25
PMF	0.109	0.124	0.138	0.151	0.152
CMF	0.203	0.211	0.223	0.226	0.226
CDL	0.193	0.209	0.224	0.236	0.238
DLMR–DAE	0.318	0.352	0.376	0.381	0.382
HBSADE	0.341	0.376	0.382	0.398	0.399
DRR-Max	0.510	0.654	0.698	0.701	0.731

Table 5 F1 for Book-crossing data set

Algorithm	*N*=5	*N*=10	*N*=15	*N*=20	*N*=25
PMF	0.106	0.105	0.103	0.111	0.099
CMF	0.169	0.159	0.149	0.150	0.147
CDL	0.171	0.176	0.143	0.155	0.148
DLMR–DAE	0.240	0.244	0.241	0.218	0.218
HBSADE	0.303	0.293	0.300	0.293	0.285
DRR-Max	0.317	0.371	0.378	0.354	0.350

Table 6 F1 for Amazon-b data set

Algorithm	*N*=5	*N*=10	*N*=15	*N*=20	*N*=25
PMF	0.105	0.106	0.102	0.097	0.095
CMF	0.192	0.185	0.166	0.158	0.153
CDL	0.193	0.197	0.160	0.158	0.157
DLMR–DAE	0.306	0.309	0.302	0.290	0.290
HBSADE	0.326	0.331	0.318	0.313	0.313
DRR-Max	0.580	0.596	0.585	0.586	0.606

experimental results, we can draw the following conclusions. First, the recall and precision of the PMF model is lowest among of these models. Second, CDL improves recommendation performance by introducing a denoising self-encoder, making its recommendation accuracy slightly higher than the PMF. Then, compared with PMF and CDL, CMF does not improve the recommendation accuracy. Although CMF incorporates a composite deep learning CNN model in capturing the correlation between the user's pre and post behavioral information, it has limited effect on improving accuracy. Next, the DLMR–DAE outperforms the PMF, CDL and CMF methods by exploring the fixed interests of users. While for HBASE, it is more effective than DLMR–DAE. This is due to that the HBASE model is enhanced by a hybrid Bayesian overlay auto-decoder model. Finally, our proposed DRR-Max model generally outperforms other five models, expect that the only one results of *Recall*@5 was slightly lower than the DLMR–DAE and HABASE.

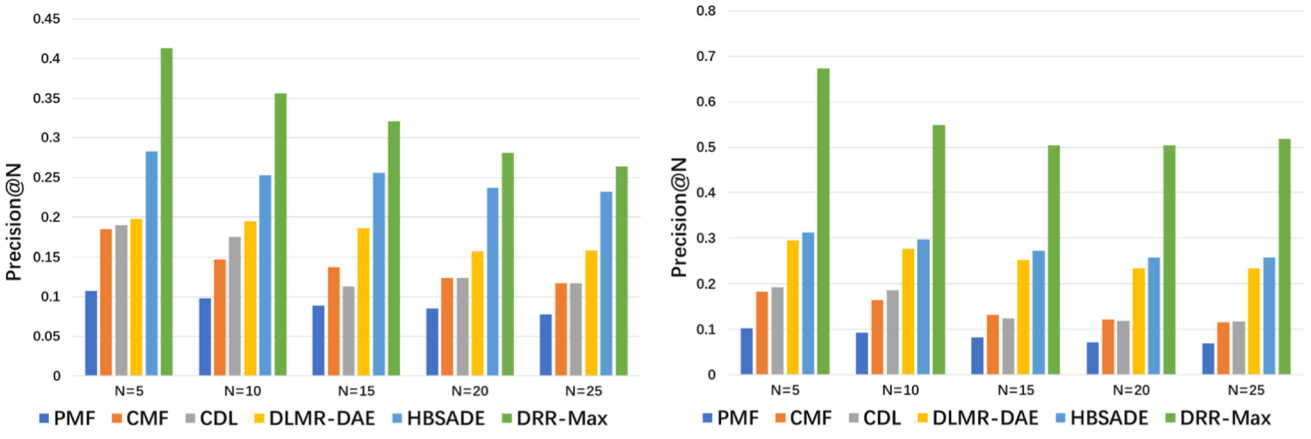
Tables 5 and 6 show the values of *F*1 all the models for different recommended list lengths. The results show that the DRR-Max model is better than other five models, such as CDL, CMF, PMF, DLMR–DAE and HBSADE.

Figure 4 shows the precision metrics of the PMF, CMF, CDL, DLMR–DAE, HBSADE and DRR-Max models on the Book-crossing data set and Amazon-b data set. The DRR-Max has the highest precision values on the two data sets among all models. As shown in Fig. 4, the recommendation accuracy tends to decrease as the length of recommendation list increases. At the same time, the accuracy of small batch recommendations is generally higher than that of large batch recommendations.

Figure 5 shows the results of taking recall as the metric on the Book- crossing data set and Amazon-b data set. The DDR-max model have the larger recall values than other five models except that the length of recommended list equals 5. Unlike the precision in Fig. 4, it can be seen that as the length of recommended list become large, the recall of the model rises.

Figure 6 shows the results on the Book-crossing and Amazon-b data sets, respectively, when the evaluation metric is *F*1. We can see from Fig. 6 that the value of *F*1 decreases when the commendation list *N* increases.

For three metrics used in this paper, we find that they have similar curves of change when the length of commendation list increases. Meanwhile, as shown in Figs. 4 to 6, the proposed model DDR-Max outperforms other five models.



(a) Book Crossing dataset

(b) Amazon-b dataset

Fig. 4 Procession for Book-crossing and Amazon-b data sets

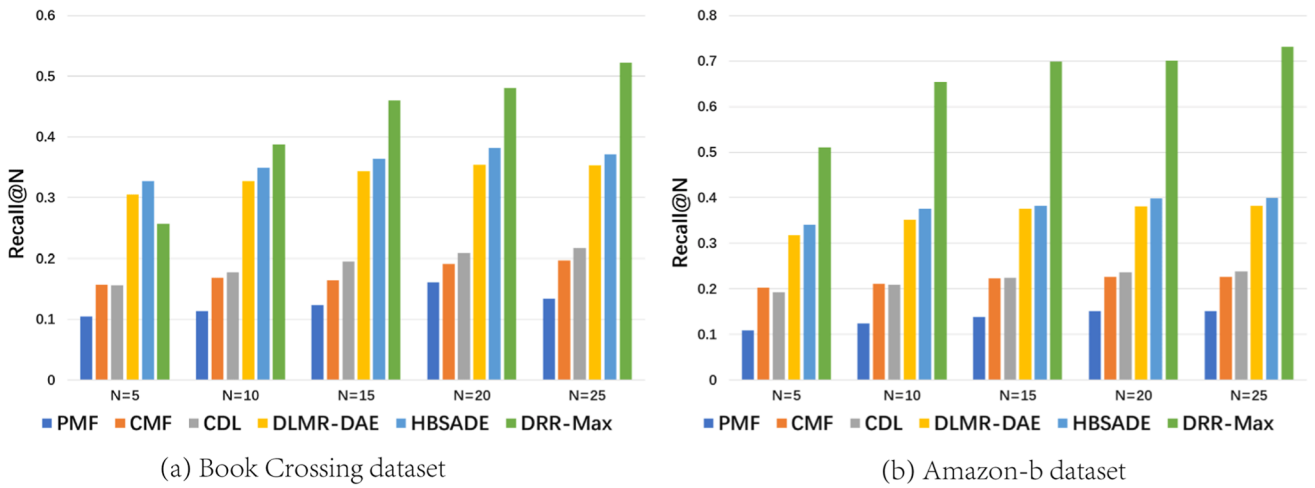


Fig. 5 Recall for Book-crossing and Amazon-b data sets

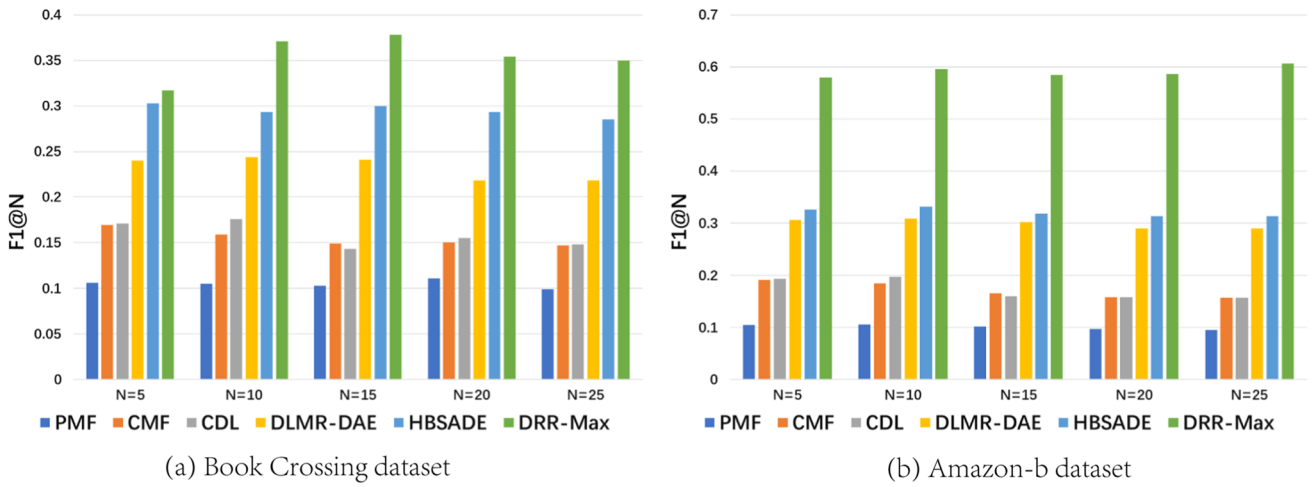


Fig. 6 F1 for Book-crossing and Amazon-b data sets

Table 7 Reward for online mode

Data set	N=5	N=10	N=15	N=20	N=25
Book-crossing	0.5045	0.5029	0.5023	0.5017	0.5015
Amazon-b	0.4983	0.4990	0.4999	0.5010	0.5013

5.5.2 Online Mode Experiment

For the online experiments on Book-crossing and Amazon-b, we use the reward function *Reward* as the evaluation function. At the same time, we also use different sizes of the recommended items list to evaluate the proposed model. Here, we do not compare our proposed model with other models, due to the simulating online experiment. The experiment results are given in Table 7. As shown in Table 7, the reward values obtained are around 0.5 in both data sets, when the length of recommended list has different values. The results reveal that our proposed model is stable as well as effective in the simulated environment of real-time user-system interaction.

6 Conclusion

In this paper, we propose a recommendation model based on deep reinforcement learning (DRR-Max). The specially designed state generation module can obtain the long-term and short-term interest changes of users according to the user's history interaction projects, and generate the current state information of users at the same time. At the same time, we designed two kinds of training experiments, offline and online. Finally, we evaluated our DRR-Max models on two real public data sets:

Book-crossing and Amazon-b. Compared with PMF, CDL, CMF, DLMR–DAE and HBSADE models, DRR-Max model shows good performance in precision, recall and F1. The proposed DRR-Max model also has the advantage of updating the network in real-time when it is deployed.

Future works should focus on extracting multi-dimension information of recommendation items and the users. This information can be used to further classify recommendation items and the users. In additional, more detailed feature information also needs to be obtained to provide the more accurate recommendations.

Declarations

Conflicts of Interest The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Resnick, P., Iacovou, N., Suchak, M., et al.: Grouplens: An open architecture for collaborative filtering of netnews[C]//Proceedings of the. ACM Conf. Comput. Supported Cooperative Work. **1994**, 175–186 (1994). https://doi.org/10.1145/192844.192905
2. Gomez-Uribe, C.A., Hunt, N.: The netflix recommender system: Algorithms, business value, and innovation. ACM Trans. Manage. Inform. Syst. (TMIS) **6**(4), 1–19 (2015). https://doi.org/10.1145/2843948
3. Zhao, X., Zhang, L., Xia, L. et al.: Deep reinforcement learning for list-wise recommendations[J]. arXiv preprint arXiv:1801.00209, 2017. https://doi.org/10.48550/arXiv.1801.00209
4. Zhao, X., Xia, L., Zhang, L. et al.: Deep reinforcement learning for page-wise recommendations. in Proceedings of the 12th ACM Conference on Recommender Systems. 2018: 95-103. https://doi.org/10.1145/3240323.3240374
5. Rostami, M., Oussalah, M., Farrahi, V.: A novel time-aware food recommender-system based on deep learning and graph clustering. IEEE Access (2022). https://doi.org/10.1109/ACCESS.2022.3175317
6. Ahuja, R., Solanki, A., Nayyar, A.: Movie recommender system using k-means clustering and k-nearest neighbor[C]//2019 9th International Conference on Cloud Computing, Data Sci. Eng. (Confluence). IEEE, 2019: 263-268. http://doi.org/10.1109/CONFLUENCE.2019.8776969
7. Phorasim, P., Yu, L.: Movies recommendation system using collaborative filtering and k-means. Int. J. Adv. Comput. Res. 2017, 7(29): 52. http://dx.doi.org/10.19101/ IJACR.2017.729004
8. Kushwaha, N., Sun, X., Singh, B., et al.: A Lesson learned from PMF based approach for Semantic Recommender System[J]. J. Intell. Inform. Syst. **50**(3), 441–453 (2018). https://doi.org/10.1007/s10844-017-0467-2
9. Zheng, G., Zhang, F., Zheng, Z., et al.: DRN: a deep reinforcement learning framework for news recommendation. Proc. World Wide Web Conf. **2018**, 167–176 (2018). https://doi.org/10.1145/3178876.3185994
10. Zhao, X., Zhang, L., Ding, Z. et al.: Recommendations with negative feedback via pairwise deep reinforcement learning[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018: 1040-1048. https://doi.org/10.1145/3219819.3219886
11. Liu, F., Guo, H., Li, X. et al.: 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: 384-392. https://doi.org/10.1145/3336191.3371858
12. Shani, G., Heckerman, D., Brafman, R.I.: An MDP-based recommender system. J. Mach. Learn. Res. **6**, 1265–1295 (2005)
13. Taghipour, N., Kardan, A.: A hybrid web recommender system based on q-learning[C]//Proceedings of the. ACM Symp. Appl. Comput. **2008**, 1164–1168 (2008). https://doi.org/10.1145/1363686.1363954
14. Peters, J., Schaal, S.: Natural actor-critic[J]. Neurocomputing **71**(7–9), 1180–1190 (2008). https://doi.org/10.1016/j.neucom.2007.11.026
15. Deng, J., Guo, J., Wang, Y.: A novel K-medoids clustering recommendation algorithm based on probability distribution for collaborative filtering. Knowl.-Based Syst. **175**, 96–106 (2019). https://doi.org/10.1016/j.knosys.2019.03.009
16. Goldberg, D., Nichols, D., Oki, B.M., et al.: Using collaborative filtering to weave an information tapestry. Commun. ACM **35**(12), 61–70 (1992). https://doi.org/10.1145/138859.138867
17. Chen, Y.C., Hui, L., Thaipisutikul, T.: A collaborative filtering recommendation system with dynamic time decay. J. Supercomput. **77**(1), 244–262 (2021). https://doi.org/10.1007/s11227-020-03266-2
18. Liao, C.L., Lee, S.J.: A clustering based approach to improving the efficiency of collaborative filtering recommendation. Electron. Commerce Res. Appl. **18**, 1–9 (2016). https://doi.org/10.1016/j.elerap.2016.05.001
19. Koohi, H., Kiani, K.: User based collaborative filtering using fuzzy C-means[J]. Measurement **91**, 134–139 (2016). https://doi.org/10.1016/j.measurement.2016.05.058
20. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Trans. Inform. Syst. (TOIS) **22**(1), 143–177 (2004). https://doi.org/10.1145/963770.963776
21. Polatidis, N., Georgiadis, C.K.: A multi-level collaborative filtering method that improves recommendations[J]. Expert Syst. Appl. **48**, 100–110 (2016). https://doi.org/10.1016/j.eswa.2015.11.023
22. Tian, Y., Zheng, B., Wang, Y., et al.: College library personalized recommendation system based on hybrid recommendation algorithm[J]. Proc. CIRP **83**, 490–494 (2019). https://doi.org/10.1016/j.procir.2019.04.126
23. Cai, X., Hu, Z., Zhao, P., et al.: A hybrid recommendation system with many-objective evolutionary algorithm[J]. Expert Syst. Appl. **159**, 113648 (2020). https://doi.org/10.1016/j.eswa.2020.113648
24. Zhang, S., Yao, L., Sun, A., et al.: Deep learning based recommender system: a survey and new perspectives[J]. ACM Comput. Surveys (CSUR) **52**(1), 1–38 (2019). https://doi.org/10.1145/3285029
25. Forouzandeh, S., Rostami, M., Berahmand, K.: A hybrid method for recommendation systems based on tourism with an evolutionary algorithm and topsis model. Fuzzy Inform. Eng. **14**(1), 26–50 (2022). https://doi.org/10.1080/16168658.2021.2019430

26. Zhang, L., Luo, T., Zhang, F., et al.: A recommendation model based on deep neural network. IEEE Access **6**, 9454–9463 (2018). https://doi.org/10.1109/ACCESS.2018.2789866
27. Nassar, N., Jafar, A., Rahhal, Y.: A novel deep multi-criteria collaborative filtering model for recommendation system[J]. Knowl.-Based Syst. **187**, 104811 (2020). https://doi.org/10.1016/j.knosys.2019.06.019
28. Naumov, M., Mudigere, D., Shi, H.J.M. et al.: Deep learning recommendation model for personalization and recommendation systems[J]. arXiv preprint arXiv:1906.00091, 2019. https://doi.org/10.48550/arXiv.1906.00091
29. Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding[C]//Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. 2018: 565-573. https://doi.org/10.1145/3159652.3159656
30. Li, J., Ren, P., Chen, Z. et al.: Neural attentive session-based recommendation[C]//Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. 2017: 1419-1428. https://doi.org/10.48550/arXiv.1711.04725
31. Zhang, S., Tay, Y., Yao, L. et al.: Next item recommendation with self-attentive metric learning. Thirty-Third AAAI Conference on Artificial Intelligence. 2019, 9. https://doi.org/10.48550/arXiv.1808.06414
32. Wu, S., Tang, Y., Zhu, Y. et al.: Session-based recommendation with graph neural networks. in Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33(01): 346-353. https://doi.org/10.1609/aaai.v33i01.3301346
33. Fang, H., Zhang, D., Shu, Y., et al.: Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations[J]. ACM Trans. Inform. Syst. (TOIS) **39**(1), 1–42 (2020). https://doi.org/10.1145/3426723
34. Hidasi, B., Karatzoglou, A., Baltrunas, L. et al.: Session-based recommendations with recurrent neural networks[J]. arXiv preprint arXiv:1511.06939, 2015. https://doi.org/10.48550/arXiv.1511.06939
35. Hidasi, B., Quadrana, M., Karatzoglou, A. et al.: Parallel recurrent neural network architectures for feature-rich session-based recommendations[C]//Proceedings of the 10th ACM conference on recommender systems. 2016: 241-248. https://doi.org/10.1145/2959100.2959167
36. Guo-zhen, L.I.U., Hong-long, C.: Convolutional Memory Graph Collaborative Filtering. J. Beijing Univ. Posts Telecommun., 44(3): 21. https://journal.bupt.edu.cn/EN/Y2021/V44/I3/21
37. Huang, L., Fu, M., Li, F., et al.: A deep reinforcement learning based long-term recommender system[J]. Knowl.-Based Syst. **213**, 106706 (2021). https://doi.org/10.1016/j.knosys.2020.106706
38. Liu, F., Tang, R., Li, X. et al.: Deep reinforcement learning based recommendation with explicit user-item interactions modeling. arXiv preprint arXiv:1810.12027, 2018. https://doi.org/10.48550/arXiv.1810.12027
39. Wang, H., Wang, N., Yeung, D.Y.: Collaborative deep learning for recommender systems. in Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. 2015: 1235-1244. https://doi.org/10.1145/2783258.2783273
40. Salakhutdinov, R., Mnih, A.: Probabilistic Matrix Factorization, Advances in Neural Information Processing Systems 20 (NIPS'07), pp. 1257–1264, 2008
41. Zhou, W., Li, J., Zhang, M. et al.: Deep learning modeling for top-n recommendation with interests exploring. IEEE Access, 2018, 6: 51440-51455. https://doi.org/10.1109/ACCESS.2018.2869924
42. Sivaramakrishnan, N., Subramaniyaswamy, V., Viloria, A., et al.: A deep learning-based hybrid model for recommendation generation and ranking. Neural Comput. Appl. **33**(17), 10719–10736 (2021). https://doi.org/10.1007/s00521-020-04844-4