Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 5564234, 12 pages
https://doi.org/10.1155/2021/5564234

Research Article

Tag-Aware Recommender System Based on Deep Reinforcement Learning

Zhiruo Zhao,[1] **Xiliang Chen**,[1] **Zhixiong Xu**,[2] **and Lei Cao**[1]

[1]*Command & Control Engineering College, Army Engineering University of PLA, Nanjing CO 210000, China*
[2]*Army Academy of Border and Coastal Defense, Xian 710100, China*

Correspondence should be addressed to Lei Cao; caolei.nj@foxmail.com

Received 14 January 2021; Revised 22 April 2021; Accepted 7 May 2021; Published 28 May 2021

Academic Editor: Salman saleem

Recently, the application of deep reinforcement learning in the recommender system is flourishing and stands out by overcoming drawbacks of traditional methods and achieving high recommendation quality. The dynamics, long-term returns, and sparse data issues in the recommender system have been effectively solved. But the application of deep reinforcement learning brings problems of interpretability, overfitting, complex reward function design, and user cold start. This study proposed a tag-aware recommender system based on deep reinforcement learning without complex function design, taking advantage of tags to make up for the interpretability problems existing in the recommender system. Our experiment is carried out on the MovieLens dataset. The result shows that the DRL-based recommender system is superior than traditional algorithms in minimum error, and the application of tags have little effect on accuracy when making up for interpretability. In addition, the DRL-based recommender system has excellent performance on user cold start problems.

1. Introduction

With the increasing amount of information and access to information getting more and more smooth, users' choice towards goods, movies, and restaurants has significantly increased. On the one hand, mass information brings more convenience; on the other hand, information overload brings the trouble of overchoice as well. The recommender system is the information filtering tool that deals with such problem through providing users information with guiding significance in a highly personalized manner [1].

As a subarea of machine learning, deep reinforcement learning-based recommender systems have gained significant attention by overcoming drawbacks of traditional methods and achieving high recommendation quality. Traditional algorithms regard recommendation as a statistic process, while DRL-based recommender algorithms solve the dynamic changes of interest and distribution of users and items. Due to MDP modeling and cumulative rewards, long-term returns are considered to improve user viscosity. The application of the deep neutral network effectively solved sparse data issues in the recommender system.

There are two main DRL algorithms applied in the recommender system, including DQN and actor-critic, which are elaborated in related works. Currently, most successfully applied methods are based on DQN. Considering the Q value-based deep reinforcement learning algorithm is only suitable for low-dimensional and discrete motion spaces, as it is well known that DQN was first proposed in Atari games, which only have four actions. In the rating prediction problem studied in our study, which has ten specific ratings, the Q value-based method is no longer proper. However, the actor-critic-based deep reinforcement learning algorithm is not limited to discrete motion space and even can handle continuous motion space, so the algorithm in this study is under the framework of actor-critic. This trend and the importance of topic motivated us to prepare this study. To be specific, we apply DDPG as our basic algorithm.

However, every coin has two sides. The deep neutral network lacks interpretability and leads to overfitting.

Reinforcement learning needs complex reward function design and hardly handle with user cold start problem. In order to handle with problems caused by the combination of above methods, we propose a tag-aware recommender system based on deep reinforcement learning.

First, modeling recommendation questions (rating predictions) based on deep reinforcement learning. The deep reinforcement learning approach models recommendation issues as a dynamic process. In spatial, it is scalable as the number of users and items increases. In time, it adapts to the dynamic changes in user interests, not only taking short-term returns into account but also long-term returns. Besides, complex data preprocessing is not necessary for the processing of datasets, and the algorithm can automatically learn feature representations from scratch [2].

Second, tag information is used to make up for the interpretability problems existing in the recommender system. By Wikipedia's definition, a label is a nonhierarchical keyword used to describe information, which can be used to describe the semantics of the item. Depending on who labels the item, there are generally two types of labeling applications: one asks the author or expert to label the item and the other allows the regular user to label the item, the latter also called user-generated content. When a user tags an item, the label describes the user's interests on the one hand and the semantics of the item on the other. Users apply labels to describe their views on items, so labels are an important link between users and items. Labels also reflect the interests of users as an important data source; the effective use of labels to improve the quality of personalized recommendation results is of great help [3]. Douban does make good use of label data, increasing both the diversity and interpretability of recommendations.

Finally, user cold start problems [4]. The recommender system needs to predict user's future behavior and interest according to the user's historical behavior and interests, so a large amount of user behavior data become an important part and prerequisite. Designing a personalized recommender system without a large amount of user data is a cold start issue. This study focused on the problem of user cold start, which mainly solves the problem of how to make personalized recommendations for new users. Deep reinforcement learning can dig into the potential connection between user characteristics and item characteristics. Hence, it has potential advantages in solving cold start problems.

Our contributions are listed as follows:

(1) Apply deep reinforcement learning for rating prediction, adapting to the dynamics of users and items, and taking long-term returns into account. To be specific, we use the DDPG algorithm innovatively, which is seldom used in the past.

(2) Complex user label data are cleaned and filtered to make up for the interpretability of the recommender system

(3) The trained neural network has learned the nonlinear relationship between users and items, and meanwhile, we used double network architecture to prevent overfitting, resulting in better adaptability for new users. So, the user cold start issue is relieved to some degree.

The rest of this study is organized as follows. Section 2 briefly reviews the work related to the combination of deep reinforcement learning and recommender systems. Section 3 first defines the recommendation problem according to the deep reinforcement learning and then uses the DDPG algorithm to predict ratings. Section 4 carries out experiment on the MovieLens 20M dataset, and the results show that our algorithm is superior than traditional algorithms in minimum error and has excellent performance on user cold start problems. Section 5 concludes the work in this study and puts forward directions of future work.

2. Related Work

The recommender system is aimed at predicting users' preference on items and recommend items that users may be interested in automatically [5, 6]. Recommendation algorithms are usually classified into three categories [5, 7]: collaborative filtering, content-based, and hybrid recommender system. Collaborative filtering makes recommendations according to users' or items' historical records, either explicit or implicit. Content-based recommendation is on the basis of items and users auxiliary information, such as voice, images, and videos. Hybrid recommendation integrates at least two different recommendation algorithms [7, 8].

Since the Netflix prize competition, different researchers from different countries have come up with numerous rating prediction algorithms. Traditional algorithms include averaging prediction: predictions are made by calculating the average of the ratings. Domain-based approach: predictions are calculated by the similarity of users or items [9]. With the development of machine learning, the latent factor model [10] and matrix factorization [11] are proposed, the essence of which is to study how to complete the rating matrix by the method of dedimensionality. The representative algorithms include SVD, LFM, and SVD++ [12], which fuse time information on the basis of SVD.

Reinforcement learning operates on a trial-and-error paradigm [13]. The basic model is composed of the following components: agents, environments, states, actions, and rewards. The combination of deep neural networks and reinforcement learning formulate DRL which have achieved human-level performance across multiple domains such as games and self-driving cars. Deep neural networks enable the agent to learn from scratch. Common DRL algorithms include DQN, DDQN, dueling DQN, and DDPG.

Recently, DRL has obtained good results [14–16] in the recommender system. Zhao et al. [17] explored the page-wise recommendation scenario with DRL; the proposed framework deep page is able to adaptively optimize a page of items based on user's real-time actions. On this basis, the list-wise method is proposed further [18], and these two articles mainly solve the sorting problem in the recommender system, applying the DDPG framework. Zhao et al.

[19] proposed a DRL framework, DEERS, for recommendation with both negative and positive feedbacks in a sequential interaction setting and especially highlight the importance of negative feedback. This study also demonstrates the effectiveness of proposed framework in real-world e-commerce setting. Zheng et al. [20] proposed a news recommender system, DRN, with DRL to tackle the following three challenges: (1) dynamic changes of news content and user preference, (2) single feedbacks, and (3) diversity of recommendations. This study not only considers click labels or rating into consideration but also take user viscosity into account. Chen et al. [21] proposed a robust deep Q-learning algorithm to address the unstable issue with two strategies: stratified sampling replay and approximate regretted reward. The former idea solves the problem from sample aspect while the latter from reward aspect. DQN-based algorithm alleviates the problem of distribution shifting in dynamic environment, but needs complex reward function. Chen et al. [22] get a more predictive user model and learn the reward function in a way consistent with the user model. Learned rewards function benefits reinforcement learning in a more principled way, rather than relying on hand-designed rewards. The user model makes it possible for model-based RL and online fit for new users, which address the user cold start problem. Although complex reward functions no longer need to be built when using user models, the design of reward functions is still required during the user model building phase. Choi et al. [14] proposed solving the cold start problem with RL and biclustering. This study uses biclustering to improve cold start problem and provides interpretability for the recommender system. Munemasa et al. [15] proposed using DRL for stores recommendation. The state of art survey divides the DRL-based recommender system into three categories according to different DRL algorithms; they are DQN, actor-critic, and reinforce [23].

3. Methods

Considering the behavior of user rating movies is typical of sequential decision, which is in accord with the delayed feedback in reinforcement learning, and we apply reinforcement learning to model recommendation problems. In this study, the dataset of user rating records are viewed as the environment, and the agent needs to perceive the environment when predicting ratings. Reinforcement learning is usually modeled in the form of the Markov decision process (MDP), which is a tuple $\langle S, A, P, R, \gamma \rangle$, so our model is defined as follows.

3.1. Problem Definition

3.1.1. State Space. The state should be able to represent explicit features of users and movies, respectively, and implicit features between users and movies. Based on MovieLens datasets, the dimension of each state is 28 and is sorted in chronological order by rating timestamp. Suppose $s_t \varepsilon S$,

$$s_t = \begin{pmatrix} \text{userId}_t, \text{movieId}_t, \text{ratingtimestamp}_t, \text{imdbId}_t, \text{tmdbId}_t, \\ \text{Genres}_t^{20}, \text{tagId}_t, \text{tagtimestamp}_t, \text{relevance}_t \end{pmatrix}. \tag{1}$$

3.1.2. Action Space. Our goal is to predict users' rating on movies, so we regard ratings as actions directly. The scale of ratings ranges from 0.5 to 5 in half-star; thus, there are 10 discrete ratings in total. Therefore, the action space has 10 actions.

3.1.3. Reward Function. The key of rating prediction is to enhance the accuracy. The larger the difference between predicted rating and actual rating is, the smaller the reward is. On the country, the smaller the difference between predicted rating and actual rating is, the larger the reward is. The reward function in this article is a subtraction of the difference between the prediction rating and the true rating. Since user ratings are the only feedback used, this article does not require complex reward function design and reward shaping.

$$\text{Reward} = e^{-x}. \tag{2}$$

3.1.4. Discount Factor γ. $\gamma \varepsilon [0, 1]$ when $\gamma = 0$, and the recommender system only takes immediate reward into consideration, and when $\gamma = 1$, all future rewards are fully counted.

3.2. DDPG-Based Rating Prediction Algorithm. The full name of DDPG [24] is deep deterministic policy gradient, a combination of actor-critic and DQN algorithms [25]. Deep means using the experience pool and double network structure applied in DQN to promote effective neural network learning. Deterministic, that is, actor no longer outputs the probability of each action, but rather a specific action, which helps us learn in the continuous motion space. The theory basis of our proposed algorithm is based on DDPG, as follows.

Figure 1 shows the network structure of DDPG.

We call the two networks in actor are the action estimation network (rating prediction network) and action reality network. We call the two networks in critic are the state reality network and state estimation network.

DDPG applies a double network structure similar to DQN, and both actor and critic have TargetNet and EvalNet. It is important to emphasize here that we only train the parameters of the action estimation network (rating prediction network) and the state estimation network, while the parameters of the action reality network and the state reality network are copied by the first two networks at a certain time.

First of all, on critic's side, the learning process on critic's side is similar to that of DQN, and we all know that networks in DQN are learned based on the following loss functions,

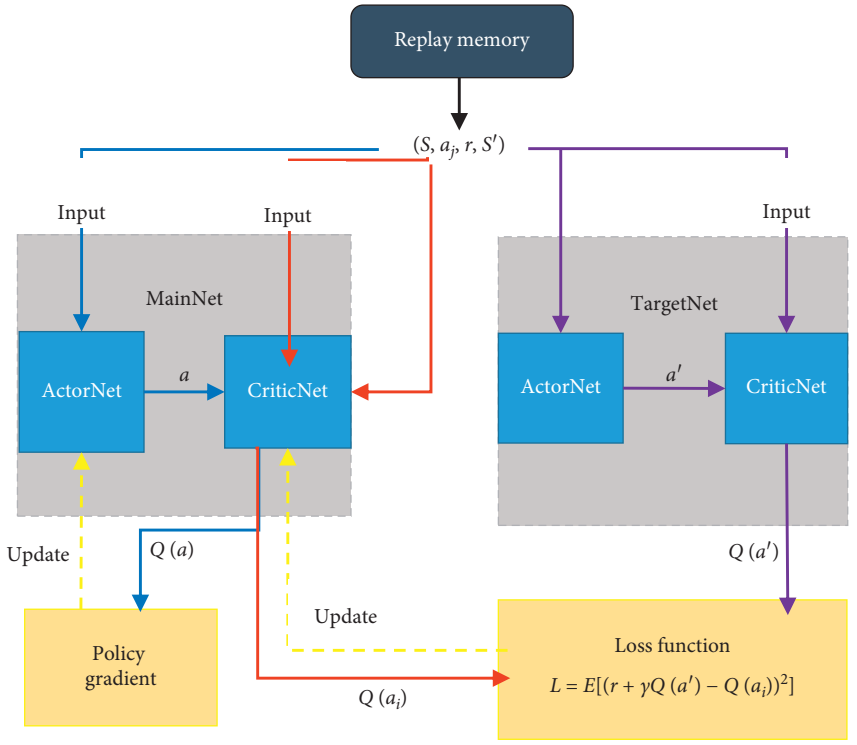

Figure 1: The network structure of DDPG.

namely, the real Q value and the estimated Q value square loss:

$$R + \gamma \max_a Q(s', a) - Q(s, a), \tag{3}$$

where $Q(s, a)$ is obtained from the state estimation network, and a is the action passed over by the action estimation network (rating prediction network).

$R + \gamma \max_a Q(s\prime, a)$ is the real Q value. Instead of using greedy strategy to select action a, we directly get action a through the action reality network.

In general, the training of the critic's state estimation network is based on the square loss of the real Q value and estimated Q value. The estimated Q value is obtained after inputting the current state s and action a, which is outputted by the action estimation network (rating prediction network) to state the estimation network. The real Q value is obtained after putting the reality reward R, the next state $s\prime$, and the action $a\prime$ of the action reality network into the state reality network and then calculated the discount value.

Second, on the actor's side, in this study, we estimate the parameters of the action estimation network (rating prediction network) according to the following formula [24]:

$$\begin{aligned} \nabla_{\theta^\mu} J &\approx E_{s_t \sim \rho^\beta}\left[\nabla_{\theta^\mu} Q\left(s, a|\theta^Q\right)|_{s=s_t, a=\mu(s_t|\theta^\mu)}\right] \\ &= E_{s_t \sim \rho^\beta}\left[\nabla_a Q\left(s, a|\theta^Q\right)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu\left(s|\theta^\mu\right)|_{s=s_t}\right]. \end{aligned} \tag{4}$$

Let us set an example to explain this formula. Suppose to the same state, the action estimation network (rating prediction network) predicts two different ratings $a1$ and $a2$ and gets two feedback Q values from the state estimation network: $Q1$ and $Q2$. Assuming $Q1 > Q2$, that is, rating 1 is closer to the true value and then, according to the idea of policy gradient, increases the probability of action 1 and decreases the probability of action 2. Based on this, actor wants to get as large a Q value as possible. Therefore, the loss of actor can be simply understood as the greater the feedback Q value is, the less the loss is, or the less the feedback Q value is, the greater the loss is.

In addition, the traditional DQN uses a TargetNet network parameter update called "hard" mode, that is, assigning network parameters in EvalNet to TargetNet every certain steps. While, DDPG applies a "soft" mode of TargetNet network parameter updates, that is, each step updates the parameters in the TargetNet network a little bit. This method of parameter updating has been tested, showing that the stability of learning can be greatly improved.

Algorithm 1 is the DDPG-based rating prediction algorithm.

Algorithm 1. DDPG-based rating prediction algorithm.

(1) Randomly initialize the critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ

(2) Initialize the target network $Q\prime$ and $\mu\prime$ weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$

(3) Initialize reply buffer R

(4) **For** episode $e \in 1, 2, 3, \ldots, M$ **do**

(5) Initialize a random process noise for action exploration

(6) Receive initial record s_1

(7) **For** $t \in 0, 1, \ldots$ **do**

(8) Predict rating $a_t = \mu(s_t|\theta^\mu + \text{noise})$ according to the current policy and exploration noise

(9) Apply rating a_t and observe reward r_t and next record s_{t+1}

(10) Store transition (s_t, a_t, r_t, s_{t+1}) in R

(11) Sample a random minibatch of N transitions from R

(12) Set $y_i = r_i + \gamma Q\prime(s_{i+1}, \mu\prime(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$

(13) Update critic by minimizing the loss: $L = (1/N)\sum_i (y_i - Q(s_i, a_i|\theta^Q)^2)$

(14) Update the actor policy using the sampled gradient:

$$\nabla_{\theta^\mu}\mu|_{s_t} \approx (1/N)\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_t}$$

(15) Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^Q$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^\mu$$

(16) End for

(17) End for

4. Experiment

4.1. Dataset. MovieLens datasets are widely used in recommendation research. Our experiment employs the MovieLens 20M dataset, which contains 138493 users' 20 million ratings and tag apps for 27278 movies. Only users with at least 20 ratings are included.

Different from former datasets, 20M datasets do not include any demographic information (age, gender, occupation, and zip code), which is stopped being collected in the site, but include tag applications [26]. Besides, 20M includes a table mapping MovieLens movie IDs to movie IDs in two external sites to allow dataset users build more complete content-based representations of the items.

Table 1 is the overview of ML-20M dataset.

Before the experiment, we preprocessed the dataset:

(1) Tags are words or short phrases applied by users to movies. This study does not use word2vec or any other NLP methods but directly selected 1127 labels most commonly used, according to tags' initials distributing ID number and directly apply tagId as a feature.

(2) This study only selects users who have both ratings and tags for a movie

(3) All features are normalized

(4) Records include both tag and rating

Specifically, our dataset is sorted by rating timestamp in a chronological order, the top 80% data are used for training, and the last 20% data are used for testing.

In order to test the user cold start problem, users in the test set are divided into two parts. 502 users were old users (existing in the training set), and the remaining 960 users were new users (not in the training set). There are 21227 records for old users and 21902 records for new users.

Our experiment datasets are given in Table 2.

The preprocessing of experimental data in this study refers to TRSDL [27]; although the evaluation measures are all the same, due to the different data preprocessing methods, the experimental results are not comparable.

4.2. Evaluation Measures

4.2.1. MAE (Mean Absolute Error). MAE is used to measure the difference between the true rating and the estimated rating of recommendation algorithms.

$$\text{MAE} = \frac{1}{n}\sum_{i\varepsilon U, j\varepsilon I}\left|p_{ij} - r_{ij}\right|. \tag{5}$$

4.2.2. RSME (Root Mean Squared Error). RSME is the evaluation criterion used by Netflix prize. The smaller the value of RSME, the more accurate the algorithm is.

$$\text{RSME} = \sqrt{\frac{1}{n}\sum_{i\varepsilon U, j\varepsilon I}\left(p_{ij} - r_{ij}\right)^2}. \tag{6}$$

4.3. Compared Method. This study selects some classic algorithms in the recommender system for comparative analysis.

4.3.1. Normal Predictor. Algorithm predicting a random rating based on the distribution of the training set is assumed to be normal.

The prediction r_{ui} is generated from a normal distribution $N(\mu, \sigma^2)$, where μ and σ^2 are estimated from the training data using maximum likelihood estimation:

$$\mu = \frac{1}{\left|R_{\text{train}}\right|}\sum_{r_{\text{ui}}\varepsilon R_{\text{train}}} r_{\text{ui}},$$

$$\sigma = \sqrt{\sum_{r_{\text{ui}}\varepsilon R_{\text{train}}} \frac{(r_{\text{ui}} - \mu)^2}{\left|R_{\text{train}}\right|}}. \tag{7}$$

4.3.2. Coclustering. A collaborative filtering algorithm is based on coclustering [28]. Basically, users and items are assigned some clusters C_u, C_i, and some coclusters C_{ui}. The prediction r_{ui} is set as

$$r_{\text{ui}} = \overline{C_{\text{ui}}} + (\mu_u - \overline{C_u}) + (\mu_i - \overline{C_i}), \tag{8}$$

where $\overline{C_{\text{ui}}}$ is the average rating of cocluster C_{ui}, $\overline{C_u}$ is the average rating of u's cluster, and $\overline{C_i}$ is the average rating of i's cluster.

4.3.3. KNN Basic. A basic collaborative filtering algorithm. The prediction r_{ui} is set as

Table 1: ML-20M dataset.

Name	Data range	Rating scale	Users	Movies
ML-20M	1/1995–3/2015	0.5–5, half-stars*	138493	27278
	Ratings	Tag apps	Density (rating)	Density (tag)
	20000263	465564	0.54%	0.012%

*MovieLens changed from a 1–5 full-star scale to a 0.5–5 half-star scale on Feb 18, 2003.

Table 2: Experiment dataset.

Name	Users	Movies	Records	Tags
Dataset	5510	7525	214129	1127
Train set	4550	6802	171000	1127
Test set	1462	4030	43129	1098

$$r_{\mathrm{ui}} = \frac{\sum_{j \in N_u^k} \mathrm{sim}(i,j) \cdot r_{\mathrm{uj}}}{\sum_{j \in N_u^k} \mathrm{sim}(i,j)}. \tag{9}$$

4.3.4. KNN with Means. A basic collaborative filtering algorithm, taking into account the mean ratings of each user. The prediction r_{ui} is set as

$$r_{\mathrm{ui}} = \mu_i + \frac{\sum_{j \in N_u^k(i)} \mathrm{sim}(i,j) \cdot \left(r_{\mathrm{uj}} - \mu_j\right)}{\sum_{j \in N_u^k(i)} \mathrm{sim}(i,j)}. \tag{10}$$

4.3.5. KNN with Baseline. A basic collaborative filtering algorithm taking a baseline rating into account. The prediction r_{ui} is set as

$$r_{\mathrm{ui}} = b_{\mathrm{ui}} + \frac{\sum_{j \in N_u^k(i)} \mathrm{sim}(i,j) \cdot \left(r_{\mathrm{uj}} - \mu_{\mathrm{uj}}\right)}{\sum_{j \in N_u^k(i)} \mathrm{sim}(i,j)}. \tag{11}$$

4.3.6. Slope One. A simple yet accurate collaborative filtering algorithm [29]. The prediction r_{ui} is set as

$$r_{\mathrm{ui}} = \mu_u + \frac{1}{\left|R_i(u)\right|} \sum_{j \in R_i(u)} \mathrm{dev}(i,j), \tag{12}$$

where $R_i(u)$ is the set of relevant items, i.e., the set of items j rated by u that also have at least one common user with i. $\mathrm{dev}(i,j)$ is defined as the average difference between the ratings of i and those of j:

$$\mathrm{dev}(i,j) = \frac{1}{\left|U_{\mathrm{ij}}\right|} \sum_{u \in U_{\mathrm{ij}}} r_{\mathrm{ui}} - r_{\mathrm{uj}}. \tag{13}$$

4.3.7. SVD++. The famous SVD algorithm is popularized by Simon Funk during the Netflix prize. When baselines are not used, this is equivalent to probabilistic matrix factorization [30]. The prediction r_{ui} is set as $r_{\mathrm{ui}} = \mu + b_u + b_i + q_i^T p_u$. If user u is unknown, then the bias b_u and the factors p_u are assumed to be zero. The same applies for item i with b_i and q_i [31, 32].

The SVD++ algorithm is an extension of SVD taking into account implicit ratings.

The prediction r_{ui} is set as

$$r_{\mathrm{ui}} = \mu + b_u + b_i + q_i^T \left(p_u + \left|I_u\right|^{-1/2} \sum_{j \in I_u} y_i \right), \tag{14}$$

where the y_i terms are a new set of item factors that capture implicit ratings. Here, an implicit rating describes the fact that a user u rated an item j, regardless of the rating value.

4.3.8. NMF. A collaborative filtering algorithm is based on nonnegative matrix factorization. This algorithm is very similar to SVD. The prediction r_{ui} is set as

$$r_{\mathrm{ui}} = q_i^T p_u, \tag{15}$$

where user and item factors are kept positive. Our implementation follows that suggested in NMF [33], which is equivalent to [34] in its nonregularized form. Both are direct applications of NMF for dense matrices.

4.3.9. DQN. The neural network (supervised learning) used by DQN is trained using a variant Q-learning algorithm, using SGD to update the weights and replay mechanism that are used to eliminate relevance between data by randomly sampling in the past transitions.

The update formula for the Q value is as follows:

$$Q(S,A) \leftarrow (1-\alpha) * Q(S,A) + \alpha * \left[R + \gamma * \max_a Q(S',a)\right]. \tag{16}$$

4.4. Parameter Setting. We decide the parameters setting according to experience, grid search, and random search. Replay memory size is 10000, batch size is 32, gamma is 0.9, learning rate is 0.001, batch size is 32, tau is 0.001, and episode is 10000.

4.5. Experimental Results. Our experiment is carried out on the processed ML-20M dataset. First, we use the traditional algorithm and DQN as baselines to make comparisons. Then, the tag-aware DDPG algorithm proposed in this study

Table 3: Minimum errors of various algorithms.

Algorithms	MAE	RSME
Normal predictor	1.0422	1.3296
Coclustering	0.6142	0.8221
KNN basic	0.4210	0.6593
KNN with means	0.4564	0.6578
KNN with baseline	0.4141	0.6189
Slope one	0.4231	0.6288
NMF	0.4470	0.6478
SVD++	0.3816	0.5620
DQN	0.4218	0.6256
DDPG (tag-free)	0.3720	0.5432
DDPG (tag-aware)	0.3900	0.5577

Table 4: Performance of DDPG.

RSME	Tag-free	Tag-aware	Cold start
Minimum ($Q1$)	0.54321	0.55771	0.49387
1/4 value ($Q2$)	0.8436	0.85068	0.81359
Median ($Q3$)	0.93408	0.93447	0.92291
3/4 value ($Q4$)	1.02588	1.04036	1.04641
Maximum ($Q5$)	1.49223	1.50439	1.57043
Interquartile range ($Q4$–$Q2$)	0.18228	0.18968	0.23283
Range ($Q5$–$Q1$)	0.94902	0.94668	1.07656

is employed to calculate the error; meanwhile, in order to verify whether tags have effect on error, this study also calculates the error without tags to make comparative analysis. Finally, we select a test set which only contains new users to independently verify the user cold start issue.

Table 3 provides the minimum errors of various algorithms. SVD++ performs best among all traditional algorithms, whose MAE and RSME are 0.3816 and 0.5620. Our algorithm is slightly lower than that of SVD, where the MAE and RSME of tag-free reach 0.3720 and 0.5432 and the MAE and RSME of tag-aware reach 0.3900 and 0.5577. The comparison between DQN and DDPG also shows the superiority of the later.

It is vivid that the best results of our algorithm are superior than traditional methods. In addition to the advantage in reducing error, the DDPG-based recommender system is more scalable when the number and characteristics of users and items enlarge and can adapt to the dynamic changes of users and items as well. What is more, deep learning is good at digging potential connections between users and items, which provide a better idea for optimizing the long-term user experience.

With RSME as the evaluation indicator, DDPG's performance is given in Table 4.

Judging from the performance of DDPG, although it exceeds the traditional algorithm in the minimum value, the robustness is poor. To be specific, the range between minimum value and maximum value is large. Besides, the use of tag apps has little effect on error, but adds interpretability to the recommender system. What is more, when all users in test set are new users, DDPG shows excellent performance. More details will be analyzed.

4.5.1. Tag-Free and Tag-Aware. Table 5 and Figure 2 separately show the comparison results of tag-free and tag-aware from table format and figure formats.

Since reinforcement learning learns from scratch, the initial predicted ratings are rather random, causing the training error very large at first. However, with the increase of training time, reinforcement learning gradually learned the correct strategy. The error is decreased and stabilized.

Figures 3 and 4 show the RSME of tag-free and tag-aware, respectively. The average RSME of tag-free is 0.9448, and the best RSME is 0.5274. The average RSME of tag-aware is 0.9456, and the best RSME is 0.5577. The results show that the application of tag has little effect on the accuracy; however, it makes up for the drawback of interpretability existing in the recommender system.

Figures 5 and 6 show the distribution of RSME of tag-free and tag-aware. The error distribution follows the normal distribution, and the number of errors on the left side of the mean is greater than the right side, that is, most errors are concentrated in the interval with smaller errors. It can be seen that DDPG algorithms tend to have smaller errors, which means, more accurate.

4.5.2. Cold Start. Table 6 and Figure 7 separately show the results of cold start from table format and figure formats.

Figure 8 shows the distribution of RSME of cold start. When the test set only contains new users, the accuracy of DDPG is still very high. It can be speculated that deep reinforcement learning can be a good solution to the problem of user cold start. This is something the former algorithms cannot solve.

Table 5: Comparison results of tag-free and tag-aware.

Name	MAE		RSME	
	Average	Best	Average	Best
Tag-free	0.7165	0.3720	0.9448	0.5432
Tag-aware	0.7143	0.3900	0.9456	0.5577

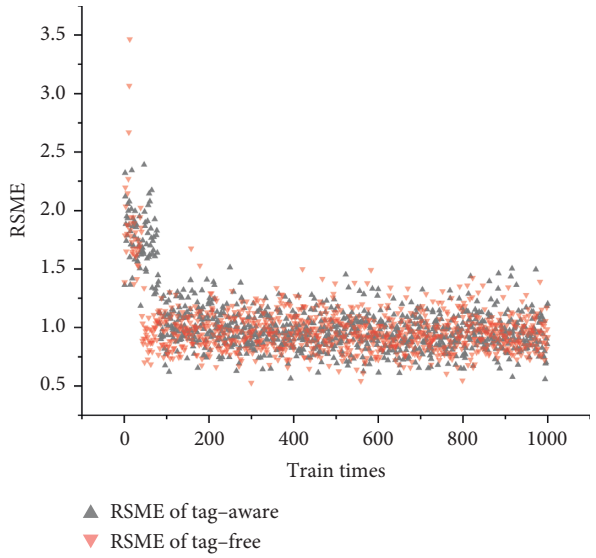

Figure 2: Comparison results of tag-free and tag-aware.

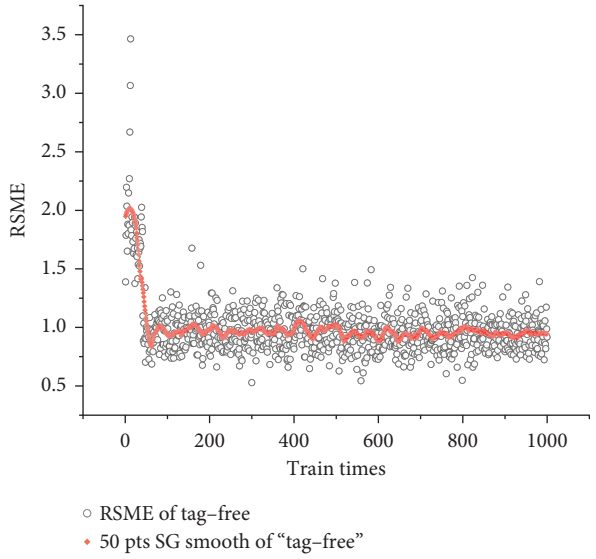

Figure 3: RSME of tag-free.

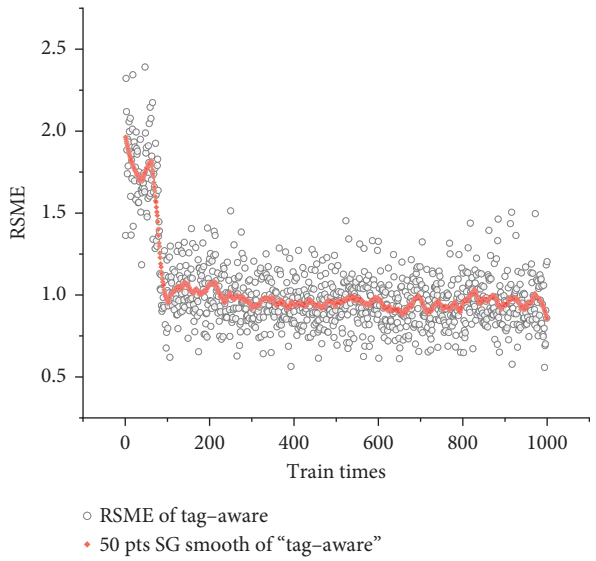

Figure 4: RSME of tag-aware.

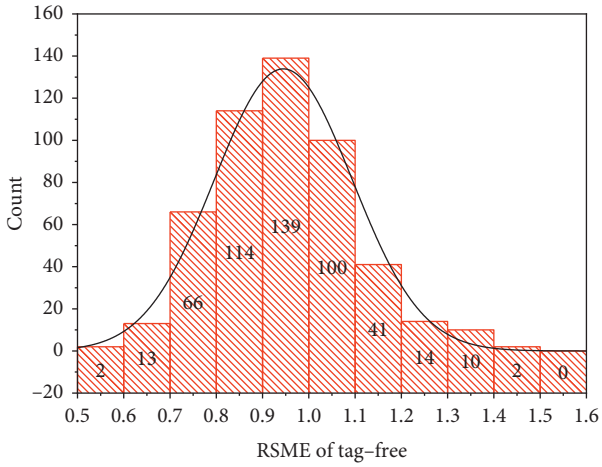

Figure 5: The distribution of RSME of tag-free.

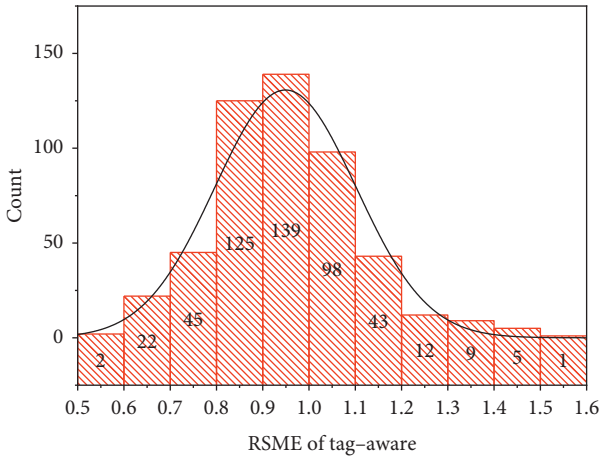

Figure 6: The distribution of RSME of tag-aware.

Table 6: The results of cold start.

Name	MAE		RSME	
	Average	Best	Average	Best
Cold start	0.7044	0.3600	0.9388	0.4939

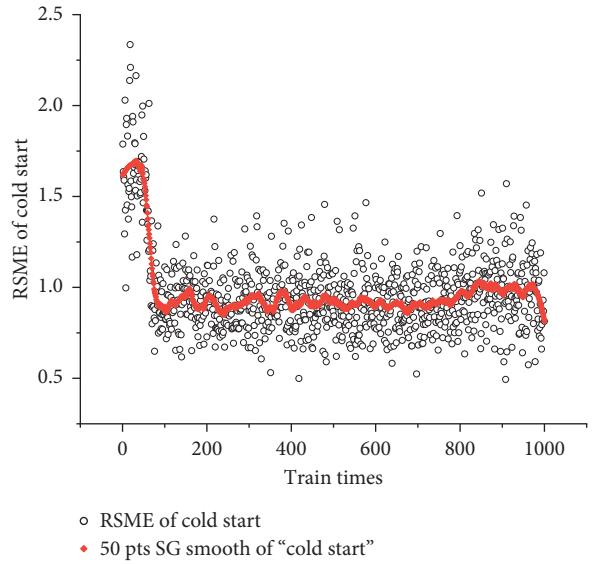

Figure 7: RSME of cold start.

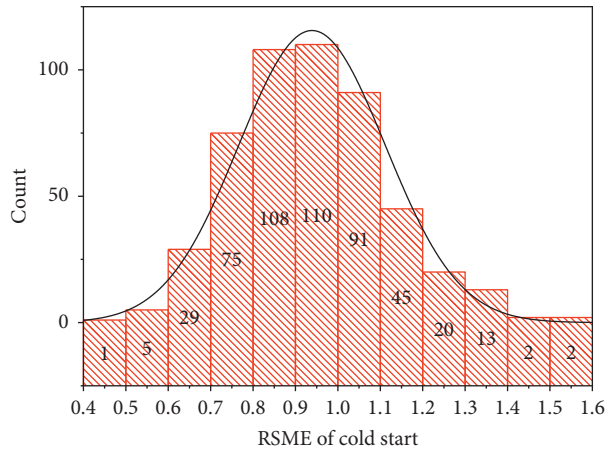

Figure 8: The distribution of RSME of cold start.

DDPG shows a lower error in dealing with the user cold start problem, which shows that the method adopted in our study has a good effect on the problem of overfitting.

The error distribution follows the normal distribution, and DDPG algorithms also tend to have smaller errors.

To sum up, the DDPG-based recommender system has three advantages:

(a) Enhance accuracy and reduce errors

(b) Increase explanatory

(c) Reduce overfitting and solve cold start problems

5. Conclusion

The combination of deep reinforcement learning and recommender systems has become a popular trend, and Internet giants such as Google and Alibaba both have performed a lot in theoretical exploration and engineering practice in. In this study, the DDPG algorithm is applied to predict the rating in the recommender system. Since the basic algorithm of DDPG is generally used to deal with large-scale continuous action, this study first discretizes the continuous action, which is the rating of movies. Although

the average error is higher than traditional algorithms, the minimum error is much smaller than the existing recommendation algorithm, and the results of this experiment tend to have smaller errors. Then, without increasing the error, tag is used to make up for the interpretability of the recommender system. Finally, on the issue of user cold start, the experiment proves that the recommendation algorithm used in this study has smaller errors, and it also has a good effect on the overfit problem.

For future work, we have the following directions. (1) Scalability. This study uses the MovieLens 20M dataset, and we can continue to research on the 25M and latest datasets to explore the scalability problems. (2) Robustness. Although the error of the DDPG algorithm converges to a great result, the error range is large; hence, there is room for improvement of the robustness. (3) Parameters. The DDPG algorithm requires a lot of tuning, which is a common disease of machine learning. We want to propose more adaptable recommended algorithms.

Data Availability

The data used to support the findings of this study are available in MovieLens datasets (https://grouplens.org/datasets/movielens/).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61806221).

References

[1] J. B. Schafer, J. A. Konstan, and J. Riedl, "E-commerce recommendation applications," *Applications of Data Mining to Electronic Commerce*, vol. 5, no. 1/2, pp. 115–153, 2001.

[2] S. Zhang, L. Yao, A. Sun, and T. Yi, "Deep learning based recommender system: a survey and new perspectives," *ACM Computing Surveys*, vol. 52, no. 1, pp. 1–38, 2017.

[3] Y. Zuo, J. Zeng, M. Gong, and L. Jiao, "Tag-aware recommender systems based on deep neural networks," *Neurocomputing*, vol. 204, pp. 51–60, 2016.

[4] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based hybrid recommendation for cold start problem," in *Proceedings of the 2016 IEEE 14th International Conference on Dependable, Autonomic and Secure Computing*, pp. 874–877, Auckland, New Zealand, 2016.

[5] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[6] F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems: Introduction and Challenges Recommender Systems Handbook*, Springer, Berlin, Germany, 2015.

[7] D. Jannach, M. Zanker, F. Alexander, and G. Friedrich, *Recommender Systems: An Introduction*.Cambridge University Press, Cambridge, UK, 2010.

[8] R. Burke, "Hybrid recommender systems: survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[9] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[10] T. Bansal, D. Belanger, and A. McCallum, "Ask the guru: multi-task learning for deep text recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, vol. 107–114, Boston, MA, USA, 2016.

[11] M. V. Baalen, "Deep matrix factorization for recommendation," Master's thesis, University of Amsterdam, Amsterdam, Netherlands, 2016.

[12] S. Zhang, L. Yao, and X. Xu, "AutoSVD++: an efficient hybrid collaborative filtering model via contractive autoencoders," in *Proceedings of the 40th International ACM SIGIR Conference*, Shinjuku, Japan, 2017.

[13] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Article ID 7540, 2015.

[14] S. Choi, H. Ha, U. Hwang, C. Kim, J.-W. Ha, and S. Yoon, "Reinforcement learning based recommender system using biclustering technique," 2018, https://arxiv.org/abs/1801.05532.

[15] I. Munemasa, Y. Tomomatsu, K. Hayashi, and T. Takagi, "Deep Reinforcement Learning for Recommender Systems," in *Proceedings of the 2018 International Conference on Information and Communications Technology*, Yogyakarta, Indonesia, 2018.

[16] X. Wang, Yi Wang, D. Hsu, and Ye Wang, "Exploration in interactive personalized music recommendation: a reinforcement learning approach," *ACM Transactions on Multimedia Computing, Communications, and Applications TOMM*, vol. 11, no. 1, pp. 1–22, 2014.

[17] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for page-wise recommendations," 2018, https://arxiv.org/abs/1805.02343.

[18] X. Zhao, L. Zhang, Z. Ding, D. Yin, Y. Zhao, and J. Tang, "Deep reinforcement learning for list-wise recommendations," 2018, https://arxiv.org/abs/1801.00209.

[19] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and 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*, London, UK, 2018.

[20] G. Zheng, F. Zhang, Z. Zheng et al., "DRN: a deep reinforcement learning framework for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*, Lyon, France, 2018.

[21] S.-Y. Chen, Y. Yang, D. Qing, J. Tan, H.-K. Huang, and 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*, London, UK, 2018.

[22] X. Chen, S. Li, H. Li, Y. Qi, and L. Song, "Generative adversarial user model for reinforcement learning based recommendation system," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 21, Long Beach, CA, USA, 2019.

[23] M. M. Afsar, T. Crump, and B. Far, "Reinforcement learning based recommender systems: a survey," 2021, https://arxiv.org/abs/2101.06286.

[24] T. P. Lillicrap, J. J. Hunt, P. Alexander et al., "Continuous control with deep reinforcement learning," 2016, https://arxiv.org/abs/1509.02971.

[25] D. Silver, G. Lever, N. Heess, D. Thomas, W. Daan, and R. Martin, "Deterministic Policy Gradient Algorithms," in *Proceedings of the 31st International Conference on Machine Learning*, Beijing, China, 2014.

[26] F. Maxwell Harper, J. A. Konstan et al., "The MovieLens datasets: history and context," in *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, pp. 1–19, 2015.

[27] N. Liang, H.-T. Zheng, J.-Y. Chen, A. Sangaiah, and C.-Z. Zhao, "TRSDL: tag-aware recommender system based on deep learning-intelligent computing systems," *Applied Sciences*, vol. 8, no. 5, p. 799, 2018.

[28] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," in *Proceedings of the Fifth IEEE International Conference on Data Mining*, Hong Kong, China, 2005.

[29] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," in *Proceedings of the 2005 SIAM International Conference on Data Mining*, pp. 21–23, Newport Beach, CA, USA, 2007.

[30] R. R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorizations," in *Proceedings of the Neural Information Processing Systems*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., vol. 20, pp. 1257–1264, Vancouver, Canada, 2008.

[31] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2019.

[32] L. Baltrunas, B. Ludwig, and F. Ricci, "Matrix factorization techniques for context aware recommendation," in *Proceedings of the RecSys'11ACM Conference on Recommender Systems*, Chicago, IL, USA, 2012.

[33] R. Liao, Y. Zhang, J. Guan, and S. Zhou, "CloudNMF: a MapReduce implementation of nonnegative matrix factorization for large-scale biological datasets," *Genomics, Proteomics & Bioinformatics*, vol. 12, no. 1, pp. 48–51, 2014.

[34] M. Heiler and C. Schnörr, "Learning sparse representations by nonnegative matrix factorization and sequential cone programming," *Journal of Machine Learning Research*, vol. 7, no. 3, pp. 1385–1407, 2006.