Tag-Aware Recommender System Based on Deep Reinforcement Learning

Zhiruo Zhao[1], Lei Cao[1], Xiliang Chen[1], Zhixiong Xu[1]
[1]Command & Control Engineering College, Army Engineering University of PLA, Nanjing, CO 210000 China

Abstract: Recently, the application of deep reinforcement learning in 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 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 paper 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 recommender system. Our experiment is carried out on MovieLens dataset. The result shows that, DRL based recommender system is superior than traditional algorithms in minimum error and the application of tags has little effect on accuracy when making up for interpretability. In addition, DRL based recommender system has excellent performance on user cold start problems.

Keywords: recommender system; tag-ware; deep reinforcement learning; user cold start

1. Introduction

With the increasing amount of information and access to information getting more and more smooth, users' choice towards goods, movies, restaurants, etc. has significantly increased. On one hand, mass information brings more convenience. On the other, information overload brings the trouble of over-choice as well. Recommender system are information filtering tools that deal with such problem through providing users information with guiding significance in a highly personalized manner[1].

As a sub area 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 deep neutral network effectively solved sparse data issues in recommender system.

However, every coin has two sides. Deep neutral network lacks interpretability and tends to overfitting. Reinforcement learning need complex reward function design, and hardly handle with user cold start problem. Currently, most algorithms combined reinforcement learning with recommender system are based on DQN. Considering the Q-value based deep reinforcement learning algorithm is only suitable for low-dimensional, 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 paper, which has ten specific ratings, the Q-value based method is no longer proper. However, Actor Critic based deep reinforcement learning algorithm is not limited to discrete motion space, and even can handle continuous motion space, so algorithm in this paper is under the framework of Actor Critic. To be specific, we apply DDPG as our basic algorithm.

The research of recommender system includes Top-N recommendation and rating prediction, this paper focuses on rating prediction. In this paper, we propose a tag-aware recommender system based on deep reinforcement learning.

Firstly, 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's 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 data sets and the algorithm can automatically learn feature representations from scratch[6].

Secondly, tag information is used to make up for the interpretability problems existing in recommender system. By Wikipedia's definition, a label is a non-hierarchical 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, 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[7]. Douban does make good use of label data, increasing both the diversity and interpretability of recommendations.

Finally, user cold start problems.[8] Recommender system need 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 becomes an important part and prerequisite. Designing a personalized recommender system without a large amount of user data is a cold start issue. This paper 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.
2. Use tag apps to make up for the interpretability of recommender system.
3. Resolve the user cold start issue.

The rest of this paper 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 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 paper and put forward directions of future work.

2. Related work

2.1. Traditional recommender algorithms and rating prediction

From GroupLens to Netflix, then to Yahoo! Music's KDD Cup, rating prediction has always been a hotspot in recommender system. The basic dataset for rating prediction is the user-rating dataset. The dataset consists of users' rating records, each record is a triple (u, i, r), indicating that user u gives item i a rating r. Because it is impossible for users to rate all items and new users come up every day, the key of rating prediction is to predict unknown user-rating records through history records. For example, there has two users AB, three movies CDE. User A's rating to movie C is 2, to movie D is 5 and movie E is not rated. User B is a new user without any rating records. When User A browses the web seeing Movie E, we want to help User A decide whether to see this movie by a predicted rating. When User B browses the web, we want to help the user decide which movie to watch by predicting user B's rating towards movie CDE.

Recommender systems predict users' preference on items and recommend items that users may be interested in automatically [9][12]. Recommendation algorithms are usually classified into three categories[9][11]: 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, like voice, images and videos. Hybrid recommendation integrates at least two different recommendation algorithms[10][11].

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[2]. With the development of machine learning, latent Factor Model[3] and Matrix Factorization[4] are proposed, the essence of which is to study how to complete the rating matrix by the method of de-dimensionality. The representative algorithms include SVD, LFM, and SVD++[5], which fuses time information on the basis of SVD.

2.2 DRL based recommender system

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.

Recently, DRL has obtained good results[14][15][16] in 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 recommender system, applying the DDPG framework. Zhao et al.

[19]proposed a DRL framework, DEERS, for recommendation with both negative and positive feedback in a sequential interaction setting, and especially highlight the importance of negative feedback. This paper also demonstrate the effectiveness of proposed framework in real-world e-commerce setting. Zheng et al. [20] [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; (3) diversity of recommendations. This paper not only consider 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 solve the problem from sample aspect while the latter from reward aspect. DQN based algorithm alleviates the problem of distribution shifting in dynamic environment, but need complex reward function. Chen et al.[22] get a more predictive user model and learn the reward function in a way consistent with user model. Learned rewards function benefits reinforcement learning in a more principled way, rather than relying on hand-designed rewards. 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, but 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 bi-clustering. This paper using bi-clustering to improve cold start problem and provide interpretability for recommender system. Munemasa et al [15] proposed using DRL for stores recommendation.

3. Method

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

3.1. problem definition

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 $\mathrm{s}_\mathrm{t} \in \mathrm{S}$,

$$s_t = (userId_t, movieId_t, ratingtimestamp_t, imdbId_t, tmdbId_t,\\ Genres_t^{20}, tagId_t, tagtimestamp_t, relevance_t)$$

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.

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 subtract 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.

$$reward = e^{-x}$$

Discount factor γ**:** $\gamma \epsilon [0,1]$when $\gamma = 0$, 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[23] is Deep Deterministic Policy Gradient, a combination of Actor-Critic and DQN algorithms. 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.

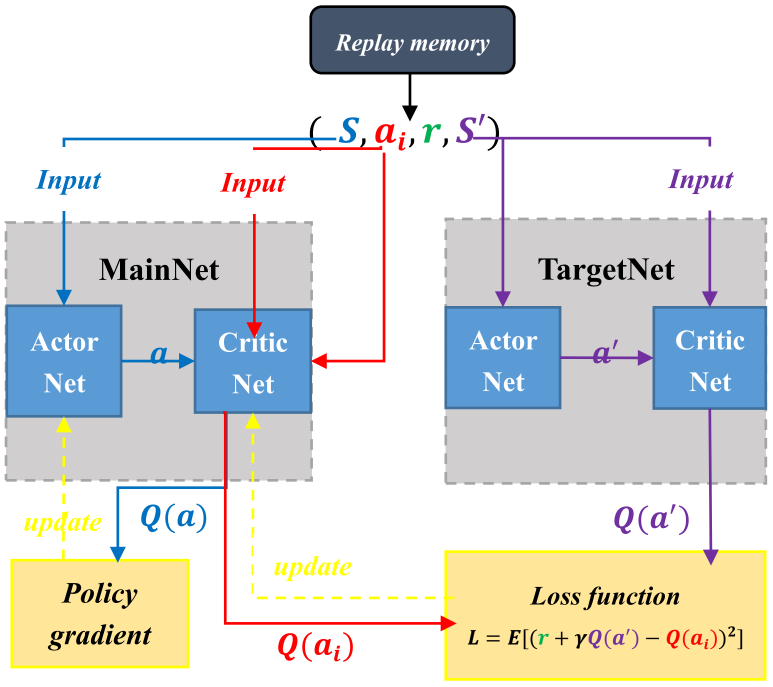


Figure 1 The network structure of DDPG

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

DDPG applies a double network structure similar to DQN, both Actor and Critic have target-net and eval-net. 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, namely the real Q value and the estimated Q value square loss:

$$R + \gamma max_a Q(s', a) - Q(s, a)$$

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

$R + \gamma max_a Q(s', a)$ 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 Critic's state estimation network is based on the square loss of 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 action estimation network (rating prediction network) to state estimation network. The real Q value is obtained after putting the reality reward R, the next state s' and the action a' of the action reality network into the state reality network and then calculated the discount value.

Secondly, on the Actor's side, in this paper, we estimate the parameters of the action estimation network (rating prediction network) according to the following formula[23]:

$$\nabla_{\theta^\mu} J \approx E_{s_t \sim \rho^\beta}[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^\mu)}]$$

$$= E_{s_t \sim \rho^\beta}[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_t}]$$

Let’s 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. 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 target-net network parameter update called 'hard' mode, that is, assigning network parameters in eval-net to target-net every certain steps. While DDPG applies a 'soft' mode of target-net network parameter updates, that is, each step updates the parameters in the target-net network a little bit. This method of parameter updating has been tested and showing that the stability of learning can be greatly improved.

Algorithm 1 DDPG based rating prediction algorithm

```
1   Randomly initialize critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights
    $\theta^Q$ and $\theta^\mu$
2   Initialize the target network $Q'$ and $\mu'$ weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
3   Initialize reply buffer $R$
4   **For** episode $e \in 1,2,3,\dots M$ **do**
5     Initialize a random process Noise for action exploration
6     Receive initial record $s_1$
7     **For** $t \in 0,1,\dots$ **do**
8       Predict rating $a_t = \mu(s_t|\theta^\mu + Noise)$ 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 $y_i = r_i + \gamma Q'(s_{i+1,} \mu'(s_{i+1,} | \theta^{\mu'}) | \theta^{Q'})$
13        Update critic by minimizing the loss: $L = \frac{1}{N} \sum_i \left( y_i - Q(s_i, a_i | \theta^Q)^2 \right)$
14        Update the actor policy using the sampled gradient:
              $\nabla_{\theta^\mu} \mu|_{s_t} \approx \frac{1}{N} \sum_i \nabla_a Q(s, a | \theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_t}$
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, zip code), which is stopped being collected in the site, but include tag applications[24]. 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.

Name	Data Range	Rating Scale	Users	Movies
ml-20m	1/1995-3/2015	0.5-5, half-stars*	138,493	27,278
	Ratings	Tag Apps	Density (Rating)	Density (Tag)
	20,000,263	465,564	0.54%	0.012%

Table 1 ml-20m dataset

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

Before the experiment, we preprocessed the dataset:

1. Tags are words or short phrases applied by users to movies. This paper 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 paper 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 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 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 21,227 records for old users and 21,902 records for new users.

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

Table 2 Experiment dataset

The preprocessing of experimental data in this paper refers to TRSDL[32], 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

MAE (mean absolute error)

MAE is used to measure the difference between the true rating and the estimated rating of recommendation algorithms.

$$MAE = \frac{1}{n} \sum_{i \epsilon U, j \epsilon I} |p_{ij} - r_{ij}|$$

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.

$$RSME = \sqrt{\frac{1}{n} \sum_{i \epsilon U, j \epsilon I} (p_{ij} - r_{ij})^2}$$

4.3. compared method

This paper selects some classic algorithms in the recommender system for comparative analysis.

Normal predictor: Algorithm predicting a random rating based on the distribution of the training set, which 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:

$$\mu = \frac{1}{|R_{train}|} \sum_{r_{ui} \epsilon R_{train}} r_{ui}$$

$$\sigma = \sqrt{\sum_{r_{ui} \epsilon R_{train}} \frac{(r_{ui} - \mu)^2}{|R_{train}|}}$$

Co-clustering: A collaborative filtering algorithm based on co-clustering[31]. Basically, users and items are assigned some clusters C_u, C_i, and some co-clusters C_{ui}. The prediction r_{ui} is set as:

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

where $\overline{C_{ui}}$ is the average rating of co-cluster 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.

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

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

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

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

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

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

Slope one: A simple yet accurate collaborative filtering algorithm[30]. The prediction r_{ui} is set as:

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

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. $dev(i,j)$is defined as the average difference between the ratings of i and those of j:

$$dev(i,j) = \frac{1}{|U_{ij}|}\sum_{u\in U_{ij}} r_{ui} - r_{uj}$$

SVD++: The famous SVD algorithm, as popularized by Simon Funk during the Netflix Prize. When baselines are not used, this is equivalent to Probabilistic Matrix Factorization.[25]The prediction r_{ui} is set as: $r_{ui} = \mu + b_u + b_i + q_i^T p_u$. 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[26][27].

The SVD++ algorithm, an extension of SVD taking into account implicit ratings. The prediction r_{ui} is set as:

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

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.

NMF: A collaborative filtering algorithm based on Non-negative Matrix Factorization. This algorithm is very similar to SVD. The prediction r_{ui} is set as:

$$r_{ui} = q_i^T p_u$$

where user and item factors are kept positive. Our implementation follows that suggested in NMF[28], which is equivalent to [29] in its non-regularized form. Both are direct applications of NMF for dense matrices.

4.4. Experimental results

Our experiment is carried out on the processed ml-20m dataset. First, we use the traditional algorithm as baselines to make comparisons. Then tag-aware DDPG algorithm proposed in this paper are employed to calculate the error, meanwhile, in order to verify whether tags have effect on error, this paper 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.

The minimum errors of various algorithms are as below:

Algorithms	MAE	RSME
Normal predictor	1.0422	1.3296
Co Clustering	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
DDPG (tag-free)	0.3720	0.5432
DDPG (tag-aware)	0.3900	0.5577

Table 3 Minimum errors of various algorithms

Table 3 shows 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, the MAE and RSME of tag-aware reach 0.3900 and 0.5577.

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

With RSME as the evaluation indicator, DDPG's performance is shown in the table below:

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

Table 4 Performance of DDPG

Judging from the performance of DDPG, although it exceeds the traditional algorithm in the minimum value, the robustness of which is poor, and the range of 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's more, when all users in test set are new users, DDPG shows excellent performance. More detail will be analyzed as below.

a) 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

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

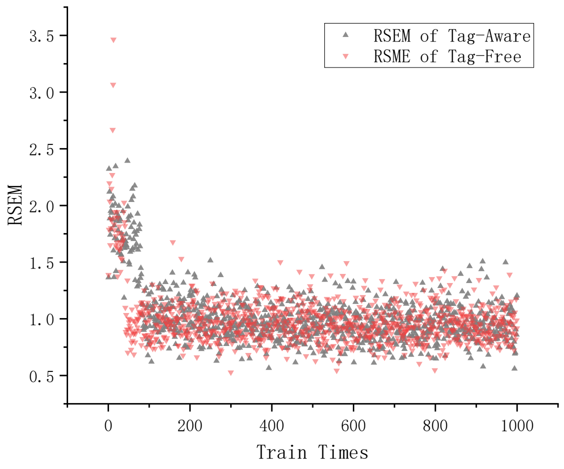
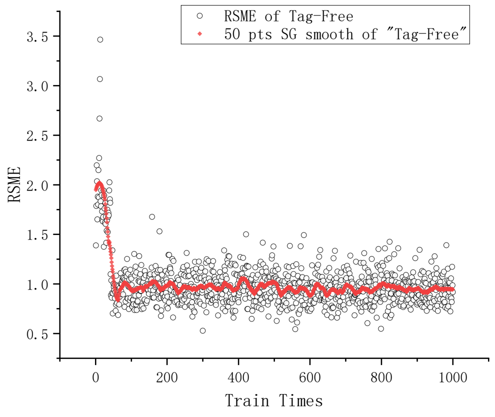


Figure 2 Comparison results of tag-free and tag-aware



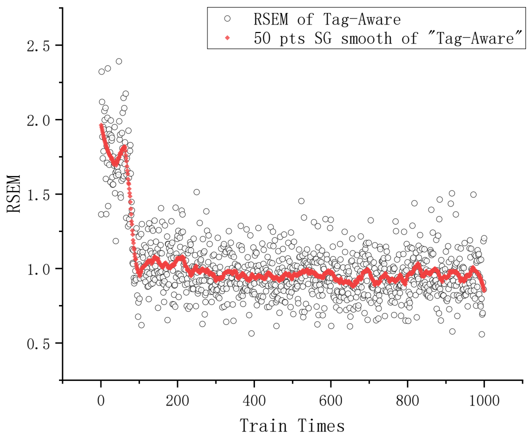
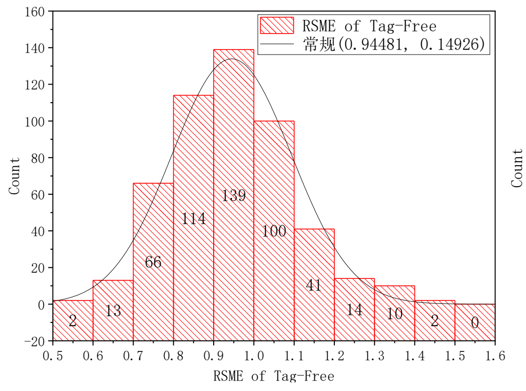


Figure 3 RSME of tag-free **Figure 4 RSME of tag-aware**

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 decreased and stabilized.

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 recommender system.



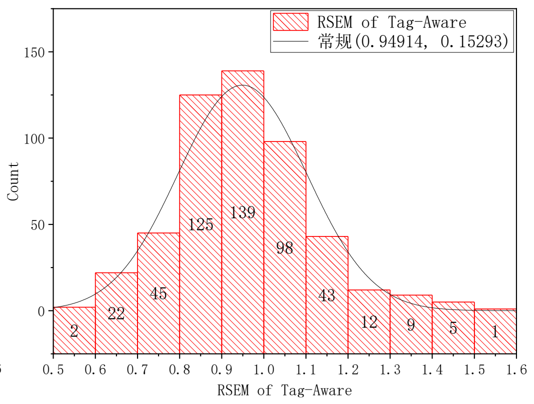


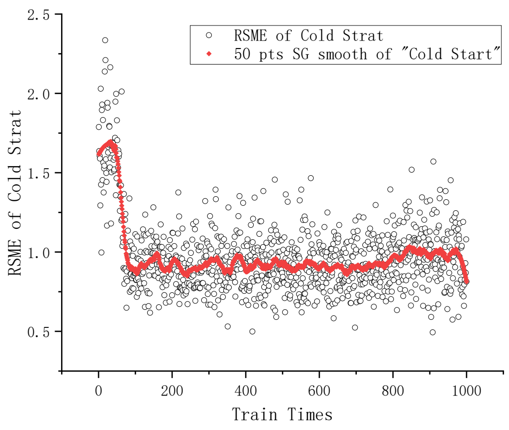
Figure 5 The distribution of RSME of tag-free **Figure 6 The distribution of RSME of tag-aware**

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.

b) Cold start:

Name	MAE		RSME	
	average	best	average	best
Cold Start	0.7044	0.3600	0.9388	0.4939

Table 6 The results of Cold Start



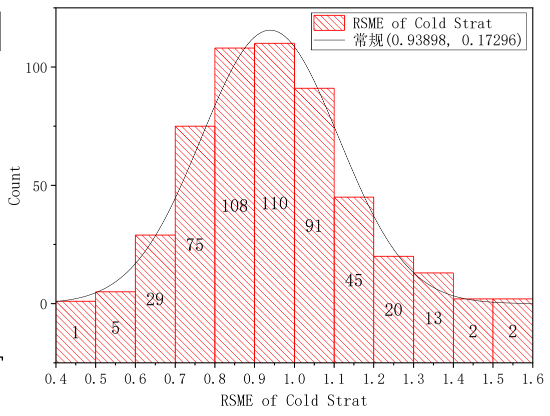


Figure 7 RSME of cold start **Figure 8 The distribution of RSME of cold start**

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.

DDPG shows a lower error in dealing with the user cold start problem, which shows that the method adopted in our paper 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.

5. Conclusion

The combination of deep reinforcement learning and recommender systems has become a popular trend, and internet giant like Google and Alibaba both have done a lot in theoretical exploration and engineering practice in. In this paper, 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 paper 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 paper 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 paper uses the MovieLens 20M dataset, we can continue to research on the 25M and latest datasets to explore the scalability problems. 2) Robustness. Although the error of 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.

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

References

[1] Schafer JB, Konstan JA, Riedl J (2001) E-commerce recommendation applications. Data Min Knowl Discov 5(1/2):115–153.

[2] Linden G, Smith B, York J (2003) Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput 7(1):76–80

[3] Trapit Bansal, David Belanger, and Andrew McCallum (2016) Ask the gru: Multi-task learning for deep text recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems. 107–114

[4] Baalen MV (2016) Deep matrix factorization for recommendation. Master's thesis, University of Amsterdam

[5] Shuai Zhang, Lina Yao, and Xiwei Xu (2017) AutoSVD++: An Efficient Hybrid Collaborative Filtering Model via Contractive Autoencoders.

[6] Zhang S , Yao L , Sun A , et al. (2017) Deep Learning Based Recommender System: A Survey and New Perspectives[J]. ACM Computing Surveys

[7] Yi Zuo, Jiulin Zeng, Maoguo Gong, and Licheng Jiao. (2016). Tag-aware recommender systems based on deep neural networks. Neurocomputing 204, 51–60

[8] Jian Wei, Jianhua He, Kai Chen, Yi Zhou, and Zuoyin Tang(2016) Collaborative filtering and deep learning based hybrid recommendation for cold start problem. IEEE, 874–877.

[9] Gediminas Adomavicius and Alexander Tuzhilin (2005) 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 17, 6, 734–749.

[10] Robin Burke (2002) Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction 12, 4, 331–370

[11] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich (2010) Recommender systems: an introduction.

[12] Francesco Ricci, Lior Rokach, and Bracha Shapira (2015) Recommender systems:

introduction and challenges. In Recommender systems handbook. 1–34.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, and others (2015) Human-level control through deep reinforcement learning. Nature 518, 7540, 529.

[14] Sungwoon Choi, Heonseok Ha, Uiwon Hwang, Chanju Kim, Jung-Woo Ha, and Sungroh Yoon (2018) Reinforcement Learning based Recommender System using Biclustering Technique. arXiv preprint arXiv:1801.05532.

[15] Isshu Munemasa, Yuta Tomomatsu, Kunioki Hayashi, and Tomohiro Takagi (2018) Deep Reinforcement Learning for Recommender Systems.

[16] Xinxi Wang, Yi Wang, David Hsu, and Ye Wang (2014) Exploration in interactive personalized music recommendation: a reinforcement learning approach. TOMM 11, 1, 7

[17] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang (2018) Deep Reinforcement Learning for Page-wise Recommendations. arXiv preprint arXiv:1805.02343.

[18] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Dawi Yin, Yihong Zhao, Jiliang Tang(2018) Deep Reinforcement Learning for List-wise Recommendations. ACM

[19] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin (2018) Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning. arXiv preprint arXiv:1802.06501.

[20] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li (2018) DRN: A Deep Reinforcement Learning Framework for News Recommendation. In WWW. 167–6.

[21] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang (2018) Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In SIGKDD. 1187–1196

[22] Chen, Xinshi, S. Li, H. Li, S. Jiang, Yuan Qi and L. Song. Generative Adversarial User Model for Reinforcement Learning Based Recommendation System.21 ICML, 2019.

[23] Timothy P. Lillicrap∗ , Jonathan J. Hunt∗ , Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver & Daan Wierstra (2016) Continuous control with deep reinforcement learning, ICLR

[24] F, MAXWELL, HARPER, et al. (2015) The MovieLens Datasets: History and Context[J]. Acm Transactions on Interactive Intelligent Systems.

[25] R. R. Salakhutdinov and A. Mnih. (2008)Probabilistic matrix factorizations. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, Advances in Neural Information Processing Systems, volume 20, pages 1257–1264.

[26] Koren Y , Bell R , Volinsky C. (2019) Matrix Factorization Techniques for Recommender Systems[J]. Computer, 42(8):30-37.

[27] Baltrunas L , Ludwig B , Ricci F . (2012) Matrix factorization techniques for context aware recommendation[C] RecSys'11ACM conference on recommender systems.

[28] Liao R , Zhang Y , Guan J , et al. (2014) CloudNMF: A MapReduce Implementation of Nonnegative Matrix Factorization for Large-scale Biological Datasets[J]. Genomics Proteomics & Bioinformatics, 12(1):48-51.

[29] Heiler M , Christoph Schnörr. (2006) Learning Sparse Representations by NonNegative

Matrix Factorization and Sequential Cone Programming[J]. Journal of Machine Learning Research, 7(3):1385-1407.

[30] Lemire D , Maclachlan A . (2007) Slope One Predictors for Online Rating-Based Collaborative Filtering[J]. Computer Science:21--23.

[31] George T , Merugu S . (2005) A scalable collaborative filtering framework based on co-clustering[C]IEEE International Conference on Data Mining. IEEE.

[32] Liang N , Zheng H T , Chen J Y , et al. (2018)TRSDL: Tag-Aware Recommender System Based on Deep Learning–Intelligent Computing Systems[J]. Applied Sciences, 8(5):799.