Deep Reinforcement Learning based Recommend System using stratified sampling

Zhiruo Zhao and Xiliang Chen
Command & Control Engineering College, Army Engineering university, No. 1, HaiFu Road, Guang Hua Road, Qin Huai District, Nanjing City, Jiangsu Province, 210007, China
Email: 383618393@qq.com

Abstract. Typically personalized movie recommendation algorithms often adopt a static view of recommend process and only take current rewards into consideration. Thus, they are hard to adapt to the dynamic change of users and items. In this paper, we propose a movie recommend system based on deep reinforcement learning to better accommodate the dynamic property when users’ distribution or interest changes. Firstly, we adopt nature DQN algorithm to set up baseline. Second, under the framework of nature DQN, we use Double DQN to solve overestimation and indeed reduce error. Besides, we use stratified sampling rather than random sampling to accelerate convergence. Finally, by testing on Movielens dataset, the experimental results shows that our algorithm is superior to traditional algorithms, and also comparable to the latest algorithms.

1. Introduction
In the era of information explosion, information overload has become more and more serious with the increasing way of information publishing. Especially, in the field of video display, according to Youtube’s data released in 2013, the total length of video content uploaded by users has already reached 100 hours per minute. Netflix, a movie rental site, even offers millions of dollars in a reward for recommendation algorithms and architectures which can raise performance of recommend system up by 10%. It’s obviously that personalized online movie recommendation is essential to improve users’ experience.

Currently widely used recommendation algorithms include content-based method[1][2][3], collaborative filtering method[5][6] and hybrid method[7][8][9]. But there are still some problems in them: Firstly, the existing algorithm is difficult to deal with the dynamic change of recommend information, which is embodied in two aspects: one is the characteristics of information and the candidate set of information usually change with time, the other is the interest of users may change with time as well. Although some researchers have built online updating recommendation models, these models usually only optimize instant rewards and didn’t consider the impact of this optimization on future cumulative rewards. Second, the existing algorithm cannot fully make use of the entire attributes of users and items to excavate the possible coupling relation because the attributes of users differ greatly from items.

The essence of recommend system is to help users make correct and reasonable decision, considering reinforcement learning can capture the real-time change, it provides a good way for us to solve problem mentioned above. In literature Reinforcement Learning beyond Games: To Make a Difference in Alibaba, Alibaba builds an adaptive online recommendation model based on deep reinforcement learning. By means of continuous reinforcement learning and model optimization, Ali
establishes a recommend decision engine. By real-time extraction and analysis of the features of mass users and products, the matching efficiency get highly improved. In paper the application of reinforcement learning in the recommendation system in Taobao, Markov Decision Process (MDP) is proposed to model the real time recommend system. Moreover, in order to solve inaccurate evaluation strategy resulted by the changing distribution of users, and poor strategy to be considered as good strategy because of user groups varying widely, stratified sampling instead of random sampling has been proposed[4].

This paper is based on Movielens-1M dataset. Firstly we establish a virtual interactive environment of movie recommendation. Then we model the behaviour of uses rating movies based on deep reinforcement learning method. The main idea of our model is as follows:
1. We propose nature DQN as basic algorithm and set up baseline.
2. Double q-learning is introduced to relieve problem of overestimating the value function. Then, we establish a new movie recommendation model based on double DQN.
3. Stratified sampling instead of random sampling is applied to deal with the problem of large variation of user distribution and user groups.

![Figure 1. Deep Reinforcement Learning Recommend System](image)

Our recommend system can be shown in figure 1. In our system, environment is made up of three datasets given by Movielens-1M: 'Users', 'Movies' and 'Ratings'. Our recommendation algorithms, DQN, acts as the agent. The state, which is ordered by timestamp, contains features of users and movies. Action is defined as the rating of a movie and reward is defined as a function of the difference value between real score and forecast score.

The rest of this paper is organized as follow: in section 2, we introduce related work. Section 3 demonstrates our recommend model. In section 4, we analysis experimental results. Finally, we show the conclusion and our intending future work in Section 5.

2. Related Work

2.1. Reinforcement Learning

Reinforcement Learning (RL) is a machine learning method in which the system learns from the environment to maximize rewards. Standard reinforcement learning and can be described as Markov decision process (MDP). MDP can be defined as a five-tuple group $(S, A, \pi, R, \gamma)$. At each time step, agent receives the state $s_t$, and then selects an action from the possible action set $A$ according to policy $\pi$. $\pi$ is a mapping from $s_t$ to $a_t$. Accordingly, agent will receive the next state and a reward. The whole process will continue until agent reaches final state and restarts. The returned value $R_t=\sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the cumulative reward calculated at the time step $t$ using the discount factor $\gamma \in (0, 1)$. The goal of agent is to maximize expected earnings at each state $s_t$.

The basic idea of reinforcement learning algorithm is to estimate the action value function by using Bellman equation as iterative update.
This iterative algorithm converges to the optimal action value function $Q_i \rightarrow Q^*$ when $i \rightarrow \infty$ [13]. In practice, because action value functions are estimated individually for each sequence and difficult to be obtained, so function approximations are usually used to estimate action value functions:

$$Q(s, a; \theta) = Q^*(s, a)$$

Linear function approximation is usually used, and sometimes nonlinear function approximation such as neural network is used. When neural network is adopted, $\theta$ denotes the approximate function of neural network with weight $\theta$.

The parameter $\theta$ of action value function $Q(s, a; \theta)$ is calculated by minimizing loss function. The loss function is defined as:

$$L_i(\theta_i) = E_{s', \epsilon} (y_i - Q(s, a; \theta_i))^2$$

$s'$ is the state after $s$, $y_i = E_{s', \epsilon} (r + \gamma \max_{a'} Q^*(s', a'; \theta_{i-1}))$. To obtain the optimal action is to optimize the loss function $L_i(\theta_i)$ under the condition that the parameters $\theta_{i-1}$ of the previous network are fixed. The gradient formula is obtained by differentiating the parameters of the loss function:

$$\nabla_{\theta} L_i(\theta_i) = E[(r + \gamma \max_{a'} Q^*(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)] \nabla_{\theta} Q(s, a; \theta_i)$$

2.2. Experience Replay in DQN

In 2013 DQN algorithm has been proposed by Mnih V on the basis of NIPS[12]. For one thing, deep neural network are used to approximate the action value function. For another, experience replay mechanism are used, the data obtained from the exploration environment is stored in the form of Memory Unit $(s_t, a_t, r_{t+1}, s_{t+1})$ and then randomly selecting samples from Experience Replay Memory to update (train) neural network parameters. This algorithm is called Nature DQN. However, in experiments with nature DQN, researchers found that there is an unstable phenomenon in the approximation of $Q$ function, and the main reasons are as follows: First, once the data of the observation sequence is relevant, it will lead to the failure of the optimization algorithm which is based on gradient descent. Second, even the slight change in $Q$ function of the training can lead to a drastic change in strategy, making the algorithm hard to converge. The experience replay mechanism in nature DQN solves the relativity problem of the observation sequence. It first stores the data in the exploration environment, then randomly selects samples from the stored data to update the parameters of the deep neural network. The solution of the second problem is put forward by Mnih V in 2015[12]. By means of target DQN iterative updating, the correlation of data is further reduced:

$$\min(r + \gamma \max_a Q(s_{j+1}, a_j; \theta) - Q(s_j, a_j; \theta))^2$$

This method uses the parameter delay update of the deep Q network to reduce the influence of the jitter of Q Network on training, and reduce the relativity between Q network and target Q network.

2.3. Double DQN

The idea of Double Q-learning is to separate the action selecting network from the action evaluating network, thus reduces overestimation. Double DQN is a combination of Double Q-learning and DQN:

$$Y_t^{DDQN} = r + \gamma Q(S_{t+1}, \arg\max Q(S_{t+1}, a; \theta_t); \theta_t)$$

In this equation, $\theta_t$ is updated in real time and copied to $\theta_{t+1}$ at every C step.
3. Intelligent Recommendation Algorithm Based on Deep Reinforcement Learning

3.1. Problem Definition

Considering the behavior of user scoring movies has sequential decision properties, which accords with the delayed feedback in reinforcement learning, we apply reinforcement learning to establish our recommendation model. Reinforcement learning is usually modeled in the form of Markov decision process (MDP), so our model is defined as the follows[11]:

3.1.1. Define the state space

In this paper, user's rating of movie is viewed as the environment and agent needs to perceive the environment when making decision. The state should be able to represent users and long-term features of movies. The features of user include UserID, Gender, Age, Occupation, Zip-code and the features of movie include MovieID, Genre. Also, because all rating records are sorted by time sequence, it’s necessary to add timestamp in state. Suppose s represents the state, there are.

\[ S = (UserID_1, MovieID_1, Gender_1, Age_1, Occupation_1, Zip-code_1, Genre_1 \ldots UserID_n, MovieID_n, Gender_n, Age_n, Occupation_n, Zip-code_n, Genre_n) \]

3.1.2. Define the action space

The goal of our paper is to predict user's rating on movies. Then, based on the forecast rating, we can recommend movies to users. Because the rating of a user for a movie is an integer value in \([1, 5]\), the last layer of DQN network output the Q value of 5 actions.

3.1.3. Setting the reward function

Once the state space and action space are defined, the state transition function is also determined, so there remains reward function needed to be defined. Since we’ve known user's real rating on movies, it’s easy to compare forecast rating with real rating. When difference between forecast rating and real rating is large, the environment will receive a negative reward. And the larger this difference is, the lower reward value will be given. On the contrary, when forecast rating is close to real rating, the environment will receive a positive reward. And the smaller the difference is, the higher reward is given. Considering both forecast ratings and real ratings are integers in \([1, 5]\), our reward function is defined as below:

\[
R = \begin{cases} 
3, & d=0 \\
0, & d=1 \\
-1, & d=2 \\
-2, & d=3 \\
-3, & d=4 \\
\end{cases}
\]

\[ d = |\text{real rating} - \text{forecast rating}| \]

In order to improve the accuracy of our algorithm, we introduce more information into the reward function to enlarge the distinction of different actions. For a movie score, movies with the same attributes might have a diametrically opposed score. Therefore, on the basis of the original, we add the historical attribute of the movie to the definition of reward function, and enrich the information by the Reward Shaping. The idea of Reward Shaping is to introduce some prior knowledge in the original reward function to speed up the convergence of the reinforcement learning algorithm. Simply, we can define the reward value of ‘Select Action a on state s and move to s’ as

\[ R(s, a, s') = R_0(s, a, s') + \phi(s) \]

\( R(s, a, s') \) is the original reward function, \( \phi(s) \) is a function that contains a priori knowledge, also called a Potential Function. We can regard \( \phi(s) \) as the Local Objective in the learning process. Here is a specific definition:
φ(s)=e^{-(r_{ui}-\bar{r}_i)}

\( r_{ui} \) is the user u's predictive rating for movie i, \( \bar{r}_i \) is the average user's score on the movie i in training set.

Based on the above definition, our reinforcement learning simulation environment on basis of Movie lens data is established in this paper.

3.2. DQN Recommend System

On account of the dynamic feature of movies recommendation and the need to estimate future reward, we apply a Deep Q-Network to predict the movie ratings scored by users. The reward is defined as a function of the difference value between the actual score and the forecast score. Therefore the total reward is:

\[ Q(s,a) = r_{immediate} + \gamma r_{future} \]

In this function, state s is made up of user features, movie features and timestamp. Action is a score of prediction. \( r_{immediate} \) represents the rewards for current situation, while \( r_{future} \) represents the agent’s projection of future rewards. \( \gamma \) is a discount factor to balance the relative importance of immediate rewards and future rewards. After figuring out the meaning of each parameter we use DDQN algorithm to predict the total rewards by taking action at time t:

\[ Y_{s,a,t}^{DDQN} = r_{a,t+1} + \gamma Q(S_{a,t+1}, \text{argmax}_a Q(S_{a,t+1}, a'; \theta_t); \theta_t') \]

In this function, \( r_{a,t+1} \) represents the immediate reward by taking action a. \( \theta_t \) and \( \theta_t' \) are two different sets of parameters of the DQN. In this formulation, agent will choose an action as a forecast rating on some given state.

3.3. A Method of Mini Batch Construction Based on Stratified Sampling

Usually, in the deep RL, relay memory is used to preserve historical samples, like Nature DQN. In previous problems, the distribution of environment was always fixed. However, in our problem, the agent is a recommendation system, the environment is the majority of users and the strategy is to predict the user's score on a movie. Because the distribution of users changes fast, some new problems arise. For example, because of the significant interest differences existing among different types of users, wrong policy estimate is likely to be proposed when the user's distribution changes. To release this problem, we decide to use stratified sampling replay in place of random replay. Similarly, there will still have a replay memory to hold previous samples. However, unlike random replay, we have stratified sampling on certain metrics, such as sampling according to the user's gender and age, rather than randomly, thus having a smaller variance.

The Double DQN based algorithm is given by Algorithm 1.

---

**Algorithm 1**: Movie Recommendation algorithm

**Require**:  
D: empty reply pool, N_: maximum capacity of reply pool  
\( \theta \): parameter of the network, \( \theta^- \): copy of \( \theta \)  
N_b: size of training batch, SRS: stratified random sampling

1. Initialize action-value network \( Q \) with random weights \( \theta \)
2. Initialize the target neural network buffer \( (Q_i)_{i=1}^L \)
3. For episode \( e \in 1,2,3, \ldots M \) do
4. \hspace{1em} For \( t \in 0,1, \ldots \) do
5. \hspace{2em} With probability \( \varepsilon \) select a random action \( a_t \), otherwise \( a_t = \text{argmax}_a Q(s_t,a; \theta) \)
6. \hspace{2em} Execute action \( a_t \) in environment and observe reward \( r_t \) and next state \( s_t+1 \) and store transition tuple \( (s_t,a_t,r_t,s_t+1) \) in D
7. \hspace{1em} Use stratified sampling to sample a mini batch: \( N_b \) tuples \( (s_t,a_t,r_t,s_t+1) \sim \text{SRS}(D) \)
4. Experiment

4.1. Dataset
This experiment employs the Movie lens 1M data set, which contains 6,040 users’ 1 million rating records for 3,952 movies. Because our experiment has temporal correlation, user's old behaviour is used as the training set and the new behaviour is the test set. Specifically, after sorting all of the scoring record by time sequence, the top 80% score records is used as training set, and the last 20% records is the test set.

4.2. Evaluation Measures
This experiment employs the Movie lens 1M data set, which contains 6,040 users’ 1 million rating records. Supposing real rating is $r_{ui}$ and forecast rating is $r_{ui}'.$

RSME:

$$\text{RSME} = \sqrt{\frac{\sum_{(u,i) \in T} (r_{ui} - r_{ui}')^2}{|\text{Test}|}}$$

The goal of our recommend system is to find out the best model to minimize the RSME of test set.

4.3. Results and Analysis
In this part, we analyse the effectiveness of our Double DQN model in comparison with the nature DQN model and also draw the cumulative reward curve of reinforcement learning. Besides, we make comparison with traditional algorithms as well.

1) Superior to nature DQN: We evaluate the effectiveness of our Double DQN model in comparison with the nature DQN model on Movie lens 1m data sets, and the results are showed as below. Figure 2 and Figure 3 respectively shows the RSME of nature DQN and Double DQN. The results in Figure 4 indicate that Double DQN model reduce the error by 4.4% regarding RMSE compared with nature DQN model. The improvements obtained by Double DQN model therefore demonstrate the importance of effectiveness of our way to mine the information behind the attributes and solve overestimation.
2) Accelerating Convergence: Figure 5 and Figure 6 show that by using stratified sampling instead of random sampling, the convergence rate is faster.

3) Superior to traditional algorithms and comparable to latest algorithms: By referring to latest papers based on MF, their RSME has been reduced to about 0.84, which is slightly better than ours.\textsuperscript{[11]} However, as a newly proposed recommendation algorithm under completely different frame, our algorithm compared with matrix factorization (MF) which has been improved for many years, only has few difference. Thus, it’s obvious that deep reinforcement learning based recommendation algorithm has great advantages.

5. Conclusion
In this paper, we propose a deep reinforcement learning based recommend system using stratified sampling to do movie recommendation. Different from previous methods, our method can adapt to the dynamic change of users’ interest and distribution. After establishing nature DQN based recommend model, we introduce double q-learning to make modified, and rebuilding improved Double DQN based recommend model. In addition, we apply stratified sampling rather than random sampling into our model and consequently, the convergence rate is faster. Experiments have shown that our method can effectively improve the recommendation accuracy and our algorithm is comparable to latest algorithms. However, like most other recommend systems, our model is tested on historical data for predictive accuracy. That is, the system doesn’t test how users’ behavior influenced by it.\textsuperscript{[4]} To carry out such experiment, we should employ our model in a real site with real users to see whether this system is effective or not. Thus, in our future work, we hope to do some online test and make real time interaction with users.

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

7. References
[1] Wouter IJntema, Frank Goossen, Flavius Frasincar, and Frederik Hogenboom. 2010. Ontology-based news recommendation. In Proceedings of the 2010 EDBT/ICDT Workshops. ACM, 2010:1-6.

[2] Kompan M, Bieliková M. Content-Based News Recommendation\textsuperscript{[C]}// E-Commerce and Web Technologies, International Conference, Ec-Web 2010, Bilbao, Spain, September 1-3, 2010. Proceedings. DBLP, 2010:61-72.

[3] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual bandit approach to personalized news article recommendation. In Proceedings of the 19th international conference on World wide web. ACM, 661–670

[4] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. Journal of Machine Learning Research 6, Sep (2005), 1265-1295.
[5] Benjamin Marlin and Richard S Zemel. 2004. The multiple multiplicative factor model for collaborative filtering. In Proceedings of the twenty-first international conference on Machine learning. ACM, 73.

[6] Steffen Rendle. 2010. Factorization machines. In Data Mining (ICDM), 2010 IEEE 10th International Conference on. IEEE, 995–1000.

[7] Lei Li, Dingding Wang, Tao Li, Daniel Knox, and Balaji Padmanabhan. 2011. SCENE: a scalable two-stage personalized news recommendation system. In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. ACM, 125–134.

[8] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In Proceedings of the 15th international conference on Intelligent user interfaces. ACM, 31–40.

[9] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In Proceedings of the 16th international conference on World Wide Web. ACM, 271–280.

[10] Zheng G, Zhang F, Zheng Z, et al. DRN: A Deep Reinforcement Learning Framework for News Recommendation[C]// World Wide Web Conference. 2018:167-176.

[11] Zhao X, Zhu C, Cheng L. Coupled Bayesian Matrix Factorization in Recommender Systems[C]// IEEE International Conference on Data Science and Advanced Analytics. IEEE, 2018:1564-1574

[12] V Mnih, K Kavukcuoglu, D Silver, AA Rusu, J Veness. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540):29-33.

[13] Mozer S, M C, Hasselmo M. Reinforcement Learning: An Introduction[J]. IEEE Transactions on Neural Networks, 2005, 16(1):285-286.