Movie Recommendation via Deep Reinforcement Learning

Fatma Betül Yazıcı*
*Department of Computer Engineering
İstanbul Technical University, Turkey, Email: betulyazicii@gmail.com

***Abstract*—Nowadays, the recommendation system has made finding the things easy that we need. Movie Recommender systems have been developed to model users preferences. State-of-the-art methods have two limitations. First one these methods considering the recommendation as a static procedure and ignoring the dynamic interactive nature between users and the recommender systems, second limitation is focusing on the immediate feedback of recommended items and neglecting the long-term rewards. To address the two limitations, in this paper, propose a recommendation framework based on deep reinforcement learning. The experimental results based on the well-known MovieLens dataset demonstrate the effectiveness of the proposed framework.**

***Index Terms*—Recommendation,Reinforcement Learning, Actor Critic Architecture, Markov Decision Process**

I. Introduction

Recommendation systems have become a crucial part of almost all online movie platforms. Movie recommendation systems purpose at helping movie lovers by suggesting what movie to watch without having to go through the long process of choosing from a large set of movies which go up to thousands and millions that is time consuming and confusing. A typical interaction between the recommender system and its users is users are recommended a page of items and they provide feedback with rating movies, and then the system recommends a new movie list. Thus, the recommendation process is a sequential process.

In this paper, using Actor Critic architecture continuously improving recommender systems strategies during the interactions with users. The system gives movie lovers more chance to explore new movies which are not popular which may match with their interests.

The paper is organized as follows. In Section II related work different techniques on recommendation systems will be mentioned. In Section III proposed methodology will be explained in detail with the subheadings Markov Decision Process and Actor Critic Architecture. Describes the testing procedure and experimental results in Section IV. Conclusion and final remarks are given in Section V.

II. RELATED WORK

There are various methods exist for recommendation. In this section, we briefly review works related to recommendation systems. We will review this section as State-of-the-art Recommendation Techniques and RL based Recommendation Techniques.

A. State-of-the-art Recommendation Techniques

Different kinds of recommendation techniques are proposed in the past a few decades to improve the performance of recommender systems, including collaborative filtering, content-based filtering and matrix factorization.

- **Collaborative Filtering:** CF assumes that users may be interested in items selected by people who share similar interaction records with them. The interaction can be classified explicitly [1], [2], like ratings, or implicitly [3], [4] such as click and view. There are two different approaches collaborative filtering exist. First approach User-based collaborative Filtering computes the correlation with all other users for each item and aggregate the rating of highly correlated users. Second approach Item-based collaborative Filtering computes for each user item the correlation with all other item and aggregates for each user the ratings for item that are already highly correlated. The approaches are expensive and time consuming due to the diversity of opinions of large number of users.
- **Content-Based Filtering:** Content-based filtering recommends items on the basis of comparison between the content of the items and a user profile [5]. With content-based filtering produced candidate items with the user profile is essentially matching them with the user's previous records. Therefore, this approach tends to recommend items that are similar to items liked by a user in the past [6].
- **Matrix Factorization:** MF models characterize both items and users by vectors in the same space, which are inferred from the observed user-item interactions [7], [8]. User and item interactions represented in a form of a matrix. In matrix each row represents each users, while each column represents different movies. One strength of matrix factorization is the fact that sparse problem since not every user is going to watch every movies.

B. RL-Based Recommendation Techniques

Model-free reinforcement learning technique has been used in the development of recommendation systems in recent years.The model-free RL techniques can be divide into two categories policy-based and value-based.

- **Policy-Based:** Policy-based approaches goal to generate a policy, of which the input is a state, and the output is an action. [9], [10], [11] These works apply deterministic policies, which generates an action directly. In the policy based network outputs a continuous action representation, and the recommendation is generated by ranking the items with their scores, which are computed by a pre-defined function with the action representation and the item embeddings as input.
- **Value-Based:** Value-based approaches goal to select best action maximum Q-value over all the possible actions. [12], [13] to model Q-value of a state-action pair. Value-based approaches need to evaluate the Q-values of all the actions under a specific state, which is very inefficient when the number of actions is large.

To make RL based recommendation techniques suitable for large-scale items scenario, in this paper, we propose the a framework Actor-Critic which using the policy based approaches.

III. Proposed Method

In this section, we will define the sequential interactions between users and a recommender system as a Markov Decision Process (MDP) and then build recommender system via Actor-Critic based reinforcement learning.

A. Markov Decision Process

An MDP is a model for sequential stochastic decision problems [14]. An MDP is defined as (S, A, P, R, γ).

State Space S : A state space $\mathrm{s}_t = s_t^1, ..., s_t^N \in S$ is defined as the browsing history of a user, i.e., previous N items that a user rated before time t. State space as a list of N last interacted items. In this experiments we will consider N = 200.

Action Space A: An action $\mathrm{a}_t = a_t^1, ..., a_t^K \in A$ is to recommend a list of items to a user at time t based on current state, where K is the number of items the systems recommends to user each time. In this experiments we will consider K=4.

Transition probability P: Transition probability $p(\mathrm{s}_{t+1}|s_t, a_t)$ defines the probability of state transition from s_t to s_{t+1} when agent takes action a_t.

Reward R: Given the recommendation based on the action a and the state s, the user will provide feedback with rating. The recommender receives immediate reward $R(s, a)$ according to the user's feedback.

Discount factor γ: $\gamma \in [0, 1]$ is a factor measuring the present value of long-term rewards. In the case of γ = 0, the recommender considers only immediate rewards but long-term rewards are ignored. On the other hand, when γ = 1, the recommender treats immediate rewards and long-term rewards as equally important.

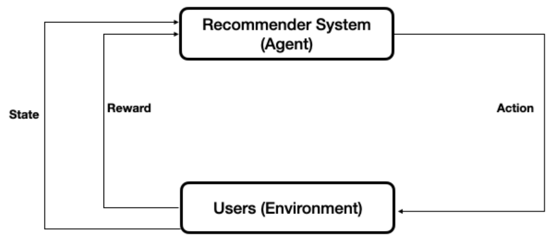

Fig. 1. Agent-Environment Interactions in MDP

Figure 1 shows the agent-environment interactions in MDP formulation. Considering the current user state and immediate reward to the previous action, the recommender takes an action. An action is a continuous parameter vector. Taking such an action, the parameter vector is used to determine the ranking scores of all the candidate items, by performing inner product with item embeddings. All the candidate movie items are ranked according to the computed scores and Top-K items are recommended to the user.

B. Actor-Critic Reinforcement Learning

In this section, we will propose a multi-agent reinforcement learning framework for the movie recommendation problem. The multi-agent RL framework with Actor-Critic architecture is showed in Figure 2.

The goal of the actor is to suggest recommendations based on users historical browsing behaviors (state). Actor generates a deterministic optimal action according to the current state, and the Critic generate outputs the Q-value of this state-action pair.

The goal of the actors is to suggest recommendations based on users' historical browsing behaviors (state), which should address two challenges: (i) how to capture users' dynamic preference in one recommendation session, and (ii) how to generate recommendations according to the learned users' preference. To tackle these challenges, we develop a two-stage framework, where the first stage (i.e. actor) aims to learn users' dynamic preferences, and the second stage targets to generate recommendations.

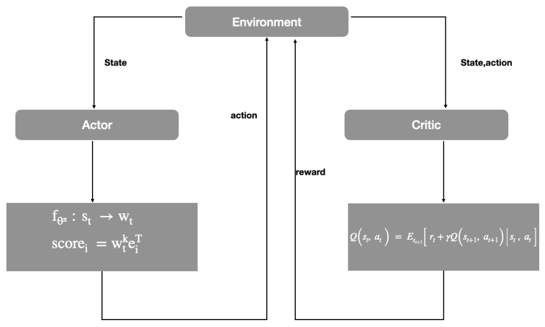
Environment
State
State,action
action
Actor
Critic
reward
$f_{\theta^\pi} : s_t \rightarrow w_t$
$score_i = w_t^k e_i^T$
$Q(s_t, a_t) = E_{s_{t+1}}[r_t + \gamma Q(s_{t+1}, a_{t+1}) | s_t, a_t]$

Fig. 2. An overview of the proposed Actor-Critic Framework

Actor network has two challenges: (i) how to capture users dynamic preference in recommendation session and (ii) how to generate recommendations according to the learned users preference. To tackle these challenges, actor consist of two steps. First, state-specific scoring function parameter generating and second action generating. Actor network models state-specific scoring function parameter generating step maps the current state $\mathrm{s}_t = s_t^1, ..., s_t^N$ to a list of weight vectors $\mathrm{w}_t = w_t^1, ..., w_t^K$ as follows:

$$f_{\theta^\pi} : s_t \rightarrow w_t \tag{1}$$

where f_{θ^π} is a function parametrized by θ^π mapping from the state space to the weight representation space. Using deep neural networks as the parameter generating function to capture users' sequential browsing behaviors.

In figure 2, the item representations $\mathrm{e}_1, \cdots, \mathrm{e}_i$ are dense and low-dimensional vectors, which are pre-trained based on users browsing history via word embedding [15], where the clicked items in one recommendation session are treated as a sentence, and each item is treated as a word. Next the action-generating step based on the aforementioned scoring function parameters. In this step generate recommendations according to users' preference learned by the Actor. To be more specific, a similarity function between the representations of user's current preference at and each candidate item is proposed to calculate a similarity score:

$$score_i = w_t^k e_i^T \tag{2}$$

Then we select the item with the highest similarity score as the output of the second step of actor, i.e., the next item to be recommended according to users' current preference.

Critic takes state s_t, action a_t as input and compute the Q(s_t, a_t) that evaluates the future cumulative rewards starting from the current state and action. According to the Q(s_t, a_t), the actors will update their parameters to generate more accurate recommendations. Optimal policy should satisfy: $\mathrm{a}^*(s) = argmaxQ^*(s, a)$.

Q-value function computed in Eq.(3) as follows:

$$Q(s_t, a_t) = E_{s_{t+1}}[r_t + \gamma Q(s_{t+1}, a_{t+1})|s_t, a_t] \tag{3}$$

where the Q-value function is the expected return based on state s_t and the action a_t

In critic network, action value function Q can be optimized by minimizing the mean-squared Bellman error loss function.

$$L = E[(Q(s, a) - (r + \gamma Q_{target}(s^{'}, a_{target}(s^{'}))))^2] \tag{4}$$

Reward calculated the similarity of each existing historical state-action pair in the memory. The first term measures the state similarity and the second term evaluates the action similarity. Parameter α controls the balance of two similarities.

$$CosineSimilarity = \alpha \frac{\mathrm{s_t s_i^T}}{||\mathrm{s_t}||.||\mathrm{s_i}||} + (1-\alpha)\frac{\mathrm{a_t a_i^T}}{||\mathrm{a_t}||.||\mathrm{a_i}||} \tag{5}$$

IV. EXPERIMENTAL RESULTS

Dataset: We experiment with real-world dataset namely MovieLens. [16] The MovieLens (100k) consist of 943 users, 1682 movie items and 100000 rating.

Evaluation Metric: In this work, Hit Rate evaluation metric used to evaluate our models. Hit rate is defined as the fraction of the predicted movies that users actually watched. Hit rate using to check that among the list of movies predicted, what percentage of them are actually successfull.

In this experiment recommendation list size choose K=4.

TABLE I
PERFORMANCE COMPARISON OF ALL METHODS

Method Name	HR@4
Collaborative Filtering	0.0551
Content-Based Filtering	0.0015
Matrix Factorization	0.0194
Reinforcement Learning	0.0424

In Figure 3, the average score of the films recommended with reinforcement learning is shown on the left picture, and the average score of the films recommended randomly is shown on the right.

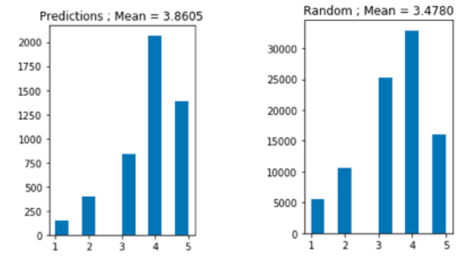

Fig. 3. Comparison of Prediction and Random Recommendations

V. CONCLUSION

In this paper, we propose a deep reinforcement learning based Actor-Critic framework. Emperimental results compared with real world dataset reinforcement learning and state-of-the-art methods. Due to the limited data set, the results were measured with hit rate metric and 4 movies.

REFERENCES

[1] X. Amatriain, J. M. Pujol, and N. Oliver, "I like it... i like it not: Evaluating user ratings noise in recommender systems," in International Conference on User Modeling, Adaptation, and Personalization. Springer, 2009, pp. 247–258.
[2] G. Jawaheer, M. Szomszor, and P. Kostkova, "Comparison of implicit and explicit feedback from an online music recommendation service," in proceedings of the 1st international workshop on 16 information heterogeneity and fusion in recommender systems, 2010, pp. 47–51.
[3] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in 2008 Eighth IEEE International Conference on Data Mining. Ieee, 2008, pp. 263–272.

[4] C. Wang, H. Zhu, C. Zhu, C. Qin, and H. Xiong, “Setrank: A setwise bayesian approach for collaborative ranking from implicit feedback,” in Proceedings of the AAAI Conference on Artificial Intelligence, 2020.

[5] M. Balabanovi, Y. Shoham, “Fab: Content Based,Collaborative Recommendation”. “Magazine Communications of the ACM” Volume 40 Issue 3, pp.66-72, 1997

[6] P. Lops, M. De Gemmis, and G. Semeraro, “Content-based recommender systems: State of the art and trends,” in Recommender systems handbook. Springer, 2011, pp. 73–105.

[7] M. Deshpande and G. Karypis, “Item-based top-N recommendation algorithms,” ACM Trans. Inf. Syst., vol. 22, no. 1, pp. 143–177, 2004.

[8] J. Wang, A. P. De Vries, and M. J. Reinders, “Unifying user-based and item-based collaborative filtering approaches by similarity fusion,” in SIGIR. ACM, 2006, pp. 501–508.

[9] X. Zhao, L. Zhang, Z. Ding, D. Yin, Y. Zhao, and J. Tang, “Deep reinforcement learning for list-wise recommendations,” CoRR, vol. abs/1801.00209, 2018.

[10] Y. Hu, Q. Da, A. Zeng, Y. Yu, and Y. Xu, “Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application,” CoRR, vol. abs/1803.00710, 2018

[11] G. Dulac-Arnold, R. Evans, P. Sunehag, and B. Coppin, “Reinforcement learning in large discrete action spaces,” CoRR, vol. abs/1512.07679, 2015.

[12] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin, “Recommendations with negative feedback via pairwise deep reinforcement learning,” CoRR, vol. abs/1802.06501, 2018.

[13] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, “DRN: A deep reinforcement learning framework for news recommendation,” in WWW 2018, Lyon, France, April 23-27, 2018, 2018, pp. 167–176

[14] Richard S Sutton and Andrew G Barto. 1998. Reinforcement learning: An introduction. Vol. 1. MIT press Cambridge.

[15] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In Advances in neural information processing systems. 2177–2185.

[16] F. M. Harper and J. A. Konstan, “The MovieLens datasets,” ACM Transactions on Interactive Intelligent Systems, vol. 5, no. 4, pp. 1–19, Dec. 2015.https://grouplens.org/datasets/movielens/100k/