Reinforcement Learning based Recommender System using Biclustering Technique

Sungwoon Choi
Seoul National University
Korea University
nebulach23@gmail.com

Heonseok Ha
Seoul National University
heonseok.ha@gmail.com

Uiwon Hwang
Seoul National University
uiwon.hwang@snu.ac.kr

Chanju Kim
Clova AI Research
NAVER Corporation
chanju.kim@navercorp.com

Jung-Woo Ha
Clova AI Research
NAVER Corporation
jungwoo.ha@navercorp.com

Sungroh Yoon*
Seoul National University
sryoon@snu.ac.kr

ABSTRACT

A recommender system aims to recommend items that a user is interested in among many items. The need for the recommender system has been expanded by the information explosion. Various approaches have been suggested for providing meaningful recommendations to users. One of the proposed approaches is to consider a recommender system as a *Markov decision process* (MDP) problem and try to solve it using reinforcement learning (RL). However, existing RL-based methods have an obvious drawback. To solve an MDP in a recommender system, they encountered a problem with the large number of discrete actions that bring RL to a larger class of problems. In this paper, we propose a novel RL-based recommender system. We formulate a recommender system as a gridworld game by using a biclustering technique that can reduce the state and action space significantly. Using biclustering not only reduces space but also improves the recommendation quality effectively handling the cold-start problem. In addition, our approach can provide users with some explanation why the system recommends certain items. Lastly, we examine the proposed algorithm on a real-world dataset and achieve a better performance than the widely used recommendation algorithm.

CCS CONCEPTS

• **Information systems** → *Collaborative filtering*; • **Computing methodologies** → *Artificial intelligence*;

KEYWORDS

Recommender System, Reinforcement Learning, Markov Decision Process, Biclustering

ACM Reference Format:
Sungwoon Choi, Heonseok Ha, Uiwon Hwang, Chanju Kim, Jung-Woo Ha, and Sungroh Yoon. 2018. Reinforcement Learning based Recommender System using Biclustering Technique. In *Proceedings of Workshop on Multi-dimensional Information Fusion for User Modeling and Personalization (IFUP'18).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.475/123_4

*Corresponding author

IFUP'18, Feb 2018, Los Angeles, USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

1 INTRODUCTION

As the choice of users increases, the importance of recommender systems that assist in decision making is increasing day by day. Recommender systems are introduced in a variety of domains, and the performance of recommender systems is directly related to the interests of the company or individual. Previously, recommender systems have achieved great success with a method called collaborative filtering (CF). CF is one of the most popular techniques in the recommender system domain. The objective of CF is to make a personalized prediction about the preferences of users using the information about other users who have similar interests for items.

One disadvantage of CF is that it considers only one of the two dimensions (*i.e.*, users or items), which often makes it difficult to detect important patterns that otherwise could be captured by considering both dimensions. In addition, the data matrix a typical recommender system has to handle is sparse and high-dimensional, because there are a large number of available items, many of which are never purchased or rated by the users. These two facts led to the developments of biclustering-based recommender systems, some of which have shown superior performance to conventional CF approaches [1, 5, 9, 16, 19]. Biclustering, also known as co-clustering [4, 18], two-way clustering [6], and simultaneous clustering [8], aims to find subsets of rows and columns of a given data matrix [3]. The big difference between clustering and biclustering is that clustering derives a global model, whereas biclustering produces a local model [7, 10].

Another disadvantage of CF is that it is static, therefore it is usually not possible to reflect a user's response in real time. Therefore, an MDP-based recommender system is proposed [14]. They use a discrete state MDP model to maximize the utility function that takes into account the future interactions with their users. In their work, they suggest the use of an n-gram predictive model for generating the initial MDP. They consider the actions of the MDP as a recommendation for an item. This leads to a large action space which makes it difficult to solve the MDP problem.

In this paper, we propose a new recommendation algorithm using biclustering and RL. We reduce state and action space by using a biclustering technique which renders the MDP problem

easy to solve. Using biclustering not only reduces space but also improves the recommendation quality of the cold start problem. Moreover, it can be explained to users why the system recommends certain items.

The paper is structured as follows. In Section 2 we review the necessary background on MDP and RL. In Section 3 we define the problem. In Section 4 we describe the proposed approach. Section 5 provide an empirical evaluation of the actual recommender system based on the two Movielens datasets. We discuss the paper in Section 6 then we conclude the paper in Section 7.

2 PRELIMINARY

Markov Decision Processes : An MDP is a model for sequential stochastic decision problems [15]. An MDP model is specified by a tuple of states, actions, a reward function, a transition function, and a discount factor. The agent stays in a particular state $s_t \in S$ for each discrete time step $t \in \{0, 1, 2, ...\}$. After the choice of an action $a_t \in A$, the agent moves to the next state s_{t+1} by calling a transition function $\mathrm{T}(s_t, a_t)$. At the same time the agent receives a reward r_t from the environment by reward function $\mathrm{R}(s_t, a_t, s_{t+1})$. Based on a policy $\pi(s)$, the action a is selected in a certain state s. MDP can be solved by RL. RL aims to find the optimal policy π^* that maximizes the exptected cumulative reward G which is called return. In RL, the optimal policy can be learned by a state-action value function $\mathrm{Q}_\pi(s, a)$ which means the expected value of the return G obtained from episodes starting from a certain state s with the action a. $\mathrm{Q}_\pi(s, a)$ can be expressed as follows:

$$\mathrm{Q}_\pi(s, a) = E_\pi\{G_t | s_t = s, a_t = a\} \tag{1}$$

$$= E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a\right\} \tag{2}$$

where γ is a discount factor ($0 < \gamma \leq 1$).

3 PROBLEM DEFINITION

We consider a recommender system as an MDP problem that can be formalized in a gridworld. Figure 1 describes the overview of the formalization. A gridworld is a 2D environment in which an agent can move in four directons at a time. Typically, the goal in a gridworld is that the agent navigates to some location by maximizing the return. In our case, the agent and the state are considered as a user and a group of items, respectively. User movement in the gridworld means getting new recommendations from the group of items. Moreover, the reward can be considered as a user's satisfaction for the recommended items. At first, we need to be specify the environment of the MDP. In this paper, we assume that we have obtained n^2 biclusters from the user and item matrix $B = (U, I)$. We describe the environment in more detail below:

State Space S : Gridworld has $n \times n = n^2$ distinct states. Each state $s = (U, I)$ includes a user set U and an item set I which are obtained from biclustering. The start state can be any state. Outside of the gridworld cannot be moved to.

Action Space A : The agent can choose from up to four actions to move around: up, down, left, right.

Transition Function $\mathbf{T}(s_t, a_t)$: Gridworld is deterministic.

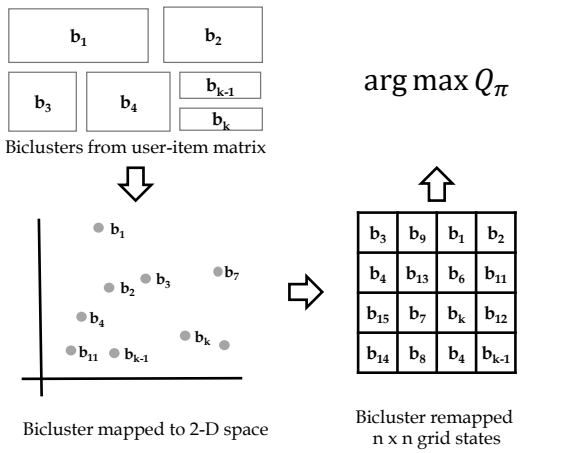

Figure 1: Overview of the proposed method

Reward Function $\mathbf{R}(s_t, a_t, s_{t+1})$: A reward r_t is also deterministic and is determined by the proposed reward function as follows:

$$\mathrm{R}(s_t, a_t, s_{t+1}) = \mathrm{Jaccard\_Distance}(U_{s_t}, U_{s_{t+1}}) \tag{3}$$

$$= \frac{|U_{s_t} \cap U_{s_{t+1}}|}{|U_{s_t} \cup U_{s_{t+1}}|}. \tag{4}$$

The agent receives between 0 to 1 reward through the calculation of the Jaccard distance with the user vectors of two states s_t, s_{t+1}. In this environment, reward is deterministic function of state-action pair. As the two states have more same users, the reward approaches to 1. The similarity of the two item vectors of the states is not considered as a reward, because we do not want to recommend only a small number of items when moving the state.

4 PROPOSED APPROACH

The proposed method is composed of four parts: constructing the states, learning the Q-Function, generating recommendations, and updating the model online.

4.1 Constructing the States

In the next step, each state is mapped to one of n^2 biclusters, so that each state has an item set and a user set. The mapping is performed based on the distance between the user vector of the bicluster and the states of the gridworld that can be considered as a two-dimensional (2D) euclidean space. However, the user vector of the bicluster is not 2D so it is converted to a 2D space using a dimensionality reduction technique. Now our goal is to map the user vectors to the 2D gridworld by minimizing the total distance. It is an NP-hard problem, hence we propose simple greedy algorithm. It is almost same as the traveling salesman problem. After calculating similarities between n^2 user vectors and gridworld, the user vectors are mapped to the nearest gridworld point one by one.

4.2 Learning the Q-Function

In this gridworld environment, the agent is looping over all states and evaluating the Q function for each of the four possible actions.

Algorithm 1: Generating Recommendations

input : state-space S, action-space A, policy π, transition function T, # candidate start states k, a user
output : recommended the list of items

```
select top-k states {s_1, .., s_k} with high similarity to a user;
for i ← 1 to k do
    s ← s_i;
    while at least one items to recommend do
        recommend items based on s;
        a ← ε-greedy with π(s);
        execute action a;
        s' ← T(s, a);
        s ← s';
```

Q-learning [17] and SARSA [13] are frequently used for this problem. We tested both algorithms in this study. Moreover, any state in our environment can be the start state. Then, the policy is updated to select the actions that maximize the Q value at each state. Moreover, we use the ϵ-greedy method for balancing exploration and exploitation [15]. The policy is described as follows:

$$\pi(s) = \begin{cases} \text{random action from } A & \text{if } \xi < \epsilon \\ \arg\max_{a \in A} Q(s,a) & \text{otherwise} \end{cases} \quad (5)$$

Then, an optimal policy can be found by the Bellman equation. Executing these updates repeatedly is guaranteed to converge to the optimal policy [15]. In other words, this corresponds to actions that guide the user to obtain good recommendations, while maximizing rewards.

4.3 Generating Recommendations

Algorithm 1 describes a procedure for generating recommendations for a user. Unlike the other tabula methods, all the states in this environment can be the starting state. To set the starting state, the jaccard distance between all states and the user is calculated. The state with the highest similarity to the user becomes the starting state. Then, the algorithm attempts to recommend the item with the ϵ-greedy based policy until there are no more recommended items.

4.4 Updating the Model Online

One of the major advantages of the proposed model is that the user feedback is reflected the states online. This makes the value of the reward function different. As the value of the reward function changes, the optimal policy may change. For example, when a user who has recommended an itemset in state s_t is satisfied with the items, the system immediately adds that user to the userset U_{s_t} of the corresponding state s_t. If the user are satisfied with the item recommended in the next state s_{t+1}, the size of $U_{s_{t+1}}$ is also increased by 1. As a result, $\mathrm{R}(s_t, a_t, s_{t+1})$ increases from $|U_{s_t} \cap U_{s_{t+1}}| \,/\, |U_{s_t} \cup U_{s_{t+1}}|$ to $|U_{s_t} \cap U_{s_{t+1}}| + 1 \,/\, |U_{s_t} \cup U_{s_{t+1}}| + 1$. Using the algorithm proposed in this paper, it is possible to update the state space in real time, and since the reward value changes according to the updated state space, the recommendation can be changed according to the current trend of the users.

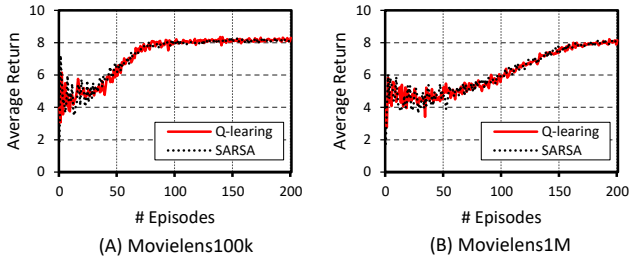

Figure 2: Learning curve on Movielens dataset

5 EXPERIMENTS

5.1 Dataset

OpenAI Gym [2] is used to experiment the proposed algorithm. Gym has a collection of environments so that the proposed reinforcement learning can be easily implemented. In addition, two Movielens datasets are used for evaluating our algorithm. One is a dataset of 943 users and 1682 items including 100,000 ratings. The other one has 1,000,209 ratings with 6,040 users and 3,900 items. Both data sets represent the preferences of users as ratings from 1 to 5. Two Movielens datasets were binarized applying a threshold of three rating points, as done in many studies. 80% of the dataset was used as the training set and the remaining was used to test the algorithm. To evaluate the algorithms in the cold start conditions, each test user rating was deleted leaving only 10%.

5.2 States Setup

To obtain the biclusters from the matrix, we use two well-known biclustering algorithms: Bimax [11], Bibit [12]. These two biclustering algorithms have the minimum number of rows and columns of the biclusters as input parameter. We vary the parameter values and obtain biclusters of various shapes and sizes. Subsequently, we randomly select a total of n^2 biclusters to map to the gridworld. Finally, the state space in the gridworld is completed using the proposed greedy algorithm. In our experiment, n is 20 and 30 on the two Movielens datasets so that the total number of states is 400 and 900, respectively.

5.3 Q-Learning versus SARSA

Q-learning and SARSA are used to find the optimal policy π. Q-learning is an off-policy based method while SARSA is an on-policy based method. In this paper, we used both methods and evaluated which performed better in our environment. Figure 2 demonstrates the learning curve on the two Movielens datasets. The average return is measured by the number of episodes. The performance of the two algorithms are almost identical in the experiment.

5.4 Performance on the Cold-Start Problem

In this paper, we use ranking metrics to evaluate proposed algorithm. The two most popular ranking metrics are precision and recall. Given a top-N recommendation list I_N, precision and recall are defined as follows:

Table 1: P@30 and R@30 Comparison

	Movielens_100k		Movielens_1M	
	P@30	R@30	P@30	R@30
Global-average	0.153	0.102	0.161	0.094
User-based	0.187	0.129	0.212	0.119
Item-based	0.193	0.132	0.220	0.124
Proposed	**0.246**	**0.169**	**0.277**	**0.155**

$$\text{P@N} = \frac{|I_N \cap I_D|}{N} \qquad \text{R@N} = \frac{|I_N \cap I_D|}{|I_D|} \tag{6}$$

where I_D is the items that algorithm should predict. The precision and recall are calculated by averaging the precision and recall over all the users.

We evaluate the standard algorithms based on which of the items are actually hidden by the user in the test data. Table 1 shows the results for P@30 and R@30 on the two Movielens datasets. The general observation is that the proposed algorithm outperforms the other recommendation methods under the cold-start condition.

5.5 Explainable Recommendation

When recommending an item group with this proposed algorithm, the reason for the recommendation can be explained to users by informing the corresponding state s together *e.g.* group of items I_s or group of users U_s. Providing a reason for a recommendation to a user in a real system can increase the recommendation reliability. In a model-based based recommender system, the recommendations are not explainable. In addition, existing bicluster based recommendation systems are not usually suitable for providing explanations because the bicluster size is usually very large.

6 DISCUSSION

In this paper, we assume that n^2 good quality of biclusters are given. However, biclustering is heuristically found in most cases, due to the fact that it is an NP-hard problem. Therefore, the performance related to providing a recommendation depends largely on the biclusters. We leave finding an optimal bicluster as a part of future research. Moreover, we have reduced the action space to four, top, bottom, right and left, but this action space can move in 8 directions or more directions in a multi-dimensional space instead of 2D space. It can be easily extended. Of course, the larger the action space, the greater the computational complexity and the better the accuracy of the recommendation. The method of mapping the bicluster to the state space is crucial for the quality of the recommendation. In addition, we are unable to test with various evaluation methods such as coverage, novelty, etc., but it is expected that the value of coverage and novelty will be acceptable based on ϵ value.

7 CONCLUSION

We have proposed a novel algorithm using RL and biclustering to mitigate the cold-start, online-learning, and explainable recommendation problems. We formulate a recommender system as a gridworld game by using a biclustering technique that reduces the state and action space significantly. Using biclustering not only reduce the space but also improves the recommendation quality of the cold start problem. Moreover, the system can explain to users why the system recommends certain items. We examine the proposed algorithm on the real world dataset and achieved better performance than standard recommender technique. We expect that this algorithm will be useful for recommending items in actual commercial applications.

ACKNOWEDGEMENT

This work was supported in part by a research grant from Naver Corporation and in part by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-IT1601-05.

REFERENCES

[1] Faris Alqadah, Chandan K Reddy, Junling Hu, and Hatim F Alqadah. 2015. Biclustering neighborhood-based collaborative filtering method for top-n recommender systems. *Knowledge and Information Systems* 44, 2 (2015), 475–491.
[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. (2016). arXiv:arXiv:1606.01540
[3] Yizong Cheng and George M Church. 2000. Biclustering of expression data.. In *Ismb*, Vol. 8. 93–103.
[4] Inderjit S Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 269–274.
[5] Thomas George and Srujana Merugu. 2005. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE, 4–pp.
[6] Gad Getz, Erel Levine, and Eytan Domany. 2000. Coupled two-way clustering analysis of gene microarray data. *Proceedings of the National Academy of Sciences* 97, 22 (2000), 12079–12084.
[7] Yongkweon Jeon and Sungroh Yoon. 2015. Multi-threaded hierarchical clustering by parallel nearest-neighbor chaining. *IEEE Transactions on Parallel and Distributed Systems* 26, 9 (2015), 2534–2548.
[8] Rebecka Jörnsten and Bin Yu. 2003. Simultaneous gene clustering and subset selection for sample classification via MDL. *Bioinformatics* 19, 9 (2003), 1100–1109.
[9] Kenneth Wai-Ting Leung, Dik Lun Lee, and Wang-Chien Lee. 2011. CLR: a collaborative location recommendation framework based on co-clustering. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 305–314.
[10] Sara C Madeira and Arlindo L Oliveira. 2004. Biclustering algorithms for biological data analysis: a survey. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on* 1, 1 (2004), 24–45.
[11] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. 2006. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics* 22, 9 (2006), 1122–1129.
[12] Domingo S Rodriguez-Baena, Antonio J Perez-Pulido, and Jesus S Aguilar-Ruiz. 2011. A biclustering algorithm for extracting bit-patterns from binary datasets. *Bioinformatics* 27, 19 (2011), 2738–2745.
[13] Gavin A Rummery and Mahesan Niranjan. 1994. *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering.
[14] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, Sep (2005), 1265–1295.
[15] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.
[16] Panagiotis Symeonidis, Alexandros Nanopoulos, Apostolos N Papadopoulos, and Yannis Manolopoulos. 2008. Nearest-biclusters collaborative filtering based on constant and coherent values. *Information retrieval* 11, 1 (2008), 51–75.
[17] Christopher John Cornish Hellaby Watkins. 1989. *Learning from delayed rewards*. Ph.D. Dissertation. King's College, Cambridge.
[18] Sungroh Yoon, Luca Benini, and Giovanni De Micheli. 2007. Co-clustering: a versatile tool for data analysis in biomedical informatics. *IEEE Transactions on Information Technology in Biomedicine* 11, 4 (2007), 493–494.
[19] Daqiang Zhang, Ching-Hsien Hsu, Min Chen, Quan Chen, Naixue Xiong, and Jaime Lloret. 2014. Cold-start recommendation using bi-clustering and fusion for large-scale social recommender systems. *IEEE Transactions on Emerging Topics in Computing* 2, 2 (2014), 239–250.