Playlist Recommendation Based on Reinforcement Learning

Binbin Hu, Chuan Shi, Jian Liu

Playlist Recommendation based on Reinforcement Learning

Binbin Hu[1], Chuan Shi[1], and Jian Liu[1]

[1]Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications,Beijing, China 100876
hubinbin@bupt.edu.cn,shichuan@bupt.edu.cn,fullback@yeah.net

Abstract. Recently, there is a surge of recommender system to alleviate the Internet information overload. A number of recommendation techniques have been proposed for many applications, among which music recommendation is a kind of popular Internet services. Unlike other recommendation services, music recommendation needs to consider the interaction and content information, as well as the inherent correlation and feedback among music playlist. Thus, in this paper, we model music recommendation as a Markov Decision Process, and consider the music recommendation as a playlist recommendation task. Along this line, we propose a novel reinforcement learning based model, called RLWRec, to exploit the optimal strategy of playlist. Two novel strategies are designed to solve the curse of state space and efficient music recommendation. Experiments on real dataset validate the effectiveness of our proposed method.

Keywords: music recommendation, reinforcement learning, Markov decision process

1 Introduction

With the rapid development of information technology and Internet, human society has gradually entered the era of information overload, where users have more and more difficulties in accessing information in Interest. Thus recommender systems [7] arise at the historic moment, which utilize user-item interaction and/or content information associated with users and items [11] to help users to predict preference and make reasonable recommendation.

Recommender systems have been widely used in various applications [10, 5, 2]. Among them, the music recommendation is a very popular web service, which recommends songs from huge corpus by matching songs with user preference [15]. Existing recommendation techniques, such as collaborative filtering, matrix factorization and so on [3, 15, 12, 9, 17], typically utilize ratings of users on items (may also include some additional information, e.g., social relations and geography) to infer user preference and make proper recommendation. In these methods, there are weak ties among adjacent items. Moreover, there are rare feedbacks of users on items, and these feedbacks have little swift effects on subsequent items. We think these methods may not be suitable for music recommendation, since there are some specific characteristics in the process of listening

to music. We know that people listen to music that reflect their feelings. And the music playlist has the inherent and consistent feelings. In addition, there are many meaningful feedbacks of users on songs, such as "skip","listen","download" and "collect". These feedbacks reflect different preferences of users on songs.

In order to consider the song correlation, advanced methods introduce Markov decision process [6] and reinforcement learning [13], and regard the recommendation task as decision problem. Chi et al. [1] models the automatic playlist generation problem as Markov decision process and learns the user preference with reinforcement learning. Furthermore, considering the influence between songs, Liebman et al. [4] relate learning individual preferences with holistic playlist generation and propose a novel reinforcement-learning framework DJ-MC to recommend song sequences. Moreover, Wand et al. present a reinforcement learning framework based on Bayesian model to balance the needs to explore user preferences and to exploit this information for recommendation [15].

In this paper, similarly, we model the recommender system as Markov decision process, and employ reinforcement learning to solve it. However, our research differs from state-of-the-art models in several ways: (1) We integrate the user feedback into model as well as considering the influence between songs. (2) We propose a state compression method to capture enormous state space of MDP. Meanwhile, we design a recommendation strategy to make a trade-off between accuracy and coverage of recommendation. With these two designs, we propose the RLWRec model based on Q-learning with ϵ-greedy strategy. The experiments on real music datasets show the effectiveness of our model. Meanwhile, we analyze the influence of user listening frequency and the window size on our model.

2 Preliminary

In this section, we describe notations used in the paper and present some preliminary knowledge.

Markov decision process is a expansion of Markov chain, whose difference is the actions and rewards to join. In general, MDP is defined as a quintuple form $M =< S, A, T, R, \gamma >$, where S is the set of states, A is the set of actions, $T : S \times S \times A \to \mathbb{R}$ depicts the function of state transition, representing that state s will transfer to state s' in probability $Pr(s'|s, a)$ when performing action a, $R : S \times S \times A \to \mathbb{R}$ is the reward function, representing that the rewards of performing action a in state s and reaching state s' is $R(s', s, a)$. $\gamma(0 \leq \gamma \leq 1)$ depicts the attenuation of rewards, the smaller γ means the more importance of immediate rewards. The key of MDP is to find a optimal strategy $\pi : S \to A$, representing a mapping of states to actions, i.e. the optimal action $a = \pi(s)$ in each state.

Reinforcement learning is the name of a set of methods and algorithms for controlling agents to automatically improve their performance by tying to maximize the rewards received from environment. After modeling the real problem as MDP, we can apply reinforcement learning technology to estimate optimal strategy. Q-learning [16] and SARSA [8] are the most widely employed in actual

problems, which are both primarily concerned with estimating the value of performing any action in each state and dynamically update the learned strategy.

3 Reinforcement Learning based Playlist Recommendation Model

In this section,we describe our recommendation problem and model the recommendation task as MDP, and then propose the *R*einforcement *L*earning with *W*indow for *Rec*ommendation (called RLWRec).

3.1 Problem Description

Listening music on the music platform is actually an interactive process: the music platform recommends a song for us, and we can choose a series of actions as feedback for the music platform, such as skip, listen, download, collect and so on. As a consequence, we can obtain the sequence of interaction between music platform and users, which can be considered as a MDP. Therefore, the recommendation task can be summarized as follows: based on a music corpus and the interactive records, we train the recommender system to recommend songs which will follow user preferences.

3.2 Recommendation Framework based on Reinforcement Learning

In this section, we model the recommendation task as MDP. Concretely, we will model states, actions and reward function.

Modeling States and Actions In order to learn user preferences from interactive process, we have to model user's states to reflect his or her listening history. Inspired by the N-Gram model, which predicts the next word while knowing last N words, we model user's states as sliding window of size K, which preserves the K songs recently listened. And we approximatively regard the state as user's listening history on the music platform. Thus, we concretely define the music corpus $M = \{m_1, m_2, ..., m_n\}$, and then the state space of our model can be represented as $S = \{(m_i, m_{i+1}, ..., m_{i+K-1}) | \forall j \in [i, i+K-1], m_j \in M\}$. With the definition of states, we regard each song as an action. Therefore, each time our model executes an action a (recommending a song) once, user's state will transfer, the sliding window of state s will slide a step backward, and take the song recommended into window, thus reach a new state s'. For an example, we assume user's current state $s = (m_1, m_2, ..., m_K)$, the recommender system recommends a song m_{K+1} for the user according to the past strategy, then the user's state will transfer to $s' = (m_2, m_3, ..., m_{K+1})$. Thus, the action space of our model can be represented as $A = M$.

Modeling Reward Function The basis of reinforcement learning lies in the rewards the agent receives, and how it updates state and action values [14]. As for our recommender system, users will express their feelings via a series of implicit or explicit feedback when a song is recommended. Hence, we need to construct a reward function to integrate user's various feedback and measure the achievement to the recommendation goal. In this paper, in consideration

of real dataset, we only extract three feedback information: listen, collect and download. Thus, we design a reward function as follows:

$$R(f) = \sum_{i=1}^{3} w_i * I(f == F_i), \tag{1}$$
$$F = \{listen, collect, download\},$$

where F is a feedback set, I is boolean function, w_i is weight, mapping user's feedback to discrete numerical space.
With the definition of the elements of MDP, we can summarize our model as Fig. 1.

Fig. 1. Model recommender system as MDP.

3.3 Model Challenges

Although we have modeled the recommendation task as MDP, how to solve is not a trivial problem. We will still face the following challenges.

State Space Challenge Our model preserve user's latest K songs with sliding window of size K, which approximates to user's listening history. However, the model will face a serious problem – the curse of state space. Each state of MDP is composed of K songs, so that our model's state space is $|M|^K$ ($|M|$ represents the size of song set). In terms of time and space complexity, the exponential state space cannot be accepted. Hence, we present a state compression algorithm to transform individual songs to song clusters for further model learning.

Recommendation Challenge We apply reinforcement learning algorithm in MDP problem, which will estimate a series of strategy for recommender system. Thus, it is necessary to select a proper strategy for a recommendation. In addition, picking a song from a song cluster is another important issue we have to consider. Therefore, we design a novel recommendation strategy based on tree structure to improve the recommendation performance.

4 Model Learning

4.1 State Compression based on Collaborative Filter

In order to reduce the dimensionality of state space for efficient training and predicting, we propose a state compression algorithm based on collaborative filter. The basic idea is that it clusters songs according to the similar user's performance, and then replaces songs to song clusters in the model learning process.

User Clustering Our model is mainly used for personalized recommendation of single user. Hence, we extract a user's listening logs from dataset for model training, we name this user as su. We construct each user's feature vector, consisting of user's feedback (listen, download and collect) for all songs in dataset, which reflects user's music preference. Then, we measure similarity between su and each user in dataset. After similarity measure, we extract top k similar users, whose feedback will be feature for song clustering later.

$$sim(u) = \frac{su \cdot u}{|su| * |u|}, \tag{2}$$

where u is arbitrary user except su, $sim(u)$ depicts music preference similarity between u and su.

Song Clustering In the song clustering process, we transform songs to feature vector of $k + k'$ dimensions consisting of user's feedback feature and song's own feature. User's feedback feature has k dimensions, which are top k similar users' performance (0: never listened, 1: listened, 2: collected, 3: downloaded) for the song. Song's own feature has k' dimensions, including singer, release time, popularity, language, gender and so on. Consequently, we use k-means for clustering, resulting in N song clusters.

RLWRec adopts state compression based on collaborative filter, consisting of user clustering and song clustering, transforms $|M|$ songs to N song clusters ($N << |M|$). Thus, song clusters can replace songs to construct state space in modeling and training, which tend to immensely reduce the size of state space.

4.2 Learning Algorithm

We choose Q-learning as our learning algorithm. This method, combining dynamic programming and Monte Carlo's idea, is an efficient model-free reinforcement learning algorithm. Moreover, we bring ϵ-greedy strategy in learning process, which can make a trade-off between exploration and exploitation.

***Q*-learning** Q-learning is primarily concerned with estimating an evaluation of performing specific actions in each state, known as Q-values, which not only consider action's immediate rewards but also takes accumulative rewards into account. We denote the Q-value of performing specific action a in state s as $Q(s, a)$. And we design the Q-value update rule as follows:

$$Q_n(s, a) = (1 - \alpha_n)Q_{n-1}(s, a) + \alpha_n[R(s, a) + \gamma \max_{a'} Q_{n-1}(s', a')] \tag{3}$$

with

$$\alpha_n = \frac{1}{1 + VisitCount(s, a)}, \tag{4}$$

where $R(s, a)$ represents the immediate reward of performing action a in state s, s', a' are next state and action of s, γ controls the attenuation of accumulative rewards. Moreover, we bring in α_n, related to the count of performing same action in same state. This rule can partly reveal user's music preference and accelerate the coverage with the decreasing value of α_n.

ϵ-greedy Strategy The training of reinforcement learning is a process of trying. So when we choose an action in a state, we should consider two aspects: (1) knowing the rewards of each action. (2) choosing the action of most reward recently. Thus, reinforcement learning has two strategies: exploration strategy (trying as many actions as possible) and exploitation strategy(choosing the best action given current information). Obviously, the two strategies are conflicting. Hence, we take advantage of ϵ-greedy strategy, which is an eclectic collection of above two strategies. ϵ-greedy strategy guarantees the trade-off between exploration and exploitation based on a probability ϵ: exploring in the probability of ϵ and exploiting in the probability of 1-ϵ.

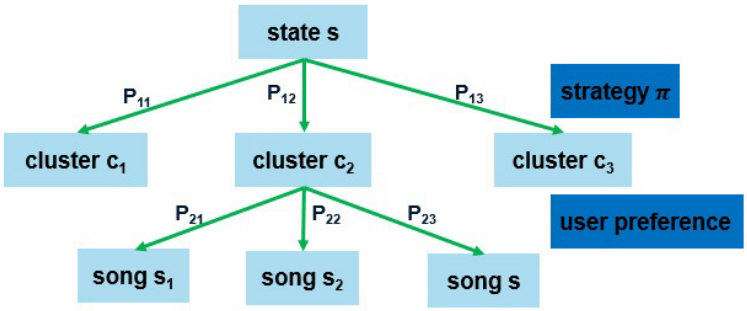


Fig. 2. Recommendation strategy based on tree structure.

4.3 Recommendation Strategy

As mentioned above, Q-learning can evaluate the true rewards of performing each action in each state, knowing as Q-value. Thus, we can get the list of Q-value of actions in each state. Thus, we need design a recommendation strategy to choose a song cluster and then recommend a song for the specific user. First of all, for the specific user su, we let him set a score for each song in the dataset, representing the preference of song of su, denoted as $preference(m)$. We give the calculation rule as follows:

$$preference(m) = \sum_{f \in F} count_f * w_f, \tag{5}$$
$$F = \{listen, collect, download\},$$

where $count_f$ represents the count of user performing feedback f for song m, w_f is the preference weight of user performing feedback f for song m. Meanwhile, in the process of recommendation, we also can update user's preference with the timely feedback, as follows:

$$preference_n(m) = preference_{n-1}(m) + w_f. \tag{6}$$

Based on user's preference score, our model can recommend for user by serval strategies. One naive idea is recommending by the maximal Q-value and

preference score. However this strategy will seriously bring down the coverage of recommendation. Moreover, we can combine this strategy with ϵ-greedy strategy to improve the coverage of recommendation. In order to get better performance further, we design a recommendation strategy based on tree structure: recommending song cluster according to the probability of Q-value and recommending song according to the probability of preference score, as shown in Fig. 2. For example, assume in the state s, the recommender system has the probability of P_{12} to choose the song cluster c_2 and further recommends the song s_2 with the probability of P_{22}. Recommending based on probability can not only guarantee accuracy of recommendation but also improve the coverage of recommendation.

5 Experiment

In this section,we will validate the effectiveness and traits of our model by conducting a series of experiments.

5.1 Dataset

In this paper, we use a real music dataset for experiment, including song data and user interaction record of six months. This music dataset consists of 349949 users, 10842 songs and 5652232 feedback (listen, collect, download). Meanwhile, this music dataset includes song's essential features, such as artist, publish time, initial popularity and so on. Based on the raw data, we cut the dataset into three smaller dataset according to user's listening frequency. These three datasets are as follows: low frequency dataset (the frequency of songs listened by users is smaller than 300), medium frequency dataset (between 400 and 700), and high frequency dataset(larger than 800). And in these datasets, we filter out some special users (listen too many or too few) and guarantee the number of users at around 600 and the number of songs at around 10000.

5.2 Comparison Methods and Metrics

Because there are few methods integrating user feedback for music recommendation, and advanced methods based on reinforcement learning [1, 4, 10] don't use quantitative indicators to measure the recommendation performance, we design the following methods as baselines.

• **RandRec**. Randomly choose songs from user's listening history and recommend for user.

• **CFRec**. Recommend songs for user according to other users with similar music preference.

• **PopRec**. Calculate song's popularity and randomly recommend the top$k$ popular songs for user.

In the experiment, we set the sliding window size $k = 3$, the attenuation factor and the exploration probability in Q-learning γ=0.8 and ϵ=0.7. Meanwhile, optimal parameters are set for other algorithms.

In this paper, we calculate the accuracy and coverage of recommendation. Similar with the traditional F_1 score, we combine the accuracy and coverage and use the *Score* to evaluate the performance of list prediction.

$$Accuracy = \frac{|\{s \in L_p \quad and \quad s \in L_t\}|}{|L_p|}, \tag{7}$$

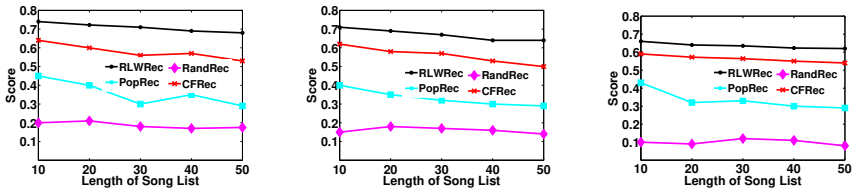
$$Coverage = \frac{set(L_p) \cap set(L_t)}{|L_t|}, \tag{8}$$

$$Score_{list} = \frac{2 * Accuracy * Coverage}{Accuracy + Coverage}, \tag{9}$$

where L_p is the song list predicted by models, and L_t is the actual listening list. A lager $Score$ means a better performance.

5.3 Effectiveness Experiments

In order to validate the effectiveness of our model, we will compare RLWRec to other baselines. For each dataset, we use 90% of data as training and the rest of the dataset for testing. Moreover, the random selection is carried out 10 times independently and the average results are illustrated in Fig. 3.



(a) Low_Frequency (b) Medium_Frequency (c) High_Frequency

Fig. 3. Performance comparisons on three datasets

It is clear that our RLWRec model achieves significant performance improvements compared to other algorithms. It indicates the benefits of recommendation with reinforcement learning. It also shows that RLWRec is more stable with the growth of song list length. Moreover, as the increment of song list length, the performance of RLWRec will gradually converge and tend to be steady, which reveals that the stabilization of users performance of music in a period of time.

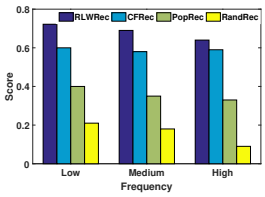


Fig. 4. Performance on different frequency

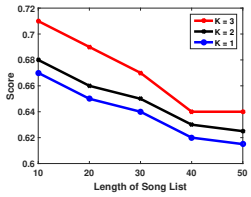


Fig. 5. Performance with different window sizes

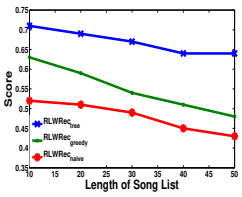


Fig. 6. Performance with different recommendation strategies

5.4 Influence of user's listening frequency

In the above experiments, comparing different frequent datasets, we can find that most methods are affected by the listening frequency. In this section, we further

study the influence of user's listening frequency to the prediction performance. We do the same experiments, and compare the performances of predicting the song list of length 20.

As illustrated in Fig. 4, it is clear that most of models will be influenced by user's listening frequency. The performance of RLWRec will slightly decrease as the growth of user's listening frequency. It can be inferred that our model's training state space will increase as the growth of user's listening frequency, which leads to the difficulty of choosing the right action in a specific state. As for the recommendation methods base on other users' behaviors, such as CFRec and PopRec, the influence is more slight.

5.5 Influence of the window size

RLWRec, based on reinforcement learning, models the recommendation task as a MDP, where we use the sliding window of size K to preserve the K songs user recently listened as user's current listening history. In this section we design a experiment to analyze the influence of the window size K. Due to the restriction of computing ability and memory space, we set K as 1, 2, 3 respectively for experiments and other parameters remain unchanged. The result is shown in Fig. 5.

The result shows that the siding window size can affect our model to some extent. It is obvious that the best result is achieved when the size of window is 3. The reason lies in the window of smaller size (i.e. less listening history) will not hold enough information in memory to make better recommendation. Meanwhile, a large window tend to provide our model a longer memory and achieve a significant performance. However, it will cause a larger state space and need more computing time.

5.6 Influence of different recommendation strategies

In the section 5.3, we show three recommendation strategies in our model: native recommendation strategy, recommendation with ϵ-greedy strategy and our recommendation strategy based on tree structure. In this section, we do experiments in the medium frequency dataset to show the influence of the three recommendation strategies. The result is shown in the Fig. 6.

It is clear that our recommendation strategy based on tree structure well outperforms other recommendation strategies. It indicates that our recommendation strategy can balance the accuracy and coverage of recommendation and improve the performance of the recommender system. Moreover, it also shows that the performance of our recommendation is more stable with the growth of the length of song list. Because the native recommendation strategy will restrict the coverage of the recommender system and the ϵ-greedy strategy will bring in the uncertainty and decrease the accuracy of the recommender system.

6 Conclusion

In this paper, we integrate the user feedback and influence between songs in recommender system and model it as Markov decision process. Then we design a novel reinforcement learning framework RLWRec to generate music playlist. Moreover, we propose the state compression method based on collaborative filter

for the dimensionality reduction and design a recommendation strategy based on tree structure to improve the accuracy and coverage of recommendation. Experiments on real datasets verifies the effectiveness of our model.

References

1. Chi, C.Y., Tsai, R.T.H., Lai, J.Y., Hsu, J.Y.j.: A reinforcement learning approach to emotion-based automatic playlist generation. In: 2010 International Conference on Technologies and Applications of Artificial Intelligence. pp. 60–65. IEEE (2010)
2. Konstan, J.A., Riedl, J., Borchers, A., Herlocker, J.L.: Recommender systems: A grouplens perspective. In: Recommender Systems: Papers from the 1998 Workshop (AAAI Technical Report WS-98-08). pp. 60–64 (1998)
3. Koren, Y., Bell, R., Volinsky, C., et al.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (2009)
4. Liebman, E., Saar-Tsechansky, M., Stone, P.: Dj-mc: A reinforcement-learning agent for music playlist recommendation. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. pp. 591–599. International Foundation for Autonomous Agents and Multiagent Systems (2015)
5. Mobasher, B., Dai, H., Luo, T., Sun, Y., Zhu, J.: Integrating web usage and content mining for more effective personalization. In: International Conference on Electronic Commerce and Web Technologies. pp. 165–176. Springer (2000)
6. Puterman, M.L.: Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons (2014)
7. Resnick, P., Varian, H.R.: Recommender systems. Communications of the ACM 40(3), 56–58 (1997)
8. Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems. University of Cambridge, Department of Engineering (1994)
9. Schedl, M., Schnitzer, D.: Location-aware music artist recommendation. In: International Conference on Multimedia Modeling. pp. 205–213. Springer (2014)
10. Shani, G., Heckerman, D., Brafman, R.I.: An mdp-based recommender system. Journal of Machine Learning Research 6(Sep), 1265–1295 (2005)
11. Shi, C., Liu, J., Zhuang, F., Philip, S.Y., Wu, B.: Integrating heterogeneous information via flexible regularization framework for recommendation. Knowledge and Information Systems. pp. 1–25 (2016)
12. Song, Y., Dixon, S., Pearce, M.: A survey of music recommendation systems and future perspectives. In: 9th International Symposium on Computer Music Modeling and Retrieval (2012)
13. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction, vol. 1. MIT press Cambridge (1998)
14. Taghipour, N., Kardan, A., Ghidary, S.S.: Usage-based web recommendations: a reinforcement learning approach. In: Proceedings of the 2007 ACM conference on Recommender systems. pp. 113–120. ACM (2007)
15. Wang, X., Wang, Y., Hsu, D., Wang, Y.: Exploration in interactive personalized music recommendation: a reinforcement learning approach. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 11(1), 7 (2014)
16. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis, University of Cambridge England (1989)
17. Yoshii, K., Goto, M., Komatani, K., Ogata, T., Okuno, H.G.: Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In: ISMIR. vol. 6, p. 7th (2006)