A Hybrid Recommendation for Music Based on Reinforcement Learning

Yu Wang(✉)

School of Electronics Engineering and Computer Science, Peking University, Beijing, China
wangyu18@pku.edu.cn

Abstract. The key to personalized recommendation system is the prediction of users' preferences. However, almost all existing music recommendation approaches only learn listeners' preferences based on their historical records or explicit feedback, without considering the simulation of interaction process which can capture the minor changes of listeners' preferences sensitively. In this paper, we propose a personalized hybrid recommendation algorithm for music based on reinforcement learning (PHRR) to recommend song sequences that match listeners' preferences better. We firstly use weighted matrix factorization (WMF) and convolutional neural network (CNN) to learn and extract the song feature vectors. In order to capture the changes of listeners' preferences sensitively, we innovatively enhance simulating interaction process of listeners and update the model continuously based on their preferences both for songs and song transitions. The extensive experiments on real-world datasets validate the effectiveness of the proposed PHRR on song sequence recommendation compared with the state-of-the-art recommendation approaches.

Keywords: Music recommendation · Hybrid recommendation · Reinforcement learning · Weighted matrix factorization · Markov decision process

1 Introduction

Recommendation systems have become indispensable for our daily life to help users navigate through the abundant data in the Internet. As the rapid expansion of the scale of music database, traditional music recommendation technology is difficult to help listeners to choose songs from such huge digital music resources. How to manage and recommend music effectively in the massive music library has become the main task of music recommendation system [1].

The mainstream recommendation algorithms can be classified as content-based [2, 3], collaborative filtering [5, 25], knowledge-based [6] and hybrid ones [7]. The collaborative filtering methods recommend items to users by exploiting the taste of other similar users. However, the cold-start and data sparse problem is very common in collaborative filtering. In knowledge-based approaches, users directly express their requirements and the recommendation system tries to retrieve items that are analogous to the users' specified requirements. The content-based recommendation approaches are to find items similar to the ones that the users once liked, and the content information or

H. W. Lauw et al. (Eds.): PAKDD 2020, LNAI 12084, pp. 91–103, 2020.
https://doi.org/10.1007/978-3-030-47426-3_8

expert label of items is also needed, but it does not require a large number of user-item rating records [4]. In order to improve performance, above methods can be combined into a hybrid recommendation system. The hybrid approach we use is feature augmentation, which takes the feature output from one method as input to another.

Nowadays, reinforcement learning [8] becomes one of the most important research hotspots. It mainly focuses on how to learn interactively, obtain feedback information in the action-evaluation environment, and then improve the choices of actions to adapt to the environment. In this paper, we propose a personalized hybrid recommendation algorithm for music based on reinforcement learning (PHRR). Based on the idea of hybrid recommendation, we utilize WMF-CNN model which uses content and collaborative filtering to learn and predict music features, and simulate listeners' decision-making behaviors by model-based reinforcement learning process. What's more, we establish a novel personalized music recommendation model to recommend song sequences which match listeners' preferences better. Our contributions are as follows:

- Our proposed PHRR algorithm combines the method of extracting music features based on WMF-CNN process with reinforcement learning model to recommend personalized song sequences to listeners.
- We make innovative improvements to the method of learning listeners' decision-making behaviors. And we promote the accuracy of model-learning by enhancing the simulation of interaction process in the reinforcement learning model.
- We conduct experiments in the real-world datasets. The experimental results show that the proposed PHRR algorithm has a better recommendation performance than other comparison algorithms in the experiments.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents details about the proposed PHRR algorithm. Section 4 introduces experimental results and analyses. In Sect. 5, we conclude our work.

2 Related Work

The recommendation system for music service differs from that for other service (such as movies or e-books), because the implicit user preferences on music are more difficult to track than the explicit rating of items in other applications. Besides, users are more likely to listen a song several times. In recent years, music recommendations have been widely studied in both academia and industry. Since music contains an appreciable amount of textual and acoustic information, several recommendation algorithms model users' preferences based on extracted textual and acoustic features [24].

What's more, the advanced recommendation approaches start to apply reinforcement learning [8] to the recommendation process, and consider the recommendation task as a decision problem to provide more accurate recommendations. Wang et al. [11] proposed a reinforcement learning framework based on Bayesian model to balance the exploration and exploitation of users' preferences for recommendation. To learn user preferences, it uses a Bayesian model that accounts for both audio content and the novelty of recommendations. Chen et al. [12] combined interest forgetting mechanism with Markov

models because people's interest in earlier items will be lost from time to time. They believed that discrete random state was represented by random variables in Markov chain. Zhang et al. [15] took the social network and Markov chain into account, and proposed a PRCM recommendation algorithm based on collaborative filtering. Taking the influence of song transitions into account, Liebman et al. [13] added the listeners' preferences for the transitions between songs to the recommendation process and proposed a reinforcement learning model named DJ-MC. Hu et al. [14] integrated users' feedback into the recommendation model and proposed a Q-Learning based window list recommendation model called RLWRec based on greedy strategy, which traded off between the precision and recall of recommendation. It is a model-free reinforcement learning framework, and it has the data-inefficient problem without model.

Different from the previous research, we focus more on simulating interaction process of listeners based on their implicit preferences for songs and song transitions. Our main aim is to capture the changes of listeners' preferences sensitively in the recommendation process and promote the recommendation quality of music.

3 Our Approach

3.1 Music Feature Extraction

As the song transition dataset is not large enough to train a good model, we can do "transfer learning", i.e. the WMF-CNN process, from the larger Million Song Dataset [22]. To extract the music features, we use WMF [9, 17] to compute the feature vectors of some songs, which is an improved matrix factorization approach for implicit feedback datasets. The feature vectors calculated by WMF are used to train the CNN model [18] to learn the feature vectors of all other songs. Each song's feature vector only needs to be trained once, so it doesn't take a long time to train. Suppose that the play count for listener u listening to song i is r_{ui}, for each listener-song pair, we define a preference variable p_{ui} and a confidence variable c_{ui} (α and ϵ are hyper-parameters, and are set as 2.0 and 1e-6 respectively):

$$p_{ui} = \begin{cases} 1, & r_{ui} > 0 \\ 0, & r_{ui} = 0 \end{cases} \tag{1}$$

$$c_{ui} = 1 + \alpha log\left(1 + \epsilon^{-1} r_{ui}\right) \tag{2}$$

The preference variable p_{ui} indicates whether listener u has ever listened to song i or not. if it is 1, we assume that listener u may like song i. The confidence variable c_{ui} measures the extent to which listener u likes song i. The song with a higher play count is more likely to be preferred. The objective function of WMF contains a confidence weighted mean squared error term and an L2-regularization term, given by Eq. 3.

$$\min_{x^* y^*} \sum\nolimits_{u,i} c_{ui} \left(p_{ui} - x_u^T y_i\right)^2 + \lambda \left(\sum\nolimits_u ||x_u||^2 + \sum\nolimits_i ||y_i||^2\right) \tag{3}$$

where λ is the regularization parameter set as 1e-5, x_u is the latent feature vector of listener u and y_i is the latent feature vector of song i.

In this paper, we use ResNet [26] as our CNN model, the input of the CNN model is mel-frequency cepstral coefficient spectrum (MFCC) [19] of songs, including 500 frames in the time dimension and 12 frequency-bins in the frequency dimension. The output vectors are the *20-dimensional* predicted latent feature vector of songs. The objective function of CNN is to minimize the mean squared error (MSE) and weighted predict error (WPE), given by Eq. 4 (θ representing the model parameters).

$$min_\theta \sum_i ||y_i - y_i'||^2 + \sum_{u,i} c_{ui} \left(p_{ui} - x_u^T y_i'\right)^2 \tag{4}$$

where y_i is the feature vector of song i calculated by WMF, and y_i' is the predicted vector of song i by the CNN model.

3.2 Problem Description

We model the reinforcement learning based music recommendation problem as an improved Markov decision process (MDP) [10], which is denoted as a five-tuple (S, A, P, R, T). And the framework is shown in Fig. 1. Given the song set $M = \{a_1, a_2, \ldots, a_n\}$, the length of song sequences to recommend is defined as k and the mathematical description of the MDP model for music recommendation is as follows.

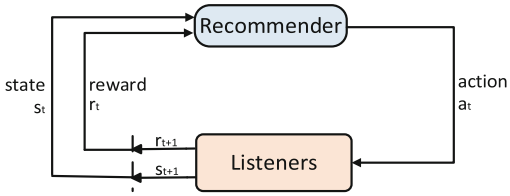


Fig. 1. Reinforcement learning framework for music recommendation.

State set S. The state set denoted as $S = \{(a_1, a_2, \ldots, a_i)|1 \leq i \leq k\}$ is the set of recommended song sequences including all intermediate states. A state $s \in S$ is a song sequence in the recommendation process.

Action set A. The action set A is the actions of listening to songs in M, denoted as $A = \{listening\ to\ song\ a_i | a_i \in M\}$. An action $a_i \in A$ means listening to song a_i.

State transition probability function P. We use abbreviated symbols $P(s, a, s') = 1$ to indicate that when we take action a in state s, the probability of transition to s' is 1, and 0 otherwise, i.e. $P((a_1, a_2, \ldots, a_i), a, (a_1, a_2, \ldots, a_i, a)) = 1$.

Reward function R. The reward function $R(s, a)$ obtains the reward value when listener takes action a in state s, and each listener has a unique reward function. One of our key problems is how to calculate the reward function of new listeners effectively.

Final state T. The final state denoted as $T = \{(a_1, a_2, \ldots, a_k)\}$ is the final recommended song sequence of length k.

Solving the MDP problem means to find a strategy $\pi : S \rightarrow A$, so that we can get an action $\pi(s)$ for a given state s. With the optimal strategy π^*, the highest expected total reward can be generated. However, the listener's reward function is unknown, so the basic challenge of song sequence recommendation is how to model R effectively.

3.3 Listener Reward Function Model

Towards our recommendation problem, the probability function P is already known, so the only unknown element is the reward function R. Most literatures about music recommendation only consider the listeners' preferences for songs, without considering their preferences for song transitions. The reward function R should consider the listeners' preferences both for songs and song transitions, as shown in Eq. 5.

$$R(s, a) = R_s(a) + R_t(s, a) \tag{5}$$

where $R_s : A \rightarrow R$ is the listener's preference reward for song a, and $R_t : S \times A \rightarrow R$ is the listener's preference reward for the song transition from song sequence s to song a.

Listener Reward for Songs. After extracting the features in Sect. 3.1, we obtain a *20-dimensional* song feature vector. Then we use the binarized feature vector by sparse coding of the feature vector to represent the song's features. As the feature vector is *20-dimensional*, it has 20 descriptors. Each descriptor can be represented as *m-bit* binarized feature factors, so the binarized feature vector of song a denoted as $\theta_s(a)$ is a *20 m-dimensional* vector. What's more, each listener has preference factors corresponding to binarized song feature factors respectively. For a *20 m-dimensional* binarized feature vector θ_s, listener u has a *20 m-dimensional* preference vector $\Phi_s(u)$. Therefore,

$$R_s(a) = \Phi_s(u) \cdot \theta_s(a) \tag{6}$$

Listener Reward for Song Transitions. When the listener listens to song a_j after song a_i, we note the reward function as $r_t(a_i, a_j)$. The song transition reward $R_t : S \times A \rightarrow R$ is based on a certain song sequence s and the next-song a to listen, as shown below.

$$R_t(s, a) = R_t((a_1, \ldots, a_{t-1}), a_t) = \sum_{i=1}^{t-1} \frac{1}{i^2} r_t(a_{t-i}, a_t) \tag{7}$$

In Eq. 7, the probability of the i-th song having influence on transition reward is $1/i$. And its influence is attenuated over time, so the i-th song's influence is reduced to $1/i$ times the original. As a result, the coefficient $1/i^2$ is the product of these two $1/i$ [13].

The calculation equation of $r_t(a_i, a_j)$ is similar to $R_s(a)$, as shown in Eq. 8. We use the sparse coding of song transition feature vector to generate the binarized feature vector $\theta_t(a_i, a_j)$. Each descriptor can be represented as m^2*-bit* binarized feature factors. Similar to $\Phi_s(u)$, listener u has a *20* m^2*-dimensional* preference vector $\Phi_t(u)$ for the *20* m^2*-dimensional* binarized feature vector $\theta_t(a_i, a_j)$.

$$r_t(a_i, a_j) = \Phi_t(u) \cdot \theta_t(a_i, a_j) \tag{8}$$

Listener Preference for Songs. We keep the listener's historical song sequence whose length is longer than k_s. In order to make $\Phi_s(u)$ tend to be uniform, we initialize each factor of the vector $\Phi_s(u)$ to $1/(k_s + bins)$, where $bins$ indicates the discretization granularity of song feature and the value is same as m above. For each song a_i in the listener's historical song sequence, $\Phi_s(u)$ adds $1/(k_s + bins) * \theta_s(a_i)$ iteratively so the feature of song a_i can be learned. After $\Phi_s(u)$ is calculated, we normalize $\Phi_s(u)$ so that the weights of *m-bit* factors corresponding to each descriptor sum to 1 respectively.

Listener Preference for Song Transitions. Similar to the process of $\Phi_s(u)$, the length of song transition sequence is $k_s - 1$ noted as k_t. In order to make $\Phi_t(u)$ tend to be uniform, we also initialize each factor of the vector $\Phi_t(u)$ to $1/(k_t + bint)$, and the value of $bint$ is $bins * bins$. Obviously, the song transition pattern of historical song sequence is the best transition pattern that listener prefers. For each transition from a_i to a_j in historical song sequence,$\Phi_t(u)$ adds $1/(k_t + bint) * \theta_t(a_i, a_j)$ iteratively. After $\Phi_t(u)$ is calculated, we normalize $\Phi_t(u)$ in the same way as $\Phi_s(u)$.

3.4 Next-Song Recommendation Model

In order to reduce the time and space complexity of processing, we utilize the hierarchical searching heuristic method [20] to recommend next-song. And search is only performed from the search space where R_s is relatively high (line 1). Besides, we take the horizon problem similar to the Go algorithm into account, which chooses the first step of the path with highest total reward as the next step (lines 9-14).

Algorithm 1: *Recommend next-song by hierarchical searching*

Input: Song dataset M; the length of horizon h
Output: The next-song to recommend *recSong*
1. Select upper median of M as M^* based on R_s
2. $bestList \leftarrow []$, $highestReward \leftarrow -\infty$
3. **while** computational power not exhausted **do**
4. $list \leftarrow []$
5. **for** $i = 1$ to h **do**
6. *songType* ← select randomly from *songTypes*(M^*) (avoid repetition)
7. Add *songType (noted as* st_i*)* to *list*
8. **end for**
9. $allReward := R_s(st_1) + \sum_{i=2}^{h}(R_t((st_1, \ldots, st_{i-1}), st_i) + R_s(st_i))$
10. **if** *allReward* > *highestReward* **do**
11. *highestReward* := *allReward*
12. *bestList* := *list*
13. *songT* := first song type of *bestList*
14. **end if**
15. **end while**
16. *recSong* ← select the song with highest R_s from *songT* (avoid repetition)
17. **return** *recSong*

Since the song space is too large, it is not feasible to select songs from the complete song dataset M. To alleviate this problem, we cluster songs by song type to reduce the complexity of searching (line 6). Clustering by song type is achieved by applying δ-medoids algorithm [21], which is a method for representative selection.

3.5 Song Sequence Recommendation and Update Model

To recommend song sequence, we define r_{adj} as $log(r_i/\overline{r})$, which determines the direction and size of update (lines 2-5). If r_{adj} is a positive value, it means that the listener

likes the recommended song and the update direction is positive, and vice versa. And the relative contributions of the song reward R_s and the song transition reward r_t to their total reward are calculated as w_s and w_t respectively, as shown in Eq. 9 and Eq. 10.

$$w_s = \frac{R_s(a_i)}{R_s(a_i) + r_t(a_{i-1}, a_i)} \tag{9}$$

$$w_t = \frac{r_t(a_{i-1}, a_i)}{R_s(a_i) + r_t(a_{i-1}, a_i)} \tag{10}$$

$$\Phi_s = \frac{i}{i+1} \cdot \Phi_s + \frac{1}{i+1} \cdot \theta_s \cdot w_s \cdot r_{adj} \tag{11}$$

$$\Phi_t = \frac{i}{i+1} \cdot \Phi_t + \frac{1}{i+1} \cdot \theta_t \cdot w_t \cdot r_{adj} \tag{12}$$

Besides, the preference vector Φ_s and Φ_t are updated based on r_{adj}, w_s and w_t, and need to be normalized. This update process considers the changes of listener's interest over time and balances the degree of trusting history with new rewards (line 6-7).

Algorithm 2: *Song sequence recommendation*

Input: Song dataset M; length of song sequence to recommend K
Output: Recommended song sequence *songlist*
1. **for** $i = 1$ to K **do**
2. Use Algorithm 1 to select song a_i
3. The i-th song's total reward $r_i := R_s(a_i) + \sum_{j=1}^{i-1} \frac{1}{j^2} r_t(a_{i-j}, a_i)$
4. $\bar{r} :=$ average$(\{r_1, \ldots, r_{i-1}\})$
5. $r_{adj} := \log(r_i/\bar{r})$
6. Update w_s, w_t, Φ_s, Φ_t using Eq.9 - Eq.12
7. Per $f \in descriptors$, normalize Φ_s^f, Φ_t^f
8. **end for**
9. **return** *songlist* $= \{a_1, \ldots, a_K\}$

4 Experiment

4.1 Datasets

Million Song Dataset. Million Song Dataset (MSD) [22] is a dataset of audio feature for 1 million songs, providing powerful support for the CNN model to learn and extract music features. The dataset is available at http://labrosa.ee.columbia.edu/millionsong/.

Taste Profile Subset Dataset. Taste Profile Subset Dataset [22] as shown in Table 1 is in the form of *listener-song-play count* triple, providing a sufficient amount of dataset for WMF. The dataset is available at https://labrosa.ee.columbia.edu/millionsong/.

Historical Song Playlist Dataset. The dataset is collected from the music website Yes.com [23], which is available at http://lme.joachims.org/. As shown in Table 2, it contains 51,260 historical song sequences.

Table 1. A dataset of listener-song-playcount

#Listeners	#Songs	#Triplets
1000000	380000	48712660

Table 2. Listeners' historical song playlist dataset

#Playlists	#Songs	#Song Transitions
51260	85262	2840554

4.2 Comparison Algorithms and Evaluation Methods

Comparison Algorithms. We compare PHRR with baselines as below. For historical song playlist dataset, we use 90% of the dataset for training and the rest 10% for testing.

PHRR-S: PHRR-S algorithm is just the PHRR recommendation algorithm without taking song transitions into account.

DJ-MC [13]: DJ-MC algorithm is a reinforcement learning model added the listeners' preferences for the transitions between songs to the recommendation process.

PRCM [15]: PRCM algorithm is a collaborative filtering recommendation algorithm taking the social network and Markov chain into account.

PopRec [16]: PopRec algorithm recommends the most popular songs.

RandRec: RandRec algorithm recommends songs randomly.

Evaluation Methods. Our evaluation metrics include hit ratio of the recommended next-songs and F1-score of the recommended song sequences.

Hit Ratio (HR). We calculate hit ratio of the recommended next-songs for evaluation. In the historical song sequence dataset, the first n songs of each song sequence are used to recommend the n+1th songs. We compare the recommended n+1th song with the true n+1th song in the actual song sequence. If it is same, it is hit, otherwise it's not hit.

F1-Score (F1). The second evaluation indicator we use is F1-score. F1-score combines precision and recall of recommendation, and the *Score* calculated by Eq. 13 – Eq. 15 is used to evaluate the effect of song sequence recommendation.

$$Precision = \frac{\left|\left\{a \in S_p \cap a \in S_t\right\}\right|}{\left|S_p\right|} \tag{13}$$

$$Recall = \frac{\left|\left\{a \in S_p \cap a \in S_t\right\}\right|}{\left|S_t\right|} \tag{14}$$

$$Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{15}$$

where S_p represents the recommended song sequences, S_t represents the song sequences presented in the historical song sequence dataset and a indicates a song.

4.3 Experimental Results on Hit Ratio

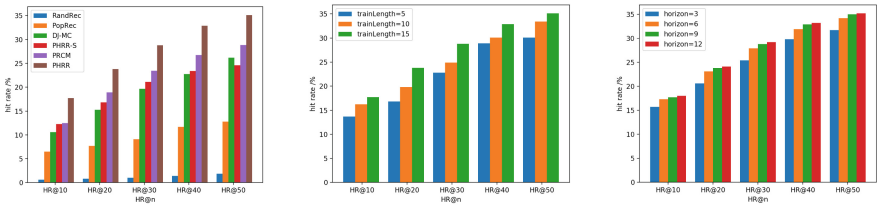
The proposed PHRR algorithm is a recommendation algorithm to recommend song sequences. In this comparison experiment, the recommendation effects are measured by calculating the hit ratio of the recommended next-songs.

Performance Comparison on Hit Ratio. HR@k is the hit rate of the most probable k songs of the recommended next-songs. The results of hit ratio of above recommendation algorithms are shown in Table 3, and the best results are boldfaced.

Table 3. Performance comparison on hit ratio

Algorithm	HR@10	HR@20	HR@30	HR@40	HR@50
PHRR	**0.1787**	**0.2394**	**0.2896**	**0.3302**	**0.3520**
PRCM	0.1255	0.1892	0.2356	0.2681	0.2897
DJ-MC	0.1232	0.1685	0.2110	0.2341	0.2462
PHRR-S	0.1016	0.1534	0.1965	0.2278	0.2619
PopRec	0.0651	0.0773	0.0916	0.1174	0.1289
RandRec	0.0060	0.0083	0.0101	0.0142	0.0186

Effect of Training Length of Song Sequence on Hit Ratio. Reinforcement learning process is based on the feedback of interactive information and simulates the decision behavior of listeners. The longer the length of training song sequence is, the more simulated interactions are in reinforcement learning process. The experimental results of the effect of training length on hit ratio are shown in Fig. 2(b).



(a) (b) (c)

Fig. 2. Experimental results on hit ratio. (a) Comparison experimental results on hit ratio. (b) Effect of training length of song sequence on hit ratio. (c) Effect of horizon length on hit ratio.

Effect of Horizon Length on Hit Ratio. We consider the horizon problem similar to the Go algorithm when recommending next-song, that is, we choose the first song of the song sequence with highest total reward as the next-song (Algorithm 1). The experimental results of effect of horizon length on hit ratio are shown in Fig. 2(c).

Experimental Results and Analyses. The result of Fig. 2(a) shows that hit ratio of PHRR is 7% higher than PRCM, 10% higher than DJ-MC, 11% higher than PHRR-S, 20% higher than PopRec, and the hit ratio of RandRec is as low as 1%. The results of Fig. 2(b) indicates that when the training sequence length n is 15, the hit ratio is higher than when n is 10 or 5. The longer the training sequence length is, the higher the hit ratio of the recommended next-songs is, and the recommendation result will be more accurate. Figure 2(c) shows that, as the horizon length increasing, the hit ratio of the recommended next-songs also tends to be higher.

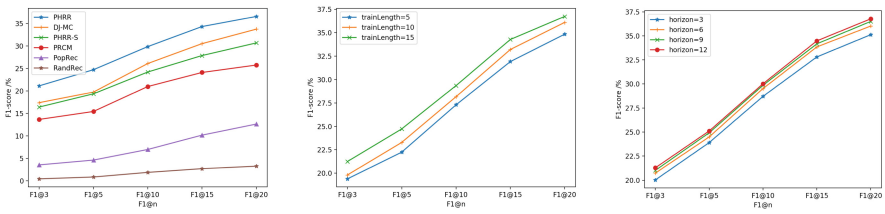
4.4 Experimental Results on F1-Score

In this section, we use F1-score as an evaluation indicator to measure the effect of above algorithms on song sequence recommendation.

Performance Comparison on F1-Score. The results of F1-score of above recommendation algorithms are shown in Table 4. F1@k represents the F1-score of the recommended song sequence whose length is k, and the best results are boldfaced.

Table 4. Performance comparison on F1-score

Algorithm	F1@3	F1@5	F1@10	F1@15	F1@20
PHRR	**0.2113**	**0.2472**	**0.2986**	**0.3432**	**0.3657**
DJ-MC	0.1738	0.1974	0.2610	0.3052	0.3374
PHRR-S	0.1640	0.1935	0.2421	0.2787	0.3068
PRCM	0.1365	0.1542	0.2098	0.2411	0.2576
PopRec	0.0354	0.0461	0.0697	0.1016	0.1262
RandRec	0.0042	0.0083	0.0186	0.0269	0.0325



(a) (b) (c)

Fig. 3. Experimental results on F1-score. (a) Comparison experimental results on F1-score. (b) Effect of training length of song sequence on F1-score. (c) Effect of horizon length on F1-score.

Effect of Training Length of Song Sequence on F1-Score. Compared with other reinforcement learning based algorithms, the proposed PHRR promotes the precision by enhancing the simulation of interaction process. The experimental results of the effect of song sequence training length on F1-score are shown in Fig. 3(b).

Effect of Horizon Length on F1-Score. In the next-song recommendation stage (Algorithm 1), we only recommend the first song of the song sequence with highest total reward, instead of recommending this entire song sequence. Because as noise accumulating during the self-updating process, the variation of the model would be larger. The experimental results of the effect of horizon on F1-score are shown in Fig. 3(c).

Experimental Results and Analyses. As shown in Fig. 3(a), F1-score of PHRR is 4% higher than DJ-MC, 6% higher than PHRR-S, 11% higher than PRCM and 20% higher than PopRec on average. PHRR enhances simulating listener's interaction in the reinforcement learning process, while other algorithms don't consider it. Figure 3(b) presents that, when the song sequence training length n is 15, F1-score is higher than when n is 10 or 5. The longer training length can bring more chances to enhance the simulation of interaction. Figure 3(c) indicates that as the horizon length increasing, F1-score shows a slight higher. The horizon length shouldn't be too long, because too long horizon length is not significantly useful for improving the effect but increases the complexity.

5 Conclusion

In this paper, we propose a hybrid recommendation algorithm for music based on reinforcement learning (PHRR) to recommend higher quality song sequences. WMF and CNN are trained to learn song feature vectors from the songs' audio signals. Besides, we present a model-based reinforcement learning framework to simulate the decision-making behavior of listeners, and model the reinforcement learning problem as a Markov decision process based on listeners' preferences both for songs and song transitions. To capture the minor changes of listeners' preferences sensitively, we innovatively enhance the simulation of interaction process to update the model more data-efficiently. Experiments conducted on real-world datasets demonstrate that PHRR has a better effect of music recommendation than other comparison algorithms.

In the future, we will incorporate more human behavioral characteristics into the model. We also want to analyze the role of these characteristics for recommendation.

Acknowledgments. We would like to thank Kan Zhang and Qilong Zhao for valuable discussions. This work is supported by the National Key R&D Program of China (No. 2019YFA0706401), National Natural Science Foundation of China (No. 61672264, No. 61632002, No. 61872399, No. 61872166 and No. 61902005) and National Defense Technology Strategy Pilot Program of China (No. 19-ZLXD-04-12-03-200-02).

References

1. Bawden, D., Robinson, L.: The dark side of information: overload, anxiety and other paradoxes and pathologies. J. Inf. Sci. **35**(2), 180–191 (2008)
2. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web. LNCS, vol. 4321, pp. 325–341. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72079-9_10
3. Aaron, V.D.O., Dieleman, S., Schrauwen, B.: Deep content-based music recommendation. In: NIPS, vol. 26, pp. 2643–2651 (2013)
4. Brunialti, L.F., Peres, S.M., Freire, V., et al.: Machine learning in textual content-based recommendation systems: a systematic review. In: SBSI (2015)
5. Fletcher, K.K., Liu, X.F.: A collaborative filtering method for personalized preference-based service recommendation. In: ICWS, pp. 400–407 (2015)
6. Koenigstein, N., Koren, Y.: Towards scalable and accurate item-oriented recommendations. In: RecSys, pp. 419–422 (2013)
7. Yao, L., Sheng, Q.Z., Segev, A., et al.: Recommending web services via combining collaborative filtering with content-based features. In: ICWS, pp. 42–49 (2013)
8. Francois-Lavet, V., Henderson, P., Islam, R., et al.: An introduction to deep reinforcement learning. Found. Trends Mach. Learn. **11**(3–4), 219–354 (2018)
9. Li, H., Chan, T.N., Yiu, M.L., et al.: FEXIPRO: fast and exact inner product retrieval in recommender systems. In: SIGMOD, pp. 835–850 (2017)
10. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, Hoboken (2014)
11. Wang, X., Wang, Y., Hsu, D., et al.: Exploration in interactive personalized music recommendation: a reinforcement learning approach. In: TOMM, pp. 1–22 (2013)
12. Chen, J., Wang, C., Wang, J.: A personalized interest-forgetting markov model for recommendations. In: AAAI, pp. 16–22 (2015)
13. Liebman, E., Saartsechansky, M., Stone, P.: DJ-MC: A reinforcement-learning agent for music playlist recommendation. In: AAMAS, pp. 591–599 (2015)
14. Hu, B., Shi, C., Liu, J.: Playlist recommendation based on reinforcement learning. In: ICIS, pp. 172–182 (2017)
15. Zhang, K., Zhang, Z., Bian, K., et al.: A personalized next-song recommendation system using community detection and markov model. In: DSC, pp. 118–123 (2017)
16. Ashkan, A., Kveton, B., Berkovsky, S., et al.: Optimal greedy diversity for recommendation. In: IJCAI, pp. 1742–1748 (2015)
17. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: ICDM, pp. 263–272 (2008)
18. Kim, P.: Convolutional Neural Network. In: MATLAB Deep Learning, pp. 121–147 (2017)
19. On, C.K., Pandiyan, P.M., Yaacob, S., et al.: Mel-frequency cepstral coefficient analysis in speech recognition. In: Computing & Informatics, pp. 2–6 (2006)
20. Urieli, D., Stone, P.: A learning agent for heat-pump thermostat control. In: AAMAS, pp. 1093–1100 (2013)
21. Liebman, E., Chor, B., Stone, P.: Representative selection in nonmetric datasets. Appl. Artif. Intell. **29**(8), 807–838 (2015)
22. Bertin-Mahieux, T., Ellis, D.P.W., Whitman, B., et al.: The million song dataset challenge. In: ISMIR (2011)
23. Chen, S., Xu, J., Joachims, T.: Multi-space probabilistic sequence modeling. In: KDD, pp. 865–873 (2013)

24. Zhang, S., Yao, L., Sun, A., et al.: Deep learning based recommender system: a survey and new perspectives. ACM Comput. Surv. **52**(1), 1–38 (2019)
25. Wu, Y., Dubois, C., Zheng, A.X., Ester, M.: Collaborative denoising auto-encoders for top-n recommender systems. In: WSDM, pp. 153–162 (2016)
26. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016)