# Hybrid SkipAwareRec: A Streaming Music Recommendation System

Rui Ramos[1(✉)], Lino Oliveira[1], and João Vinagre[1,2,3]

[1] INESC TEC, Porto, Portugal
rui.j.ramos@inesctec.pt
[2] Joint Research Centre - European Commission, Seville, Spain
[3] Faculty of Sciences - University of Porto, Porto, Portugal

**Abstract.** In an automatic music playlist generator, such as an automated online radio channel, how should the system react when a user hits the skip button? Can we use this type of negative feedback to improve the list of songs we will playback for the user next? We propose SkipAwareRec, a next-item recommendation system based on reinforcement learning. SkipAwareRec recommends the best next music categories, considering positive feedback consisting of normal listening behaviour, and negative feedback in the form of song skips. Since SkipAwareRec recommends broad categories, it needs to be coupled with a model able to choose the best individual items. To do this, we propose Hybrid SkipAwareRec. This hybrid model combines the SkipAwareRec with an incremental Matrix Factorisation (MF) algorithm that selects specific songs within the recommended categories. Our experiments with Spotify's Sequential Skip Prediction Challenge dataset show that Hybrid SkipAwareRec has the potential to improve recommendations by a considerable amount with respect to the skip-agnostic MF algorithm. This strongly suggests that reformulating the next recommendations based on skips improves the quality of automatic playlists. Although in this work we focus on sequential music recommendation, our proposal can be applied to other sequential content recommendation domains, such as health for user engagement.

**Keywords:** Sequential recommendation · Next best action recommendation · Implicit negative feedback · Reinforcement learning

## 1 Introduction

Music helps in our intellectual development, stimulating creativity, and provides a means of expressing ourselves.[1] Humans spend a lot of time listening to it,[2] and the songs that we choose to listen to are influenced by a number of factors, but the most obvious are our individual taste and preferences. However, given

[1] https://www.betterup.com/blog/benefits-of-music. Accessed: 2023-03-27.
[2] https://musicalpursuits.com/music-streaming/. Accessed: 2023-03-27.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
N. Moniz et al. (Eds.): EPIA 2023, LNAI 14115, pp. 275–287, 2023.
https://doi.org/10.1007/978-3-031-49008-8_22

the huge music catalogues available today in most music streaming services, choosing the next song to listen to is not always easy.

Our contribution, Hybrid SkipAwareRec, is a proposal of a next-action recommendation system for music that interacts in real time with the user. It is divided in two parts: (1) SkipAwareRec, a Reinforcement Learning (RL) Recommender System whose objective is to recommend the best possible next music categories, by collecting user feedback—positive and negative—during the interaction of the recommender agent with the user, and (2) Hybrid SkipAwareRec, that combines SkipAwareRec with an incremental Matrix Factorisation (MF) algorithm that selects the items belonging to the recommended category that best match user preferences. Figure 1 illustrates how these two components are combined in an RL setting. In our problem, Actions correspond to music categories composed of subsets of items, and ISGD is an incremental MF algorithm for personalised item recommendation. The figure illustrates our final setting—Hybrid SkipAwareRec—, in which we use SkipAwareRec to predict the best next category in the streaming session. We then reduce the candidate space such that ISGD considers only items belonging to that category. To account for errors of SkipAwareRec in category prediction, we add to the resulting recommended items some additional recommendations obtained from ISGD considering the whole item space. We provide further details of this hybrid setting in Sect. 3.3.

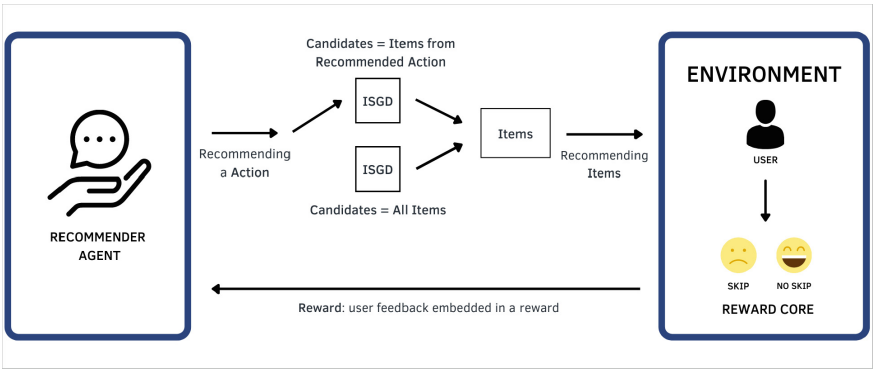


**Fig. 1.** Hybrid SkipAwareRec overview

In the remainder of the paper we provide an overview of related work in Sect. 2, followed by the description of our methodology and proposed solution in Sect. 3. Our results are presented and briefly discussed in Sect. 4. Section 5 summarises the main conclusions and indicates future research directions.

## 2 Related Work

The ability of recommender systems to adapt their behaviour to individual preferences is becoming more and more valuable [3]. Recently, the iterative recom-

mendation approach has been successfully applied to different real-world recommendation tasks in several specific scenarios, such as e-learning, recommendation of music, movies, news, and professional skills [7]. It has also been shown that exploiting negative feedback can be beneficial in music recommendation [1,8].

### 2.1 Recommendations with Negative Implicit Feedback

There are several types of user feedback that can be used to update the models responsible for generating recommendations. Most existing contributions exploit explicit feedback—which effectively demonstrates the user's feelings towards the recommendations—and positive implicit feedback—users' positive interests acquired through indirectly monitoring its behaviour. When it comes to negative implicit feedback, there are few works that cover this type of user response. Furthermore, the few contributions that cover this type of feedback use it in boolean form, that is, without quantification, just existence or not.

Regarding non-sequential recommendation paradigms, there are some works that focus on natural—not artificial—negative implicit feedback. Peska and Vojtáš [9] studied the implication of using implicit negative feedback in e-commerce oriented recommendation systems, using page visits, mouse movements, scrolling, among others. In all test cases of their work, the combined negative preferences were superior to other methods for larger train sets, suggesting that negative preferences may become important while trying to sort already good objects. Lee et al. [5] introduced a mechanism to infer negative implicit feedback and test the feasibility of this type of user response as a way to represent what users want in the context of a job recommendation system, using as negative implicit feedback the action of opening a job offer and closing it, rather than saving it. Compared to cases using only positive preferences, the distinction between good and bad jobs was significantly clear when using negative preferences.

Regarding sequential recommendation using negative implicit feedback, Zhao et al. [12] proposed a pairwise Deep RL for the recommendation framework with the ability to continuously improve its strategies during interactions with users by collecting both positive and negative feedback. In their work, authors separated negative and positive items during learning—they model the sequential interactions between users and a recommender system as an Markov Decision Process (MDP) and leverage RL to automatically learn the optimal strategies by recommending items on a trial-error basis and receive reinforcement of these items from user feedback.

### 2.2 Sequential Music Recommendation

The vast majority of contributions in sequential music recommendation are based on RL. The problem is modelled as a MDP with the goal of obtaining a controlled environment to apply an RL task. Contributions differ through different components of the MDPs and stages of the algorithm they use to recommend.

In the work of Hu et al. [4], each state represents a sequence of songs clusters listened by the user, and listening to a song is considered an action. As a reward, authors used a function of user listening, collecting, and downloading songs, and the RL algorithm used was Q-Learning. Similarly, Chi et al. [2] also use clusters of songs—researchers categorised all the songs in the experimental dataset into four classes manually pre-annotated by users with correspondence to their emotion class. Authors also used implicit feedback in their work, with its reward function consisting of implicit feedback—listening time and number of replays—and explicit feedback—ratings. The algorithms used were SARSA and Q-Learning. There are two contributions that are quite similar, both Wang [11] and Liebman [6] use song sequence to represent states, the listening of a song as a representation of an action and a function of user's preferences over individual songs and song transitions as a reward function. Wang uses a tree-search heuristic algorithm, while Liebman uses a Policy-gradient algorithm.

## 3 Methodology and Proposed Solution

Hybrid SkipAwareRec is divided into three main tasks: the generation of a set of actions that are used for recommendation, the recommendation of an action and the recommendation of specific items.

### 3.1 Action Set Generation

In a recommendation problem, one of the most important tasks is to learn user preferences. The tastes of each user can be represented by topics, or categories, that represent sets of similar specific items in a given context. For example, in a TV content recommendation context, it is important to understand what the users favourite topics are, such as comedy, sports, action, among others.

An RL approach with all available items represents a giant action space and is therefore very difficult to model. The task of grouping similar items into categories also greatly reduces the search space, enabling the RL algorithm to work with a small action set.

To generate the action set that correspond to musical categories, we use standard K-Means, because of its scalability and convergence guarantees. Through various song attributes, such as duration, year of release, acoustics, beat strength, resonance, danceability, among many others that can be chosen by whoever performs the task, it is possible to create a set of categories resulting from the grouping of songs.

### 3.2 Next Best Action Recommendation

In a real scenario, a recommender system is in constant interaction with a user. The recommendation task can be seen as an iterative process in which the user receives a recommended item and provides feedback to the system, allowing that in the next iteration the recommendation model incorporates the feedback

received and makes a new recommendation, closer to the user's current preferences.

When listening to music, the choice of the next song to listen to is conditioned by several factors, with user preferences being the most important one to consider. With this in mind, the goal of this stage is to use negative implicit feedback, more specifically song skips, to train an RL algorithm that learns user preferences during music streaming sessions and generates good category recommendations based on this learning. Figure 2 shows the general diagram of this process.

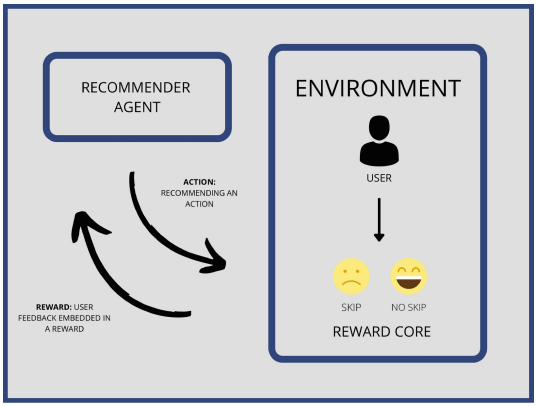


**Fig. 2.** Iterative Learning via Implicit Negative Feedback

In an RL problem, there is a learner and decision maker—the agent—that interacts with the environment. In our case, the environment is the user of the system that receives and reacts to recommendations—provides rewards and a new state based on the agent's actions. Thus, in this type of learning, we do not explicitly fit an agent to a dataset such as in a supervised learning task, but rather present it with rewards that can be either positive, neutral or negative.

**Agent and Environment** In our MDP, the environment is completely observable, i.e., the agent directly observes the state of the environment. This allows us to make the analogy to a recommender system that directly observes user's behaviour. The agent is the recommender system whose actions correspond to recommendations to the user. The user observes this action and provides implicit feedback, which is used as input to a function that outputs a reward. After an action has been performed, a new state is transitioned to and the process repeats until the session is over.

**States and Actions** The set of states $S_n$ in an MDP defines the context of the environment, i.e., it is a set of tokens that represent every state $S_i$ that the

agent can be in. An action $A$, on the other hand, is what causes the environment to change its state. In our problem setting, an action represents the listening of a song from a certain musical category. We define the states as the set of the previous $K$ actions performed by the user—i.e. the categories of the previous songs. This way, we can preserve historical information regarding the transition between musical categories by the user, and at the same time be able to focus on recent actions. Our model also allows state resets, to account for context switches in which the session is interrupted—a situation where the sequence of actions is uninformative to the recommender. An example of a context switch is the change of playlist or streaming source.

**Reward** The goal of an RL algorithm is to maximise the accumulated reward. Rewards are given by the output of a real-valued function that models how good or promising certain actions are in certain states. Essentially, it indicates "how good it is to choose/be in a certain state". A Reward $R_{S_t}$ is received for transitioning to a state $S_t$. The objective in RL is to choose action $A$ so as to maximise $R_{t+1}, R_{t+2}, R_{t+3}, \ldots$ in order to get the best long-term accumulation of rewards.

In our problem setting, the central element and focus of the reward function is the implicit negative feedback in the form of song skips. Furthermore, this information is quantified, since we have access to different types of skips, depending on when the skip is made. We assigned different weights to the feedback depending on its moment of skip—we gave higher reward values to skips that were transmitted later, or even not transmitted at all.

Two analyses were conducted on data from the Spotify Sequential Skip Prediction Challenge,[3] which showed that skip behaviour is not related to the time of day or type of music listening context. In addition, a survey was conducted which confirmed the intuition that the value of a skip in a user's own private playlist is different from a skip in other playlists (e.g. public or other users'). Only 2.4% of users who responded to the survey stated that they skip a song in their own private collection because they do not like it. The remaining respondents gave reasons such as "I do not want to listen to that specific song at the moment" and "I am trying to find a specific song through the jumps between songs" to skip songs in this type of context. Thus, we used this insight to shape our reward function differently for the user's own collections and for other contexts.

Table 1 shows the values of the reward function defined for the problem.

[3] https://www.aicrowd.com/challenges/spotify-sequential-skip-prediction-challenge. Accessed: 2023-03-28.

**Table 1.** Reward function output

| Skip moment | Reward—user collection | Reward—other context |
|---|---|---|
| 1 | 0.7 | 0.1 |
| 2 | 0.7 | 0.3 |
| 3 | 0.7 | 0.7 |
| 4 | 0.9 | 0.9 |
| Not skipped | 1 | 1 |

**Reinforcement Learning Algorithm** To solve the problem of sequentially choosing the next music category, from which the recommendations will arise, we used Q-Learning, a value-based off-policy temporal-difference (TD) RL algorithm whose goal is to learn the value of an action in a given state, by learning the action-value function. This algorithm finds an optimal policy to maximise the expected value of the total reward over any and all successive steps from the current state.

This algorithm was chosen because of its great advantage of being able to compare the expected utility of available actions without requiring a model of the environment. With this type of model-free approaches, we can update the whole value function and policy with every new sample.

**Reinforcement Learning Mechanism** In an iterative and sequential application it is mandatory that the model is constantly updated with each new iteration. To do this, we use the properties of RL to update our model each time the user transmits feedback on a given item. Whenever an information is read or received that a user has chosen an action, in addition to the state transition performed in the MDP, the Q-Table in Q-Learning is also updated. The Next Best Action (NBA) recommendation at step is obtained by choosing the action with the highest Q-value in a certain state.

In order to solve the Exploration vs Exploitation problem, an $\epsilon$-Greedy strategy is used to balance the two techniques during recommendations. This paradigm consists of randomly selecting, with a defined probability, whether to exploit or explore: 1-$\epsilon$ of the time the algorithm exploits and $\epsilon$ of the time the algorithm explores. With this, the user will receive recommendations aligned with their already demonstrated preferences, as well as new and different recommendations that allow the agent to discover new information about his/her preferences.

**Recommendation Strategy** There are two stages involved in our algorithm for recommending the NBA. In an initial phase, the algorithm is only fed with data in order to build its Q-Table. Considering that there is enough real data to perform a good learning, it is valuable not to start recommendations right away after the start of learning because, as the name of the approach suggests, the

algorithm learns by reinforcement, and with little interaction, its performance will not produce good results. It is preferable to have an initial training phase with real historical data, rather than using the approach right away to generate recommendations with few interactions performed.

In a second phase of the RL approach, the connection of the model is now only and exclusively with a specific user. This way, the algorithm when it is framed with a single user already has a good basis of how to recommend, and starts to mould itself to that user in the long term.

### 3.3 Next Best Items Recommendation

In Sect. 3.2, we describe SkipAwareRec, an RL algorithm that recommends the next best item category that corresponds to the action, at each state, that maximises the expected accumulated reward. The idea is then to use this category to narrow down the recommendation candidates space. For the actual item recommendation, any general-purpose collaborative filtering algorithm that learns from implicit feedback can be used in our problem. In this work, we use ISGD [10], an incremental MF algorithm for implicit positive-only feedback. The ability to learn incrementally fits well with our sequential learning setting, since both the RL and the MF algorithms can learn online, in parallel, from the same sequence of user actions.

**Hybrid SkipAwareRec** The most straightforward combination of SkipAwareRec with ISGD would be to simply have the latter recommend items from within the category recommended by the first. However, as we show in Table 2, this is not the best approach. For example, let us imagine a hypothetical scenario where actions/categories correspond to music genres and SkipAwareRec recommends Jazz. There may be songs from the Blues genre that are also good recommendations for the user. Solely using the NBA approach, all Blues songs will be simply ignored. Maybe more importantly, if SkipAwareRec fails to predict the correct category, the recommendation is guaranteed to fail.

To address this, we adapt the use of the ISGD algorithm to generate a Top K recommendations, whose K/2 recommendations come from the recommended NBA and K/2 recommendations from all existing items in the data space, as illustrated in Fig. 3. If the recommendations of the two ISGDs differ, we have a total of K recommendations. Otherwise, no recommendations are added, and we recommend only the unique ones in the list resulting from the junction of the two sets, resulting in a total of recommendations between K/2 and K-1. This way, we present the user with less limited and concentrated recommendations. We call this approach Hybrid SkipAwareRec.

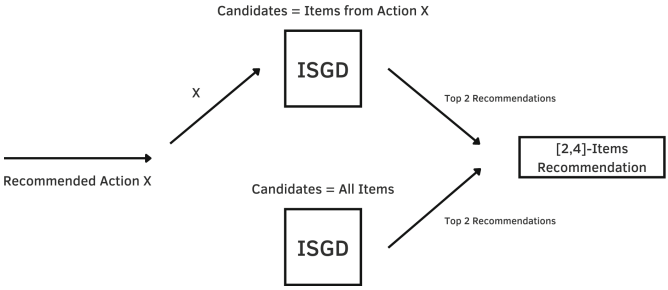


**Fig. 3.** Example of Hybrid SkipAwareRec with K = 4

**Recommendation Strategy** Similarly to the algorithm proposed to recommend the NBA, this paradigm also has 2 stages. In the first stage, the recommendation model is fed only with historical data from several users. This first phase runs in parallel with the first phase of SkipAwareRec.

The second stage starts when the algorithm is delivered to online users. ISGD continues to recommend and collect information about user iterations with items, in order to update its model in real time, thus preserving the incrementality of the solution overall.

## 4 Experiments and Results

### 4.1 Data Setup and Model Training

To evaluate our solution, we used data from the Spotify Sequential Skip Prediction Challenge—1,000,003 entries, where each entry represents the listening of a song in a given session, 59,062 sessions, of a length between 10 and 20 songs each, and a total of 174,768 individual songs associated with all these sessions. We only use data from premium sessions, since only these users are allowed to control the order of songs. We performed clustering to generate the set of possible actions/categories, resulting in a total of 23 categories, chosen using the Elbow Method. With this categorisation, there is a mode of 4 clusters per session.

To train the SkipAwareRec and ISGD models, we broke the sessions data into a training set with 85% of the data and a test set with the other 15%. We feed SkipAwareRec with the training data containing skip information. With this, we build sequential states, for each session, and update our Artificial Intelligence (AI) model. In parallel, the same training dataset was used to build the ISGD model—but without skip information—that will also be incrementally updated in a next stage.

The dataset contains 5 different types of skip, relating to different moments of the songs, which were used for the definition of the MDP reward function. After this learning stage, the RL-based model is ready to recommend personalised actions by reading an initial action by a given user. After connecting to a single user, the algorithm starts to learn his/her preferences and adapting to

the individual user. The ISGD model continues learning and providing recommendations in the test phase.

### 4.2 Evaluation

We evaluated each recommendation in each iteration read in the different sessions. Whenever new information (user, item) is received, a recommendation is generated. If this recommendation is consistent with the item read, it is considered a *hit*, otherwise it is a *miss*. This evaluation equation can be described by:

$$Hit\ Ratio\ (HR) = \frac{\#\ Guessed\ Recommendations}{\#\ Recommendations} * 100$$

Table 2 demonstrates the results of two ISGD models. The first ISGD recommends 2 items only from the category generated by SkipAwareRec, while the second recommends 2 items from the entire universe of songs. The ISGD algorithm that recommends from the whole item set performs better. This happens because when the NBA recommendation fails, the entire recommendation fails.

**Table 2.** Iterative items recommendations results in all recommendations moments

| Recommendation algorithm | Guessed recommendations | Number of recommendations | HR (%) |
|---|---|---|---|
| ISGD with SkipAwareRec items | 6626 | 107864 | 6.14 |
| ISGD with all items | 8043 | 107864 | 7.46 |

Given this result, we wanted to check the results of the SkipAwareRec+ISGD combination only when SkipAwareRec hits the right category—Table 3 shows these results. In this scenario, the ISGD algorithm that recommends only items from the generated NBA is much higher (almost double) than the ISGD that recommends items from the entire universe of songs.

**Table 3.** Iterative items recommendations results when SkipAwareRec guesses the next action

| Recommendation algorithm | Guessed recommendations | Number of recommendations | HR (%) |
|---|---|---|---|
| ISGD with SkipAwareRec items | 6626 | 31398 | 21.10 |
| ISGD with all items | 3493 | 31398 | 11.12 |

This insight is what has driven us to devise Hybrid SkipAwareRec. Given that when getting the category right, the ISGD that recommends only items from that same category is much better, and that overall, the ISGD that recommends items from the music universe is slightly better, we decided to create the hybrid strategy. We put together the recommendations from both ISGDs presented and

arrive at the results in Table 4. As can be seen, the Hybrid SkipAwareRec had a performance of 10.36%, considerably higher than standalone ISGD in Table 2. The chart in Fig. 4 presents a performance comparison of the three different approaches.

**Table 4.** Iterative items recommendations results with Hybrid SkipAwareRec

| Recommendation algorithm | Guessed recommendations | Number of recommendations | HR (%) |
|---|---|---|---|
| Hybrid SkipAwareRec | 11176 | 107864 | 10.36 |

Finally, it is important to note that we are recommending a maximum of 4 songs, out of a universe of a total of 174,768 songs. For reference, a random recommender would have a hit probability of 0.0023%.

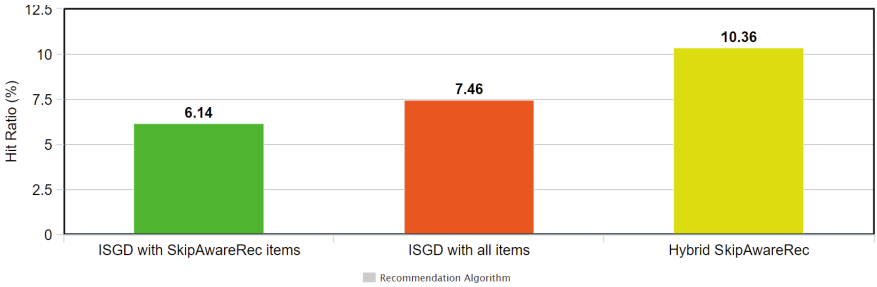


**Fig. 4.** Performance comparison of the different approaches

## 5 Conclusions and Future Work

We present in this paper a sequential recommendation algorithm for music that relies on real time user feedback in the form of song listens and song skips. Our algorithm is divided in two parts: (1) SkipAwareRec, a sequential NBA recommendation algorithm based on RL that selects the best actions corresponding to item categories, and (2) Hybrid SkipAwareRec, that combines SkipAwareRec with ISGD, an incremental MF algorithm that selects the items belonging to the recommended category that best match user preferences.

Our offline results show that our approach clearly outperforms a standalone version of the MF algorithm. This result encourages many future work directions. One research path would be to experiment with more publicly available datasets—unfortunately not abundant—, preferably with longer sessions that allow the algorithm to learn more about the users. Another direction would be to focus on the algorithmic approach, such as studying the impact of the design

choices of the reward function, using different RL approaches, clustering algorithms with different levels of granularity, as well as experimenting with other incremental baseline algorithms. Finally, we would like to evaluate SkipAwareRec online, with real users.

An adaptation of SkipAwareRec is also being implemented with the goal of recommending customised content to individual users in a healthcare use case.

**Acknowledgements.** This work is co-financed by Component 5—Capitalisation and Business Innovation, integrated in the Resilience Dimension of the Recovery and Resilience Plan within the scope of the Recovery and Resilience Mechanism (MRR) of the European Union (EU), framed in the Next Generation EU, for the period 2021–2026, within project HfPT, with reference 41.

The third author contributed to the technical work in this paper while affiliated with INESC TEC and the University of Porto and to the writing of the paper while affiliated with the Joint Research Centre of the European Commission.

## References

1. Chao, D.L., Balthrop, J., Forrest, S.: Adaptive radio: achieving consensus using negative preferences. In: Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work, GROUP 2005, Sanibel Island, Florida, USA, November 6–9, 2005, pp. 120–123. ACM (2005)
2. 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: Proceedings of the 2010 Conference on Technologies and Applications of Artificial Intelligence TAAI2010. National Taiwan University, Yuan Ze University (2010)
3. den Hengst, F., Grua, E.M., el Hassouni, A., Hoogendoorn, M.: Reinforcement learning for personalization: a systematic literature review. Data Sci. **3**(2), 107–147 (2020)
4. Hu, B., Shi, C., Liu, J.: Playlist recommendation based on reinforcement learning. In: Intelligence Science I—Second IFIP TC 12 International Conference, ICIS 2017, Shanghai, China, 25–28 Oct 2017, Proceedings, volume 510 of IFIP Advances in Information and Communication Technology, pp. 172–182. Springer, Berlin (2017)
5. Lee, D.H., Brusilovsky, P.: Reinforcing recommendation using implicit negative feedback. In: User Modeling, Adaptation, and Personalization, 17th International Conference, UMAP 2009, formerly UM and AH, Trento, Italy, 22–26 June 2009. Proceedings, Vol. 5535 of Lecture Notes in Computer Science, pp. 422–427. Springer, Berlin (2009)
6. 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, AAMAS 2015, Istanbul, Turkey, 4–8 May 2015, pp. 591–599. ACM (2015)
7. Lin, Y., Liu, Y., Lin, F., Wu, P., Zeng, W., Miao, C.: A survey on reinforcement learning for recommender systems. CoRR (2021). arXiv:abs/2109.10665
8. Park, M., Lee, K.: Exploiting negative preference in content-based music recommendation with contrastive learning. In: RecSys '22: Sixteenth ACM Conference on Recommender Systems, Seattle, WA, USA, 18–23 Sept 2022, pp. 229–236. ACM (2022)

9. Peska, L., Vojtás, P.: Negative implicit feedback in e-commerce recommender systems. In: 3rd International Conference on Web Intelligence, Mining and Semantics, WIMS '13, Madrid, Spain, 12–14 June 2013, p. 45. ACM (2013)
10. Vinagre, J., Jorge, A.M., Gama, J.: Fast incremental matrix factorization for recommendation with positive-only feedback. In: User Modeling, Adaptation, and Personalization - 22nd International Conference, UMAP 2014, Aalborg, Denmark, 7–11 July 2014. Proceedings, Vol. 8538 of Lecture Notes in Computer Science, pp. 459–470. Springer, Berlin (2014)
11. Wang, Y.: A hybrid recommendation for music based on reinforcement learning. In: Advances in Knowledge Discovery and Data Mining—24th Pacific-Asia Conference, PAKDD 2020, Singapore, 11–14 May 2020, Proceedings, Part I, Vol. 12084 of Lecture Notes in Computer Science, pp. 91–103. Springer, Berlin (2020)
12. Zhao, X., Zhang, L., Ding, Z., Xia, L., Tang, J., Yin, D.: Recommendations with negative feedback via pairwise deep reinforcement learning. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, 19–23 Aug 2018, pp. 1040–1048. ACM (2018)