Reinforcement Learning for Recommender Systems

Priyansh Shukla
Department of Mechanical Engineering
Indian Institute of Technology, Ropar
Ropar, India
2018meb1022@iitrpr.ac.in

Parnavi Shrikhande
Department of Mechanical Engineering
Indian Institute of Technology, Ropar
Ropar, India
2018med1007@iitrpr.ac.in

***Abstract*—With the number of songs as well as music streaming websites being released each year growing exponentially the need for a good music recommender system is now more than ever. This paper presents a hybrid recommender system which incorporates the contextual multi-armed bandit problem wherein each arm is a feature vector containing the users' preferences and the song's details. Our model makes use of context-based filtering methods which can further be boosted with collaborative filtering to recommend the best possible songs to the user. The model recommends songs to the user, asks the user to rate the songs (reward), and based on the ratings, it computes the Euclidean distance between the features of the top-rated songs and other non-rated songs and further recommends a new set of songs which have the least distance. This paper also presents a solution to deal with the cold-start problem which arises in most recommender systems. Our model performs highly and is capable of reliably recommending good songs based on the user's rating history.**

Keywords—Music Recommender System, Reinforcement Learning, Contextual Multi-Armed Bandits, Content-Based Recommenders.

I. Introduction

Within the last decade, there has been a drastic transformation in the music industry, with thousands of songs being released each year, all varying from one end of the spectrum to the other in terms of their genre and valence. In addition to this, there have also been numerous new artists coming up in this industry, introducing newer genres thus, adding to the already vast music library. Due to this surge, the traditional ways of exploring, finding and liking music have faded away. Just until couple of decades ago, songs were recommended mostly by radio, and the other mode being word of mouth. In the past, one could always ask the owner of the record store for music recommendations. Now, however, we can no longer expect that the teenager behind the cash register will be an expert in new music, or even be someone who listens to music at all.

Let's also consider the music consumption behavior which has evolved largely in the recent years. Due to technological improvements, personal music collections have grown and users want to explore more genres of songs, thus resulting in an overwhelming number of choices which the users have to face. The Paradox of choice, as stated by Schwartz [1], says that users often become paralyzed and doubtful when provided with such an overwhelming number of choices. Thus, there arises a need to remove some of these choices, especially those song which based on the user history, the user typically would not listen to. This can be achieved by giving personalized recommendations to the users based on their previously liked songs. Thus, strong and reliable recommendation systems are more important now than ever.

Many of the previous papers written on music recommendation systems revolve around classic machine learning and deep learning algorithms, while only a few papers cover the aspect of implementing these systems using reinforcement learning. The use of reinforcement learning (RL) is very beneficial as the recommender can be modelled around sequential user interaction for optimizing the users' overall satisfaction. Thus, RL based recommender systems have the ability to continuously update their recommendations during the interactions with the listeners, until the model finally converges to recommending the best songs based on the users' listening history. In addition to this, leveraging RL models such as Multi-Armed Bandits (MAB) allows our model to not only exploit the history of the users' liked songs and recommend similar songs, but also allows it to recommend new songs and artists belonging to different genres every now and then, so that the listeners can explore more of their music tastes.

II. Literature Review

In this section, we briefly describe the previous works done in related fields, which can be classified under two categories, first being recommender models, which do not implement RL and models which do.

In the first category, we consider the traditional recommender systems which do not incorporate reinforcement learning techniques to design their model. The most successful and widely-used technique is Collaborative Filtering (CF), which is based on the rationale that for users having similar tastes, most relevant songs can be recommended by comparing their co-ratings on particular songs and then recommending liked songs of one user to the others. However, conventional CF based methods are very likely to suffer from data scarcity. An advanced CF technique is matrix factorization (MF) wherein these models ([2], [3], [4], [5]) characterize songs and users by vectors and reduce the dimensions of the matrix, thus solving the data scarcity issue.

Another common approach is content-based filtering **Error! Reference source not found.**, which compares different features of the user's liked items and then uses these features to recommend items having similar features. Hybrid Recommender systems combine these two or more techniques so as to generate a more complex recommender system. The only requirement here is a rich dataset that accurately captures the item's features.

Logistic regression models and its variants [7] have also been implemented by considering the recommender model to be a binary classifier. However, these models impose another problem of not being able to generalize the feature interactions which rarely appear.

Another solution which has been worked upon is deep learning-based recommender system [8], [9], [10], [11]. In this, the model captures more complicated features, thus allowing for more complex data representations in higher layers.

Now let's consider the second category of recommender models, i.e., those which leverage reinforcement learning for training their models. Due to the high time complexity of model-based techniques, most of the previous works rely upon model-free techniques.

In value-based approaches, out of all the actions, the one with maximum Q-value is selected as the best action. In [12], the model takes into consideration the users' positive as well as negative feedback. In [13], Dueling Q-network is utilized to model Q-value of a state-action pair. However, in these approaches, it is very inefficient to evaluate the Q-values of a large number of actions.

Policy-based approaches ([14], [15], [16]) output a continuous action representation, and the recommendation is generated by ranking the items with their scores which are computed by a pre-defined function with the action representation and item embeddings as the inputs.

III. Preliminaries

i. Types of Recommender Systems

As discussed before, there are a quite a few types of recommendation methods which are popularly being used in such models. The first method is demographic filtering, which is used to identify the kind of users that like a certain item. This technique classifies the user profiles in clusters according to some personal data (for ex., age), geographic data and psychographic data (interests, lifestyle, etc.). The second and the most commonly used technique is collaborative filtering. This approach predicts the user preferences for songs based by learning past user-song relationships. For this, the user gives a feedback, or a rating, to the model, which then recommends songs based on the feedback that other users have provided. One of the sub-divisions of CF is Item-Based Neighborhood method. This method exploits the similarity among items by looking at a set of songs which different users have co-rated and then computes the similarity among the other songs to decide whether or not to recommend it.

The third method is Content-Based Filtering (CB). In this approach, the model collects information describing the songs (such as the energy and the danceability of the song) and then, based on the user's choices, it predicts the song which the user could like by computing the Euclidean Distance between the different features of the rated songs and unrated songs. Our algorithm majorly makes use of this method thus making the model a CB Recommender.

ii. Euclidean Distance

The Euclidean distance is calculated using the following formula:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

where x and y are vectors where x contains the song features according to the user's preferences (such as danceability, energy etc.) and y contains the features of the songs which are to be recommended.

iii. Weighted Euclidean Distance

The weighted Euclidean distance is calculated using the following formula:

$$D_{weighted}(w,x,y) = \sqrt{\sum_{i=1}^{n} w_i(x_i - y_i)^2}$$

where x, y, and w are vectors where x contains the song features according to the user's preferences (such as danceability, energy etc.), y contains the features of the songs which is to be recommended, and w contains the weights in the value approximation function for each feature.

iv. Multi-Armed Bandits

In Multi-Armed Bandits, the agent needs to choose between exploring and exploiting its choices so as to maximize the future rewards. The name originates with analogy to the slot machine, or "one-armed bandit", except that it has k levers instead of one. Each action selection is like a play of one of the slot machine's levers, and the rewards are the payoffs for hitting the jackpot.

When we maintain the estimated rewards obtained after taking each action, then there will be at least one action whose estimated reward is the highest. These actions are called greedy actions and choosing these greedy actions is knows as exploitation. Exploitation allows maximization of reward in one step. But there is a possibility that some other action might lead to an even greater reward. Thus, choosing these actions is necessary to maximize long term rewards and is known as exploration.

In the Multi-Armed Bandit problem, a proper balance needs to be struck as to when the model should explore and when it should exploit so as to maximize the future reward.

v. Contextual Multi-Armed Bandits

In contextual multi-armed bandit problem, before making a choice, the agent is shown a n-dimensional feature vector, based on which, the agent chooses the most appropriate action to maximize the reward.

In our model, we use two feature vectors, one which contains the user's desired values for features such as *danceability*, *acousticness*, *valence*, etc. and the other which contains the songs' values of the features.

IV. Dataset

The dataset was acquired from Spotify's API using *spotipy* library and contained metadata and features for 10,000 tracks was downloaded. The features included the confidence scores of a track's acousticness, danceability, valence, etc. Here's the documentation for the dataset parameters.

V. Proposed Model

The problem was modeled as a contextual multi-armed bandit problem with the user preferences and track features as the context.

1. **Value function approximation** – Value of any state can be approximated by (-1) weighted Euclidean distance and the feature values makes for the approximate policy function. The ideal values of these features can be learnt using a deep neural network.

$$V(song) = -D_{weighted}(w, song, user)$$

2. **Exploitation-exploration** – An epsilon-greedy approach was used to deal with this dilemma. The song with least distance is recommended (call argmax() on values) with a probability of 1-ϵ, and a random song is recommended with ϵ probability at each instance. ϵ ranges from 0.3 to 0.1 exponential – decay with time.

$$\in = 0.1 + e^{c_1 x + c_2}$$

Where, $c_1 = -0.00287, c_2 = -1.609$ and $x = current\ time - last\_time$

3. **Reward function –** User provides a rating for each recommendation, values ranging from -10 for a bad recommendation, to +10 for a good one. This rating is then used to update the user preferences as the learning rate for doing gradient descent and the weighted Euclidean distance as the loss.

$$\alpha = 0.1 * rating$$

$$correction_f = \alpha * weights_f * (song_f - user_f)$$

$$\forall f \in features$$

4. **Distance discounting** – A set of liked Albums and Artists is maintained (songs with rating > 7). If a song belongs to a liked album, the distance is discounted by a factor of 40% and by 20% for a liked artist.

$$if\ (song_{album}\ in\ liked_{abums}):$$

$$song_{distance} \leftarrow 0.6 * song_{distance}$$

$$else\ if\ (song_{artist}\ in\ liked_{artists}):$$

$$song_{distance} \leftarrow 0.8 * song_{distance}$$

5. **Distance penalization –** Also, in order to make sure that the same song is not recommended for a while, it is penalized with an exponential function that decays over time. This penalized distance is used for recommending the songs now.

$$Penalty\ factor = 1 + e^{c_1 x + c_2}$$

Where, $c_1 = -0.289$ and $c_2 = 4.181$

VI. Experimentation

Much testing was required to get the ideal constants for exponential decay function for penalizing the distances. The final values chosen resulted in the penalization scaling factor to be ×*50* for the song just recommended, and decays to ×*1.2* after 20 recommendations.

Similarly, the constants for exponential decay of the ϵ function for exploitation and exploration were chosen to result in reduction of ϵ value from 0.3 to 0.25 after 100 recommendations.

Discounting was necessary to encourage recommendations of music from a liked album or artist too.

VII. Results and Discussion

The problem of cold start is being dealt with by initializing the user's preferences by getting their information before any recommendation is made. This information is acquired for only the features that make sense intuitively ("Do you like acoustic songs" / "Do you prefer instrumental music or vocals more", etc.), and is then scaled to initialize the user's preferences. This can be changed to a much easier experience once details of enough users are available, by say adding a FB login to get their information.

VIII. Conclusion and Future Work

The quality of recommendations heavily depends on the context of the multi-armed bandit problem i.e. the user's information and features of the track. The dataset currently uses audio processing to get these features, and is missing the metadata on the track's genre, year of release, popularity, monthly listeners, etc. These parameters are very important features of the track, and if there's a way to automate the extraction of this information for a large playlist, we can add them to later versions of this engine and be able to give much more robust suggestions.

Similarly, the context provided by the user can be also enhanced by getting information on their age, geographic location, college, etc. or by giving them a survey. Once information of enough users has been acquired, it can be used to learn more features using ANNs, the target being the final preferences obtained till now.

The value function isn't being updated (learnt) in the current version of the code and the weights in policy approximation function is just an array of ones, although a dummy function has been added to update the policy. We can use deep learning to learn these weights and further improve the quality of recommendation of the code.

The distance discounting function can also be augmented to get different discounting factors based on the no. of songs added and liked from that album and by that artist. Finally, the dataset could benefit from some cleaning. For e.g., some songs have multiple artists (comma separated) but are treated as a completely new artist by the current code.

IX. References

[1] Schwartz, Barry & Barry,. (2005). The Paradox of Choice: Why More Is Less.

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

[3] Y. Koren, R. M. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” IEEE Computer, vol. 42, no. 8, pp. 30–37, 2009.

[4] G. Linden, B. Smith, and J. York, “Amazon.com recommendations: Item-to-item collaborative filtering,” IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, 2003.

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

[6] R. J. Mooney and L. Roy, “Content-based book recommending using learning for text categorization,” in ACM DL, 2000, pp. 195–204.

[7] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos, and J. Kubica, “Ad click prediction: a view from the trenches,” in KDD 2013, Chicago, IL, USA, August 11-14, 2013, 2013, pp. 1222–1230.

[8] W. Zhang, T. Du, and J. Wang, “Deep learning over multi-field categorical data - - A case study on user response prediction,” in ECIR 2016, Padua, Italy, March 20-23, 2016. Proceedings, 2016, pp. 45–57.

[9] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang, “Productbased neural networks for user response prediction,” in ICDM 2016, December 12-15, 2016, Barcelona, Spain, 2016, pp. 1149–1154.

[10] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, “Deepfm: A factorizationmachine based neural network for CTR prediction,” in IJCAI 2017, Melbourne, Australia, August 19-25, 2017, 2017, pp. 1725–1731.

[11] H. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, “Wide & deep learning for recommender systems,” CoRR, vol. abs/1606.07792, 2016.

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

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

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

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

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