New Music Artist Recommendation System for Users with Heterogenous Tastes

Paulo Filipe Dourado
University of California Los Angelos
fdourado@g.ucla.edu

Introduction

In today's fast-paced world, we have been conditioned to search for the latest and most relevant bits of information before anyone else can find that information. Music fans, for example, often aim to discover up-and-coming artists as soon as they release their music and before the artist becomes mainstream; however, popular streaming services lack the capabilities to effectively recommend these new artists to listeners both as soon as their music is released and according to the listeners personal music preferences. This paper discusses an approach to the "new artist" problem by implementing a well known paper's Independent Bandit Algorithm (IBA) and Ranked Bandit Algorithm (RBA). We then make a small modification to the IBA which results in a dramatic decrease in the convergence time to the theoretical limit.

The paper I have selected to base my recommender system is called *"A Fast Bandit Algorithm for Recommendations to Users with Heterogeneous Tastes"* by Pushmeet Kohli, Mahyar Salek and Greg Stoddard and can be found **here**. The paper studies scenarios where there is no prior information about the quality of the content in the system. In my recommender system, we had no prior information about the new artists arising each day. That being so, the algorithm formulated by Kohli and colleagues proved to be highly effective in its application.

System Description

Each day when our recommender updates, we submit to it a list of N new music artists found with limited to no popularity. We submit this list without regard for genre, current popularity, or number of songs.

When a user arrives at our recommender system, they request that we present a few artists for them to listen to. The system then returns k artists where k is a number we have experimentally shown to provide good MAB performance, while not overwhelming the user with too many choices. The user is then allowed to click on whichever artist they choose. Once the user has finished listening to an artist, they are quickly asked how much they enjoyed the artist's music. An answer in the affirmative will return a feedback value of 1 to the system; a response in the negative will return a feedback value of 0. Figure 1 shows an implementation of the described recommender system.

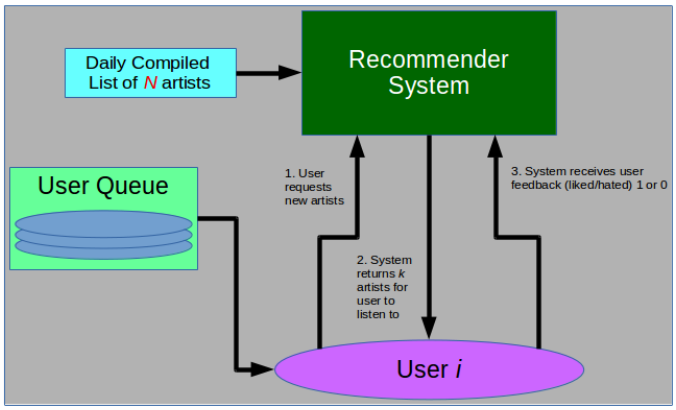

Figure 1: Recommender System Block Diagram

Algorithm Description

The paper I used as a basis for my recommender system refers to two MAB implementation algorithms: Ranked Bandit Algorithm (RBA) and Independent Bandit Algorithm (IBA). The IBA, developed by Kohli, Salek, and Stoddard (2013), is shown to outperform the RBA. Kohli, Salek, and Stoddard state that their IBA achieves a provable regret of $O(knlog(T)$ where k is the number of MAB instances (i.e., number of recommended artists to a user), n is the number of artists submitted to the system, and T is the time interval. The authors point out that their algorithm outperforms that of the RBA as the regret for the RBA is $O(k\sqrt{Tnlog(n)})$.

Both algorithms have the same fundamental structure. Each user j can be represented by a $\{0,1\}^n$ - vector X^j, where $X^j_i = 1$ indicates that user j finds artist i relevant. At each time period t a random user

is selected with a given relevance vector X^t. Let S^t denote a set of k artists we are recommending to the user. Finally, we denote the payoff for showing set S^t to a user with relevance vector X^t as the following:

$$F(S^t, X^t) = \begin{cases} 1 & \text{if } X_i^t = 1 \text{ for some } i \in S^t, \\ 0 & \text{otherwise.} \end{cases}$$

The payoffs are then fed into k MABs whose arms are all of the artists available for recommendation.

The RBA is show in Algorithm 1. As the algorithm shows, we only attain a reward for artist i if it was the first click. In other words, we only get the reward if the user listended and had a positive response to that artist first; subsequent artists in the set get no reward. We also pay no attention to artist repetitions in the recommended set, which functionally we would never replicate in a real world setting as the user would likely not appreciate the duplication from an aesthetic point of view.

Algorithm 1 Ranked Bandit Algorithm (RBA)

MAB_i: Bandit for slot i
for $t = 1...T$ **do**
  $s_i \leftarrow$selectArtist(MAB_i, N)
  $S^t \leftarrow \cup_i s_i$
  Display S^t to user, receive feedback vector X^t
  Feedback:

$$z_i = \begin{cases} 1 & \text{if artist } s_i \text{ was the first click,} \\ 0 & \text{otherwise.} \end{cases}$$

  update(MAB_i, z_i)
end for

The IBA is shown in Algorithm 2. There are two major differences between the IBA and RBA, and they are 1) that the IBA does not contain artist duplicates in the set that is shown to the user, and 2) all MABs get rewards if the user likes a selected artist in the set. We do not only give a reward to the first artist as as is the protocol with the RBA.

Algorithm 2 Independent Bandit Algorithm (IBA)

MAB_i: Bandit for slot i
for $t = 1...T$ **do**
  $S_i^t = \emptyset$
  for $i = 1...k$ **do**
    $S_i^t \leftarrow$selectArtist$(MAB_i, N \setminus S_{t-1}^t)$
  end for
  Display S^t to user, receive feedback vector X^t
  Feedback:

$$z_i = \begin{cases} 1 & \text{if artist } s_i \text{ was clicked on,} \\ 0 & \text{otherwise.} \end{cases}$$

  update(MAB_i, z_i)
end for

The last algorithm I would like to discuss is my own. Part of the assignment requested that we make a small modification to the algorithms we implemented. Mine is shown in Algorithm 3. The algorithmic modification I performed was to the ϵ-Greedy MAB underlying the IBA. Through my simulation, I was very pleased with the IBA ϵ-Greedy performance and wanted to expand on its success. I began to wonder if IBA ϵ-Greedy could somehow benefit from an extended exploration phase at the beginning of the simulation and then tappering to a stable ϵ. I tested my theory by linearly decreasing the value of ϵ over L time steps starting at an ϵ of 1 and ultimately ending on a final small value of ϵ. I discuss the results of my modification in the simulation section of the current paper, but suffice it to say, my modification significantly learned much faster than the other MAB algorithms and converged on the theoretical limit of the data set much faster. Figure 2

Algorithm 3 Independent Bandit Algorithm with Modified ϵ-Greedy

MAB_i : ϵ-Greedy Bandit for slot i
$\epsilon \leftarrow$Desired final ϵ between 0 and 1
$L \leftarrow$Number of time steps until we reach ϵ
for $t = 1...T$ **do**
  $S_i^t = \emptyset$
  $y = -\frac{(1-\epsilon)t}{L} + 1$ ▷ Linearly decreasing ϵ
  for $i = 1...k$ **do**
    $S_i^t \leftarrow$selectArtist$(MAB_i, y, N \setminus S_{t-1}^t)$
  end for
  Display S^t to user, receive feedback vector X^t
  Feedback:

$$z_i = \begin{cases} 1 & \text{if artist } s_i \text{ was clicked on,} \\ 0 & \text{otherwise.} \end{cases}$$

  update(MAB_i, z_i)
end for

Simulation Results

The data set used to validate my recommender system is from last.fm. This dataset contains ¡user, artist, plays¿ tuples collected from Last.fm API found **here**, using the user.getTopArtists() method. This dataset does not adhere to the recommender system I have been describing thus far. In my system, the artists are unknown to the user until the moment they listen to them, whereas this dataset does the opposite: it gives us the most played artists from 360k users. If we were able to erase a users musical memory and make the assumption that a user would have listened to and liked their favorite artist if we had introduced that artist to them first, then our dataset actually works well. The real reason is the assumption that a certain user has a unique inherent taste and so by knowing ahead of time what they will like, we can compute rewards for recommending a user that fits within their "taste".

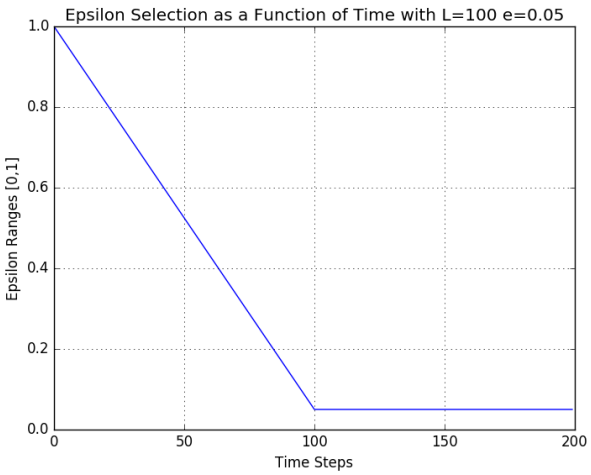

Figure 2: Tappered ϵ Function Applied to Modified IBA ϵ-Greedy

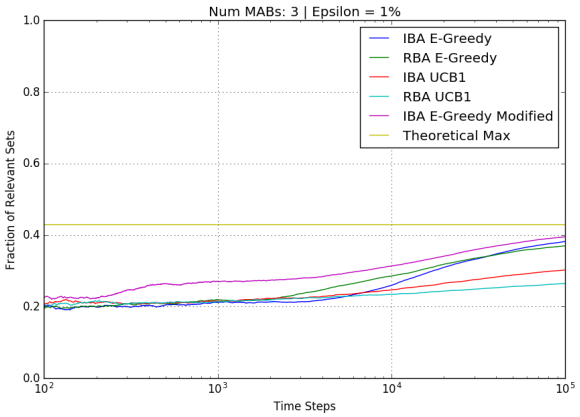

Figure 3: Simulation Results of $k = 3$ Artist Recommendations, $\epsilon = 0.01$ and $L = 100$

For my simulations, I decided to try and replicate the results of Kohli, Salek, and Stoddard (2013) before moving on to my own algorithmic implementation. The performance of the algorithms was measured by the percent of sets that contained a least one relevant article to the user, denoted in the Figures as "Fraction of Relevant Sets". I also calculated and displayed a Theoretical Limit to the simulation based on the dataset. I took the k most popular artists in the dataset and found which percent of users had ever listened to them before (equivelent to liking them in our interpretation of the dataset). This is the natural theoretical limit of the dataset as the IBA and RBA will naturally find the set of artists that attains the highest percentage of user satisfaction. Each IBA or RBA simulation was run 50 times and the results were averaged.

IBA vs RBA performance shows in Figures 3 and 4 does not appear to be significantly different for the $k = 3$ case. Arguably, the reason for this is the algorithms look nearly identical as k approaches 1 due to the first-click reward approach in the RBA. I also noted that in all cases, a very low value of $\epsilon = 0.01$ yields very nearly identical performance as well. Naturally, this signals that the IBA only performs better than the RBA for relatively larger values of k and ϵ. This is not necessarily inconsistent with the results shown in Kohli, Salek, and Stoddard's paper, which uses a fixed $k = 5$ for many different article recommendation datasets. Nonetheless, the asypmtotic performance of my simulations for the $k = 5$ case is consistent with their findings, and we can assume that if they had varied the k and ϵ parameters as I had then they would have seen the same behavior.

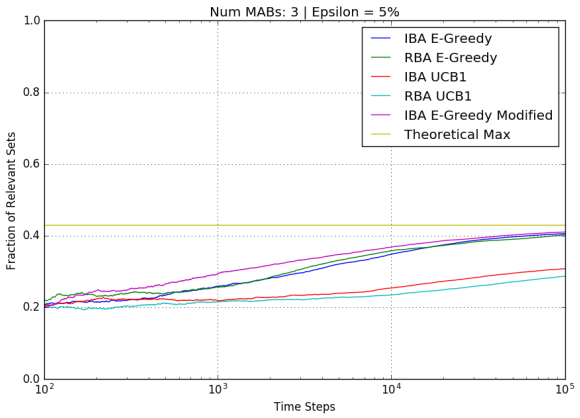

Figure 4: Simulation Results of $k = 3$ Artist Recommendations, $\epsilon = 0.05$ and $L = 100$

IBA vs RBA performance shown in Figures 5 and 6 shows us that the IBA tends to approach the limit earlier in time than the RBA implementation regardless of the underlying MAB. This is consistent with the results in Kohli, Salek, and Stoddard (2013). Furthermore, I also noted that UCB1 does not perform well for this type of recommender system as it learns much slower than an ϵ-Greedy. Kohli, Salek, and Stoddard (2013) state that this is not surprising as UCB is hindered by limited exploration in the beginning. The low exploration rate of the greedy solution allows for faster convergence in earlier slots.

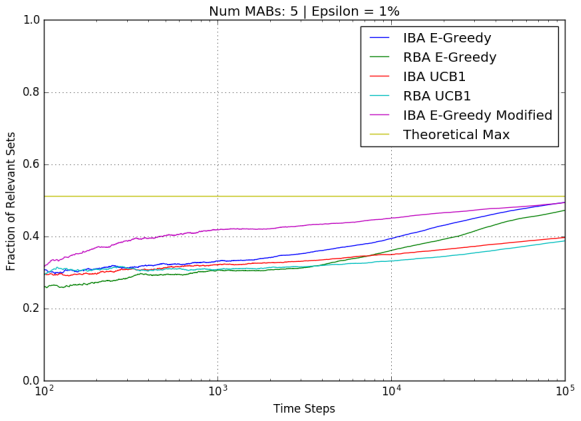

Figure 5: Simulation Results of $k = 5$ Artist Recommendations, $\epsilon = 0.01$ and $L = 100$

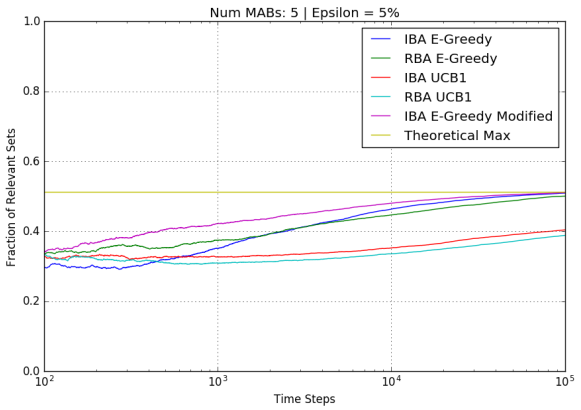

Figure 6: Simulation Results of $k = 5$ Artist Recommendations, $\epsilon = 0.05$ and $L = 100$

The last important result is that of my own algorithm: the modified greedy IBA. My algorihm was the best solution in all cases. It converges to the max 3 order of time magnitude earlier than any other algorithm. The approach I took to implement this algorithm was based on the idea that early exploration for this type of recommender system is necessary and that a fixed exploration rate would not allow for liberal exploration in the initial condition. Therefore, I created the monotonically decreasing ϵ algorithm illustrated in the current paper.

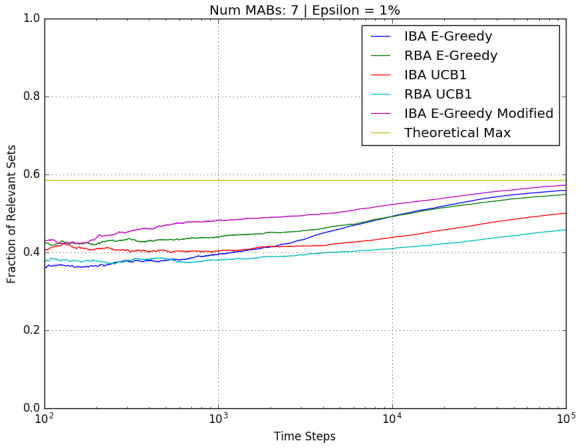

Figure 7: Simulation Results of $k = 7$ Artist Recommendations, $\epsilon = 0.01$ and $L = 100$

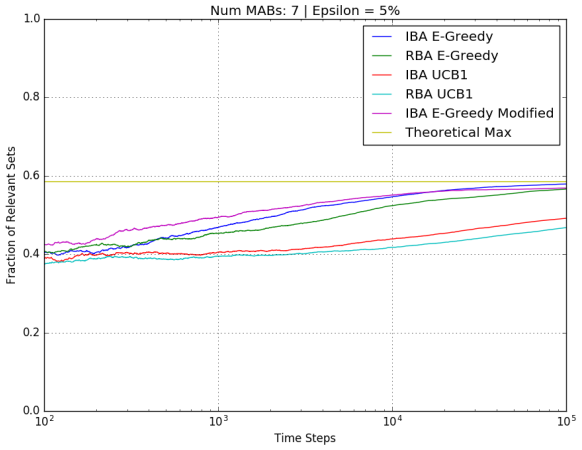

Figure 8: Simulation Results of $k = 7$ Artist Recommendations, $\epsilon = 0.05$ and $L = 100$

Conclusion

In conclusion, I have presented some simple online algorithms to solve the artist recommedation problem discussed at the beginning of this paper. I was able to successfully replicate the results of Kohli, Salek, and Stoddard (2013) for a different dataset and also found new behavior by varying algorithm parameters which had not yet been explored. Of all the algorithms implemented and tediously simulated, my own "Monotonically Decreasing ϵ" (MDE) algorithm outperformed all the rest due to the IBA architecture developed by Kohli and colleagues. In the future, I would like to explore different ϵ tapering schemes as they showed much promise in this area of research.