Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Adapting to User Preference Changes in Interactive Recommendation [*]

Negar Hariri
DePaul University
Chicago, IL
nhariri@cs.depaul.edu

Bamshad Mobasher
DePaul University
Chicago, IL
mobasher@cs.depaul.edu

Robin Burke
DePaul University
Chicago, IL
rburke@cs.depaul.edu

1 Abstract

Recommender systems have become essential tools in many application areas as they help alleviate information overload by tailoring their recommendations to users' personal preferences. Users' interests in items, however, may change over time depending on their current situation. Without considering the current circumstances of a user, recommendations may match the general preferences of the user, but they may have small utility for the user in his/her current situation. We focus on designing systems that interact with the user over a number of iterations and at each step receive feedback from the user in the form of a reward or utility value for the recommended items. The goal of the system is to maximize the sum of obtained utilities over each interaction session. We use a multi-armed bandit strategy to model this online learning problem and we propose techniques for detecting changes in user preferences. The recommendations are then generated based on the most recent preferences of a user. Our evaluation results indicate that our method can improve the existing bandit algorithms by considering the sudden variations in the user's feedback behavior.

2 Introduction

Users' interests in items may change depending on their current situation. For example, users may have different preferences for music depending on their current mood or activity. Similarly, users' preferences for travel packages can depend on the current season or the weather conditions. In these and many other similar applications, recommendations are not useful for the user if they are generated based on previous selections and preferences of the user without considering his/her current situation.

In this paper, we focus on a common setting where change of interests of a user is not explicitly available and the system should infer this information based on user feedback data. In this case, the system tracks the user's set of interactions over time and detects any significant changes in the user's behavior. User interaction data can include various types of feedback given by the user to the recommended items such as ratings or clicking on the recommendations. Detecting the preference changes enables the recommender to distinguish between the new and old preferences of the user and to generate the recommendations that match the user's current interests.

We investigate this problem in the framework of an interactive recommender system that can dynamically adapt to changes in users' interests. At each step of the interaction, the system presents a list of recommendations and receives feedback indicating recommendation utilities. The main objective of the proposed recommendation approach is to identify the recommendation list at each step such that the average obtained utility is maximized over the interaction session with a user. Choosing the best recommendation list at each step requires a strategy in which the best trade-off between *exploitation* and *exploration* is achieved. During an exploitation phase, the recommender system generates the recommendation list that maximizes the immediate utility for the current step. During exploration, the system may recommend items that do not optimize the immediate utility, but reduce the uncertainty in modeling a user's preferences and eventually help maximize the utility over the whole session. This problem can be formulated as a *multi-armed bandit problem (MAB)* which has been studied in various systems and applications [Chapelle and Li, 2011][Li *et al.*, 2010]. For the application of personalized recommendation, most of the existing bandit algorithms assume that a user's interests in items are static over time despite the fact that change of preferences can greatly impact the utility or usefulness of the recommendations for a user. We address this problem by proposing a change detection algorithm that predicts sudden variations in a user's preferences based on his/her feedback on the recommendations. These changes are then considered by our bandit recommendation method when generating the recommendations for a user.

Our evaluation results show that by considering users' change of interests in a bandit strategy, the recommender can provide more interesting and useful recommendations to users.

3 Related Work

Bandit algorithms have been widely used for the application of producing personalized recommendations [Chapelle and Li, 2011], [Li *et al.*, 2010]. The news recommender introduced in [Li *et al.*, 2010] adapts to the temporal changes

[*]This paper was invited for submission to the Best Papers From Sister Conferences Track, based on a paper that appeared in Eighth ACM Conference on Recommender Systems, RecSys '14.

of existing content. In the proposed system, both users and items are represented as sets of features. Features of a user can include demographic attributes as well as features extracted from the user's historical activities such as clicking or viewing items while item features may contain descriptive information and categories. They formulate this problem as a bandit problem with linear payoff and introduced an upper confidence bound bandit strategy which is shared among all the users. Similarly, the system in [Chapelle and Li, 2011] uses a bandit algorithm based on Thompson sampling for the task of news recommendation and display advertisement. Similar to [Li *et al.*, 2010], several features are used to represent the user and the items and the recommender learns a single bandit strategy for all users while representing each user as a fixed set of features. Our approach is different from [Li *et al.*, 2010] and [Chapelle and Li, 2011] as we are learning a bandit strategy for each user. While these two methods focus on adapting to the changes in items popularities, our system captures and adapts to the changes in users' interests. Representing each user as a fixed set of features, as in [Li *et al.*, 2010] and [Chapelle and Li, 2011] would prevent us from detecting changes in the users' preferences and capturing his/her current interests. Similarly, other bandit strategies such as [Deshpande and Montanari, 2013] and [Abbasi-Yadkori *et al.*, 2011] disregard the variations in user preferences and give all the interactions the same weight in defining the bandit strategy. Unlike these methods, our approach enables us to monitor changes in the users' behaviors and to produce recommendations that match the current interests of the user.

Tailoring the recommendations to match the current preferences of users follows the general idea of context-awareness [Adomavicius *et al.*, 2011] in recommender systems. Integrating context in the recommendation model, when this information is explicitly given to the system, has been investigated in various research work [Panniello *et al.*, 2009], [Baltrunas *et al.*, 2011], [Adomavicius *et al.*, 2005]. However, there has been less focus on interactional systems where context should be inferred from users' interactions. This is partly due to the fact that it is more difficult to capture the contextual information when it is not known to the system at design time.

4 Interactive Recommendation

Interactive recommenders have been extensively used in a variety of application domains. These systems usually rely on an online learning algorithm that gradually learns users' preferences. At each step of the interaction, the system generates a list of recommendations and observes the user's feedback on the recommended items indicating the utility of the recommendations. The goal of such a system is to maximize the total utility obtained over the whole interaction session.

4.1 Multi-Arm Bandit Approach to Online Recommendation

Choosing the best strategy to select the recommendation list at each step can be formulated as a *multi-armed bandit problem (MAB)*. Consider a gambler who can choose from a row of slot machines known as one-armed bandits. The gambler receives a reward from playing each of these arms. To maximize his or her benefit, the gambler should decide on which slot machines to play, the order of playing them and the number of times that each slot machine is played. A similar type of decision problem occurs in many real-world applications including interactive recommender systems. For an interactive recommender, the set of arms corresponds to a collection of items that can be recommended at each step, and the rewards correspond to the user's feedback, such as ratings or a clickthrough on a recommended item.

An important consideration in a bandit algorithm is the trade-off between *exploration* and *exploitation*. Exploitation is referred to the phase where the available information gathered during the previous steps is used to choose the most profitable item. During exploration, the most profitable alternative may not necessarily be picked but the algorithm may choose other items in order to acquire more information about the preferences of the user which would help to gain more rewards in future interactions.

Several techniques have been proposed for solving the multi-armed bandit problem. Optimal solutions for multi-armed bandit problems are generally difficult to obtain computationally. However, various heuristics and techniques have been proposed to compute sub-optimal solutions effectively. The ϵ*-greedy* approach is one of the most famous and widely used techniques for solving the bandit problems. At each round t, the algorithm selects the arm with the highest expected reward with probability $1 - \epsilon$, and selects a random arm with probability ϵ. The *Upper Confidence Bounds (UCB)* class of algorithms, on the other hand, attempt to guarantee an upper bound on the total regret (difference of the total reward from that of an optimal strategy).

Thompson Sampling is another heuristic that plays each arm in proportion to its probability of being optimal. As we are using this heuristic in our system, we briefly describe it in more details. In the next section, we will discuss how this heuristic is used in our system to generate the recommendation list at each interaction step.

The reward (or utility) distribution for each item can depend on various factors including characteristics of the item, general preferences of the user, and the current interests of the user. Let $p(r|a, \theta)$ represent the utility distribution for item a, and θ indicate the unknown parameter characterizing the distribution. Also, let $E_r(r|a, \theta)$ represent the expected reward for the item a for the given θ.

At the first step of an interaction, the set of observations, denoted by D, is empty and $p(\theta|D)$ is initialized with a prior distribution for θ. As more rewards are observed at each step, Bayesian updating is performed to update the θ distribution based on the observed rewards.

Algorithm 1 describes the Thompson sampling procedure [Chapelle and Li, 2011], [Scott, 2010]. At each step, θ is drawn as a sample from $p(\theta|D)$. The expected reward for each of the items is then computed and the arm having maximum expected reward is played and the reward for that arm is observed. The selected arm and the observed reward is then added to the observations set.

Algorithm 1 Thompson Sampling

```
D=$\emptyset$
for $t = 1$ to $T$ do
    Draw $\theta^t \sim P(\theta|D)$
    Select $a_t = argmaxE_r(r|a, \theta^t)$
    Observe reward $r_t$
    D = D $\cup(a_t, r_t)$
end for
```

4.2 Generating the Recommendations

We adapt Thompson sampling heuristic as the bandit strategy for generating a recommendation list at each step of interaction with a user. In this setting, θ which characterizes the utility distribution for each item, represents a user's preference model. It is a k-dimensional random vector drawn from an unknown multivariate distribution. The user model is updated after each interaction.

Linearly parameterized bandits[Rusmevichientong and Tsitsiklis, 2010] are a special type of bandits where the expected reward of each item is a linear function of θ. It has been shown that Thompson sampling with linear rewards can achieve theoretical regret bounds that are close to the best lower bounds [Agrawal and Goyal, 2013]. In our approach we also assume that the rewards follow a linear Gaussian distribution.

Given d observations, let $r \in R^d$ be the observed rewards for the recommended items and let F represent a $d \times k$ constant matrix containing the *features* of the recommended items. For example, given two observations (a_1, r_1) and (a_2, r_2), $r = [r_1, r_2]$ has two elements r_1 and r_2 and $F = [f_{a_1}, f_{a_2}]$ where f_{a_i} represents the k-dimensional feature vector for item a_i. The item features can be obtained in various ways. They can be extracted based on the available content data for the items or can be extracted from the users' feedback using different techniques such as matrix factorization. Given a training data set represented as a matrix of users' preferences on items, we apply Principal Component Analysis to represent each item in a k-dimensional space (where $k \ll$ the number of users in the training data).

Assuming a linear Gaussian distribution, we have the following prior and likelihood distributions:

$$p(\theta) = \mathcal{N}(\theta; \mu_\theta, \Sigma_\theta) \quad (1)$$

$$p(r|\theta) = \mathcal{N}(r; F\theta, \Sigma_r) \quad (2)$$

Also, we assume Σ_θ and Σ_r are diagonal matrices with diagonal values of σ_θ, and σ_r respectively.

Given a linear Gaussian system, the posterior $p(\theta|r)$ can be computed as follows (see [Murphy, 2012] for a proof):

$$p(\theta|r) = \mathcal{N}(\mu_{\theta|r}, \Sigma_{\theta|r}) \quad (3)$$

$$\Sigma_{\theta|r}^{-1} = \Sigma_\theta^{-1} + F^T \Sigma_r^{-1} F \quad (4)$$

$$\mu_{\theta|r} = \Sigma_{\theta|r}[F^T \Sigma_r^{-1}(r) + \Sigma_\theta^{-1}\mu_\theta] \quad (5)$$

At each step t, θ^t is drawn as a sample from the posterior distribution. The expected reward for each item x is then estimated as $r_x = f_x \cdot \theta$, where f_x represents the extracted features for the item x. The items are then ranked based on the estimated expected reward values. The item with the highest reward is recommended to the user.

4.3 Adaptation to Changes in Users' interests

This section describes our approach for adapting the recommendations to changes in users' interests. Our method has two main steps. The first step, is to detect preference changes and the second step is to adapt the recommendations based on the detected changes.

Our system includes a change detection module that tracks users' interactions and detects any sudden and significant changes in the users' behaviors. At each step t of interaction with a given user u, the change detection module models the user's interactions in interval $I_t = (t - N, t]$ as well as the interactions in the interval $I_{(t-N)} = (t - 2N, t - N]$, where N is a fixed pre-specified parameter representing the length of the interval. These two windows are modeled by computing the following posterior distributions:

$$W_{I_t} = p(\theta|I_t) = \mathcal{N}(\mu_t, \Sigma_t) \quad (6)$$

$$W_{I_{(t-N)}} = p(\theta|I_{(t-N)}) = \mathcal{N}(\mu_{t-N}, \Sigma_{t-N}) \quad (7)$$

The above two distributions can each be computed according to equation 3 where r, and F are substituted with the rewards and the item features of the recommendations in the corresponding window.

The distance between W_{I_t} and $W_{I_{(t-N)}}$ is used as a measure of dissimilarity between these two windows. Various measures such as KL-divergence can be used to compute the distance between the two windows. In our experiments, we used *Mahalanobis distance* which is computed as follows:

$$\Sigma = \frac{\Sigma_t + \Sigma_{t-N}}{2} \quad (8)$$

$$distance = (\mu_t - \mu_{t-N})^T \Sigma^{-1} (\mu_t - \mu_{t-N})$$

If the user feedback behavior is significantly different between the two windows, then the models for W_{I_t} and $W_{I_{(t-N)}}$ would be different. Therefore, the change detection module in our system is based on the heuristic that sudden changes in the users' feedback behavior would result in sudden changes of the distance measure computed in 8. Based on this heuristic, we compute the distance measure at each step t, and exploit a change point analysis approach to detect sudden changes in a user's feedback behavior.

In our system, we utilize the technique described in [Wayne, 2000] for change point analysis. This procedure involves iteratively applying a combination of cumulative sum charts (CUSUM) and bootstrapping to detect the changes. For each of the detected changes, a confidence level is provided which represents the confidence of the predicted change point. Given a pre-specified threshold, only those change points that have confidence level above that threshold are accepted. Also, in our adaptation of this approach, we have incorporated a *look ahead (LA)* parameter to improve the accuracy. Our change detection module accepts a change point only if there are at least LA observations after the predicted change point.

In our system, we assume a user's interactions after the change point reflect his or her most recent preferences and the interactions before the change point reflect the user's preferences in a different context. The strategy to aggregate the old and new preferences depends on the specific application in which the recommender is being used. In some applications, a user's preferences in a different context may still contain some useful information about the *general* preferences of that user. In this case, a good strategy for the recommender would be to consider all the user's preferences while giving more weight to the feedback after the change point. In other applications, a user's preferences in different contexts are almost independent. For example, the user may follow different independent tasks at different contextual states. In this situation, a good strategy would be to provide recommendations just based on the current preferences of the user, or in other words, discarding the interactions before the change of interests. The recommender presented in this paper, discards users' interactions before the last detected change point and produces the recommendations only based on the interactions in the current context.

Depending on the type of utility (binary, rating), the change point detection may falsely detect change points at earlier interactions where it starts to learn the users' preferences. To avoid this situation, we have incorporated another parameter in our model to avoid splitting the user profile if there are fewer interactions in the user's profile than the pre-specified *splitting threshold*.

In our future work, we plan to investigate adapting a bandit strategy that considers all of user's preferences. This would include, defining a utility function that while giving more weight to the current preferences of a user, also considers the interactions before the change point.

5 Evaluations

Evaluation of change adaptation in recommenders can be a challenging task. There are few publicly available datasets that explicitly contain users' change of preferences that could be used as the ground truth for evaluation. In our evaluations, we designed an experiment for simulating users' change of behaviors. Having a dataset containing users' ratings for items, to simulate sudden changes in a user's rating behavior, we built test hybrid profiles by merging two random user profiles. Treating the two selected user profiles as the rating behavior of a single hybrid user, we evaluated our method in correctly predicting the change of preferences of the user and producing recommendations matching the users' current interests.

We used the Yahoo! Music ratings of musical artists version 1.0 dataset in our experiments. This dataset represents a snapshot of the Yahoo! Music community's preferences for various musical artists. It contains over ten million ratings of musical artists given by Yahoo! Music users over the course of a one month period sometime prior to March 2004. It contains ratings in the range of 0 to 100 given by 1948882 users to 98213 artists.

The purpose of this experiment was two-fold: first, we evaluated the accuracy of our change detection approach in real-time detection of changes. Secondly, we examined the performance of our recommender compared with conventional recommendation methods that disregard user preference changes. The evaluation was performed in the framework of an interactive recommender that interacts with the user in a number of consecutive rounds. In each round, the recommender presents a recommendation and observes the user's feedback (utility) for the recommended item. The recommenders were evaluated based on the average utility gained over the session of interaction.

We followed a five-fold cross validation approach for evaluation. In each round, one of the folds was used for testing, one was used for tuning the parameters and the remaining folds were used for training the model. To simulate the changes in a user's rating behavior, we randomly selected two users u_1 and u_2 from our test data and combined them to create a hybrid user u_h. We treated each of these two users as the rating behavior of a single hybrid user in two different situations.

To simulate an interactive recommendation process, at each step of interaction, the recommender used the ratings in the hybrid test user profile to recommend a new item that has not been presented to the user before. We randomly chose an item rating from the u_1 profile and used it as the initial profile of the hybrid user. For the first $T = 30$ rounds of interaction, the observed utility for each recommended item x, was assumed to be the rating assigned to x by the user u_1. We assumed that the interest change occurs after T steps of interactions. Therefore, for the next $T = 30$ steps, the same process was repeated with the difference that the observed utility for a recommended item was set to the rating for that item in the profile of u_2. For this evaluation, we filtered the test users so that each user had at least T ratings in the user's profile.

If the user has not rated a recommended item, one option would to ignore the recommendation and discard it from the evaluation results. However, this would create a bias in the experiment. If the user rates none of the recommendations, then the recommender should be negatively scored (in comparison to a recommender that recommends items that at least are rated by the user). Setting the utility to zero for the non-rated items would also be too strict, as the user may not necessarily dislike the items that she hasn't rated yet. Another possibility would be to set the utility to average rating of all the users for that item. This also creates a bias as item popularities are not uniform. For example, if an algorithm suggests very rare items with high average ratings, it will gain a high utility score in the evaluation. A similar bias occurs if the average rating of the user is used as the utility for non-rated items. In our evaluation, if a recommended item had not been rated in the user's profile, the observed utility was set to the average rating of all items in the dataset.

The set of parameters used in this experiment for each of the algorithms are specified in table 1.

User-based kNN was used as one of our baselines for this experiment. Cosine similarity metric was used to compute the similarity of users. In each round of interaction, the kNN

Table 1: Method parameters

Method	Parameters
Bandit recommender	Number of features = 5, μ_θ = 0, σ_θ = 0.5, σ_r = 0.5
Contextual Recommender (proposed method)	Number of features = 5, μ_θ = 0, σ_θ = 0.5, σ_r = 0.5, Change detection window size = 5, Change detection confidence = 0.95, Chage detection look ahead = 3, splitting threshold = 10
User-based kNN	Number of neighbors = 10

algorithm computed the item scores and recommended a new item that has the highest score.

The *optimal recommender* was chosen as another baseline to compare our method with an ideal algorithm. This recommender has full knowledge of users' preferences and the preference changes of users. For the first $T = 30$ iterations, it followed a greedy strategy and recommended new items that had the highest rating in the u_1 profile. After change of interest at $T = 30$, in each round it recommended a new item that had the highest rating in the u_2 profile. The standard bandit algorithm with Thompson sampling heuristic (as describe in section 4.2) was used as another baseline in our experiment and was trained with the parameters specified in table 1. In the remaining of the paper, we refer to this baseline as the standard bandit recommender.

Table 2 illustrates an example interaction session generated based on the standard bandit recommender for this experiment.

The recommendations that are not rated in the user's profile are marked as '-'. For each recommended artist, a few of most similar artists that have previously been presented to the user are also shown. The artist similarities are determined based on the Last.FM data. For any given artist, the Last.FM API can be used to find the most similar artists as well as the degree of similarity between the artists. In table 2, the similarity levels are abbreviated as M(medium), H(high), VH(very high), and SH(super high). As can be seen, most of the first

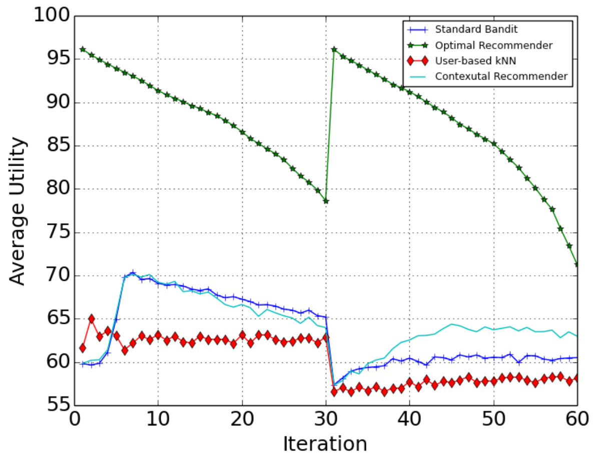

Figure 1: Experiment I-Average utility at each iteration

few (5) recommendations are not rated by u_1. After iteration 6, the recommender starts generating better recommendations as most of them have high ratings in the u_1 profile. After the simulated change of interests occurs at iteration 31, the user (u_2) has not rated most of the recommendations, showing the dramatic degradation of the recommender performance. Note that many of the recommendations are artists similar to those rated highly by u_1. For example, Foo fighters recommended at iteration 31, is highly similar to Nirvana recommended at iteration 21 or Hoobastank, recommended at iteration 32, has super high similarity to 3 Doors Down, which has received utility of 100 at iteration 27. This example shows that a traditional bandit approach cannot adapt to the changes in the users' preferences.

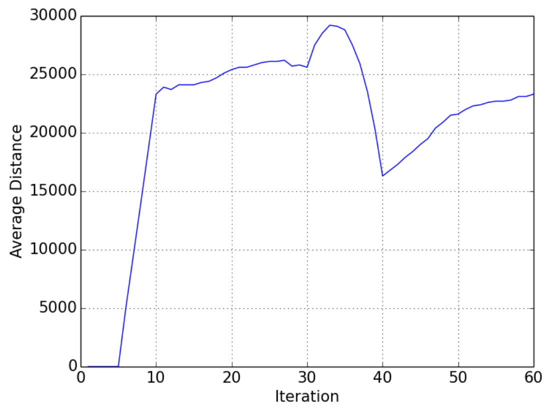

Figure 2: Expriment I-Average distance at each iteration

Figure 1 presents the average utility at each iteration computed over the set of test users. The optimal recommender achieves the highest utility of 96 at the first step and its utility decreases as more iterations pass. This can be explained by the fact that in the first T iterations items are recommended in the decreasing order of their ratings in the u_1 profile, therefore the average utility gradually decreases. After iteration 30, the optimal baseline recommends items based on their ratings in the u_2 profile and so a jump in utility can be observed at iteration 31. Both the standard bandit recommender and the contextual recommender (proposed method) need 5-6 iterations to learn users' preferences and show a dramatic improvement in the utility. As indicated in table 1, the number of factors is set to 5 for these two methods. We repeat this experiment for different number of factors. The experiments revealed that by lowering the number of factors, it takes less it-

Table 2: An example interaction session generated using the standard bandit recommender

Step	Artist Name	Utility	Similar artists
1	Chingy	-	
2	Disturbed	90	
3	Led Zeppelin	-	
4	Faith Hill	-	
5	Eminem	-	
6	Linkin Park	100	Disturbed(M), Eminem(M)
7	Staind	100	Disturbed(H)
8	Good Charlotte	-	
9	Metallica	90	Led Zeppelin(H), Disturbed(H)
10	Green Day	90	Good Charlotte(M), Linkin Park(L)
11	Red Hot Chili Peppers	90	Metallica(M)
12	The Offspring	-	Metallica(VH), Green Day(VH)
13	Korn	-	Disturbed(H), Metallica(H)
14	Evanescence	100	Disturbed(H), Linkin Park(H)
15	Blink 182	90	Good Charlotte(H), Green Day(H)
16	Limp Bizkit	50	Korn(SH), Disturbed(H)
17	Nickelback	100	Linkin Park(VH), Staind(VH)
18	Trapt	90	Staind(VH), Disturbed(M)
19	Simple Plan	-	Green Day(VH), Blink 182(H)
20	Papa Roach	90	Nickelback(VH), Linkin Park(VH)
21	Nirvana	90	Metallica(H)
22	Godsmack	100	Disturbed(SH), Staind(H)
23	P.O.D.	100	Papa Roach(VH), Staind(VH)
24	All-American Rejects	90	Good Charlotte(SH), Blink 182(VH)
25	Puddle Of Mudd	100	Staind(SH), Trapt(VH)
26	Sum 41	-	Blink 182(SH), Green Day(SH)
27	3 Doors Down	100	Nickelback(SH), Trapt(SH)
28	Incubus	100	Staind(H)
29	A.F.I.	90	Good Charlotte(L)
30	Jimmy Eat World	90	Blink 182(VH), A.F.I(H)
31	Foo Fighters	-	Nirvana(SH), Red Hot Chili Peppers(SH)
32	Hoobastank	-	Trapt(SH), 3 Doors Down(SH)
33	System Of A Down	-	Korn(H), Disturbed(H)
34	No Doubt	70	
35	U2	-	Red Hot Chili Peppers(L)
36	Alien Ant Farm	-	Incubus(SH),Puddle Of Mudd(SH)
37	Creed	-	Staind(VH), 3 Doors Down(H)
38	Smile Empty Soul	-	Trapt(SH), Staind(VH)
39	Slipknot	-	Korn(VH), System of a Down(VH)
40	Michelle Branch	-	-
41	The Ataris	-	Jimmy Eat World(VH)
42	The White Stripes	-	Nirvana(M)
43	Avril Lavigne	-	
44	Dashboard Confessional	-	Jimmy Eat World(SH)
45	Matchbox Twenty	-	-
46	New Found Glory	-	blink-182(SH)
47	Chevelle	-	Staind(VH), Trapt(H)
48	Brand New	-	Jimmy Eat World(H)
49	Nine Inch Nails	-	-
50	Switchfoot	-	
51	Goo Goo Dolls	-	3 Doors Down(H), Trapt(H)
52	Rob Zombie	-	Korn(M)
53	Audioslave	-	Foo Fighters(SH), Incubus(VH)
54	Sugarcult	-	Good Charlotte(VH)
55	Deftones	-	Korn(VH), Limp Bizkit(H)
56	The Used	-	Blink-182(H), Papa Roach(H)
57	Lifehouse	-	3 Doors Down(SH)
58	Static-X	-	Korn(VH), Disturbed(VH)
59	Fountains Of Wayne	-	
60	Taking Back Sunday	-	Jimmy Eat World(VH)

eration for the algorithm to reach its best performance. However, the highest achieved utility would be smaller as well.

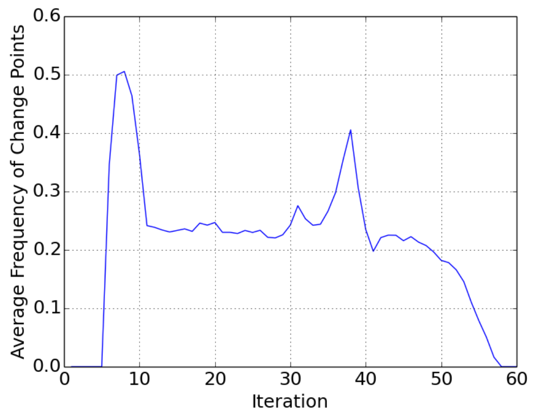

Figure 3: Experiment I-Average frequency of change points at each iteration

Figure 2 illustrates the average distance computed as in equation 8 in each round of interaction. As can be seen in the figure, at iterations 33 to 35, distance has the maximum value. As the window size for change detection is set to 5, it takes at least 3 rounds until W_{I_t} mostly contains the interactions after the change of interests and $W_{I_{t-N}}$ only has the interactions before the change of interests and therefore, the distance metric become maximized.

Figure 3 presents the average number of change points detected at each iteration. As evident in the figure, the average frequency of change points reaches the maximum value of 0.5 at iteration 6 where $W_{I_{t-N}}$ has one interaction and W_{I_t} window contains 5 (window size) interactions. Setting the splitting threshold to 10 ensures that the user profile is not split for the detected change points before round 10. Also, at iteration 38 there's another local maximum point where average frequency of change points reaches 0.4. This peak corresponds to the change of preferences in the hybrid user profile.

6 Conclusions

In this paper we have presented a novel approach for adaptation to changes in users' preferences in interactive recommender systems. In an online recommendation scenario, at each step the system provides the user with a list of recommendations and receives feedback from the user on the recommended items. Various multi-armed bandit algorithms can be used as recommendation strategies to maximize the average utility over each user's session. Our approach extends the existing bandit algorithms by considering the sudden variations in the user's feedback behavior as a result of preference changes. Our system contains a change detection component that determines any significant changes in the users' preferences. If a change is detected, the bandit algorithm based on Thompson sampling, prioritizes the observed feedback after the detected change point reflecting the current preferences of the user. The proposed recommender, discards users' interactions before the last detected change point and produces the recommendations just based on the interactions in the current context. In our future work, we plan to investigate various approaches to take into account the impact of users' interactions before the change point on the current user behavior.

References

[Abbasi-Yadkori *et al.*, 2011] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *NIPS*, pages 2312–2320, 2011.

[Adomavicius *et al.*, 2005] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, January 2005.

[Adomavicius *et al.*, 2011] Gediminas Adomavicius, Bamshad Mobasher, Francesco Ricci, and Alexander Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[Agrawal and Goyal, 2013] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. volume 28 of *JMLR Proceedings*, pages 127–135, 2013.

[Baltrunas *et al.*, 2011] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. Matrix factorization techniques for context aware recommendation. In *In ACM RecSys*, pages 301–304, 2011.

[Chapelle and Li, 2011] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *NIPS*, pages 2249–2257, 2011.

[Deshpande and Montanari, 2013] Yash Deshpande and Andrea Montanari. Linear bandits in high dimension and recommendation systems. *CoRR*, abs/1301.1722, 2013.

[Li *et al.*, 2010] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, pages 661–670. ACM, 2010.

[Murphy, 2012] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series)*. The MIT Press, 2012.

[Panniello *et al.*, 2009] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *RecSys*, pages 265–268. ACM, 2009.

[Rusmevichientong and Tsitsiklis, 2010] Paat Rusmevichientong and John N. Tsitsiklis. Linearly parameterized bandits. *Math. Oper. Res.*, 35(2):395–411, 2010.

[Scott, 2010] Steven L. Scott. A modern bayesian look at the multi-armed bandit. *Appl. Stochastic Models Bus. Ind.*, 26(6):639–658, November 2010.

[Wayne, 2000] Taylor Wayne. Change-point analysis: a powerful new tool for detecting changes. 2000.