Clustering of Bandit with Frequency-Dependent Information Sharing

Shen Yang[1,2], Qifeng Zhou[1,2](✉), and Qing Wang[3]

[1] Department of Automation, Xiamen University, Xiamen, China
yangshen@stu.xmu.edu.cn, zhouqf@xmu.edu.cn
[2] Xiamen Key Laboratory of Big Data Intelligent Analysis and Decision-Making, Xiamen, China
[3] Intelligent IT Operations, IBM T.J. Watson Research Center, New York, NY, USA
qing.wang1@ibm.com

Abstract. In today's business marketplace, the great demand for developing intelligent interactive recommendation systems is growing rapidly, which sequentially suggest users proper items by accurately predicting their preferences, while receiving up-to-date feedback to promote the overall performance. Multi-armed bandit, which has been widely applied to various online systems, is quite capable of delivering such efficient recommendation services. To further enhance online recommendations, many works have introduced clustering techniques to fully utilize users' information. These works consider symmetric relations between users, i.e., users in one cluster share equal weights. However, in practice, users usually have different interaction frequency (i.e., activeness) in one cluster, and their collaborative relations are unsymmetrical. This brings a challenge for bandit clustering since inactive users lack the capability of leveraging these interaction information to mitigate the cold-start problem, and further affect active ones belonging to one cluster. In this work, we explore user activeness and propose a frequency-dependent clustering of bandit model to deal with the aforementioned challenge. The model learns representation of each user's cluster by sharing collaborative information weighed based on user activeness, i.e., inactive users can utilize the collaborative information from active ones in the same cluster to optimize the cold start process. Extensive studies have been carefully conducted on both synthetic data and two real-world datasets indicating the efficiency and effectiveness of our proposed model.

Keywords: Interactive recommendation system · Weighed clustering · Multi-armed bandit model

1 Introduction

Personalized recommendation system (RecSys), aiming to provide users valuable information and services accurately from massive amounts of data, plays a significant role in information services of modern society. Specifically, the main focus of RecSys is to be capable of recommending items to satisfy users' requirements, so as to maximize some metrics (e.g., exposure, user satisfaction or product profit) [5]. For this reason, traditional recommendation algorithms [1] like collaborative filtering-based recommendation [10,14] and content-based recommendation [19], have emerged, which predict items that users are interested in based on historical data, and the predictions are at an individual level.

The offline training and online testing pattern of traditional recommendation algorithms results in their inappropriateness to provide high-quality online recommendations under the rapidly changing scenes of information services, e.g., cold start problems [15]. The situation of online recommendation is similar to the multi-armed bandit problem [2,18], that is, when new items or users are introduced into the system, RecSys needs to strike a balance between satisfying user interests and exploring new items to maximize long-term benefits. As a kind of reinforcement learning algorithm, bandit model can effectively solve the exploitation and exploration problem of cold start in online recommendations, and have become a popular recommendation algorithm, especially contextual bandit models [21]. LinUCB proposed in [12] is one classic contextual bandit model. With the development of social networks, researchers found that the implementation of clustering and collaborative filtering among users can make fuller use of feedback to obtain better recommendations [3,7,14,22]. The main idea of these works is to discover the underlying user clusters and share information within it for more accurate recommendations. However, current clustering-based bandit models suppose symmetric relations exist between the same-cluster users and share equal weights as well, resulting in the same collaborative effect among users [13]. To put it simply, inactive users who have low interaction frequency can pose negative impact on the collaborative effect of the same-cluster users, leading to the uncertainty of recommendation. Therefore, it is valuable for inactive users to learn more information from active ones, while active users filter out "harmful" information from inactive ones in the same cluster.

In this paper, we study the asymmetric relations between users in a cluster and propose a new $\underline{\text{freq}}$uency-dependent $\underline{\text{c}}$lustering $\underline{\text{b}}$andit model (FreqCB) to solve the aforementioned issues. We use the information of user activeness (i.e., interaction) for recommendation in the bandit setting, and the information sharing process is carried out through user activeness. In this way, inactive users can be hardly affected active ones from the same cluster by avoiding the collaborative effect, while inactive users can leverage active users' information to obtain a recommendation that fits their cluster's interests. Extensive studies have been conducted on both synthetic data and real-world datasets to indicate the effectiveness of FreqCB.

Our main contributions are as follows:

- We develop a method for capturing the influence of user activeness to optimize (e.g., speed up) the cold start process and get better recommendations.
- We propose a frequency-dependent clustering bandit model under the framework of upper-confidence bound. The model learns representation of each user's cluster by sharing collaborative information weighed based on user activeness.

- We demonstrate the effectiveness of our proposed model in extensive studies conducted on both synthetic data and real-world datasets.

2 Related Work

In this section, we highlight our studies and compare them with existing works related to our approach, including clustering-based bandit models and collaborative filtering-based bandit models.

Researchers have found that dividing users into different clusters and customizing the bandits for each cluster can enhance the bandit models' performance. The technique of online clustering of bandit is first studied in [7], which is inspired by [4] encoding social relationships in a graph for clustering bandit and other earlier works. [4] proposes GOB.Lin which runs a bandit model on a network and makes the nodes of the graph share information with its neighbor nodes by introducing the graph's Laplacian matrix. But GOB.Lin has its limitations: it is the solution only for networked bandit problems and the clusters can't be updated adaptively with the varying of user's parameter vectors. [7] follows up and designs CLUB, an adaptive clustering of bandit strategy on graph. CLUB deletes edges between users whose interests are no longer similar to each other and forms a new cluster on the new graph after edges deleting. SCLUB in [13] is a generalized version of CLUB. SCLUB proposes new splitting, merging, and updating methods on clustering to identify the underlying clusters faster and more accurate. Besides, [17] develops DYNUCB, utilizing k-means clustering method [16] to dynamically cluster all users based on the similarity of their parameter vectors. But the number of underlying clusters is unknown and needs to be specified as a hyper parameter.

Based on [7,9] proposes distributed confidence ball algorithms for solving linear bandit problems in peer to peer networks with limited communication capabilities. To address the challenge of uncertain estimation of user interests and user clustering, [22] proposes ClexB for the online recommendation, which is a contextual bandit policy that incorporates knowledge sharing via adaptive clustering and learning user interest via exploring clustering.

In these mentioned works, users are not grouped in a context-dependent approach. [14] proposes COFIBA which takes advantage of clustering and collaborative filtering from both user and item sides. [6] presents CAB based on the linear contextual bandit framework and implements the context-dependent feedback sharing mechanism over users in a flexible manner. LOCB [3] starts with a set of seeds (e.g., users), then updates recursively the seeds set and neighbors of each seed after pulling module.

These algorithms employ clustering or collaborative filtering on either user or item side to share information in/across clusters, which does not consider the asymmetric relationship between users. However, in a bandit-setting recommendation system with collaborative filtering, it's a universal case: active users provide more feedback to the recommendation system, whereas inactive/new users can hardly have affect on it. Hence, more noise would be brought in when using the equal weights of active and inactive users to represent a cluster. Our paper proposes FreqCB, a frequency-dependent clustering bandit model incorporating user activeness to represent the clusters.

3 Problem Formulation

The set of N users is represented by $U = \{u_1, u_2, \cdots, u_N\}$. Learning procedure goes as a sequence: At each round $t = [1, 2, \ldots, T]$, the learning agent receives a user $i_t \in U$ with a set of context vectors $H_{i_t} = \{x_1, x_2, \cdots, x_K\}$, where $\parallel x \parallel \leq 1, x \in \mathbb{R}^d$ for all x in H_{i_t}. H_{i_t} consists of the vectors of candidate arms (e.g., items) to be recommended for user i_t. The learning agent then selects one arm a_t, whose feature vector is $\bar{x} \in H_{i_t}$ to recommend for user i_t according to the historical feedback $\mathcal{Q}(t) = \{i_\tau, a_\tau, r_\tau\}_{\tau=1}^{t}$, and observes payoff $r_t \in \mathbb{R}$, which is a function of both user i_t and the recommended arm a_t. After receiving feedback, the agent updates its parameters and continue to recommend. The objective of the agent is to select a proper policy $\pi = \{a_1, a_2, \cdots, a_T\}$ to minimize the cumulative regret [11], which is defined below,

$$Regret_\pi = \sum_{t=1}^{T}(r_t^* - r_t), \tag{1}$$

where r_t^* denotes payoff of the optimal choice at time t. This goal is a long-term cumulative reward that can be regarded as the target of the multi-armed bandit problem, where each item denotes each arm. To accomplish this goal, assuming r_t is a linear function of a_t's feature, which has been successfully used in bandit problems. The function is defined as follows:

$$r_t = \omega_{i_t}^{\mathrm{T}} \bar{x} + \epsilon_{a_t} \tag{2}$$

where ω_{i_t} is a d-dimension parameter vector which indicates the preference of user i_t, $\bar{x}$ is the feature vector of arm a_t, and ϵ_{a_t} is Gaussian noise.

LinUCB, as a classic solution for multi-armed bandit problem, holds a linear combination assumption between the user's feature vector and context vector, and pulls the arm $\bar{x}$ with the largest score, which is defined below:

$$\bar{x} = \arg\max_{x_k \in H}(\omega^{\mathrm{T}} x_k + \alpha\sqrt{x_k^{\mathrm{T}} M^{-1} x_k}), \tag{3}$$

where H is the set of context vectors, x_k denotes context vector of the k-th arm in candidate arms, ω is the parameters of current user, $M \in \mathbb{R}^{d \times d}$ is the correlation matrix of current user which contains information of arms that the user pulls before, α is a hyper parameter to combine the expectation (the former term) and standard deviation (the latter term) of reward. The deviation term aims to balance the trade-off between exploration and exploitation, that is, the larger deviation, the more exploration. ω is estimated by ridge regression as follows:

$$\omega = M^{-1} b, \tag{4}$$

where $b \in \mathbb{R}^d$ is the corresponding response vector (e.g., the corresponding d click/no-click user feedback).

Different from standard bandit models, we focus on clusters among users and arms that valid information can be shared for better recommendation. Existing related works only consider symmetrical relations between users in one cluster, but asymmetric relations are more practical in reality.

4 Solution and Algorithms

In this section, we propose a frequency-dependent clustering bandit algorithm, FreqCB (as shown in Algorithm 1) to explore the effectiveness of fully utilizing users' activeness to improve the performance of online recommendation. In FreqCB, we treat the users' interaction frequency as their activeness, which makes sense practically.

FreqCB maintains vector ω, vector b and correlation matrix M for each user and each cluster at timestamp t. We set three hyperparameters α, β, and γ. β, γ are used in the procedure of splitting and merging user clusters, while α is for exploring the upper-confidence bound in the reward prediction. The interact frequency of each user is set to 1 initially for computational feasibility. The initial cluster is set as the whole users with index 1. This algorithm proceeds in stages. The g_{th} stage contains 2^g timestamps totally. In each stage, the algorithm can pay more attention on exploring inaccurate user clusters and leave accurate clusters unaffected. Every user i is set unserved in the beginning of the g_{th} stage: $ch_i = 0$; if user i is served, $ch_i = 1$. Note that the subscript t denotes the timestamp. At t timestamp, we get the user $i_t \in U$ to be served and the context vectors $H_t = \{x_1, x_2, \cdots, x_K\}$ of items available for recommendation.

The cluster $c_j (i_t \in c_j)$ can be found and the algorithm computes the bandit parameters of user cluster c_j by $\omega_{c_j} = {M_{c_j}}^{-1} b_{c_j}$ and recommend a_t, whose feature vector is $\bar{x}$, for user i_t according to the standard LinUCB form in Eq. 3:

$$\bar{x} = \arg\max_{x_k \in H_{i_t}} ({\omega_{c_j}}^{\mathrm{T}} x_k + \alpha \sqrt{x_k^{\mathrm{T}} {M_{c_j}}^{-1} x_k}). \tag{5}$$

After receiving the feedback r_t given by user i_t, the algorithm updates the information of user i_t and cluster c_j (see Algorithm 2). Note that in the information updating procedure of cluster c_j, we introduce the user activeness for the evolution of M_{c_j} and b_{c_j}. To share information between similar users, FreqCB aggregates the information of users in the same cluster c_j weighted by their interaction frequency (activeness) to capture more accurate information of cluster c_j:

$$M_{c_j} = I + \sum_{i \in c_j} \hat{f}_i (M_i - I), b_{c_j} = \sum_{i \in c_j} \hat{f}_i b_i, \tag{6}$$

where $\hat{f}_i$ is the normalized frequency relating to cluster c_j:

$$\hat{f}_i = \frac{f_i}{\sum_{q \in c_j} f_q}, i \in c_j. \tag{7}$$

Algorithm 1. FreqCB

```
1: Input: exploration hyperparameters $\alpha$, $\beta$, $\gamma$.
2: Initialize $b_i = 0 \in \mathbb{R}^d$, $M_i = I \in \mathbb{R}^{d\times d}$ and $f_i = 1$ for each $i \in U$.
3: Initialize the set of cluster indexes $S = \{1\}$.
4: Initialize the first cluster $c_1 = U, b_{c_1} = 0 \in \mathbb{R}^d, M_{c_1} = I \in \mathbb{R}^{d\times d}$.
5: for stage $g = 1, 2, \cdots$ do
6:     Set $ch_i = 0$ for user $i$ in each cluster.
7:     Set $\omega_{c_j} = {M_{c_j}}^{-1} b_{c_j}$ for each cluster $c_j$, $j$ is the cluster index in $S$.
8:     for $\tau = 1, 2, \cdots, 2^g$ do
9:         Compute current timestamp: $t = 2^g + \tau - 2$.
10:        Get user $i_t \in U$ to be served.
11:        Get get context vectors $H_{i_t} = \{x_1, \cdots, x_K\}$ for candidates to be recommended.
12:        Get the index $j$ of cluster $c_j$ that user $i_t$ belongs to.
13:        Compute the coefficient of cluster $c_j$: $\omega_{c_j} = {M_{c_j}}^{-1} b_{c_j}$.
14:        Recommend item $a_t$ corresponding to $\bar{x} = \arg\max_{x_k \in H_{i_t}} ({\omega_{c_j}}^{\mathrm{T}} x_k + \alpha\sqrt{x_k^{\mathrm{T}} {M_{c_j}}^{-1} x_k})$.
15:        Receive the feedback $r_t$ given by user $i_t$.
16:        Update the information of user $i_t$ and cluster $c_j$ by calling UPDATE.
17:        Split user $i_t$ from cluster $c_j$ and form a new cluster $c_{j'}$ by calling SPLIT.
18:        Set $ch_{i_t} = 1$.
19:        Merge similar clusters by calling MERGE.
20:    end for
21: end for
```

Algorithm 2. UPDATE

```
1: $M_{i_t} = M_{i_t} + \bar{x}\bar{x}^{\mathrm{T}}$, $b_{i_t} = b_{i_t} + r_t \bar{x}$, $f_{i_t} = f_{i_t} + 1$
2: $\omega_{i_t} = M_{i_t}^{-1} b_{i_t}$
3: Obtain $\hat{f_{i_t}}$ by normalizing the frequency of users in $c_j$.
4: $M_{c_j} = I + \sum_{i \in c_j} \hat{f_i}(M_i - I)$, $b_{c_j} = \sum_{i \in c_j} \hat{f_i} b_i$, $F_{c_j} = F_{c_j} + 1$
5: $\omega_{c_j} = {M_{c_j}}^{-1} b_{c_j}$
6: Compute $p_{c_j} = \frac{F_{c_j}}{\|c_j\| t}$
7: Compute $p_{i_t} = \frac{f_{i_t}}{t}$
```

By Eq. 6, the algorithm could assist inactive users who have few interaction records leverage from active ones for effective clustering analysis, and could prevent cluster representations from being devastated by inaccurate representations of inactive users.

If user i_t is not consistent with the current cluster c_j, FreqCB splits user i_t out (see Algorithm 3). Different with SCLUB [13], FreqCB introduces user activeness in the evolution of M_{c_j} and b_{c_j}:

$$M_{c_j} = M_{c_j} - \hat{f_{i_t}}(M_{i_t} - I), b_{c_j} = b_{c_j} - \hat{f_{i_t}}\bar{x}. \tag{8}$$

Algorithm 3. SPLIT

```
 1: Set $Q(F) = \sqrt{\frac{1+\ln(1+F)}{1+F}}$
 2: if $\|\omega_{i_t} - \omega_{c_j}\| > \beta(Q(F_{i_t}) + Q(F_{c_j}))$ or $\|p_{i_t} - p_{c_j}\| > \gamma Q(t)$ then
 3:     Set $j^{'} = \max S + 1$.
 4:     Obtain $\hat{f_{i_t}}$ by normalizing the frequency of users in $c_j$.
 5:     $M_{c_j} = M_{c_j} - \hat{f_{i_t}}(M_{i_t} - I)$, $b_{c_j} = b_{c_j} - \hat{f_{i_t}} b_{i_t}$, $F_{c_j} = F_{c_j} - f_{i_t}$
 6:     $c_j = c_j - \{i_t\}$
 7:     $M_{c_{j^{'}}} = M_{i_t}$, $b_{c_{j^{'}}} = b_{i_t}$, $F_{c_{j^{'}}} = f_{i_t}$, $c_{j^{'}} = \{i_t\}$
 8: end if
```

Algorithm 4. MERGE

```
 1: for any two served clusters $j_1$ and $j_2$ do
 2:     if $\|\omega_{c_{j_1}} - \omega_{c_{j_2}}\| < \frac{\beta}{2}(Q(F_{c_{j_1}}) + Q(F_{c_{j_2}}))$ and $\|p_{c_{j_1}} - p_{c_{j_2}}\| < \gamma Q(t)$ then
 3:         $c_{j_1} = c_{j_1} \cup c_{j_2}$
 4:         $F_{c_{j_1}} = F_{c_{j_1}} + F_{c_{j_2}}$
 5:         Obtain $\hat{f_i}$ by normalizing the frequency of users in $c_{j_1}$.
 6:         $M_{c_{j_1}} = I + \sum_{i \in c_{j_1}} \hat{f_i}(M_i - I)$
 7:         $b_{c_{j_1}} = \sum_{i \in c_{j_1}} \hat{f_i} b_i$
 8:         Delete $c_{j_2}$
 9:     end if
10: end for
```

Based on Eq. 8, the information of user i_t can be deleted from cluster c_j and other users' information in cluster c_j remains. Then two served clusters will be merged if they are consistent at some level (see Algorithm 4). Similarly, according to Eq. 6, FreqCB introduces user activeness in the evolution of new clusters. When this stage ends, FreqCB continues to the next stage until it reaches the final timestamp. β and γ are hyper parameters to control the gap between user and cluster, and the gap between different clusters.

To generate the candidate items for H_{i_t}, we randomly select $K-1$ items from all items and randomly select one item from nonzero payoff items of user i_t to form a candidate item set of size K.

Complexity of Implementation. Assuming a total of T rounds, K items in H_{i_t}, and d dimensions for vectors x_k, we can analysis the complexity of FreqCB by each round. Each recommendation takes $O(Kd^2)$ time, where estimated vector for the cluster that the current user belongs to need to be computed as well as predicted scores for items in H_{i_t}. After receiving feedback, users frequency is calculated, and both user and cluster representation are updated. Note that the number of users in the cluster is at most n. Thus, the time complexity for the update is $O(nd^2 + nd)$. Each split check and each merge check take $O(d)$, respectively.

In the rounds where all current clusters are accurate, FreqCB does not continue to split and merge after one split check and m merge check. Therefore, it costs $O((K+n)d^2+(n+m)d)$ under this circumstance. In the rounds where cluster structure is being exploring, each split takes $O(n+d^2)$ time and each merge costs $O(d^3+nd)$. Note that the number of merge times is at most $n(n-1)/2$, so the time complexity in this situation is $O(n^2d^3+(K+n+1)d^2+(n^3+n)d+n)$. And the time complexity of FreqCB for T rounds (the worst case) is

$$O(T(n^2d^3+(K+n+1)d^2+(n^3+n)d+n)).$$

In addition, the time complexity for LinUCB is $O(TKd^2)$. Since SCLUB allows split and merge for cluster exploration, its time complexity (in expectation) for T rounds is:

$$O(TKd^2+Tmd+(\frac{nd^3}{\delta_1}+\frac{n^2d^3}{\delta_2})\ln(T)),$$

where δ_1 and δ_2 are simplified parameters in [13].

5 Experiments

In this section, we evaluate FreqCB[1] on both synthetic data and two real-world datasets comparing with strong baselines.

5.1 Compared Algorithms

- **Random.** It randomly picks one of all K items in every timestep, without learning user parameters.
- **ϵ-greedy.** It randomly selects one of all K items with probability ϵ, otherwise selects the item with the highest empirical mean with probability $1-\epsilon$.
- **Lin-One.** It runs a single instance of LinUCB to serve all users, which means all users belong to the same cluster.
- **Lin-Ind.** It runs an independent instance of LinUCB per user to make a recommendation in a fully personalized style, which means each user forms one cluster by himself.
- **CAB.** It is based on the linear contextual bandit framework and incorporates collaborative filtering. Users having similar behavior form one cluster in a context dependent way, then CAB selects arms by using the information of the whole cluster.
- **SCLUB.** It incorporates collaborative filtering by graph-based clustering. User clusters are regarded as graphs and they are maintained by splitting, merging, and updating. Users in the same cluster share collaborative information with others.

[1] Our code and data is available at https://github.com/holywoodys/FreqCB.

Algorithm 5. Evaluation

```
1: Inputs: $T > 0$
2: Initial: $G_0 = 0$, a zero total reward at $t = 0$
3: for $t = 1, 2, \cdots, T$ do
4:     Get event feedback $e_t = \{i_t, a_t, r_t\}$
5:     $\mathcal{Q}(t) = \mathcal{Q}(t-1) \cup e_t$
6:     $G_t = G_{t-1} + r_t$
7:     Current CTR $= G_t/t$
8: end for
9: Outputs: final CTR $= G_T/T$
```

5.2 Experiments on Synthetic Data

Dataset Description. The synthetic data contains 100 users split into ten clusters, and 1,000 articles which are grouped into ten clusters as well. The size of each user cluster is set as 10. For article feature vectors, we allocate different numbers of 0 randomly to the first five dimensions as a seed vector of a group, and the values of other non-zero dimensions are generated from Gaussian distribution. Then, their sixth dimension is set to 1 and the vectors are transformed into unit vectors. In the sequence of recommendation, the size of candidate item set H_{i_t} is 25 ($K = 25$), and the served user i_t is generated uniformly at random over 100 users.

Evaluation Method. The synthetic data includes statistic information of user-item interactions. To mimic the real-world dataset, We allocate z items to each user and take these as items they interact with. In every synthetic data, z is fixed for every user for simplicity. We hold the setting that if a user has interacted with the recommended item, the policy gets 1 payoff. We evaluate algorithm performance by Click-Through Rate (CTR). Note that whether the cold start process is accelerated can be judged by final CTR. If the algorithm learns the user's interest faster, it can achieve more accurate recommendations in the follow-up, thereby increasing the final CTR. The concrete evaluation method is described in Algorithm 5. In our experiments, without loss of generality, hyper parameter γ and β are fixed. We test different α, a common exploration parameter that all baselines share.

Experimental Results. We run each algorithm in seven different parameter settings with $T = 50{,}000$, where α is set as $\{0.001, 0.005, 0.01, 0.07, 0.1, 0.5, 1\}$. Each setting is tested on five runs and we record the average CTRs, standard deviation, max and min CTR. The best results for each algorithm are shown in Table 1. We can observe that FreqCB ($\alpha = 0.07$) achieves the best CTR over all results. Specifically, the best result of FreqCB improves over SCLUB's best result by 14.4%. The reason is that FreqCB enables active users to filter out the "harmful" information from inactive users, and inactive users to learn more from active users. Therefore, the cluster representation ω_j can be improved.

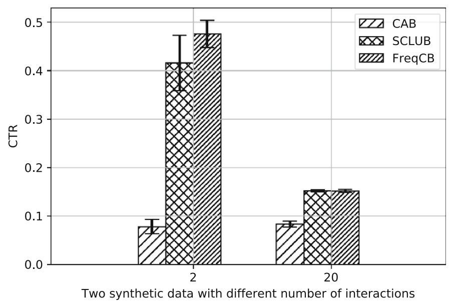


Fig. 1. CTR on different synthetic data. The abscissa represents the numbers of interactions. To mimic real-world data, in our settings, numbers of interactions z denote the numbers of items that each user interacts with.

To study the influence of users' activeness, we test clustering bandit model on different synthetic data and report their best results in Fig. 1. The horizontal axis denotes user activeness which reflects the interaction frequency in the synthetic data. Figure 1 shows that when the user activeness is at a low level, our model outperforms other clustering bandit models, and when the user's activeness becomes higher, the performance of our model tends to be consistent with SCLUB.

Table 1. CTR on synthetic data.

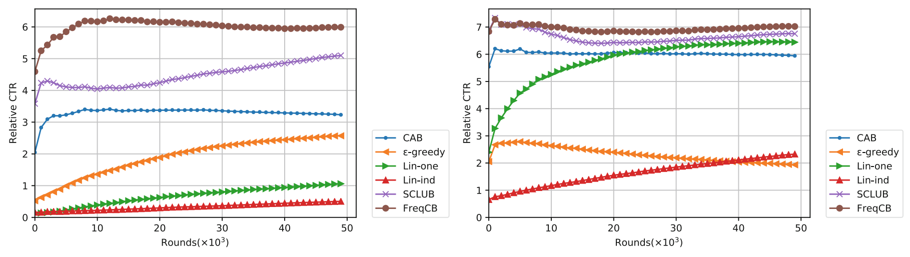
Algorithm	Mean	std	max	min
Random	0.04234	0.000504	0.04282	0.04160
Lin-one ($\alpha = 0.1$)	0.30435	0.012441	0.31776	0.28692
Lin-ind ($\alpha = 0.01$)	0.14651	0.016498	0.16880	0.12972
CAB ($\alpha = 0.001$)	0.07792	0.014791	0.09840	0.06388
SCLUB ($\alpha = 0.01$)	0.41366	0.060439	0.49294	0.35398
FreqCB ($\alpha = 0.07$)	**0.47581**	0.028275	0.50606	0.42456

5.3 Experiments on Real-world Dataset

Dataset Description. The following experiments are conducted on two real-world datasets:

(1) **LastFM** dataset is extracted from the music streaming service Last.fm. This dataset contains a social network with 1,892 users, 12,717 bi-directional user friend relations, and 17,632 artists. There are 11,946 tags used to tag artists by users. We preprocess tags by breaking down them into single words, then remove special symbols and convert duplicate tags into one, e.g., we consider the tag "metal metal metal" as "metal". The dataset contains the tag assignments of artists provided by each particular user, so we collect all tags that an artist has

been tagged to create a TF-IDF context vector $x \in \mathbb{R}^d$, which uniquely represents the current artist, and utilize Principal Component Analysis for dimension reduction. Besides, the dataset contains information about artists that users have listened to, so we utilize this information to create binary payoffs: if a user listened to an artist at least once the payoff is 1, otherwise the payoff is 0. The way of generating recommendation candidates is mentioned in Sect. 4.



(a) Relative CTR on Last.FM (b) Relative CTR on MovieLens

Fig. 2. The relative CTR of the proposed model and baselines on the two datasets, Last.FM and MovieLens.

(2) **25 m MovieLens** dataset is an extension of [8], which contains 25 million ratings and more than 1 million tag applications for 62,423 movies by 162,541 users. For simplicity, we choose randomly 1,000 users from the users whose rating numbers $\geq$100 for experiments. To avoid the prior information for bandit models, we use the same method as LastFM dataset does to get movie features. The binary payoff is given by 5-star rating: if the rating is more than 3, the payoff is 1, otherwise the payoff is 0.

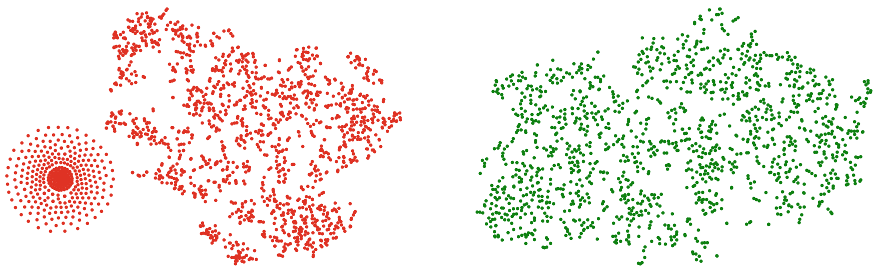
Experimental Results. In our experiments, the interact frequency of each user is set to 1 initially. When a user is served, his interact frequency updates according to Algorithm 2. We test two datasets with T = 50,000 samples and record the CTRs per thousand. For all algorithms, the exploration parameter α is set as $\{0.1, 0.3, 0.5, 0.7, 1.0\}$, respectively. The evaluation policy is described in Algorithm 5. Each dataset has been run five times and we compute the average relative CTR [20] for comparison. Experimental results are shown in Fig. 2.

For both two real-world datasets, our model FreqCB outperforms SCLUB as well as other strong baselines. The final CTR of FreqCB improves over SCLUB by 17.45% and 3.94% on LastFM and MovieLens, respectively.

For LastFM, in the beginning, Lin-one and Lin-ind perform worse because the users' feedback is not enough to estimate users' parameters accurately. As recommendation proceeds, Lin-one outperforms Lin-ind in the long run due to its better collaborative effect over all users. Compared to MovieLens dataset, the performance of baselines without applying collaborative personalization (e.g., Lin-one and Lin-ind) is relatively poor. This can be attributed to the generation way of LastFM dataset [6]. The interactions between users and music are few and lots of songs to be played are generated by the music streaming service of Last.fm. Hence, the nonzero payoff music may not be the one the user likes actually and the collaborative effect over all users is weak.

For MovieLens dataset, users' interactions are frequent, so the models incorporating collaboration between users (e.g., Lin-one, CAB, SCLUB, FreqCB) achieve better performance than others (e.g., ϵ-greedy, Lin-ind).

Figure 3 provides the visualization of cluster representations derived from SCLUB and FreqCB on LastFM dataset, respectively. The given results are under each model's best parameter setting. SCLUB acquires 1,891 clusters and obtains 0.201 CTR, while FreqCB achieves 1,411 clusters and obtains 0.261 CTR. From the visualization of two models, we find that SCLUB cannot effectively merge some similar clusters, especially the clusters shown at the bottom left of Fig. 3a. Figure 3b exhibits discernible clustering effect, which qualitatively indicate that FreqCB is capable of leveraging asymmetric collaborative signals to have better cluster representations.



(a) 1,891 clusters from SCLUB (b) 1,411 clusters from FreqCB

Fig. 3. Visualization of the learned t-SNE transformed cluster representations derived from SCLUB and FreqCB. Each point represents a user cluster from Last.FM dataset.

6 Conclusion

In this paper, we propose a frequency-dependent clustering bandit model for personalized online recommendations. Our model is capable of achieving collaborative filtering effectively by considering user activeness. The experimental results on both synthetic data and real-world datasets indicate that our model has promising results compared with other state-of-the-art models when considering the frequency of user interactions with the system.

There are some remained directions worthwhile to be explored. One future direction is to extend our solution to some other clustering of bandit strategies, where loosening the hard assumption that one user only belongs to one group is a prospective way to improve the quality of recommendation. Another interesting direction is to apply both users and items clustering for better interactive collaboration.

Acknowledgement. This work is partially supported by China Natural Science Foundation under grant (No. 62171391) and the Natural Science Foundation of Fujian Province of China under grant (No. 2020J01053).

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005). https://doi.org/10.1109/TKDE.2005.99
2. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multiarmed bandit problem. SIAM J. Comput. **32**(1), 48–77 (2002). https://doi.org/10.1137/S0097539701398375
3. Ban, Y., He, J.: Local clustering in contextual multi-armed bandits. In: Proceedings of the Web Conference 2021, pp. 2335–2346, WWW 2021. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3442381.3450058
4. Cesa-Bianchi, N., Gentile, C., Zappella, G.: A gang of bandits (2013)
5. Gangan, E., Kudus, M., Ilyushin, E.: Survey of multiarmed bandit algorithms applied to recommendation systems. Int. J. Open Inf. Technol. **9**, 12–27 (2021)
6. Gentile, C., Li, S., Kar, P., Karatzoglou, A., Zappella, G., Etrue, E.: On context-dependent clustering of bandits. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1253–1262. PMLR, 6–11 August 2017. https://proceedings.mlr.press/v70/gentile17a.html
7. Gentile, C., Li, S., Zappella, G.: Online clustering of bandits. In: Xing, E.P., Jebara, T. (eds.) Proceedings of the 31st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 32, pp. 757–765. PMLR, Beijing, China, 22–24 June 2014. https://proceedings.mlr.press/v32/gentile14.html
8. Harper, F.M., Konstan, J.A.: The MovieLens datasets: history and context. ACM Trans. Interact. Intell. Syst. **5**(4), 1–19 (2015). https://doi.org/10.1145/2827872
9. Korda, N., Szorenyi, B., Li, S.: Distributed clustering of linear bandits in peer to peer networks. In: Balcan, M.F., Weinberger, K.Q. (eds.) Proceedings of The 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 48, pp. 1301–1309. PMLR, New York, New York, USA, 20–22 June 2016. https://proceedings.mlr.press/v48/korda16.html
10. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009). https://doi.org/10.1109/MC.2009.263
11. Lattimore, T., Szepesvári, C.: Bandit Algorithms. Cambridge University Press, Cambridge (2020). https://doi.org/10.1017/9781108571401
12. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, pp. 661–670. Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1772690.1772758, https://doi.org/10.1145/1772690.1772758
13. Li, S., Chen, W., Li, S., Leung, K.S.: Improved algorithm on online clustering of bandits (2019)
14. Li, S., Karatzoglou, A., Gentile, C.: Collaborative filtering bandits. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2016, pp. 539–548. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2911451.2911548

15. Lika, B., Kolomvatsos, K., Hadjiefthymiades, S.: Facing the cold start problem in recommender systems. Expert Syst. Appl. **41**(4, Part 2), 2065–2073 (2014). https://doi.org/10.1016/j.eswa.2013.09.005, https://www.sciencedirect.com/science/article/pii/S0957417413007240
16. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297, Oakland, CA, USA (1967)
17. Nguyen, T.T., Lauw, H.W.: Dynamic clustering of contextual multi-armed bandits. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, pp. 1959–1962. Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2661829.2662063
18. Pandey, S., Chakrabarti, D., Agarwal, D.: Multi-armed bandit problems with dependent arms. In: Proceedings of the 24th International Conference on Machine Learning, ICML 2007, pp. 721–728. Association for Computing Machinery, New York, NY, USA (2007). https://doi.org/10.1145/1273496.1273587
19. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web. LNCS, vol. 4321, pp. 325–341. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72079-9_10
20. Wang, Q., et al.: Online interactive collaborative filtering using multi-armed bandit with dependent arms. IEEE Trans. Knowl. Data Eng. **31**(8), 1569–1580 (2019). https://doi.org/10.1109/TKDE.2018.2866041
21. Xu, X., Dong, F., Li, Y., He, S., Li, X.: Contextual-bandit based personalized recommendation with time-varying user interests. Proc. AAAI Conf. Artif. Intell. **34**(04), 6518–6525 (2020). https://doi.org/10.1609/aaai.v34i04.6125, https://ojs.aaai.org/index.php/AAAI/article/view/6125
22. Yang, L., Liu, B., Lin, L., Xia, F., Chen, K., Yang, Q.: Exploring clustering of bandits for online recommendation system. In: Fourteenth ACM Conference on Recommender Systems, RecSys 2020, pp. 120–129. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3383313.3412250