Exploring Clustering of Bandits for Online Recommendation System

Liu Yang
lyangau@cse.ust.hk
Hong Kong University of Science and Technology
Hong Kong, China

Bo Liu
brodyliu@webank.com
AI Group, WeBank
Shenzhen, China

Leyu Lin
goshawklin@tencent.com
WeiXin Group, Tencent
Beijing, China

Feng Xia
xiafengxia@tencent.com
WeiXin Group, Tencent
Beijing, China

Kai Chen
kaichen@cse.ust.hk
Hong Kong University of Science and Technology
Hong Kong, China

Qiang Yang
qiangyang@webank.com
AI Group, WeBank
Shenzhen, China

ABSTRACT

Cluster-of-bandit policy leverages contextual bandits in a collaborative filtering manner and aids personalized services in the online recommendation system (RecSys). When facing insufficient observations, the cluster-of-bandit policy could achieve more outstanding performance because of knowledge sharing. Cluster-of-bandit policy aims to maximize the cumulative feedback, *e.g.*, clicks, from users. Nevertheless, in the way of their goal exist two kinds of uncertainties. First, cluster-of-bandit algorithms make recommendations according to their uncertain estimation of user interests. Second, cluster-of-bandit algorithms transfer relevant knowledge upon uncertain and noisy user clusters. Existing algorithms only consider the first one, while leaving the latter one untouched. To address the two challenges together, in this paper, we propose the ClexB policy for online RecSys. On the one hand, ClexB estimates user clustering more accurately and with less uncertainty via explorable-clustering. On the other hand, ClexB also exploits and explores user interests by sharing information within and among user clusters. In summary, ClexB explores knowledge transfer and further aids the inferences about user interests. Besides, we provide extensive empirical experiments on both the synthetic and real-world datasets and regret analysis, further consolidating the superiority of ClexB.

CCS CONCEPTS

• **Computing methodologies** → **Online learning settings**; *Sequential decision making*; • **Information systems** → **Recommender systems**; **Collaborative filtering**.

KEYWORDS

online learning, cluster-of-bandit, recommendation system

RecSys '20, September 22–26, 2020, Virtual Event, Brazil

ACM ISBN 978-1-4503-7583-2/20/09…$15.00
https://doi.org/10.1145/3383313.3412250

ACM Reference Format:
Liu Yang, Bo Liu, Leyu Lin, Feng Xia, Kai Chen, and Qiang Yang. 2020. Exploring Clustering of Bandits for Online Recommendation System. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 22–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3383313.3412250

1 INTRODUCTION

Online recommendation system (RecSys) is one of the most popular application areas of machine learning. It customizes personalized recommendation services, like articles, music, and short videos, for each user. RecSys receives feedback, such as clicks, comments, likes, *etc.*, from users. The target of RecSys is to accurately mine the interests of users from their behavior data. Thus, RecSys could show them the desired items to make a profit. However, traditional RecSys always suffers from the exploitation-exploration dilemma. The critical cause of this dilemma is the uncertain estimation, from finite and noisy feedback, of the reward function, *e.g.*, user interests. Facing the uncertainty of user interests, algorithms should make a trade-off between exploiting the existing knowledge to make the best-known recommendation and exploring the uncharted territory. The contextual bandit algorithm [12] has been popular to solve this problem. In a contextual bandit setting of online RecSys, articles and their corresponding features, like title, category, *etc.*, are observed by the agent. At each step, the agent chooses one article to present to users according to the click history and observes if users clicked this recommendation or not. The final goal is to maximize cumulative clicks. The agent tries to know about user interests from the received finite and noisy rewards. Therefore, the learned parameters contain varying degrees of uncertainties, which gives the essence of the contextual bandit, taking those uncertainties into consideration through the exploration of the reward function. The contextual bandit prefers the known items with the higher CTR, *i.e.*, exploitation. In the meantime, the contextual bandit policies also tend to choose the items with large uncertainty, *i.e.*, exploration. In the absence of exploration, supervised learning algorithms tend to get stuck in sub-optimal solutions.

Recently, some researchers incorporate collaborative filtering (CF) effects into bandit methods for enhancing performance. Collaborative filtering is one of the dominating methods in both the

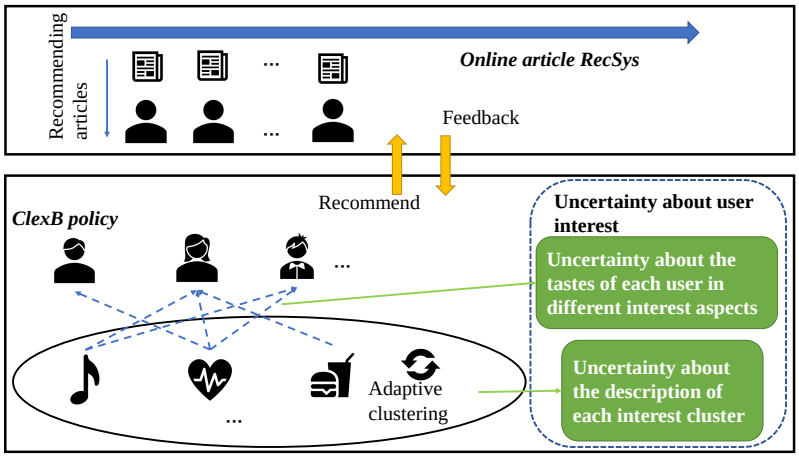


Figure 1: Illustration of the online article RecSys using ClexB. The adaptive clustering, for instance, produces different interest clusters, *e.g.*, music, health, and food. And each user has the corresponding tastes for these interest aspects. ClexB explores both interest clusters and users' tastes for these interest aspects, thus indirectly exploring user interest.

academia and industry [19]. In the view of transfer learning [17], collaborative filtering shares knowledge among similar users and, on the other side, related items. Hence, such algorithms always perform better, facing insufficient data. Most of the existing methods utilize the clustering methods to represent the idea of collaborating. Firstly, they construct user clusters via k-means clustering [16], graph-based clustering [8], and *etc.*. Then, users inter and intra the clusters share knowledge among each other. Cluster-of-bandit algorithms improve the performance of the original contextual bandit algorithms. Nevertheless, it introduces another kind of uncertainty, *i.e.*, the uncertainty of clustering.

In this paper, we consider the two aforementioned uncertainties, including the uncertain user interests and the uncertain user clustering, simultaneously, and propose a new **Cl**ustering-**Ex**plorable **B**andits (ClexB) algorithm. ClexB performs the clustering on user interests in an adaptive manner, and transfers knowledge among users, benefiting the acquaintance with user preferences accordingly. According to Fig. 1, the cluster can be regarded as the interest cluster, *e.g.*, the interest cluster of music, health, or food. More concretely, we assume that user interest is a linear combination of the parameter vectors of potential interest clusters. In other words, the interests of one user can involve different aspects. Furthermore, the coefficients of the linear combination are assumed to be sparse, which means one user has preferences and cannot cover all interest aspects in nature. For example, the first user in Fig. 1 belongs only to the interest clusters of health and food. It means that he is not into music. Besides, one user can belong to several interest clusters with different weights. ClexB explores both interest clustering and user tastes on these interest aspects, thus indirectly exploring user interest. ClexB leverages both transfer learning and bandit algorithm. The two techniques complement and help each other. On the one hand, the simultaneous exploration of user interests and user clustering improves both preference learning and knowledge transfer. Hence, ClexB is able to quickly identify the accurate user clustering, avoiding getting stuck in sub-optimal recommendations. On the other hand, transfer learning also helps bandit a lot. Theoretically speaking, the regret of a context bandit policy, *i.e.*, the cost of exploration, grows with respect to the increasing dimension of context. As a result, the majority of bandit policies suffer when facing high-dimensional context space or insufficient observations per user. With knowledge sharing, ClexB utilizes behavior data of all users to explore the parameter vectors of clusters. It also explores the low-dimensional clustering weight of each user only using personal data. When the number of user clusters is much fewer than the dimension of context, *e.g.*, user features, ClexB enormously saves exploration efforts.

In conclusion, the major challenge of our algorithm is the exploration of two mechanisms simultaneously. Especially one of them is sparse, which is also relatively rare in academia and industry. The contributions of this paper are threefold: 1) ClexB shares knowledge among related users to speed up exploration; 2) To the best of knowledge, ClexB is the first algorithm to enhance collaborative filtering efforts via exploring user clustering; 3) ClexB is empirically verified on synthetic and real-world datasets. One of the real-world datasets is collected from a popular social application *i.e.*, WeChat [1]. Meantime, ClexB enjoys rigorous theoretical analysis.

In the following sections, Section 2 reviews several relative works. Section 3 formally defines the problem and explains the ClexB algorithm. In Section 4, we empirically compare ClexB with state-of-the-art baseline models on both synthetic and real-world datasets. Besides, in section 5, we analyze ClexB theoretically, including the regret analysis. Finally, we draw conclusions in Section 6.

2 RELATED WORKS

In this section, we compare ClexB with several lines of related works, including transfer learning, multi-task learning, and contextual bandit policies.

Multi-task learning and ClexB share a similar goal, which is to enhance the overall performance of tasks via knowledge sharing.

[1] http://www.wechat.com

GO-MTL [11], for instance, is a framework for multi-task learning that enables one to share the information across the tasks selectively. It assumes that each task parameter vector is a linear combination of a finite number of underlying basis tasks and that the linear combination coefficients are sparse. However, it ignores exploring the uncertain user interests. When facing a RecSys problem with finite and noisy feedback, GO-MTL tends to get stuck in the sub-optimal actions. Besides, GO-MTL acquires the sparse parameters by optimizing the l_1 norm penalty, while ClexB makes use of a Bayesian approach. LinUCB [12] and LTS [3] are two standard linear contextual bandit algorithms. And RVM-LTS [9] proposes a scalable approach that adapts to the unknown sparse support, which borrows the ideas from LTS and RVM [20]. Furthermore, TCB [15] takes advantage of knowledge sharing. It addresses the exploitation-exploration dilemma and the cold-start problem together. TCB adopts transfer learning to improve the exploitation of a contextual bandit policy and accelerate its exploration in a target domain; on the other hand, it leverages contextual bandit policies to speed up the knowledge transfer. But, TCB is designed for a fixed source domain and target domain. In comparison, ClexB shares knowledge across multiple related tasks. Similarly, [24] and [10] also benefit from sharing knowledge via collaborative filtering, exploring the uncertainties of the PMF model [18]. They supply the guidance of exploring two parameters but have nothing to do with context, *e.g.*, user profile.

In the past few years, several works also study the cluster-of-bandit algorithms. GOB [5], CLUB [8], COFIBA [14], CAB) [7] and SCLUB [13] are a series of ongoing contributions. Among them, CAB is designed for collaborative recommendation tasks. It implements the underlying feedback sharing mechanism by estimating the neighborhood of users in a context-dependent manner. For each arm, CAB re-estimates its neighborhood, according to others' tastes of this arm. The tastes rely upon the arm's context. After one user is served using a collaborative approach, not only him but also its neighborhood is updated. And ThompCAB [4] is a Bayesian counterpart of CAB. CLUB is based on an adaptive clustering of strategy. It also regards each user as the node in the graph. CLUB starts from the complete graph and progressively erases edges based on the evolution of user interests. Users in the same cluster (connected partition) make choices all-together. And SCLUB generalizes the setting of CLUB. It allows non-uniform frequency distributions over users. Besides, SCLUB also designs new split and merge operations on clusters together with set representations of clusters to accelerate the process of identifying underlying clusters. Moreover, DynUCB [16] dynamically assigns users from one cluster to another, as their preferences change, utilizing K-means based on the similarity of user parameter vectors. Nevertheless, in comparison with ClexB, these algorithms overlook the necessity of exploring the uncertain user clustering.

3 OUR APPROACH

In this section, we first define the notations used throughout this paper. Then, we formally define cluster-of-bandit problem. Furthermore, we explain ClexB in two parts, *i.e.*, knowledge sharing via adaptive clustering and learning user interest via exploring clustering. ClexB assumes a proper parameter structure to cluster users and transfer knowledge among them. Meanwhile, exploring clustering accelerates the transfer process, learning better user preferences.

Throughout this paper, the bold uppercase symbol ($\boldsymbol{A}$), the bold lower-case symbol ($\boldsymbol{x}$), and the regular symbol (r) represent matrix, column vector, and scalar respectively. The uppercase calligraphic symbol ($\mathcal{Z}$) stands for set. We use $\boldsymbol{I}^d$ to denote the identity matrix with d rows and d columns and use $\boldsymbol{0}^d$ to stand for the d-dimensional zero vector. In addition, $\boldsymbol{\theta}$, $\hat{\boldsymbol{\theta}}$, and $\tilde{\boldsymbol{\theta}}$ denote, the fixed but unknown true parameter vector, the empirical estimate of true parameter, and the vector randomly drawn from the distribution with the mean $\hat{\boldsymbol{\theta}}$. For simplicity, the notations commonly used in the following part of this paper are shown in Table 1.

3.1 Problem Setting

Suppose there are n users, represented by $\mathcal{U} = \{1, ..., n\}$. At each time step $t = 1, 2, ..., T$ of the online recommendation, the agent faces one user $u(t) \in \mathcal{U}$ and $\mathcal{Z}(t) = \{\boldsymbol{x}_i(t) \subseteq \mathbb{R}^d\}_{i=1}^{z(t)}$, which represents the context corresponding to the available arms. The agent chooses one arm $a(t)$ to recommend for user $u(t)$ according to the historical feedback, *i.e.*, $\mathcal{H}(t-1) = \{a(\tau), r_{a(\tau)}(\tau), \mathcal{Z}(\tau)\}_{\tau=1}^{t-1}$. Then, the agent obtains the feedback $r_{a(t)}(t)$. The objective of the agent is to select the proper actions $\{a(1), ..., a(T)\}$ at different steps and minimize the cumulative regret, which is defined below, in theory:

$$R(T) = \sum_{t=1}^{T} r_{a^*(t)}(t) - r_{a(t)}(t), \tag{1}$$

where $a^*(t)$ denote the optimal arm at time t. In practice, in the absence of the optimal arm at each step, we maximize the total reward, *i.e.*, $\sum_{t=1}^{T} r_{a(t)}(t)$ instead.

Contextual bandit policies aim to maximize the cumulative clicks for a RecSys. To accomplish this goal, we should estimate users' preferences upon their historical behaviors. For instance, we want to learn n users' reading preferences of articles. And their past reading records, *i.e.*, $\{(\boldsymbol{X}_u, \boldsymbol{r}_u)\}_{u=1}^{n}$ could be the training set, where $\boldsymbol{X}_u \in \mathbb{R}^{d \times N_u}$, $\boldsymbol{r}_u \in \mathbb{R}^{N_u \times 1}$, d stands for the feature dimension of article and N_u is the sample amount. We hold the linear assumption that $\boldsymbol{r}_u = {\boldsymbol{X}_u}^T \boldsymbol{\theta}_u + \boldsymbol{\eta}$, in which $\boldsymbol{\eta}$ is a Gaussian noise vector drawn from distribution $\mathcal{N}(\boldsymbol{0}^d, \sigma \boldsymbol{I}^d)$. $\boldsymbol{\theta}_u \in \mathbb{R}^{d \times 1}$ is the parameter vector represents the reading interest of user u. Apparently, we could infer user interests $\hat{\boldsymbol{\theta}}_u$ for each independent user u using his own records.

3.2 Knowledge Sharing via Adaptive Clustering

We design an adaptive clustering method to transfer knowledge among users. It estimates user interests more accurately, especially when each user's feedback is limited. We assume there are k interest clusters and one user's parameter vector can be a linear combination of several parameter vectors of clusters. Mathematically speaking, we assume that $\boldsymbol{\Theta} = \boldsymbol{LS}$, where $\boldsymbol{\Theta} \in \mathbb{R}^{d \times n}$ is generated by stacking the parameter vectors of users, $\boldsymbol{L} \in \mathbb{R}^{d \times k}$ stands for the d-dimensional parameter vectors of interest clusters, and $\boldsymbol{S} \in \mathbb{R}^{k \times n}$ represents the proportions of users holding these interests. One specific user preference $\boldsymbol{\theta}_u$ can be obtained by $\boldsymbol{L}\boldsymbol{s}_u$, where $\boldsymbol{s}_u \in \mathbb{R}^{k \times 1}$ is just the column of $\boldsymbol{S}$. With the intervention of $\boldsymbol{L}$ and $\boldsymbol{S}$, the reward function for one specific user u is:

Table 1: Table of notations.

Symbol	Description	Symbol	Description
$\mathcal{U}$	User set	$\mathcal{Z}(t)$	Arm pool at time t
$\boldsymbol{x}_i(t)$	Context of arm i at time t	$\mathcal{H}(t-1)$	History up to time $t-1$
$a(t)$	Chosen arm at time t	$r_{a(t)}(t)$	Reward of arm $a(t)$ at time t
$\boldsymbol{\theta}_u$	Linear parameter of user u	$\boldsymbol{\eta}$	Gaussian noise vector
$\boldsymbol{L}$	Parameter vectors of interest clusters	$\boldsymbol{s}_u$	Parameter vector of cluster indicators for user u
$\boldsymbol{A}_u$	Diagonal matrix	$\boldsymbol{\alpha}_u$	Diagonal elements of $\boldsymbol{A}_u$
d	Dimension of context	k	Number of interest clusters
σ	Parameter controlling user interest coverage	v_L	Parameter controlling the exploration of $\boldsymbol{L}$
v_S	Parameter controlling the exploration of $\boldsymbol{S}$	$\Sigma(t)^{-1}$	Covariance matrix of the prior distribution for $\text{vec}(\tilde{\boldsymbol{L}}(t))$ at time t
$\boldsymbol{\Psi}_{u(t)}(t)^{-1}$	Covariance matrix of the prior distribution for $\tilde{\boldsymbol{s}}_{u(t)}(t)$ at time t	$\{(\boldsymbol{X}_u, \boldsymbol{r}_u)\}_{u=1}^n$	Behavior records of user u

$$\boldsymbol{r}_u = \boldsymbol{X}_u^T \boldsymbol{L} \boldsymbol{s}_u + \boldsymbol{\eta}. \tag{2}$$

Without the loss of generality, we should also assume $\|\boldsymbol{L}\|_F \leq 1$, $\|\boldsymbol{s}_u\| \leq 1$ and $\|\boldsymbol{x}_i(t)\| \leq 1$, for all u, i, t, to make the regret bounds scale-free. Furthermore, each user is composed of multiple but not all interest clusters. For instance, shown in Fig. 1, the first user is interested in only health and food, while the third one is not interested in food. Alternatively speaking, $\boldsymbol{S}$ is sparse. Therefore, each $\boldsymbol{\theta}_u$ is a combination of only a few columns of $\boldsymbol{L}$, which means the elements of $\boldsymbol{s}_u$ contains zeros with high probability. For any two columns $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ of $\boldsymbol{S}$, corresponding to users i and j, the overlappings between the sparsity patterns determine the number of interest aspects that the two users have in common. Users whose sparsity patterns are orthogonal to each other could be thought to have no intersections of interests. To obtain a sparse $\boldsymbol{S}$, we add the prior, *i.e.*, $p(\boldsymbol{s}_u) = \mathcal{N}(\boldsymbol{0}^k, \boldsymbol{A}_u^{-1})$, on it. $\boldsymbol{A}_u \in \mathbb{R}^{k \times k}$ is a diagonal matrix. And we define vector $\boldsymbol{\alpha}_u := [\alpha_u^1, \alpha_u^2, ..., \alpha_u^k]$ to be the diagonal elements of $\boldsymbol{A}_u$, with $A_u^{i,i} = \alpha_u^i$, i indicating the position of matrix or vector. A hyper-prior is then further assigned on $\boldsymbol{\alpha}_u$, *i.e.*, $\boldsymbol{\alpha}_u \sim \prod_{i=1}^k Gamma(0,0)$. Thus, s_u^i, which is the i^{th} element of $\boldsymbol{s}_u$, is randomly drawn from a zero-mean Gaussian distribution with variance ${\alpha_u^i}^{-1}$. Therefore, we can manipulate $\boldsymbol{\alpha}$ to control the sparsity of $\boldsymbol{S}$. We could obtain the optimal α by maximizing the marginal likelihood $p(\boldsymbol{r}|\boldsymbol{\alpha})$. For each user u, the sparse Bayesian learning is formulated as the (local) maximization of the marginal likelihood with respect to $\boldsymbol{\alpha}_u$:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}_u) &= \log\, p(\boldsymbol{r}_u|\boldsymbol{\alpha}_u, \sigma^2), \\ &= log \int_{-\infty}^{+\infty} p(\boldsymbol{r}_u|\boldsymbol{s}_u, \sigma^2)\, p(\boldsymbol{s}_u|\boldsymbol{\alpha}_u)\, d\boldsymbol{s}_u, \\ &= -\frac{1}{2}[N\, log\, 2\pi + log|\boldsymbol{C}_u| + \boldsymbol{r}_u^T \boldsymbol{C}_u^{-1} \boldsymbol{r}_u], \end{aligned} \tag{3}$$

where $\boldsymbol{C}_u = \sigma^2 \boldsymbol{I}^{N_u} + (\boldsymbol{X}_u^T \boldsymbol{L}^*) \boldsymbol{A}_u^{-1} (\boldsymbol{X}_u^T \boldsymbol{L}^*)^T$ and $\mathcal{L}(\boldsymbol{\alpha}_u)$ has a unique maximum with respect to α_u^i:

$$\alpha_u^i = \begin{cases} \frac{{p_u^i}^2}{{q_u^i}^2 - p_u^i}, & \text{if } {q_u^i}^2 > p_u^i, \\ +\infty, & \text{if } {q_u^i}^2 \leq p_u^i, \end{cases} \tag{4}$$

$$p_u^i = {\boldsymbol{\phi}_u^i}^T \boldsymbol{C}_{u-i}^{-1} \boldsymbol{\phi}_u^i, \tag{5}$$

$$q_u^i = {\boldsymbol{\phi}_u^i}^T \boldsymbol{C}_{u-i}^{-1} \boldsymbol{r}_u, \tag{6}$$

$$\boldsymbol{C}_{u-i} = \sigma^2 \boldsymbol{I}^{N_u} + \sum_{m \neq i} \alpha_u^{m-1} \boldsymbol{\phi}_u^m \boldsymbol{\phi}_u^{mT}, \tag{7}$$

where $\boldsymbol{\phi}_u^i \in \mathbb{R}^{N_u \times 1}$ is the ith column of $\boldsymbol{X}_u^T \boldsymbol{L}$. A sequential sparse Bayesian learning algorithm is used to optimize $\boldsymbol{\alpha}_u$, which orderly chooses one α_i to re-estimate until convergence. Detailed methods can be found in RVM [20] and [21]. The latter one also provides a computational speedup algorithm.

This adaptive clustering approach has two hyper-parameters, *i.e.*, k and σ. k controls the number of interest clusters, and σ manages the user's interest coverage. With a small σ, our clustering method tends to learn a broader user interest. Otherwise, the user's interests would focus on several main aspects. After we learn the accurate interest clusters and users' cluster indicators, knowledge could be shared both inter and intra the clusters.

3.3 Learning User Interest via Exploring Clustering

However, in the online environment, it's challenging to learn a functional interest clustering with finite and noisy feedback. In this subsection, we fortify the aforementioned adaptive clustering method with the exploration part.

The contextual bandit family can be approximately divided into two parties, *i.e.*, OFUL [1] and LTS [3]. The former adopts the upper confidence bound to always make the most optimistic decisions. The latter one, in each step, samples the parameter vector from a learned multivariate normal distribution, which contains a level set. Compared with LTS, the bound of OFUL is too loose to perform well in practice for sparse parameters [9]. Therefore, we make use

Algorithm 1 Clustering-explorable Bandits (ClexB)

Input: $\sigma, \lambda_L, \upsilon_L, \upsilon_S \in \mathbb{R}^+$.
Output: Policy $\pi : u, \mathcal{Z}_u \rightarrow a$.
Initialize:
$\Sigma(1) \leftarrow \lambda_L \boldsymbol{I}^{dk}$, $\text{vec}(\hat{\boldsymbol{L}}(1)) \leftarrow \boldsymbol{0}^{dk}$;
$\hat{\boldsymbol{A}}_u(1) \leftarrow \boldsymbol{I}^k$, $\boldsymbol{\Psi}_u(1) \leftarrow \hat{\boldsymbol{A}}_u(1)$, $\hat{\boldsymbol{s}}_u(1) \leftarrow \boldsymbol{0}^{dk}$, $u = 1, 2, ..., n$;
for $t = 1, 2, 3, ..., T$ **do**
  Receive user $u(t)$;
  Observe features of all arms $a \in \mathcal{Z}_{u(t)}(t) : \boldsymbol{x}_{u(t),a}(t) \in \mathbb{R}^d$;
  Sample $\tilde{\boldsymbol{L}}(t)$ according to Eq. 8;
  Sample $\tilde{\boldsymbol{s}}_{u(t)}(t)$ according to Eq. 9;
  for all $a \in \mathcal{Z}_{u(t)}(t)$ **do**
    $p_a(t) = \boldsymbol{x}_{u(t),a}(t)^T \tilde{\boldsymbol{L}}(t) \tilde{\boldsymbol{s}}_{u(t)}(t)$;
  end for
  Choose arm $a(t) = \arg\max_{a \in \mathcal{Z}_{u(t)}(t)} p_a(t)$ and observe a real-valued payoff $r_{a(t)}(t)$;
  while Not Convergence **do**
    Update $\Sigma(t+1)$ and $\hat{\boldsymbol{L}}(t+1)$ according to Eq. 10 and Eq. 11;
    Update $\boldsymbol{A}_{u(t)}(t+1)$ according to [21];
    Update $\boldsymbol{\Psi}_{u(t)}(t+1)$ and $\boldsymbol{s}_{u(t)}(t+1)$ according to Eq. 12 and Eq. 13;
  end while
end for

of the Thompson Sampling for the contextual bandit to implement the exploration of parameters.

In our Thompson Sampling solution, at every step t, we sample $\tilde{\boldsymbol{L}}(t)$ and $\tilde{\boldsymbol{s}}_{u(t)}(t)$ from the corresponding prior distribution, as followed:

$$\text{vec}(\tilde{\boldsymbol{L}}(t)) \sim \mathcal{N}(\text{vec}(\hat{\boldsymbol{L}}(t)), \upsilon_L^2 \Sigma(t)^{-1}), \tag{8}$$

$$\tilde{\boldsymbol{s}}_{u(t)}(t) \sim \mathcal{N}(\hat{\boldsymbol{s}}_{u(t)}(t), \upsilon_S^2 \boldsymbol{\Psi}_{u(t)}(t)^{-1}), \tag{9}$$

where we could give the means and covariances in an alternating manner. With $\boldsymbol{S}$ fixed, we get $\Sigma(t)$ and $\hat{\boldsymbol{L}}(t)$:

$$\Sigma(t) = \lambda_L \boldsymbol{I}^{dk} + \sum_{u=1}^{n} (\hat{\boldsymbol{s}}_u(t-1)\hat{\boldsymbol{s}}_u(t-1)^T) \otimes (\boldsymbol{X}_u(t-1)\boldsymbol{X}_u(t-1)^T), \tag{10}$$

$$\hat{\boldsymbol{L}}(t) = \Sigma(t)^{-1}[\sum_{u=1}^{n} \text{vec}(\boldsymbol{X}_u(t-1)\boldsymbol{r}_u(t-1)\hat{\boldsymbol{s}}_u(t-1)^T)]. \tag{11}$$

Similarly, with $\boldsymbol{L}$ and $\boldsymbol{A}_{u(t)}$ fixed, we get $\boldsymbol{\Psi}_{u(t)}(t)$ and $\boldsymbol{s}_{u(t)}(t)$:

$$\boldsymbol{\Psi}_{u(t)}(t) = \hat{\boldsymbol{A}}_{u(t)}(t) + (\boldsymbol{X}_{u(t)}(t-1)^T \hat{\boldsymbol{L}}(t-1))^T \cdot (\boldsymbol{X}_{u(t)}(t-1)^T \hat{\boldsymbol{L}}(t-1)), \tag{12}$$

$$\hat{\boldsymbol{s}}_{u(t)}(t) = \boldsymbol{\Psi}_{u(t)}(t)^{-1} (\boldsymbol{X}_{u(t)}(t-1)^T \hat{\boldsymbol{L}}(t-1))^T \boldsymbol{r}_{u(t)}(t-1), \tag{13}$$

where $\boldsymbol{X}_{u(t)}(t-1)$ and $\boldsymbol{r}_{u(t)}(t-1)$ are features and rewards collected by user $u(t)$ until time step $t-1$.

ClexB always chooses the arm which gives the largest prediction, using the parameters drawn from the multivariate distribution, shown as below:

$$a(t) = \arg\max_{a \in \mathcal{Z}_{u(t)}(t)} \boldsymbol{x}_{u(t),a}(t)^T \tilde{\boldsymbol{L}}(t) \tilde{\boldsymbol{s}}_{u(t)}(t). \tag{14}$$

Then, the reward $r_{a(t)}(t)$ is observed. We use this feedback to update the parameters, calculating the posterior distribution $\mathcal{N}(\text{vec}(\hat{\boldsymbol{L}}(t+1)), \upsilon_L^2 \Sigma(t+1)^{-1})$ and $\mathcal{N}(\hat{\boldsymbol{s}}_{u(t)}(t+1), \upsilon_S^2 \boldsymbol{\Psi}_{u(t)}(t+1)^{-1})$, with Eq. 10, Eq. 11, Eq. 12 and Eq. 13. And a new $\hat{\boldsymbol{A}}_{u(t)}(t+1)$ can also be solved according to [21]. This update procedure goes until convergence. As we learn an accurate $\boldsymbol{L}$, we can have a better estimation of $\boldsymbol{S}$, vice versa. The optimization of $\boldsymbol{L}$ and $\boldsymbol{S}$ benefit each other. The details of our algorithm ClexB are shown in Algorithm 1.

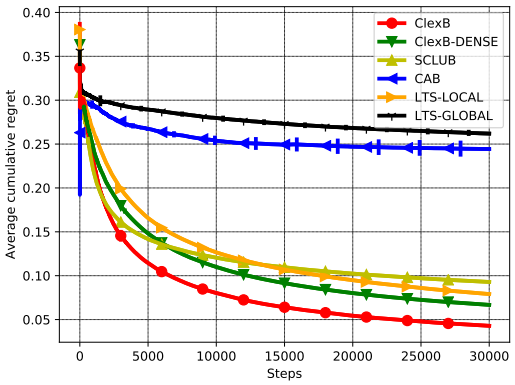
Reasonable exploration of the parameters helps to accelerate finding accurate interest clusters and further aid the learning of user interests. In the real online environment, it's always too time-consuming to optimize ClexB to convergence at each step. Thus, in practice, we just run a few iterations of parameter updates. In each round, we can randomly select several α_{ui} to renew. It is similar to the idea of gradient descent algorithm and also results in an excellent performance in the experiments.

4 EXPERIMENTS

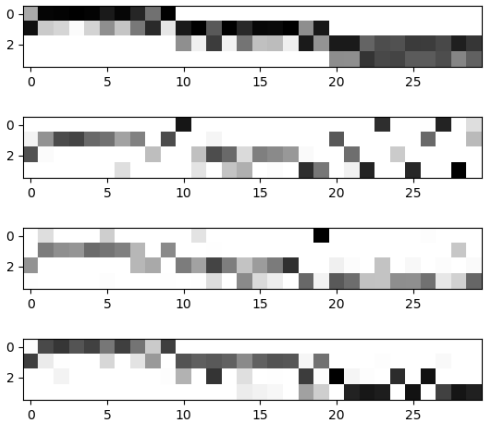
In this section, we verify ClexB's empirical performance on both synthetic and real-world datasets. We utilize the "replay" offline evaluation method introduced in [12]. For the synthetic datasets, we utilize the cumulative regret defined in Eq. 1 for evaluating performances. For the real-world datasets, without knowing the optimal arms, we use the cumulative rewards instead. We compare ClexB with five baseline models which are state-of-the-art clustering-of-bandit methods or representative online models:

- **ClexB-DENSE** stands for the dense version of ClexB. It utilizes no sparsity characteristic of the parameters, which means ClexB-DENSE doesn't make more assumptions on the prior of $\boldsymbol{S}$.
- **CAB** [7] estimates neighbors according to the predictions of users for each specific item. Users behaving similarly in the face of one item are regarded to be neighbors (in the same cluster). Neighbors take actions as well as update together.
- **SCLUB** [13] is a graph-based method. It defines update, split, and merge operations to maintain user clusters, according to user's inferences and frequency.
- **LTS-LOCAL** maintains a local LTS [3] model for each user. These separate models don't share any knowledge with each other.
- **LTS-GLOBAL** uses only one global LTS model to serve all users. And, feedback obtained by all users is used to update this global model altogether.

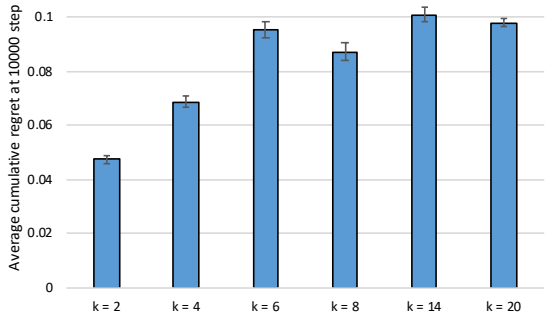
At the very beginning, ClexB and ClexB-DENSE need a 500-step warm-start to accumulate data, then utilize the Singular Value Decomposition (SVD) method for parameter initialization. During the warm-start period, ClexB and ClexB-DENSE degenerate into LTS-LOCAL. Also, to get a reliable result, all experiments are averaged for ten runs. For fair comparisons, we align the exploration of all algorithms. At first, we fix the exploration parameter



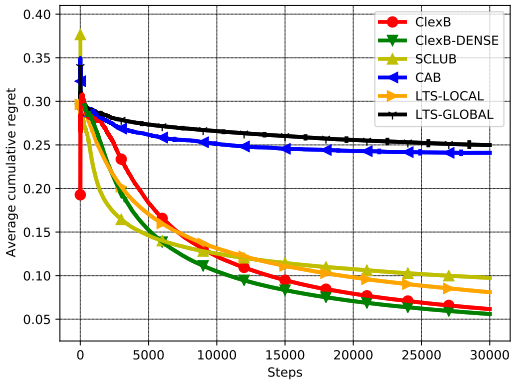
(a) Performance of ClexB on the synthetic dataset that is consistent with the sparsity assumption. It shows the changes of the average cumulative regret. As the step growing, the model with the smallest regret wins. ClexB outperforms other baseline models because it considers the uncertainties of both the user interest and user clustering. In addition, ClexB matches the sparsity assumption.



(b) Sparsity patterns of S captured by ClexB with different degrees of exploration at the step of 5000 in the synthetic experiment. From top to bottom, the first one is the original sparsity pattern. In sequence, with the other hyper-parameters staying the same, we have the sparsity patterns captured by ClexB with $v_S^2 = 0.0001$, $v_S^2 = 0.1$ and $v_S^2 = 100$. We find that the more it explores, the more accurate sparsity pattern ClexB can catch.



(c) Average cumulative regret of ClexB with different k at 1,0000 step. The other hyper-parameters remain unchanged in the synthetic experiment. The parameter k used to generate the original data and the model hyper-parameter k are set identical. A smaller k stands for a simpler parameter structure, which is easier to learn. Thus, ClexB could converge much faster.



(d) Performance of ClexB when the synthetic data conflicts with our sparsity assumption. We set a small value for σ and want to broaden the users' interest coverage. However, in the beginning, ClexB still tries to explore the sparse structure of parameters. In the end, ClexB converges to a close position to ClexB-DENSE, yet not as good as ClexB-DENSE.

Figure 2: Experiments on Synthetic Datasets.

of Thompson Sampling algorithms, *i.e.*, v^2 to be 0.1. Then, we determine the exploration parameter of UCB algorithms by aligning the performance of LinUCB-LOCAL with LTS-LOCAL. The exploration parameters of ClexB and ClexB-DENSE are aligned with LTS-LOCAL, while the exploration parameters of CAB and SCLUB are aligned with LinUCB-LOCAL. We split part of our datasets as the auxiliary datasets and perform these operations on them. Using the auxiliary datasets, we also tune all our hyper-parameters. Especially in real-world experiments, we need to try different k and σ to find the proper potential structure of data.

In the experiment of synthetic datasets, we first construct a dataset that meets our sparsity assumptions. We show that ClexB can precisely catch the sparsity patterns and outperform the baseline models. Then, in the experiment of two real-world datasets, ClexB is proven to surpass the baselines as well.

4.1 Synthetic Experiments

In the synthetic experiments, we mainly demonstrate the performance of ClexB in the artificial environment that conforms to our assumption. The data construction processes are as followed. First, we generate the matrix $\boldsymbol{L} \in \mathbb{R}^{20\times 4}$ as the parameter vectors of user clusters, with each entry randomly drawn *i.i.d.* from a Gaussian distribution, *i.e.*, $\mathcal{N}(0, \sigma)$ and $\sigma \sim \mathcal{U}(0, 1)$. Then, we generate 30 users and their parameter vectors $\boldsymbol{S} \in \mathbb{R}^{30\times 4}$, by randomly drawing each dimension from a Gaussian distribution as well. Next, given a specific sparsity, *i.e.* 0.5, we randomly replace $\boldsymbol{s}_u$'s entries with zero.

For each user, we generate 500 arms with context $\boldsymbol{X}_u \in \mathbb{R}^{500\times 20}$. We draw each context dimension from a Gaussian distribution in the same way. Then, we normalize $\boldsymbol{L}$, $\boldsymbol{s}_u$, and $\boldsymbol{x}_u$. Finally, the reward of each arm $\boldsymbol{x}_u$ is generated by $r_u = \boldsymbol{x}_u^T \boldsymbol{L}\boldsymbol{s}_u + \eta$, where η is an additive random Gaussian noise. We assume ClexB and ClexB-DENSE have already known the potential $\boldsymbol{k}$.

At each step t, we randomly choose one user to serve. All algorithms need to choose one arm from the whole arm pool of this user. In Fig. 2(a), we plot the cumulative regret *w.r.t.* the number of steps. Apparently, ClexB outperforms all the other algorithms on synthetic data. The reason is that ClexB fits our assumption perfectly. However, the other five baseline models violate the assumption more or less. ClexB-DENSE doesn't utilize the sparsity pattern of parameter $\boldsymbol{S}$. CAB maintains a single LinUCB policy for each user. Policies of similar users make recommendations together. It is useful when data is insufficient in the first few steps. However, CAB also updates the policies using similar users' rewards. It conflicts with the generation process of data and introduces additional noise. Thus, CAB obtains an inaccurate estimation of user interest. SCLUB constructs the user relationship with the graph, which moves away from our synthetic data. LTS-LOCAL neglects the parameter structure, hence converging much more slowly without knowledge sharing. And LTS-GLOBAL just shares all the feedback and loses personalization.

In light of our observations, we find that the more it explores, the more accurate sparsity pattern ClexB can capture. The reason is that ClexB must meet enough data to estimate the precise sparsity pattern. Fig. 2(b) illustrates the sparsity pattern of $\boldsymbol{S}$ that ClexB acquires at the step of 5,000. The abscissa represents 30 users, and the ordinate represents four dimensions of $\boldsymbol{s}_u$. With a more massive exploration, ClexB can collect more samples for the estimation of $\boldsymbol{A}_u$, thus learn a better sparsity pattern of $\boldsymbol{s}_u$. Shown as Fig. 2(c), we also illustrate the performance of ClexB when k changes, while other hyper-parameters stay the same. The parameter k used to generate the original data and the model hyper-parameter k are set identical. As k decreasing, ClexB converges faster and faster. The reason is that, with a small k value, ClexB could transfer knowledge through a simpler structure and explore per cluster using more data, relatively. In addition, a larger k value could add more computation cost. Therefore, in reality, we need lots of trials to find the proper value for the hyper-parameter k. Another experiment demonstrates the situation that the dataset conflicts our assumption. It means that the potential parameter $\boldsymbol{S}$ is dense. Fig. 2(d) shows that ClexB stays very close to ClexB-DENSE but is still inferior to ClexB-DENSE. ClexB spends extra effort exploring the sparse parameter structure at the beginning, even though we set a relatively small value for σ and want ClexB to focus on a broader coverage of user interests.

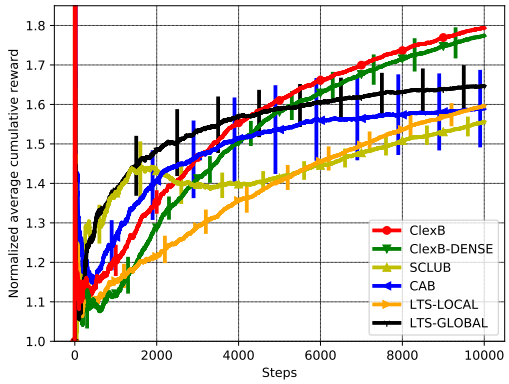
4.2 Real-World Experiments

In this subsection, we show the results of two real-world experiments. The first experiment utilizes a public dataset, *i.e.*, LastFM [2], and the second experiment verifies the performance of ClexB on the dataset collected from a mobile social application, *i.e.*, WeChat [3]. These experiments show that ClexB could capture and take advantage of the sparse pattern of user interests in practical scenarios.

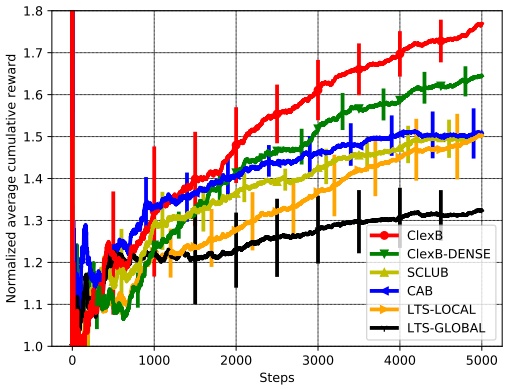
[2] www.grouplens.org/node/462
[3] http://www.wechat.com

The LastFM dataset is publicly offered by Last.fm, a big music website of the United Kingdom. This dataset is about users listening to artists, containing tags and social network information. It contains 1,892 users and 17,632 artists. At first, we cluster the original users into 30 user clusters. These user clusters are obtained by Gralus [6] using the users' social network as input. Thus, users in the same user cluster u maintain a common $\boldsymbol{s}_u$. This process accelerates the experiment's progress. All the algorithms learn on the same data settings. Therefore, in our opinion, the comparison is fair. Our data processing procedures follow the description in [7]. First, we use the term frequency-inverse document frequency (tf-idf) [22] and principal components analysis (PCA) [2] techniques sequentially to process the tags of artists. It constructs a 20-dimensional context for each artist. Then, we construct binary rewards according to whether a user listened to the artist at least once. If he did, the reward is 1; otherwise, the reward is 0. At each step t, we randomly select a user $u(t)$ and construct an arm pool $\mathcal{Z}_{u(t)}(t)$, which contains 100 arms. An arm pool comprises at least one positive arm in terms of the served user. In Fig. 3(a), we show that ClexB has an excellent performance on the LastFM dataset. As we can see, LTS-GLOBAL nearly dominates the whole experiment progress, until ClexB and ClexB-DENSE catch up and surpass it. Compared with ClexB-DENSE, ClexB benefits from its sparsity property. In terms of SCLUB and CAB, ClexB and ClexB-DENSE explore not only the user interest but also the user clustering, thus finding a better optimal solution. However, ClexB also sacrifices more in the beginning.

The WeChat dataset is extracted from the real-world news RecSys. This dataset consists of the logs generated by an online policy. The logs record the events that articles were shown to users, and that users read some of the articles. Besides, both users and articles have features. The original features of each log are the combination of users' and articles' features. Besides, the features have been preprocessed via data masking to protect users' privacy. After feature engineering, *e.g.*, one-hot encoding, the dimension of features is about several million. For each user, we regard the articles shown to him as the arms. If this user has clicked the article, the reward for pulling this arm is 1. Otherwise, the reward is 0. And the arm context $\boldsymbol{X}$ consist of both user and article features. Similarly, we adopt the PCA technique to reduce the context dimension to $d = 20$. And the horizontal normalization is applied to the arm context. We also cluster users into $n = 30$ user clusters by applying K-means on the user features to accelerate the process. At each step t of the experiment, we randomly choose a user $u(t)$ for models to serve and construct an arm pool $\mathcal{Z}_{u(t)}(t)$ with 100 relative arms. Also, the arm pool must contain at least one positive arm. Each model chooses one arm $a(t)$ to pull and receives a reward. In Fig. 3(b), we plot normalized cumulative reward *w.r.t.* the number of steps t. The cumulative reward is normalized by the random model to protect the privacy of the industrial dataset. As we can see, ClexB dominates the competition. CAB and SCLUB perform well in the beginning because of the knowledge sharing among clusters. However, they are overtaken by ClexB and ClexB-DENSE at last because of the lack of exploration about user clustering.



(a) Performance of ClexB on the LastFM dataset. The model that obtains the most rewards has the best performance. As we can see, ClexB outperforms the baseline models at last, although it sacrifices a lot for exploration in the beginning.



(b) Performance of ClexB on the industrial dataset. Similarly, ClexB also outperforms all the baselines. The good performances of ClexB in the real-world experiment indicate that ClexB could successfully capture the low-rank data structures.

Figure 3: Experiments on Industrial Datasets.

5 REGRET ANALYSIS

In this section, we briefly prove that the regret, which ClexB incurs until the variances become small enough, can be small enough as well. The core of our proof is to show that $\boldsymbol{x}_{u,a}(t)^T \hat{\boldsymbol{L}}(t)\hat{\boldsymbol{s}}_u(t)$ is concentrated around its respective mean $\boldsymbol{x}_{u,a}(t)^T \boldsymbol{L}\boldsymbol{s}_u$. It is not a jointly convex problem if we optimize $\boldsymbol{L}$ and $\boldsymbol{S}$ together. And analyzing the convergence to the actual parameters in such a non-convex problem is beyond the focus of this paper. Therefore, we investigate by solving them separately. Fixing $\boldsymbol{S}$, Lemma 5.1 illustrates the concentration result of $\boldsymbol{x}_{u,a}(t)^T \hat{\boldsymbol{L}}(t)\boldsymbol{s}_u$:

Lemma 5.1. *For any $t \geq, 0 < \delta < 1$, suppose $\|\boldsymbol{L}\|_F \leq 1$, $\|\boldsymbol{s}_u\|_2 \leq 1$, $\|\boldsymbol{x}_{u,a}\|_2 \leq 1$ and user $u(t)$ is randomly served from n users at every time step t, with probability at least $1-\delta$,*

$$\begin{aligned}&\left|\boldsymbol{x}_{u(t),a}(t)^T \hat{\boldsymbol{L}}(t)\boldsymbol{s}_{u(t)} - \boldsymbol{x}_{u(t),a}(t)^T \boldsymbol{L}\boldsymbol{s}_{u(t)}\right| \\ &\leq (\sigma\sqrt{dk\log\frac{t}{\delta}} + 1)\cdot \|\boldsymbol{s}_{u(t)} \otimes \boldsymbol{x}_{u(t),a}(t)\|_{\Sigma(t-1)^{-1}}.\end{aligned} \tag{15}$$

Proof. First let's define, for all a,

$$\begin{aligned}\xi^L(t) &= \sum_{\tau=1}^{t} \boldsymbol{s}_{u(\tau)} \otimes \boldsymbol{x}_{u(\tau),a}(\tau)\eta_\tau, \\ &= \sum_{\tau=1}^{t} \boldsymbol{s}_{u(\tau)} \otimes \boldsymbol{x}_{u(\tau),a}(\tau) \\ &\quad \cdot (r_{u(\tau),a} - \boldsymbol{s}_{u(\tau)}^T \otimes \boldsymbol{x}_{u(\tau),a}(\tau)^T \mathrm{vec}(\boldsymbol{L})).\end{aligned} \tag{16}$$

Then we can have

$$\hat{\boldsymbol{L}}(t) - \boldsymbol{L} = \Sigma(t-1)^{-1}(\xi^L(t-1) - \mathrm{vec}(\boldsymbol{L})), \tag{17}$$

and with probability at least $1-\delta$,

$$\|\xi^L(t-1)\|_{\Sigma(t-1)^{-1}} \leq \sigma\sqrt{dk\log\frac{t}{\delta}}. \tag{18}$$

Next, we could give the concentration result of $\boldsymbol{x}_{u(t),a}(t)^T \hat{\boldsymbol{L}}(t)\boldsymbol{s}_{u(t)}$, with probability at least $1-\delta$,

$$\begin{aligned}&\left|\boldsymbol{x}_{u(t),a}(t)^T \hat{\boldsymbol{L}}(t)\boldsymbol{s}_{u(t)} - \boldsymbol{x}_{u(t),a}(t)^T \boldsymbol{L}\boldsymbol{s}_{u(t)}\right| \\ &= \left|\boldsymbol{s}_{u(t)}^T \otimes \boldsymbol{x}_{u(t),a}(t)^T \Sigma(t-1)^{-1}(\xi^L(t-1) - \mathrm{vec}(\boldsymbol{L}))\right|, \\ &\leq \|\xi^L(t-1) - \mathrm{vec}(\boldsymbol{L})\|_{\Sigma(t-1)^{-1}} \|\boldsymbol{s}_{u(t)} \otimes \boldsymbol{x}_{u(t),a}(t)\|_{\Sigma(t-1)^{-1}}, \\ &\leq (\sigma\sqrt{dk\log\frac{t}{\delta}} + \|\boldsymbol{s}_{u(t)} \otimes \boldsymbol{x}_{u(t),a}(t)\|_{\Sigma(t-1)^{-1}}) \\ &\cdot \|\boldsymbol{s}_{u(t)} \otimes \boldsymbol{x}_{u(t),a}(t)\|_{\Sigma(t-1)^{-1}}, \\ &\leq (\sigma\sqrt{dk\log\frac{t}{\delta}} + 1)\cdot \|\boldsymbol{s}_{u(t)} \otimes \boldsymbol{x}_{u(t),a}(t)\|_{\Sigma(t-1)^{-1}}.\end{aligned} \tag{19}$$

□

Similarly, with $\boldsymbol{L}$ and $\boldsymbol{A}_u$ fixed, Lemma 5.2 illustrates the concentration result of $\boldsymbol{x}_{u,a}(t)^T \boldsymbol{L}\hat{\boldsymbol{s}}_u(t)$:

Lemma 5.2. *For any $t \geq, 0 < \delta < 1$, suppose $\|\boldsymbol{L}\|_F \leq 1$, $\|\boldsymbol{s}_u\|_2 \leq 1$, $\|\boldsymbol{x}_{u,a}\|_2 \leq 1$ and user $u(t)$ is randomly served from n users at every time step t, with probability at least $1-\delta$,*

$$\begin{aligned}&\left|\boldsymbol{x}_{u(t),a}(t)^T \boldsymbol{L}\hat{\boldsymbol{s}}_{u(t)}(t) - \boldsymbol{x}_{u(t),a}(t)^T \boldsymbol{L}\boldsymbol{s}_{u(t)}\right| \\ &\leq (\sigma\sqrt{k\log\frac{\bar{\alpha}_{u(t)} + (t/n)}{\delta\ {}^{s_{\alpha_{u(t)}}}\sqrt{\alpha_{u(t)}}}} + \|\boldsymbol{\beta}_{u(t)}\|_2) \\ &\cdot \|\boldsymbol{x}_{u(t),a}(t)\boldsymbol{L}\|_{\Psi_{u(t)}(t-1)^{-1}},\end{aligned} \tag{20}$$

where $s_{\alpha_{u(t)}}$, $\bar{\alpha}_{u(t)}$ and ${}^{s_{\alpha_{u(t)}}}\sqrt{\alpha_{u(t)}}$ are the number, arithmetic mean and geometric mean of none-zero elements of $\boldsymbol{\alpha}_{u(t)}$ respectively. And $\boldsymbol{\beta}_{u(t)}$ is a vector whose ith element is $\boldsymbol{\alpha}_{u(t),i}$ if $\boldsymbol{s}_{u(t),i}$ is none-zero, otherwise 0.

Proof. Following a similar procedure as the proof of Lemma 5.1, we can also have the concentration result of $\boldsymbol{x}_{u(t),a}(t)^T \boldsymbol{L}\,\hat{\boldsymbol{s}}_{u(t)}(t)$. Firstly, we define, for all a,

$$\begin{aligned}\xi^S(t) &= \sum_{\tau=1}^{t} \boldsymbol{L}^T \boldsymbol{x}_{u(\tau),a}(\tau)\eta_\tau, \\ &= \sum_{\tau=1}^{t} \boldsymbol{L}^T \boldsymbol{x}_{u(\tau),a}(\tau)(r_{u(\tau),a} - \boldsymbol{x}_{u(\tau),a}(\tau)^T \boldsymbol{L}\boldsymbol{s}_{u(\tau)}).\end{aligned} \tag{21}$$

Then and with probability $1-\delta$,

$$\begin{aligned}&\left|\boldsymbol{x}_{u(t),a}(t)^T \boldsymbol{L}\hat{\boldsymbol{s}}_{u(t)}(t) - \boldsymbol{x}_{u(t),a}(t)^T \boldsymbol{L}\boldsymbol{s}_{u(t)}\right| \\ &= \left|\boldsymbol{x}_{u(t),a}(t)^T \boldsymbol{L}\boldsymbol{\Psi}_{u(t)}(t-1)^{-1}(\xi^S(t-1) - \boldsymbol{s}_{u(t)})\right|, \\ &\leq \|\xi^S(t-1) - \boldsymbol{s}_{u(t)}\|_{\boldsymbol{\Psi}_{u(t)}(t-1)^{-1}} \|\boldsymbol{L}^T \boldsymbol{x}_{u(t),a}(t)\|_{\boldsymbol{\Psi}_{u(t)}(t-1)^{-1}}, \\ &\leq (\sigma\sqrt{k\log\frac{\bar{\alpha}_{u(t)} + (t/n)}{\delta\ s_{\alpha_{u(t)}}\sqrt{\alpha_{u(t)}}}} + \|\boldsymbol{s}_{u(t)}\|_2) \\ &\cdot \|\boldsymbol{x}_{u(t),a}(t)\boldsymbol{L}\|_{\boldsymbol{\Psi}_{u(t)}(t-1)^{-1}}, \\ &\leq (\sigma\sqrt{k\log\frac{\bar{\alpha}_{u(t)} + (t/n)}{\delta\ s_{\alpha_{u(t)}}\sqrt{\alpha_{u(t)}}}} + \|\boldsymbol{\beta}_{u(t)}\|_2) \\ &\cdot \|\boldsymbol{x}_{u(t),a}(t)\boldsymbol{L}\|_{\boldsymbol{\Psi}_{u(t)}(t-1)^{-1}},\end{aligned} \tag{22}$$

where we have an assumption that, at each time step t, n users are randomly served. Thus, each user u has a cumulative step t/n. □

Theorem 5.3. *For any $t \geq, 0 < \delta < 1$, suppose $\|\boldsymbol{L}\|_F \leq 1$, $\|\boldsymbol{s}_u\|_2 \leq 1$, $\|\boldsymbol{x}_{u,a}\|_2 \leq 1$ and user $u(t)$ is randomly served from n users at every time step t, with probability at least $1-\delta$, the total regret for ClexB in time T can is bounded by*

$$\begin{aligned}R(T) = O(&\sum_{u=1}^{n}(min\{\sqrt{s_{\alpha_u}}, \sqrt{\log N_u}\} \\ &\cdot [\sigma\sqrt{s_{\alpha_u}\log\frac{(T/n)^2\bar{\alpha}_u + (T/n)^3}{\delta\ s_{\alpha_u}\sqrt{\alpha_u}}} + \|\boldsymbol{\beta}_u\|_2]) \\ &+ dk\sqrt{T}\cdot(min\{\sqrt{dk}, \sqrt{\log\sum_{u=1}^{n} N_u}\}) \\ &\cdot (\log T + \sqrt{\log T\log\frac{1}{\delta}}))).\end{aligned} \tag{23}$$

Proof. With Lemma 5.1 and Lemma 5.2, we can prove Theorem 5.3. The regret $R(T)$, until time T, consists of two parts, $R^L(T)$ and $R^S(T)$. Using Lemma 5.1, fixing $\boldsymbol{S}$, with at least probability $1-\delta$, we can bound the first part regret as:

$$\begin{aligned}R^L(T) = O(&dk\sqrt{T}\cdot(\min\{\sqrt{dk}, \sqrt{\log\sum_{u=1}^{n} N_u}\}) \\ &\cdot (\log T + \sqrt{\log T\log\frac{1}{\delta}}))).\end{aligned} \tag{24}$$

Similarly, using Lemma 5.2, fixing $\boldsymbol{L}$, with at least probability $1-\delta$, we bound the second part as:

$$\begin{aligned}R^S(T) = O(&\sum_{u=1}^{n}(\min\{\sqrt{s_{\alpha_u}}, \sqrt{\log N_u}\} \\ &\cdot [\sigma\sqrt{s_{\alpha_u}\log\frac{(T/n)^2\bar{\alpha}_u + (T/n)^3}{\delta\ s_{\alpha_u}\sqrt{\alpha_u}}} + \|\boldsymbol{\beta}_u\|_2])).\end{aligned} \tag{25}$$

Thus, combining those two parts of regrets, we can bound the final regret as, with at least probability $1-\delta$:

$$R(T) = R^L(T) + R^S(T). \tag{26}$$

□

According to Lemma 5.1, Lemma 5.2, and Theorem 5.3, ClexB enjoys the same order of sublinear regret as LTS [3], *i.e.* $O(\sqrt{T}\log\sqrt{T})$. As we can see, ClexB achieves better performances in the experiments as well as maintains the original order of regret. One important reason is that ClexB need to explore smaller parameter space, compared with baseline models. As the context dimension d becoming larger, ClexB will converge more slowly according to our theoretical analysis. The high dimensionality of context challenges the majority of contextual bandit policies. Besides, there have been many existing works [23] trying to solve this problem. It is beyond the focus of this paper. In our opinion, ClexB will always have a better performance than the other baselines after a proper exploration.

6 CONCLUSIONS

In this paper, we point out the shortage of the existing cluster-of-bandit algorithms for the recommendation system. They only emphasize the uncertainty of user interest and intuitively cluster users into groups. Thus, they neglect the uncertainty of user clustering. To address this problem, we propose a new cluster-of-bandit algorithm, *i.e.*, ClexB. ClexB designs an adaptive clustering method to share knowledge among users. Moreover, ClexB explores both user's interest and user clustering together in the online environment. Experiments and regret analysis show ClexB's superiority over the state-of-the-art algorithms. In the future, our research will focus on optimizing ClexB's computation efficiency to bring it online for a real recommendation situation and designing the non-linear version of ClexB for better performance.

REFERENCES

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. 2011. Improved algorithms for linear stochastic bandits. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*. 2312–2320.

[2] Hervé Abdi and Lynne J Williams. 2010. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics* 2, 4 (2010), 433–459.

[3] Shipra Agrawal and Navin Goyal. 2013. Thompson sampling for contextual bandits with linear payoffs. In *Proceedings of the 30th International Conference on Machine Learning*. 127–135.

[4] Nicolò Campolongo. 2017. An empirical evaluation of context aware clustering of bandits using Thompson sampling.

[5] Nicolo Cesa-Bianchi, Claudio Gentile, and Giovanni Zappella. 2013. A gang of bandits. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*. 737–745.

[6] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. 2007. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 11 (2007), 1944–1957.

[7] Claudio Gentile, Shuai Li, Purushottam Kar, Alexandros Karatzoglou, Giovanni Zappella, and Evans Etrue. 2017. On context-dependent clustering of bandits. In *Proceedings of the 34th International Conference on Machine Learning*. 1253–1262.

[8] Claudio Gentile, Shuai Li, and Giovanni Zappella. 2014. Online clustering of bandits. In *Proceedings of the 31th International Conference on Machine Learning*. 757–765.

[9] Davis Gilton and Rebecca Willett. 2017. Sparse linear contextual bandits via relevance vector machines. In *Proceedings of the 12th International Conference on Sampling Theory and Applications*. 518–522.

[10] Jaya Kawale, Hung H Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla. 2015. Efficient Thompson sampling for online matrix-factorization recommendation. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*. 1297–1305.

[11] Abhishek Kumar and Hal Daume III. 2012. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Conference on Machine Learning*.

[12] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. 661–670.

[13] Shuai Li, Wei Chen, Shuai Li, and Kwong-Sak Leung. 2019. Improved algorithm on online clustering of bandits. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 2923–2929.

[14] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. 2016. Collaborative filtering bandits. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 539–548.

[15] Bo Liu, Ying Wei, Yu Zhang, Zhixian Yan, and Qiang Yang. 2018. Transferable contextual bandit for cross-domain recommendation. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

[16] Trong T Nguyen and Hady W Lauw. 2014. Dynamic clustering of contextual multi-armed bandits. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 1959–1962.

[17] Sinno Jialin Pan, Qiang Yang, et al. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359.

[18] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic matrix factorization. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*. 1257–1264.

[19] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The Adaptive Web, Methods and Strategies of Web Personalization*. 291–324.

[20] Michael E Tipping. 2001. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* 1, Jun (2001), 211–244.

[21] Michael E Tipping and Anita C Faul. 2003. Fast marginal likelihood maximisation for sparse Bayesian models. In *Proceedings of the 19th International Workshop on Artificial Intelligence and Statistics*.

[22] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. 2008. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems* 26, 3 (2008), 13:1–13:37.

[23] Xiaotian Yu, Michael R Lyu, and Irwin King. 2017. Cbrap: Contextual bandits with random projection. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. 2859–2866.

[24] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. 2013. Interactive collaborative filtering. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 1411–1420.