# Adaptive Linear Contextual Bandits for Online Recommendations

Anonymous Author(s)

## ABSTRACT

Contextual bandit algorithms have been successfully applied to online recommender systems by dynamically optimizing a trade-off between exploration and exploitation. However, most existing approaches recommend items to a user based on the user's all previous preferences, which neglects specific characteristics of various items in online recommendations, especially for those with a variety of preferences. To tackle this problem, in this paper, we propose an adaptive user preference filtering mechanism for online recommendations. Specifically, we develop an adaptive linear upper confidence bound algorithm, i.e., *AdaLinUCB*, to serve users with contextual bandits. We propose a *preference filtering matrix* that varies with user preferences. By filtering user preferences according to various items, we recommend the item with highest score to the user adaptively. We propose a coordinate descent method to optimize the bandit parameters with an online sliding window model. Extensive experiments conducted on two benchmark datasets verify the significant improvement of AdaLinUCB against state-of-the-art baselines. In addition, we find such preference filtering matrix has explainable clustering characteristics, which helps to visualize and interpret recommendation results.

## CCS CONCEPTS

• **Information systems → Recommender systems**;

## KEYWORDS

Contextual bandits, online recommendation, filtering user preferences

**ACM Reference format:**
Anonymous Author(s). 2018. Adaptive Linear Contextual Bandits for Online Recommendations. In *Proceedings of The Web Conference 2018, Lyon, France, April 23-27, 2018 (WWW 2018),* 10 pages.
DOI: xx.xxx/xxxx

## 1 INTRODUCTION

During online recommendations, the need to probe the user's new interest and the need to satisfy the user's interest results in an explore-exploit dilemma [2, 3, 20]. Intuitively, such dilemma can be formulated as a contextual bandit problem, which consists of a series of trials in a sequential recommendation process [31]. Here a trial indicates to a task pipeline for a contextual bandit, including recommending an item to a user, receiving a reward from the user,

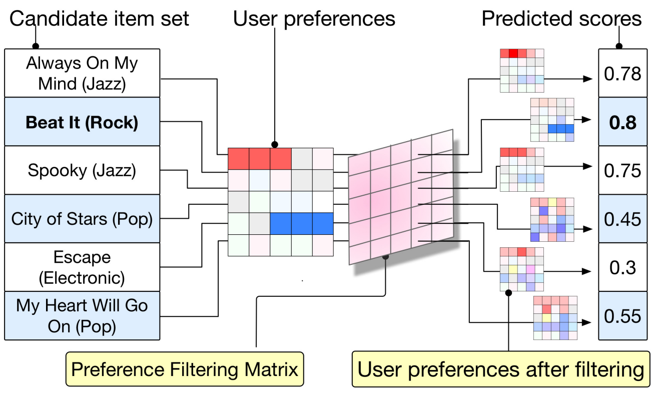


**Figure 1: An example of our online recommendation in a music portal. Recommender system recommends a song with highest score from a given candidate list. The red and blue colors refer to the user preferences on jazz and rock respectively. Other than existing bandits models, we propose a preference filtering matrix to filter user preference according to various items.**

and updating its recommendation strategy, successively. Eventually, the target of such contextual bandits recommendation is to maximize the total received reward of all trials. Recent work on contextual bandits show impressive performance in online recommendations, especially when the recommendation space is large whereas rewards are interrelated [2, 8, 19, 20].

The whole user preferences may have various characteristics referring to different items. Scoring an item using the corresponding characteristics in the whole user preferences has been proved effective in collaborative filtering [14, 36]. However, most of existing contextual bandit algorithms still make a prediction by scoring each item based on the whole user preferences, and neglect to adaptively utilizing characteristics. This inevitably results in a systematic bias, i.e., the whole user preferences have a negative influence when scoring an item.

Filtering user preferences not only improves the accuracy of the results but also improves the ranking orders of the recommendation sequence. To this end, it is necessary to propose a contextual bandit algorithm that could adaptively filter user preferences instead of using the whole user preferences for online recommendations. To tackle this problem, we consider adding a preference filtering matrix into contextual bandits and employ user preferences after filtering to score items, which can provide a more precise estimation. Shown in Figure 1, Given a music portal, we assume that a user likes rock and jazz style music. The red and blue colors refer to the user preferences on jazz and rock respectively. There are various styles of songs in the candidate item set. We add a preference filtering matrix into the existing model, and it gets six new user preferences according to different songs. Then, recommender system generates predictions by using preferences on rock to score the rocky-style song.

In this paper, we propose an adaptive linear upper confidence bound algorithm (AdaLinUCB) which uses the preference filtering matrix to filter user preferences. AdaLinUCB first filters user preferences according to different items. Then, for each item, AdaLinUCB generates the predicted score using the new user preferences. After that, AdaLinUCB selects the highest scoring item to recommend to the user, and then the user gives a feedback. Finally, AdaLinUCB uses user feedback and historical recommendation records to update user preferences and preference filtering matrix. It is challenging to learn the user preference filtering matrix, since the parameter update method of the existing contextual bandit algorithms is not suitable for this task [20, 24, 37]. Therefore, we propose a coordinate descent method to learn the user preference filtering matrix. Besides, we present an Online Sliding Window Model to dynamically manage user historical activities. It improves the accuracy of online recommendations without sacrificing efficiency.

The main contributions of this work are summarized as follows.

1. We present an AdaLinUCB algorithm to adaptively take user preferences about specific items into consideration for online recommendations.
2. We propose a coordinate descent method to optimize the bandit parameters with an online sliding window model to dynamically manage user historical activities.
3. We conduct extensive experiments on two benchmark datasets to evaluate the effectiveness of our AdaLinUCB algorithm compared with several state-of-the-art contextual bandit algorithms. We analyze the learned preference filtering matrices and find that they have explainable clustering characteristics corresponding to realistic situations.

We introduce related work in §2; in §3 we formulate our research problem. We describe our approach in §4; §5 details our experimental setup and presents the results; §6 analyzes the results; §7 concludes the paper.

## 2 RELATED WORK

To the best of our knowledge, no previous work has studied the problem of filtering user preference for contextual bandits. But there are several lines of related work: 1) contextual bandit algorithms; 2) online collaborative filtering with bandits; and 3) probability matching with bandits.

Contextual bandit algorithm is an important branch of Multi-armed bandit algorithms. Contextual bandit algorithm assumes the distributions of rewards pertaining to each item are connected by a set of parameters [6, 8, 19, 20, 34]. The setting for contextual bandit with linear rewards was first introduced in [2], where the expectation of reward for each item is assumed to be a linear function of its context vector. In the follow-up researches, Li et al. [20] proposed LinUCB to use ridge regression to compute the expected reward of each item and the corresponding confidence interval. Then, there are many variants of these models. Cesa-Bianchi et al. [8] proposed GOB.Lin that requires connected users in a network to have similar bandit parameters via a graph Laplacian based model regularization. Nguyen and Lauw [24] proposed DynUCB by introducing a dynamic clustering approach to keep a bandit customized to individual user and yet allow users to benefit from the collective set of learning items in their cluster. Wu et al. [37] proposed CoLin by explicitly modeling the underlying dependency among users via a weighted adjacency graph. This establishes a bridge to share information among different users.

There are also some recent developments that focus on online collaborative filtering and probability matching with bandits [7, 16–18, 23, 26, 27]. Zhao et al. [40] studied interactive collaborative filtering via probabilistic matrix factorization. Chapelle and Li [9] presented some empirical results and demonstrated the efficiency of Thompson sampling. May et al. [22] extended the Thompson Sampling by introducing a new algorithm, Optimistic Bayesian Sampling (OBS), in which the probability of recommending an item increases with the uncertainty in the estimate of the reward. Kawale et al. [16] developed a Thompson sampling scheme for online matrix-factorization. Nakamura [23] developed a UCB-like strategy to perform online collaborative filtering.

Contextual bandit algorithms provide solutions to the explore/-exploit dilemma [1, 4, 10, 12, 28], which exists in many real-world applications, such as display advertisements [15, 21, 29, 32] and recommender systems [5, 20, 25, 39]. It has achieved promising performance in various application scenarios [11, 20, 38]. In this paper, we propose an AdaLinUCB algorithm to adaptively use user preferences for online recommendations. It not only improves the accuracy of the result but also improves the ranking orders of the recommendation sequence.

## 3 PRELIMINARIES

### 3.1 Problem Formulation

**Table 1: Important Notations**

| Notation | Description |
|---|---|
| $\mathcal{A}$ | the candidate item set |
| $a$ | item in candidate item set |
| $\boldsymbol{x}_{t,a}$ | the feature vector of item $a$ in the $t$-th trial |
| $\boldsymbol{\theta}_u$ | the preference parameter of user $u$ |
| $\hat{\boldsymbol{\theta}}_u$ | the estimation of user preference parameter $\boldsymbol{\theta}_u$ |
| $r_{t,a}$ | the reward of recommending the item $a$ in the $t$-th trial |
| $\hat{r}_{t,a}$ | the expected reward of recommending the item $a$ in the $t$-th trial |
| $(\boldsymbol{D}_u, \boldsymbol{b}_u)$ | $\boldsymbol{D}_u$ is a matrix, whose rows correspond to item features that are observed previously, $\boldsymbol{b}_u$ is the corresponding feedback vector |

In online recommendations, recommender system needs to continuously recommend items for users. The problem setting consists of a series of trials and could be seen as a sequential recommendation problem. For the $t$-th trial, given a candidate item set $\mathcal{A}_t$, the recommender system first selects an item $a \in \mathcal{A}_t$ to recommend to user $u$. Then, it receives a reward $r_{t,a}$. After that, the recommender system uses the reward $r_{t,a}$ to update recommendation strategy for better recommendation in next trials. The goal of the algorithm is to get the maximization of the total received rewards, $\boldsymbol{R} = \sum_{t=1}^{T} r_{t,a}$.

The expected reward $\hat{r}_{t,a}$ is determined by the item feature $\boldsymbol{x}_{t,a}$ and user preference parameter $\boldsymbol{\theta}_u$.

$$\hat{r}_{t,a} = f(\boldsymbol{x}_{t,a}, \boldsymbol{\theta}_u), \tag{1}$$

where $f()$ is a predefined prediction function. The important notations are shown in Table 1.

### 3.2 Standard Contextual Bandits

In a standard contextual bandit problem, the reward of each item is assumed to be governed by bandit parameters (i.e., user preferences which are recorded by the algorithm) and context vectors of items [2, 20]. At each trial $t$, we assume the expected reward of an item $a$ is linear in its feature vector $\boldsymbol{x}_{t,a}$ ($\boldsymbol{x}_{t,a} \in \mathbb{R}^d$) with the preference parameter $\boldsymbol{\theta}_u$ from current user $u$,

$$\hat{r}_{t,a} = \boldsymbol{x}_{t,a}^\top \boldsymbol{\theta}_u. \tag{2}$$

Let $\boldsymbol{D}_u$ be a designed matrix of dimension $m \times d$ at trial $t$, whose rows correspond to $m$ items that are observed previously by user $u$, and $\boldsymbol{b}_u \in \mathbb{R}^m$ be the corresponding feedback vector, where each dimension represents the user reward. Applying ridge regression to the training data $(\boldsymbol{D}_u, \boldsymbol{b}_u)$ gets an estimate of the preferences [20],

$$\hat{\boldsymbol{\theta}}_u = (\boldsymbol{D}_u^\top \boldsymbol{D}_u + \boldsymbol{I}_{d\times d})^{-1} \boldsymbol{D}_u^\top \boldsymbol{b}_u, \tag{3}$$

where $\boldsymbol{I}_{d\times d}$ is the $d \times d$ identity matrix. When components in $\boldsymbol{b}_u$ are independent conditioned on corresponding rows in $\boldsymbol{D}_u$, Walsh et al. [33] show that with probability at least $1 - \delta$,

$$|\boldsymbol{x}_{t,a}^\top \hat{\boldsymbol{\theta}}_u - r_{t,a}| \leqslant \beta \sqrt{\boldsymbol{x}_{t,a}^\top (\boldsymbol{D}_u^\top \boldsymbol{D}_u + \boldsymbol{I}_{d\times d})^{-1} \boldsymbol{x}_{t,a}}, \tag{4}$$

where $\beta = 1 + \sqrt{ln(2/\delta)/2}$ is a constant, for any $\delta > 0$. In other words, Eq.4 gives a reasonably tight Upper Confidence Bound (UCB) [2, 3] for the expected reward of item $a$. Then, an UCB-type item-selection strategy can be derived: at each trial $t$, choose

$$a^* \overset{def}{=} \underset{a \in \mathcal{A}_t}{\arg\max} \, (\boldsymbol{x}_{t,a}^\top \hat{\boldsymbol{\theta}}_u + \beta \sqrt{\boldsymbol{x}_{t,a}^\top \boldsymbol{A}_u^{-1} \boldsymbol{x}_{t,a}}), \tag{5}$$

where $\mathcal{A}_t$ is current candidate item set, $\boldsymbol{A}_u \overset{def}{=} \boldsymbol{D}_u^\top \boldsymbol{D}_u + \boldsymbol{I}_{d\times d}$.

Algorithm 1 gives a detailed description of the linear contextual bandit algorithm and Figure 2 is the general framework of making recommendations to a single user in discrete iterations. As shown in Figure 2, the recommendation framework can be divided into three parts from left to right: item, system, and user. The item part mainly includes candidate item sets. The system part maintains user preference parameters and makes predictions about item rewards. The user part receives items which were recommended by the system part and gives a real-value feedback to system part. Specifically, for each trial in Figure 2, the item part first delivers item features to the system part. Then the recommendation system predicts rewards of items in candidate item set (in steps 8 to 10), using item features and user preference parameters. After that, recommending the item with the highest reward to user and the user will give a real evaluation of the item. Finally, the system receives user feedback and updates user preference parameters (in steps 13).

Due to the user preferences are extracted from different types of items, the problem in above standard contextual bandit algorithm is that it uses the whole user preferences for the prediction. As shown in Eq.5, when calculating the predicted reward, the algorithm uses the mixed user preferences which consist of different types of characteristics. This probably leads to suboptimal recommendations. In order to minimize this inherent noise effects, we propose an AdaLinUCB algorithm.

**Algorithm 1** Linear Contextual Bandit Algorithm

```
1:  Inputs: the parameter of exploration: $\beta \in \mathbb{R}^+$
2:  for $t = 1, 2, 3, \dots, T$ do
3:      Receive user $u$
4:      if user $u$ is new then
5:          initialize $\boldsymbol{A}_u \leftarrow \boldsymbol{I}_{d\times d}, \boldsymbol{b}_u \leftarrow \boldsymbol{0}_{d\times 1}, \boldsymbol{\theta}_u \leftarrow \boldsymbol{I}_{d\times 1}$
6:      end if
7:      Extract features of all items $a \in \mathcal{A}_t : \boldsymbol{x}_{t,a} \in \mathbb{R}^d$
8:      for all $a \in \mathcal{A}_t$ do
9:          $\hat{r}_{t,a} = \boldsymbol{x}_{t,a}^\top \boldsymbol{\theta}_u + \beta \sqrt{\boldsymbol{x}_{t,a}^\top \boldsymbol{A}_u^{-1} \boldsymbol{x}_{t,a}}$
10:     end for
11:     Select $a^* = \arg\max_{a \in \mathcal{A}_t} (\hat{r}_{t,a})$
12:     Observe the real-valued $r_{t,a^*}$ from user $u$
13:     Update user preference parameters:
            $\boldsymbol{A}_u \leftarrow \boldsymbol{A}_u + \boldsymbol{x}_{t,a^*} \boldsymbol{x}_{t,a^*}^\top$
            $\boldsymbol{b}_u \leftarrow \boldsymbol{b}_u + r_{t,a^*} \boldsymbol{x}_{t,a^*}$
            $\boldsymbol{\theta}_u \leftarrow \boldsymbol{A}_u^{-1} \boldsymbol{b}_u$
14: end for
```

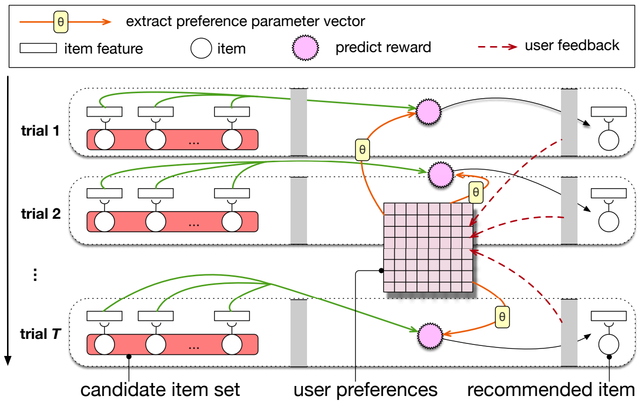


**Figure 2: The framework of contextual bandit algorithms**

## 4 ADAPTIVE LINEAR CONTEXTUAL BANDITS

In this section, we detail our proposed approach for online recommendations. We first present the adaptive linear upper confidence bound algorithm (AdaLinUCB) in §4.1. We then propose an Online Sliding Window Model to learn preference filtering matrix in §4.2.

### 4.1 AdaLinUCB

We present a new linear contextual bandit algorithm by introducing the mechanism of adaptively filtering user preferences according to specific items. Given Algorithm 1, we have user preference parameters, i.e., $\boldsymbol{\theta}_u$, $\boldsymbol{A}_u$, and $\boldsymbol{b}_u$, which are shown as follows:

$$\boldsymbol{\theta}_u = \boldsymbol{A}_u^{-1} \boldsymbol{b}_u, \tag{6}$$

$$\boldsymbol{A}_u \leftarrow \boldsymbol{A}_u + \boldsymbol{x}_{t,a^*}\boldsymbol{x}_{t,a^*}^\top, \tag{7}$$

$$\boldsymbol{b}_u \leftarrow \boldsymbol{b}_u + r_{t,a^*}\boldsymbol{x}_{t,a^*} \tag{8}$$

where $\boldsymbol{x}_{t,a}$ refers to the item feature for item $a$ in trial $t$, while $\boldsymbol{\theta}_u$ is the preference parameters for user $u$. We use a filtering matrix to filter user preferences before making predictions, which is shown in Eq 9-11.

$$\hat{r}_{t,a} = \boldsymbol{x}_{t,a}^\top \boldsymbol{\theta}_u, \tag{9}$$

$$\boldsymbol{\theta}_u = (\boldsymbol{W}_{u,x_{t,a}} \odot \boldsymbol{A}_u^{-1})\boldsymbol{b}_u, \tag{10}$$

$$\boldsymbol{W}_{u,x_{t,a}} = g(\boldsymbol{x}_{t,a})\boldsymbol{W}_u \tag{11}$$

where $\hat{r}_{t,a}$ is the expected reward. $\boldsymbol{W}_{u,x_{t,a}} \in \mathbb{R}^{d\times d}$ is the current preference filtering matrix and $\boldsymbol{W}_u \in \mathbb{R}^{d\times d}$ is the common preference filtering matrix which has been learned. Function $g()$ (see Eq. 16) is a filtering function, which is used to build the current preference filtering matrix according to item feature $\boldsymbol{x}_{t,a}$. Specifically, adaptively filtering user preferences is a general solution for contextual bandits: when user preference filtering matrix does not affect recommendations, i.e., all elements in matrix $\boldsymbol{W}_{u,x_{t,a}}$ are 1, our AdaLinUCB algorithm degenerates to those conventional linear contextual bandit algorithms.

Due to the coupling between $\boldsymbol{W}_{u,x_{t,a}}$ and $\boldsymbol{\theta}_u$ in the reward generation, we appeal to a coordinate descent algorithm to learn the preference filtering parameter $\boldsymbol{W}_{u,x_{t,a}}$ and the user preference parameter $\boldsymbol{\theta}_u$. We use the ridge regression to estimate the user preference parameter $\boldsymbol{\theta}_u$, and use the logistic regression to learn the preference filtering parameter $\boldsymbol{W}_{u,x_{t,a}}$. Specifically, the objective function of ridge regression can be written as follows [20, 35],

$$\min_{\theta_u} \sum_{t=1}^{T} (\boldsymbol{x}_{t,a}^\top \boldsymbol{\theta}_u - r_{t,a})^2 + \gamma ||\boldsymbol{\theta}_u||_2^2, \tag{12}$$

where $\gamma$ is the trade-off parameter for $\mathbf{L}2$ regularization. The objective function of logistic regression can be written as follows,

$$\min_{\boldsymbol{W}_{u,x_{t,a}}} -\frac{1}{T}\sum_{i=1}^{T}[r_{t,a} log(\boldsymbol{x}_{t,a}^\top \boldsymbol{\theta}_u) + (1-r_{t,a}) log(1-\boldsymbol{x}_{t,a}^\top \boldsymbol{\theta}_u)]. \tag{13}$$

However, maximizing the expected reward alone may result in a long term regret from not discovering a better item through exploration. Therefore, we also consider the Upper Confidence Bound (UCB) [2, 3, 35]:

$$|\hat{r}_{t,a} - r_{t,a}| \leqslant \beta\sqrt{\boldsymbol{x}_{t,a}^\top (\boldsymbol{D}_u^\top \boldsymbol{D}_u + \boldsymbol{I}_{d\times d})^{-1} \odot \boldsymbol{W}_{u,x_{t,a}} \boldsymbol{x}_{t,a}}, \tag{14}$$

where $\boldsymbol{D}_u$ is the training input, $\boldsymbol{I}_{d\times d}$ is the $d \times d$ identity matrix. $\beta$ is a parameter for the importance of exploration, for any $\delta > 0$, $\beta = 1 + \sqrt{ln(2/\delta)/2}$. To this end, we design the following item recommendation strategy by selecting the item maximizing the UCB:

$$a^* \overset{def}{=} \arg\max_{a\in\mathcal{A}_t} (\boldsymbol{x}_{t,a}^\top \boldsymbol{\theta}_u + \beta\sqrt{\boldsymbol{x}_{t,a}^\top \boldsymbol{A}_u^{-1} \odot \boldsymbol{W}_{u,x_{t,a}} \boldsymbol{x}_{t,a}}), \tag{15}$$

where $\mathcal{A}_t$ is current candidate item set, $\boldsymbol{A}_u \overset{def}{=} \boldsymbol{D}_u^\top \boldsymbol{D}_u + \boldsymbol{I}_{d\times d}$.

We refer to this linear contextual bandit algorithm as AdaLinUCB and illustrate the detailed procedure in Algorithm 2. We also describe the framework of AdaLinUCB in Figure 3. As shown in Figure 3, it extends the standard contextual bandit framework as shown in Figure 2 by introducing a preference filtering matrix. Specifically, for each user $u$, we initialize a preference filtering matrix $\boldsymbol{W}_u$, all values in $\boldsymbol{W}_u$ are 1. Simultaneously, we initialize the user preference parameters to record the characteristics of user interested items (in step 4 to 6 of Algorithm 2). For each trial in Figure 3, we first adaptively filter user preferences and calculate the predicted reward for each item in the candidate item set (in step 9 to 12 of Algorithm 2). Then, we select the item with highest predicted reward to recommend. After that, the user gives a real-valued feedback about the item (in step 14 and 15 of Algorithm 2). Finally, we use a coordinate descent method to update the user preference parameters and the preference filtering matrix (in step 16 and 17 of Algorithm 2). We develop an Online Sliding Window

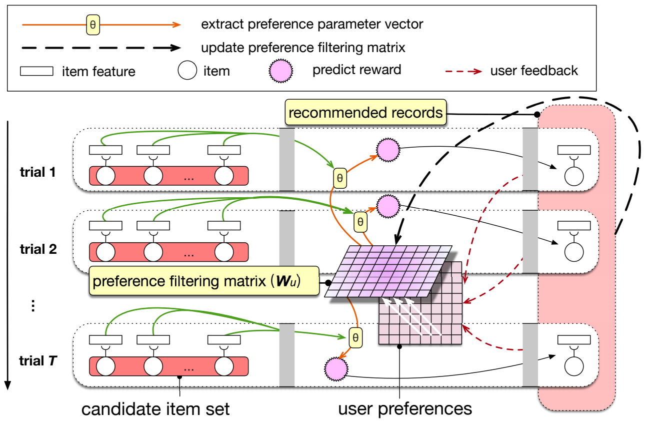


**Figure 3: Framework of adaptive linear upper confidence bound algorithm**

**Algorithm 2** AdaLinUCB

```
1: Inputs: the parameter of exploration: $\beta \in \mathbb{R}^+$
2: for $t = 1, 2, 3, \ldots, T$ do
3:     Receive user $u$
4:     if user $u$ is new then
5:         initialize $\boldsymbol{W}_u \in \mathbb{R}^{d\times d}$ (all elements are 1)
6:         initialize $\boldsymbol{A}_u \leftarrow \boldsymbol{I}_{d\times d}$, $\boldsymbol{b}_u \leftarrow \boldsymbol{0}_{d\times 1}$, $\boldsymbol{\theta}_u \leftarrow \boldsymbol{I}_{d\times 1}$
7:     end if
8:     Extract features of all items $a \in \mathcal{A}_t : \boldsymbol{x}_{t,a} \in \mathbb{R}^d$
9:     for all $a \in \mathcal{A}_t$ do
10:        $\boldsymbol{W}_{u,x_{t,a}} \leftarrow g(\boldsymbol{x}_{t,a})\boldsymbol{W}_u$
11:        $\boldsymbol{\theta}_u \leftarrow (\boldsymbol{W}_{u,x_{t,a}} \odot \boldsymbol{A}_u^{-1})\boldsymbol{b}_u$
12:        $\hat{r}_{t,a} = \boldsymbol{x}_{t,a}^\top \boldsymbol{\theta}_u + \beta\sqrt{\boldsymbol{x}_{t,a}^\top \boldsymbol{A}_u^{-1} \odot \boldsymbol{W}_{u,x_{t,a}} \boldsymbol{x}_{t,a}}$
13:    end for
14:    Select $a^* = \arg\max_{a\in\mathcal{A}_t} (\hat{r}_{t,a})$
15:    Observe the real-valued $r_{t,a^*}$ from user $u$
16:    Update user preference parameters:
           $\boldsymbol{A}_u \leftarrow \boldsymbol{A}_u + \boldsymbol{x}_{t,a^*}\boldsymbol{x}_{t,a^*}^\top$
           $\boldsymbol{b}_u \leftarrow \boldsymbol{b}_u + r_{t,a^*}\boldsymbol{x}_{t,a^*}$
17:    Update preference filtering matrix:
           $\boldsymbol{W}_u \leftarrow \boldsymbol{W}_u - \lambda_u \frac{\partial \hat{r}_{t,a}}{\partial \boldsymbol{W}_u}$
18: end for
```

Model to dynamically manage user historical activities and use Stochastic Gradient Descent (SGD) to learn the user preference filtering matrix.

### 4.2 Online Sliding Window Model

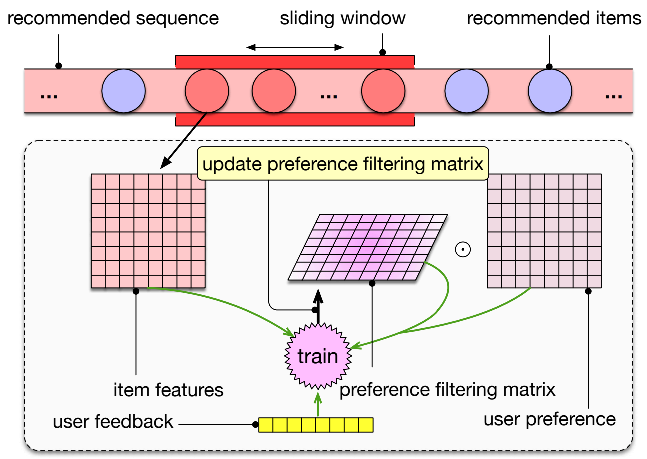


**Figure 4: Online Sliding Window Model**

So far we have introduced AdaLinUCB algorithm which could adaptively filter user preferences for online recommendations. The question then arises: how could we model the filtering function $g()$ and what kind of way to learn the preference filtering matrix without sacrificing efficiency?

We propose an Online Sliding Window Model to solve this problem as shown in Figure 4. A straightforward solution to model the filtering function is to apply a nonlinear kernel to model $g()$. Specifically, the model we adopt could be formulated as:

$$g(\mathbf{x}_{t,a}) = \sigma(\mathbf{x}_{t,a}\mathbf{x}_{t,a}^{\top}). \qquad (16)$$

We use the sigmoid function $\sigma(x) = 1/(1+e^{-x})$ to model $g()$. As shown in Figure 4, Online Sliding Window Model manages the recommended sequence and trains the preference filtering matrix in an online manner. The recommended items in the recommended sequence consist of all items in a trial, i.e., including all items which were recommended and not recommended. The size of sliding window could be adjusted according to different users. Instead of updating the matrix in every iteration with little observed data, coordinate descent can be performed in a mini-batch mode with adaptive window size. This will not affect the efficiency of online recommendations. Therefore, the Online Sliding Window Model provides an effective method of learning the filtering matrix with a low computational complexity.

## 5 EXPERIMENTS

In this section, we describe our experiments in details. §5.1 lists our research questions. §5.2 describes our experimental settings. §5.3 reports the performance comparison results and answers the research question **RQ1**. §5.4 reports the impact of Online Sliding Window Model and answers the research question **RQ2**. §5.5 reports the result of computation complexity and answers the research question **RQ3**.

### 5.1 Research questions

**RQ1** Does the AdaLinUCB method outperforms state-of-the-art baseline methods? (See §5.3)

**RQ2** What is the impact of Online Sliding Window Model for AdaLinUCB method? (See §5.4)

**RQ3** Will filtering user preferences affect the time efficiency of online recommendations? (See §5.5)

### 5.2 Experimental Setup

**Datasets.** We carry out experiments on two publicly available datasets[1]: LastFM and Delicious Bookmarks. The characteristics of the two datasets are summarized in Table 2.

**Table 2: Statistics of the evaluation datasets**

| Dataset | Users# | Tags# | Items# | (User, Item) pairs# |
|---|---|---|---|---|
| LastFM | 1892 | 9643 | 17632 | 71064 |
| Delicious | 1867 | 11619 | 69226 | 104220 |

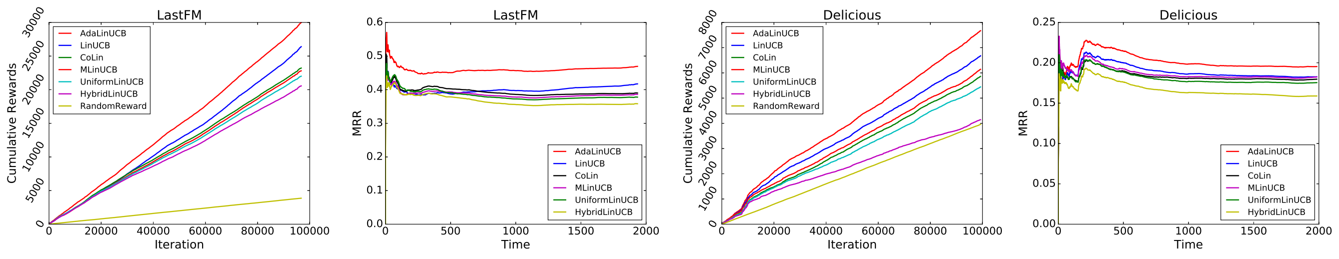
**1. LastFM**. This dataset is extracted from the music streaming service Last.fm[2] and has been widely used to evaluate contextual bandit algorithms. We use the information of "listened artists" of each user to create rewards for bandit algorithms: if a user listened to an artist at least once, the reward is 1, otherwise 0.

**2. Delicious**. This dataset is extracted from the social bookmark sharing service website Delicious[3]. We generate the rewards using the information about the bookmarked URLs for each user: the reward is 1 if the user bookmarked a particular URL, otherwise 0.

Note that the Delicious dataset is much sparser than the LastFM dataset in terms of observations per item: they contain about the same number of users, but the number of items in Delicious is almost four times larger than that in LastFM. Therefore, these two datasets provide us complementary evaluations of online recommendation in different scenarios. Following the same settings in [8, 37], we pre-processed these two datasets. First, we used all tags associated with a single item to create its TF-IDF feature vector. Then we used PCA to reduce the dimensionality of the features. In both datasets, we took the first 25 principle components to construct the context vectors, i.e., the observed feature dimension $d = 25$. We then generated the candidate item set as follows: we fixed the size of candidate item set to 25; for a particular user $u$, we picked one item from those nonzero reward items according to the whole observations in the dataset, and randomly picked the other 24 from those zero-reward items. As a result, there is only one relevant item in each item set.

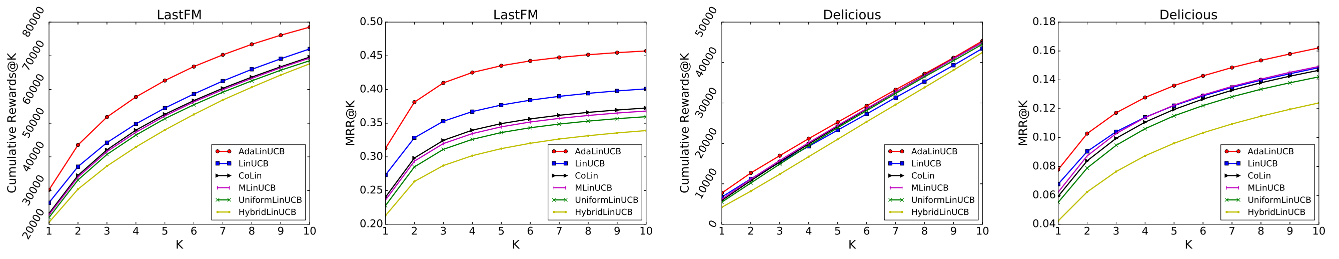
**Evaluation Protocols.** Following existing studies [24, 37], we use the *Cumulative Reward* as one evaluation metric. In addition, we

[1]https://grouplens.org/datasets/hetrec-2011/
[2]https://www.last.fm
[3]https://del.icio.us/



(a) LastFM–Cumulative Rewards (b) LastFM–MRR (c) Delicious–Cumulative Rewards (d) Delicious–MRR

**Figure 5: Performance of Cumulative Reward and MRR on the two datasets. The online sliding window size is set to 15.**



(a) LastFM–Cumulative Rewards@K (b) LastFM–MRR@K (c) Delicious–Cumulative Rewards@K (d) Delicious–MRR@K

**Figure 6: Evaluation of Top-K item recommendation where $K$ ranges from 1 to 10 on the two datasets.**

record the ranking of test items in the recommended processes. The performance of the ranked lists is judged by *Mean Reciprocal Rank* (MRR). Then, we truncate the ranked list at 10 for both metrics. The *Cumulative Reward* intuitively measures whether the test item is presented in the top-$K$ list, and the MRR measures the ranking orders of the correct items in the top-$K$ list.

**Baselines.** We perform empirical evaluations of our proposed AdaLinUCB algorithm against several related baseline algorithms.

- **CoLin** [37]. It operates at the the adjacency graph among users and shares the context and feedbacks between neighboring bandits during online update.
- **LinUCB** [20]. It selects an item based on an Upper Confidence Bound of the estimated reward with given context vectors.
- **HybridLinUCB** [20]. It extends LinUCB via a hybrid feature representation of both items and users.
- **MLinUCB** [37]. It executes LinUCB to $M$ user clusters.
- **UniformLinUCB** [37]. It executes LinUCB to all of the users.

**Parameter Settings.** We implement our AdaLinUCB algorithm based on Python and Theano. The source code of our model is available online[4]. In our experiments, our datasets are consistent with [37], so we follow their basic parameter settings. We fix the trade-off parameter $\gamma$ for **L**2 regularization to 0.3, and set the learning rate $\lambda$ to 0.03. We set the sliding window size from 1 to 15, then we record the results of cumulative rewards and item ranks. For M-LinUCB algorithm, the user cluster parameter $M$ is set to 50. For CoLin algorithm, in order to make the graph denser and make the algorithm computational feasible, we perform graph-cut to cluster users into 50 clusters (following the same setting in [8, 37]).

[4]URL suppressed. It will be shared upon publication of the paper

## 5.3 Performance Comparison (RQ1)

Figure 5 shows the performance of *Cumulative Reward* and MRR with respect to the total dataset. From the results shown in Figure 5 (a) and (c), we could see that AdaLinUCB achieves the best performance on both datasets, outperforming the state-of-the-art methods by a large margin, the improvement over LinUCB is 14.47% and 14.92% on LastFM and Delicious respectively. It's worth nothing that these two datasets are structurally different, as shown in Table 2. Delicious dataset is much sparser than LastFM dataset. Besides, the popularity of items on these two datasets differs significantly as shown in [37]: on LastFM dataset, there are a lot more popular artists whom everybody listens to than the popular websites which everyone bookmarks on Delicious dataset. Thus, the improvement on both datasets shows a strong adaptability of our algorithm. In Figure 5 (b) and (d), we show the results of MRR on both datasets. We record the ranking value of the recommended item in each iteration and use MRR to evaluate the ranking list. To make the figure more clearly, we calculate MRR in every 50 iterations. From Figure 5 (b) and (d) we can see that, AdaLinUCB demonstrates significant improvements over other baselines in terms of MRR. It outperforms LinUCB on LastFM and Delicious datasets with about 5.20% and 1.27% relative improvement respectively.

Figure 6 shows the performance of Top-$K$ recommended lists where the ranking position $K$ ranges from 1 to 10. As shown in Figure 6 (a) and (b), AdaLinUCB shows a persistent improvement on

**Table 3: Evaluation of Top-K item recommendation where K ranges from 1 to 5 on the two datasets. No-Pre stands for the result without Pre-training, Pre stands for the result with Pre-training and Result (%) stands for the improvement ratio of Pre-training *(learning rate = 0.03, training round = 5)***

| | Cumulative Reward | | | MRR | | |
|---|---|---|---|---|---|---|
| Top-$K$ | No-Pre | Pre | Result | No-Pre | Pre | Result |
| | | | **LastFM** | | | |
| 1 | 9702 | 10111 | **4.21** | 0.323 | 0.337 | **4.33** |
| 2 | 13274 | 14508 | **9.29** | 0.382 | 0.410 | **7.32** |
| 3 | 15482 | 17163 | **10.85** | 0.407 | 0.439 | **7.86** |
| 4 | 17172 | 19011 | **10.70** | 0.421 | 0.455 | **8.07** |
| 5 | 18520 | 20454 | **10.44** | 0.430 | 0.464 | **7.90** |
| | | | **Delicious** | | | |
| 1 | 2224 | 2606 | **17.17** | 0.074 | 0.086 | **16.21** |
| 2 | 3628 | 4157 | **14.58** | 0.097 | 0.112 | **15.46** |
| 3 | 4816 | 5417 | **12.47** | 0.110 | 0.126 | **14.54** |
| 4 | 6048 | 6640 | **9.78** | 0.121 | 0.136 | **12.39** |
| 5 | 7276 | 7847 | **7.84** | 0.129 | 0.144 | **11.62** |

LastFM dataset. However, as shown in Figure 6 (c) and (d), the improvement of AdaLinUCB on Delicious dataset is reduced compared with the results on LastFM dataset. Considering the sparsity of Delicious dataset, it is reasonable. Note that the trends of AdaLinUCB and LinUCB are similar on both datasets. The results of cumulative rewards and MRR keep increasing as the $K$ ranges from 1 to 10. It means that AdaLinUCB not only improves the accuracy of the result but also could improve the ranking orders of the recommendation sequence.

#### 5.3.1 *Utility of Pre-training.*

It is realizable to access the historical recommended records in most practical applications. To demonstrate the utility of pre-training for AdaLinUCB, we compare the performance of two versions of AdaLinUCB, with and without pre-training. For both datasets, we use 60000 records to do pre-training, with 30000 records to do test. We set learning rate to 0.03 and optimize it with *Stochastic Gradient Descent* (SGD). As shown in Table 3, **No-Pre** stands for the result without Pre-training, **Pre** stands for the result with Pre-training and **Result** (%) stands for the improvement ratio of Pre-training. We could clearly notice that AdaLinUCB with pre-training achieves better performance in all cases. The improvement of AdaLinUCB with pre-training is significant for both two datasets in terms of *Cumulative Reward* and MRR. This result justifies the usefulness of pre-training method for initializing AdaLinUCB.

## 5.4 Online Sliding Window (RQ2)

We use the coordinate descent method to optimize the ridge regression loss and logistic regression loss alternatively. We optimize the ridge regression loss following the existing contextual bandit algorithms [20]. In order to optimize the logistic regression loss, we propose an Online Sliding Window Model to dynamically manage user historical activities and learn the user preference filtering matrix as shown in Figure 4. Specifically, instead of updating the preference filtering matrix in every iteration, we can keep accumulating the recommended records and learning user preference filtering matrix with a reduced frequency. This would greatly reduce the computation complexity of our algorithm in large-scale deployment.

As shown in Table 4, with the increase of window size, the improvement ratio of *Cumulative Reward* rises and eventually stabilizes at around 12%. Note that when the sliding window size is set to 1, the improvement is negative on Delicious dataset. We designed validation experiments to find the reason. The results showed that we needed more training data, e.g., the sliding window size becomes larger, the result gets better as shown in Table 4. This observation verifies the differences of the two datasets, which the Delicious dataset is much sparser than LastFM dataset. This also shows that AdaLinUCB needs more observations to learn the preference filtering matrix. To illustrate the impact of different sliding window sizes for AdaLinUCB, we carried out experiments with different sliding window size. We report the performance of AdaLinUCB algorithm *w.r.t* different sliding window size in Figure 7. We could see that as the size of the sliding window increases, the results under different evaluation criteria steadily increase and gradually stabilized.

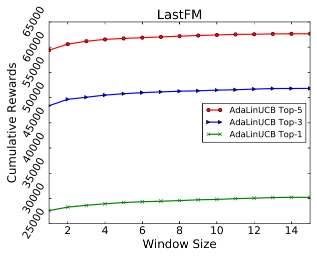
**Table 4: *Cumulative Reward* improvement performance of AdaLinUCB. Result (%) stands for the improvement ratio.**

| LastFM | | Delicious | |
|---|---|---|---|
| Window Size | Result | Window Size | Result |
| 1 | 4.53 | 1 | -1.99 |
| 2 | 7.11 | 2 | 1.71 |
| 3 | 8.47 | 3 | 4.05 |
| 4 | 9.64 | 4 | 6.23 |
| 5 | 10.37 | 5 | 7.89 |
| 6 | 11.15 | 6 | 9.67 |
| 7 | 11.56 | 7 | 10.52 |
| 8 | 12.02 | 8 | 11.27 |
| 9 | 12.63 | 9 | 12.23 |
| 10 | **12.91** | 10 | **12.85** |

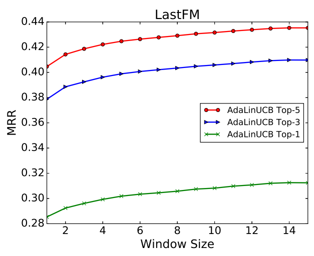
## 5.5 The Analysis of Running Time(RQ3)

There is little work on comparing computation complexity of different contextual bandit algorithms. Due to most of the contextual bandit algorithms are used to do online recommendations, it is meaningful to study the computation complexity. As shown in Algorithm 1 (see §3.2) and 2 (see §4), the contextual bandit algorithm's complexity is mainly concentrated in two stage, item selection and parameter updating. We conducted experiments to compare their runtime of item selection and parameter updating. Experiments were run on a computer with 2 core, 2.6 GHz and 8 GB RAM. In order to eliminate the influence of machine computing capability, we normalized the time consumption by the result of LinUCB.

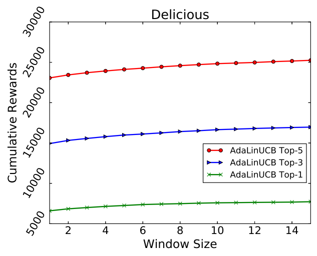
In the stage of item selection, as shown in Table 5, the time consumption of MLinUCB and UniformLinUCB are similar with LinUCB, as we mentioned in the *Baselines* (see §5.2) that MLinUCB and UniformLinUCB are the variants of LinUCB. The time consumption of AdaLinUCB is 1.69 times as long as LinUCB, the



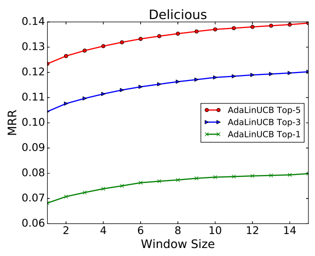
**(a) LastFM–Cumulative Rewards**



**(b) LastFM–MRR**



**(c) Delicious–Cumulative Rewards**



**(d) Delicious–MRR**

**Figure 7: Performance of Cumulative Rewards and MRR on the two datasets. The window size ranges from 1 to 15.**

**Table 5: Time consumption(*ms*) of item selection**

| Method | Item Selection | Normalized Time |
|---|---|---|
| LinUCB | 0.402 | 1 |
| MLinUCB | 0.41 | 1.02 |
| UniformLinUCB | 0.48 | 1.19 |
| **AdaLinUCB** | **0.682** | **1.69** |
| CoLin | 58.476 | 145.46 |
| HybridLinUCB | 148.748 | 370.01 |

**Table 6: Time consumption(*ms*) of parameter updating**

| Method | Parameter Updating | Normalized Time |
|---|---|---|
| LinUCB | 0.15 | 1 |
| MLinUCB | 0.186 | 1.24 |
| UniformLinUCB | 0.198 | 1.32 |
| **AdaLinUCB** | **43.684** | **291.22** |
| HybridLinUCB | 141.434 | 942.893 |
| CoLin | 1247.5 | 8316.66 |

increase comes mainly from the user preference filtering processes. Compared with CoLin and HybridLinUCB, this time consumption is negligible. From Table 5 we could see that the time consumption of CoLin and HybridLinUCB are 145 and 370 times as long as LinUCB, respectively. These two methods depend on the clustering relationship between users or the graph connection of different users which need more time to do calculation than AdaLinUCB algorithm.

In the stage of parameter updating, we report the result which the sliding window size is set to 10 as shown in Table 6. Compared with the LinUCB method, the time complexity of AdaLinUCB is high, although reaching 43*ms*, but it is acceptable for most recommender systems. The time consumption of CoLin is 8316 times as long as LinUCB when the user cluster parameter M is set to 50. And with the increase of M, the time consumption is exponential growth. Besides, we also investigate the influence of sliding window size for AdaLinUCB. Figure 8 shows the time consumption of AdaLinUCB varies with different sliding window sizes. As the sliding window size becomes larger, the time consumption increases linearly. We could conclude from the results that its computational complexity is linear with respect to the number of sliding window sizes for AdaLinUCB.

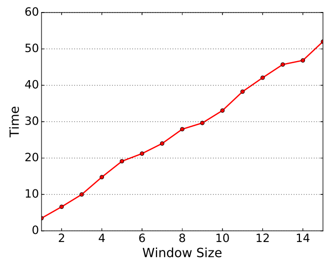


**Figure 8: Time consumption (*ms*) of AdaLinUCB with different sliding window size**

## 6 ANALYSIS

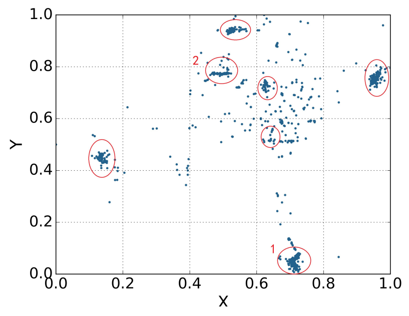
Having answered our main research questions in the previous section, we now analyze our experimental results of AdaLinUCB algorithm. We analyze the clustering relation of user preference filtering matrices. Then, we explore the relation of structure characteristics between different preference filtering matrices. We visualize the result and find that they have explainable clustering characteristics corresponding to realistic situations.

### 6.1 Clustering relation of user preference filtering matrices

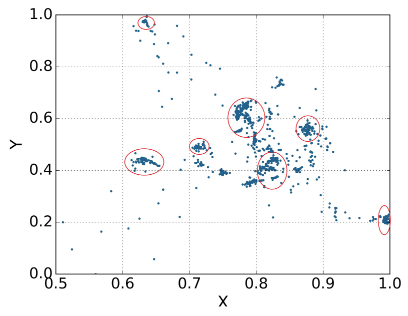
The main idea of AdaLinUCB algorithm is adaptively filtering user preferences according specific items. As far as we know, similar users hold similar interests in recommender systems. This means the user preference parameters, which are learned by the existing contextual bandit algorithm, are similar in the same dataset. Therefore, we can infer that the preference filtering matrices which are learned by Online Sliding Window Model also hold similar clustering characteristics.

First, we executed AdaLinUCB with Online Sliding Window Model on both datasets. In AdaLinUCB algorithm, each user holds a filtering matrix which is $25 \times 25$ dimensions. After continuous learning, we recorded the learning results of the preference filtering matrices. Then we truncated each row of the matrix to get a one-dimensional vector. Finally, we used t-Distributed Stochastic Neighbor Embedding[5] (t-SNE) [13, 30] to reduce the dimensionality of the vectors and visualized the result, as shown in Figure 9 (a) and

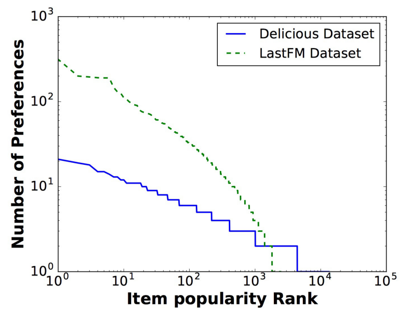
[5]https://lvdmaaten.github.io/tsne/



**(a) LastFM**

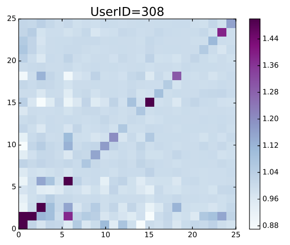


**(b) Delicious**

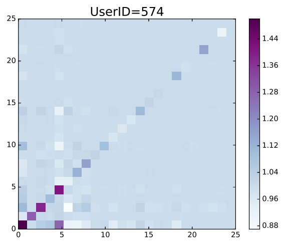


**(c) Item popularity of the two datasets**

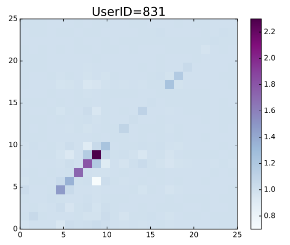
**Figure 9: Clustering relation of user preference filtering matrices by t-SNE, LastFM result contains 1892 individual user and Delicious result contains 1867 individual user. Figure (c) is item-based analysis on datasets.**



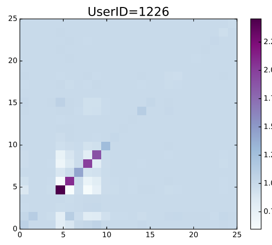
**(a) UserID = 308**



**(b) UserID = 574**



**(c) UserID = 831**



**(d) UserID = 1226**

**Figure 10: Visualization of user preference filtering matrix**

(b). For LastFM and Delicious datasets, we could see from Figure 9 (c), there are a lot more popular artists whom everybody listens to on LastFM dataset than the popular websites which everyone bookmarks on Delicious dataset. Due to LastFM dataset is much denser than the Delicious dataset, it is easier to dig out the potential relationship between users. As we can see from Figure 9 (a), the user presents a strong scattered aggregation relationship. Each cluster represents a group of users with the same interest. For Delicious dataset, as shown in Figure 9 (b), most of the users are gathered in a relatively large range in the middle. This shows that most users' interests are more similar in Delicious dataset than LastFM. Considering that websites is less popular than music artists which contains a variety of styles and each style has a large number of loyal fans, this result is reasonable.

## 6.2 Structure characteristics of preference filtering matrices

We further explore the relation of structure characteristics between different users' preference filtering matrices. It is worth nothing that the user preference filtering matrices in the same cluster is structurally similar. In order to show this result more clearly, we draw the preference filtering matrix by heat map, as shown in Figure 10. In Figure 10 (a) and (b), these two users whose ID are 308 and 574, all from cluster 1 in Figure 9 (a). In both figures, values in diagonal and lower left corner are both relatively large and it means that the structure of this two users is relatively similar. Moreover, users 308 and 574 are friends in LastFM dataset. In Figure 10 (c) and (d), these two user whose ID are 831 and 1226 from cluster 2 in Figure 9 (a) hold the similar characteristics. We analyze the user preference filtering matrices in the same cluster and find that users in the same cluster are probably friend or can be connected by their common friend. These results illustrate the correctness and effectiveness of the AdaLinUCB algorithm from another perspective.

## 7 CONCLUSION

In this work, we present an adaptive linear upper confidence bound algorithm, AdaLinUCB for online recommendations. AdaLinUCB takes advantage of preference filtering matrix to adaptively utilize user preferences during recommendations. We conducted extensive experiments on LastFM and Delicious datasets. AdaLinUCB significantly outperforms state-of-the-art methods and achieves 15% improvement over benchmarks in terms of *Cumulative Reward* metric. It improves the accuracy of online recommendations without sacrificing efficiency. Besides, based on our experimental results and subsequent analyses, we found that the clustering characteristics of learned preference filtering matrices consist with real scenarios which means AdaLinUCB could not only improve the recommendation performance but also help to understand how the recommendation results are generated.

A limitation in our work is that the dimension of the preference filtering matrix depends on the item feature vectors. As to future work, exploring more principled optimization procedures for learning AdaLinUCB, instead of gradient decent with a sliding window, should give new insights for contextual bandits algorithms. Also, reformulation of AdaLinUCB into a non-linear form may enhance the prediction performance of online recommendations.

## REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6 (2005), 734–749.

[2] Peter Auer. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3 (Nov 2002), 397–422.

[3] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47, 2-3 (May 2002), 235–256.

[4] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 1995. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposiumon Foundations of Computer Science.* 322–331.

[5] Djallel Bouneffouf, Amel Bouzeghoub, and Alda Lopes Gançarski. 2012. A contextual-bandit algorithm for mobile context-aware recommender system. In *International Conference on Neural Information Processing.* 324–331.

[6] Swapna Buccapatnam, Atilla Eryilmaz, and Ness B Shroff. 2013. Multi-armed bandits in the presence of side observations in social networks. In *52nd IEEE Conference on Decision and Control.* 7309–7314.

[7] Swapna Buccapatnam, Atilla Eryilmaz, and Ness B Shroff. 2014. Stochastic bandits with side observations on networks. *The 2014 ACM International Conference on Measurement and Modeling of Computer Systems* 42, 1 (2014), 289–300.

[8] Nicolò Cesa-Bianchi, Claudio Gentile, and Giovanni Zappella. 2013. A Gang of Bandits. In *Advances in Neural Information Processing Systems.* 737–745.

[9] Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *Advances in Neural Information Processing Systems.* 2249–2257.

[10] Wei Chu, Lihong Li, Lev Reyzin, and Robert E Schapire. 2011. Contextual bandits with linear payoff functions. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, Vol. 15. 208–214.

[11] Sarah Filippi, Olivier Cappe, Aurélien Garivier, and Csaba Szepesvári. 2010. Parametric bandits: The generalized linear case. In *Advances in Neural Information Processing Systems.* 586–594.

[12] John C Gittins. 1979. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)* (1979), 148–177.

[13] G. E. Hinton. 2008. Visualizing High-Dimensional Data Using t-SNE. *Vigiliae Christianae* 9, 2 (2008), 2579–2605.

[14] S Kabbur and G Karypis. 2014. NLMF: NonLinear Matrix Factorization Methods for Top-N Recommender Systems. In *2014 IEEE International Conference on Data Mining Workshop.* 167–174.

[15] Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. 2008. Efficient bandit algorithms for online multiclass prediction. In *Proceedings of the 25th International Conference on Machine Learning.* 440–447.

[16] Jaya Kawale, Hung H Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla. 2015. Efficient thompson sampling for online matrix-factorization recommendation. In *Advances in Neural Information Processing Systems.* 1297–1305.

[17] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th International Conference on Knowledge Discovery and Data Mining.* 426–434.

[18] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[19] John Langford and Tong Zhang. 2008. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems.* 817–824.

[20] 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.

[21] Wei Li, Xuerui Wang, Ruofei Zhang, Ying Cui, Jianchang Mao, and Rong Jin. 2010. Exploitation and exploration in a performance based contextual advertising system. In *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining.* 27–36.

[22] Benedict C May, Nathan Korda, Anthony Lee, and David S Leslie. 2012. Optimistic bayesian sampling in contextual-bandit problems. *Journal of Machine Learning Research* 13, Jun (2012), 2069–2106.

[23] Atsuyoshi Nakamura. 2015. A ucb-like strategy of collaborative filtering. In *Asian Conference on Machine Learning.* 315–329.

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

[25] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th International Conference on Machine Learning.* 784–791.

[26] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th International Conference on Research and Development in Information Retrieval.* 635–644.

[27] Aleksandrs Slivkins. 2014. Contextual bandits with similarity information. *Journal of Machine Learning Research* 15, 1 (2014), 2533–2568.

[28] Liang Tang, Yexi Jiang, Lei Li, Chunqiu Zeng, and Tao Li. 2015. Personalized recommendation via parameter-free contextual bandits. In *Proceedings of the 38th ACM International Conference on Research and Development in Information Retrieval.* 323–332.

[29] Liang Tang, Romer Rosales, Ajit Singh, and Deepak Agarwal. 2013. Automatic ad format selection via contextual bandits. In *Proceedings of the 22nd International Conference on Information & Knowledge Management.* 1587–1594.

[30] Laurens van der Maaten and Geoffrey Hinton. 2012. Visualizing non-metric similarities in multiple maps. *Machine Learning* 87, 1 (2012), 33–55.

[31] Joannes Vermorel and Mehryar Mohri. 2005. Multi-armed bandit algorithms and empirical evaluation. In *European Conference on Machine Learning.* 437–448.

[32] Andrew J Vickers, Cathee Till, Catherine M Tangen, Hans Lilja, and Ian M Thompson. 2011. An empirical evaluation of guidelines on prostate-specific antigen velocity in prostate cancer detection. *Journal of the National Cancer Institute* 103, 6 (2011), 462.

[33] Thomas J Walsh, István Szita, Carlos Diuk, and Michael L Littman. 2009. Exploring compact reinforcement-learning representations with linear regression. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence.* 591–598.

[34] Chih-Chun Wang, Sanjeev R Kulkarni, and H Vincent Poor. 2005. Bandit problems with side observations. *IEEE Trans. Automat. Control* 50, 3 (2005), 338–355.

[35] Huazheng Wang, Qingyun Wu, and Hongning Wang. 2016. Learning hidden features for contextual bandits. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management.* 1633–1642.

[36] Jason Weston, Ron J. Weiss, and Hector Yee. 2013. Nonlinear latent factorization by embedding multiple user interests. In *Proceedings of the 7th ACM Conference on Recommender Systems.* 65–68.

[37] Qingyun Wu, Huazheng Wang, Quanquan Gu, and Hongning Wang. 2016. Contextual Bandits in a Collaborative Environment. In *Proceedings of the 39th ACM International Conference on Research and Development in Information Retrieval.* 529–538.

[38] Yisong Yue and Carlos Guestrin. 2011. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems.* 2483–2491.

[39] Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning.* 1201–1208.

[40] 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.