Contextual Bandits in A Collaborative Environment

Qingyun Wu\textsuperscript{1}, Huazheng Wang\textsuperscript{1}, Quanquan Gu\textsuperscript{2}, Hongning Wang\textsuperscript{1}
\textsuperscript{1}Department of Computer Science
\textsuperscript{2}Department of Systems and Information Engineering
University of Virginia, Charlottesville VA, 22904 USA
\{qw2ky,hw7ww,qg5w,hw5x\}@virginia.edu

\section*{ABSTRACT}
Contextual bandit algorithms provide principled online learning solutions to find optimal trade-offs between exploration and exploitation with companion side-information. They have been extensively used in many important practical scenarios, such as display advertising and content recommendation. A common practice estimates the unknown bandit parameters pertaining to each user independently. This unfortunately ignores dependency among users and thus leads to suboptimal solutions, especially for the applications that have strong social components.

In this paper, we develop a collaborative contextual bandit algorithm, in which the adjacency graph among users is leveraged to share context and payoffs among neighboring users while online updating. We rigorously prove an improved upper regret bound of the proposed collaborative bandit algorithm comparing to conventional independent bandit algorithms. Extensive experiments on both synthetic and three large-scale real-world datasets verified the improvement of our proposed algorithm against several state-of-the-art contextual bandit algorithms.

\textbf{Categories and Subject Descriptors}
H.3.5 [Information Systems]: On-line Information Services; I.2.6 [Computing Methodologies]: Learning

\textbf{Keywords}
Collaborative contextual bands; online recommendations; reinforcement learning

\section{1. INTRODUCTION}
Satisfying users with personalized information plays a crucial role for online service providers to succeed in market. However, the rapid appearance of new information and new users together with the ever-changing nature of content popularity make traditional recommendation approaches, e.g., collaborative filtering [7, 25], incompetent. Modern information service systems now adopt online learning solutions to adaptively find good mappings between available content and users. During online learning, the need to focus on information that raises user interest and, simultaneously, the need to explore new information for globally improving user experience create an explore-exploit dilemma. Significant research attention has been paid on multi-armed bandit algorithms [16, 19, 4, 5], which provide a principled solution of the dilemma. Intuitively, bandit algorithms designate a small amount of traffic to collect user feedback while improving their estimation qualities in realtime.

With the available side information about users or items to be presented, contextual bandits have become a reference solution [3, 10, 21, 14]. Specifically, contextual bandits assume the expected payoff is determined by a conjecture of unknown bandit parameters and given context, which is represented as a set of features extracted from both users and recommendation candidates. Such algorithms are especially advantageous when the space of recommendation is large but the payoffs are interrelated. They have been successfully applied in many important applications, e.g., content recommendation [21, 6] and display advertising [11, 23].

However, a common practice in contextual bandit algorithms estimates the unknown bandit parameters pertaining to each user independently. This unfortunately ignores dependency among users. Due to the existence of social influence [13], e.g., content and opinions sharing among friends in a social network, exploiting the dependency among users raises new challenges and opportunities in personalized information services. For example, in many real-world applications, e.g., content recommendation in Facebook or Twitter, because of the mutual influence among friends and acquaintances, one user’s click decision on the recommended items might be greatly influenced by his/her peers. This indicates the knowledge gathered about the interest of a given user can be leveraged to improve the recommendation to his/her friends, i.e., collaborative learning. In other words, the observed payoffs from a user’s feedback might be a compound of his/her own preference and social influence he/she receives, e.g., social norms, conformity and compliance. As a result, propagating the knowledge collected about the preference of one user to his/her related peers can not only capitalize on additional information embedded in the dependency among users, which is not available in the context vectors; but also helps conquer data sparsity issue by reducing the sample complexity of preference learning (e.g., known as cold-start in recommender systems [26]). Failing to recognize such information among users will inevitably lead to a suboptimal solution.

In this work, we develop a collaborative contextual bandit algorithm that explicitly models the underlying dependency among users. In our solution, a weighted adjacency graph is constructed, where each node represents a contextual bandit deployed for a single user and the weight on each edge indicates the influence between a pair of users. Based on this dependency structure, the observed payoffs on each user are assumed to be determined by a mixture of neighboring users in the graph. We then estimate the bandit parameters over all the users in a collaborative manner: both context and received payoffs from one user are prorogated across
the whole graph in the process of online updating. The proposed collaborative bandit algorithm establishes a bridge to share information among heterogeneous users and thus reduce the sample complexity of preference learning. We rigorously prove that our collaborative bandit algorithm achieves a remarkable reduction of upper regret bound with high probability, comparing to the linear regret with respect to the number of users if one simply runs independent bandits on them. Extensive experiment results on both simulations and large-scale real-world datasets verified the improvement of the proposed algorithm compared with several state-of-the-art contextual bandit algorithms. In particular, our algorithm greatly alleviates the cold-start challenge, in which encouraging performance improvement is achieved on new users and new items.

3. METHODOLOGY

We develop a contextual bandit algorithm in a collaborative environment, where the adjacency graph among users is leveraged to share context and payoffs between neighboring bandits during online update. We provide rigorous proof of the resulting upper regret bound, which has a significant regret reduction comparing to running independent bandit algorithms on the same collection of users. In the following discussions, we will first describe the notations and our model assumptions about the collaborative bandit problem, then carefully illustrate our developed bandit algorithm and corresponding regret analysis.

3.1 Bandits in a Collaborative Setting

We consider a contextual bandit problem with a finite, but possibly large, number of arms, which correspond to the candidate item set to be presented (e.g., articles in a content recommendation system). We denote the arm set as \( \mathcal{A} \) and the cardinality of \( \mathcal{A} \) as \( K \). There are \( N \) different users in this collection. At each trial \( t \), a learner observes a given user \( u_t \) from user collection and a subset of arms from \( \mathcal{A} \), where each arm \( a \) is associated with a feature vector \( \mathbf{x}_{a,u} \in \mathbb{R}^d \) summarizing the information of both user \( u_t \) and arm \( a_t \) at trial \( t \). The learner selects an arm \( a_t \) to \( u_t \) and receives the corresponding payoff \( r_{a_t,u} \) from the user. The goal of the learner is to update its arm-selection strategy with respect to the new feedback, such that after \( T \) trials its regret with respect to the oracle arm selection strategy is minimized. In particular, the accumulated \( T \)-trial regret for all users is defined formally as,

\[
R(T) = \sum_{t=1}^{T} R_t = \sum_{t=1}^{T} (r^*_{a_t,u_t} - r_{a_t,u_t})
\]

where \( r^*_{a,t} \) is the best arm to display to user \( u_t \) at trial \( t \) according to the oracle strategy, \( r_{a_t,u_t} \) is the corresponding optimal payoff, and \( R_t \) is the regret from all users at trial \( t \).

In standard linear contextual bandit problems, the payoffs of each arm with respect to different users are assumed to be governed by a noisy version of an unknown linear function of the context vectors \([9, 21]\). Specifically, each user \( i \) is assumed to associate with an unknown parameter \( \Theta_i \in \mathbb{R}^d \) (with \( \| \Theta_i \| \leq 1 \)), which determines the payoff of \( a_t \) by \( r_{a_t,i} = \mathbf{x}_{a_t,i}^T \Theta_i + \epsilon_t \), where the random variable \( \epsilon_t \) is drawn from a Gaussian distribution \( N(0, \sigma^2) \). \( \Theta_i \) is independently estimated based on the observations from each individual user. However, due to the existence of mutual influence among users, an isolated bandit can hardly explain all the observed payoffs even for a single user. For example, the context vectors fail to encode such dependency. To capitalize on the additional information embedded in the dependency structure among users (i.e., \( \Theta \) for different users), we propose to study contextual bandit problems in a collaborative setting.

In this collaborative environment, we place the bandit algorithms on a weighted graph \( G = (V, E) \), which encodes the affinity relationship among users. Specifically, each node \( v_i \in \{V_1, ..., V_N\} \) in \( G \) hosts a bandit parameterized by \( \Theta_i \) for user \( i \); and the edges in \( E \) represent the affinity relation over pairs of users. This graph can be described as an \( N \times N \) stochastic matrix \( \mathbf{W} \). In this matrix, each element \( w_{ij} \) is nonnegative and proportional to the influence that user \( j \) has on user \( i \) in determining the payoffs of different arms. \( w_{ij} = 0 \) if and only if user \( j \) has no influence on user \( i \). \( \mathbf{W} \) is column-wise normalized such that \( \sum_{j=1}^{N} w_{ij} = 1 \) for \( i \in \{1, ..., N\} \). In this work, we assume \( \mathbf{W} \) is time-invariant and known to the learner beforehand.

Based on the graph \( \mathbf{W} \), our collaboration among bands will happen when determining the payoff of a particular arm with respect to a given user. To denote this, we define a \( d \times N \) matrix \( \Theta \), which consists of parameters from all the bandits in the graph: \( \Theta = \)
(θ_1, ..., θ_d). Accordingly, we define a context feature matrix \( X_t = (x_{a_1 t}, ..., x_{a_d t}) \), where the \( i \)-th column is the context vector \( x_{a_i t} \) for arm \( a_i \) at trial \( t \) selected for user \( u \). The collaboration among bandits characterized by the influence matrix \( W \) results in a new bandit parameter matrix \( \tilde{\Theta} = \Theta W \), which determines the payoff \( r_{a_t u_t} \) of arm \( a_t \) for user \( u_t \) at trial \( t \) by,

\[
r_{a_t u_t} = diag_{a_t}(X_t) \tilde{\Theta} W \sim N(0, \sigma^2) \tag{2}
\]

where \( diag_{a_t}(X) \) is the operation returning the \( u_t \)-th element in the diagonal of matrix \( X \).

Eq (2) postulates our additive assumption about reward generation in this collaborative environment: the reward \( r_{a_t u_t} \) is not only determined by user \( u_t \)'s own preference on the arm \( a_t \) (i.e., \( w_{u_t a_t} x_{a_t u_t} ^ T \theta_{a_t} \)), but also by the judgements from the neighbors who have influence on \( u_t \) (i.e., \( \sum_{j \neq u_t} w_{u_t j} x_{a_t u_j} ^ T \theta_{j} \)). This enables us to distinguish a user's intrinsic preference of the recommended content from his/her neighbors' influence, i.e., personalization. In addition, the linear payoff assumption in our model is to simplify the discussion in this paper; and it can be relaxed via a generalized linear model [14] to deal with nonlinear rewards.

We should note that our model assumption about the collaborative bandits is different from that specified in the GOB.Lin model [10]. In GOB.Lin connected users in the graph are required to have similar underlying bandit parameters, i.e., via graph Laplacian regularization over the learned bandit parameters. And their assumption about reward generation follows conventional contextual bandit settings, i.e., rewards are independent across users. In our setting, neighboring users do not have to share similar bandit parameters, but they will generate influence on their neighbors' decisions. This assumption is arguably more general, and it leads to an improved upper regret bound and practical performance. Theoretical comparison between these two algorithms will be rigorously discussed in Section 3.3.

### 3.2 Collaborative Linear Bandit Algorithm

To simplify the notations in our following discussions, we define two long context feature vectors and a long bandit parameter vector based on the vectorize operation \( vec(\cdot) \). We define \( X^T_{a_t} = vec(X_{a_t}) = (x^T_{a_t 1}, ..., x^T_{a_t N}) \), which is a concatenation of context feature vectors of the chosen arm \( a_t \) at trial \( t \) for all the users. And we define \( X^T_{a_t u_t} = vec(X_{a_t u_t}) \), in which \( X_{a_t u_t} \) is a special case of \( X_{a_t} \); only the column corresponding to the user \( u_t \) at time \( t \) is set to \( x_{a_t u_t} \), and all the other columns are set to zero. This corresponds to the situation that at trial \( t \) the learner only needs to interact with one user. Correspondingly, we define \( \theta = vec(\Theta) = (\theta^T_1, \theta^T_2, ..., \theta^T_N)^T \in \mathbb{R}^{dN} \) as the concatenation of bandit parameter vectors over all the users.

With the collaborative assumption about the expected payoffs defined in Eq (2), we appeal to ridge regression for estimating the unknown bandit parameter \( \theta \) for each user. In particular, we simultaneously estimate the global bandit parameter matrix \( \Theta \) for all the users as follows,

\[
\hat{\Theta} = \arg \min_{\Theta} \frac{1}{2} \sum_{t=1}^T (X^T_{a_t u_t} vec(\Theta, W) - r_{a_t u_t})^2 + \frac{\lambda}{2} tr(\Theta^T \Theta)
\]

where \( \lambda \in [0, 1] \) is a trade-off parameter of L2 regularization in ridge regression.

Since the objective function defined in Eq (3) is quadratic with respect to \( \Theta \), we have a closed-form estimation of \( \Theta \) as \( \hat{\Theta} = \sum_{t=1}^T X^T_{a_t u_t} X_{a_t u_t} W^{-1} \).

Algorithm 1 Collaborative LinUCB

1: **Inputs:** \( \alpha \in \mathbb{R}, \lambda \in [0, 1], W \in \mathbb{R}^{N \times N} \)
2: **Initialize:** \( A_t \leftarrow \lambda I, b_t \leftarrow 0, \theta_t \leftarrow A_t^{-1} b_t, C_t \leftarrow (W^T \otimes I) A_t^{-1} (W \otimes I) \)
3: for \( t = 1 \) to \( T \) do
4: Receive user \( u_t \)
5: Observe context vectors, \( x_{a_t u_t} \in \mathbb{R}^d \) for all \( u_t \in A \)
6: Take action \( a_t = \arg \max_{a \in A} X^T_{a_t u_t} vec(\theta_t, W) + \sqrt{2 \lambda tr(C_t)} \)
7: Observe payoff \( r_{a_t u_t} \)
8: \( A_{t+1} \leftarrow A_t + vec(X_{a_t u_t} W^T) vec(X_{a_t u_t} W^T)^T \)
9: \( b_{t+1} \leftarrow b_t + vec(X_{a_t u_t} W^T) r_{a_t u_t} \)
10: \( C_{t+1} \leftarrow (W^T \otimes I) A_t^{-1} (W \otimes I) \)
11: \( \theta_{t+1} \leftarrow \theta_{t} - b_{t+1} \)
12: end for

**A_t^{-1} b_t**, in which \( \theta = vec(\hat{\Theta}) \) and \( A_t \) and \( b_t \) are computed as,

\[
A_t = \lambda I + \sum_{t'=1}^T vec(X_{a_{t'} u_{t'}} W^T) vec(X_{a_{t'} u_{t'}}, W^T)^T
\]

\[
b_t = \sum_{t'=1}^T vec(X_{a_{t'} u_{t'}} W^T) r_{a_{t'} u_{t'}} \]

where \( I \) is a \( dN \times dN \) identity matrix.

The effect of collaboration among bandits is clearly depicted in the above estimation of \( \Theta \). Matrix \( A_t \) and vector \( b_t \) store global information shared among all the bandits in the graph. More specifically, the context vector \( x_{a_t u_t} \), and payoff \( r_{a_t u_t} \) observed in user \( u_t \) at trial \( t \) are propagated through the whole graph via the relational matrix \( W \). To understand this, note that \( vec(X_{a_t u_t} W^T) \) is a dense vector with projected context vectors on every user, while the original \( X_{a_t u_t} \) is a sparse vector with observations only at active users \( u_t \). Because of this information sharing, at certain trial \( t \), although some users might generate any observation yet (i.e., cold-start), they can already start from a non-random initialization of their bandit parameters \( \theta_t \). It is easy to verify that when \( W \) is an identity matrix, i.e., users have no influence among each other, the estimation of \( \Theta \) degenerates to independently computing \( N \) different \( \theta \) (since \( vec(X_{a_t u_t} W^T) = X_{a_t u_t} \)). And the mutual influence will be maximized when \( W \) is a uniform matrix, i.e., all the users have equivalent influence to each other. We have to emphasize that the benefit of this collaborative estimation of \( \Theta \) is not to just simply compute the \( \theta \) in an integrated manner; but because of the collaboration among users, the estimation uncertainty of all \( \theta \) can be quickly reduced comparing to simply running \( N \) independent bandit algorithms. This in turn leads to an improved regret bound. We will elaborate the effect of collaboration in online bandit learning with more theoretical justifications in Section 3.3.

The estimated bandit parameters \( \hat{\Theta} \) predict the expected payoff of a particular arm for each user according to the observed context feature matrix \( X_t \). To complete an adaptive bandit algorithm, we need to design the exploration strategy for each user. Our collaborative assumption in Eq (2) implies that \( r_{a_t u_t} \) across users are independent given \( X_t \) and \( W \). As a result, for any \( \sigma \), i.e., the standard deviation of Gaussian noise in Eq (2), the following inequality holds with probability at least \( 1 - \delta \),

\[
|r_{a_t u_t} - r_{a_t u_t}| \leq \alpha \sqrt{vec(X_{u_t} W^T)^T A_t^{-1} vec(X_{u_t} W^T)} \]
of this inequality can be found in the Appendix.

The inequality in Eq (6) gives us a reasonably tight upper confidence bound (UCB) for the expected payoff of a particular arm over all users in the graph \( G \), from which a UCB-style action-selection strategy can be derived. In particular, at trial \( t \), we choose an arm for user \( u_t \) by,

\[
    a_{t,u_t} = \arg\max_{a \in A} \left( \hat{\alpha}^T_{a,u_t} \text{vec}(\Theta, W) + \alpha_t \sqrt{\text{vec}(X_{u_t} W^T)^T A_a^{-1} \text{vec}(X_{u_t} W^T)} \right) 
\]

We name this resulting algorithm as Collaborative Linear Bandit, or CoLin in short. The detailed description of CoLin is illustrated in Algorithm 1, where we use the property that \( \text{vec}(X_{u_t} W^T) = (W \otimes I) \text{vec}(X_{u_t}) = (W \otimes I) X_{u_t} \) to simplify Eq (7).

Another evidence of the benefit from collaboration among users is demonstrated in Algorithm 1. When estimating the confidence interval of the expected payoff for action \( a_t \) in user \( u_t \) at trial \( t \), CoLin not only considers the prediction confidence from bandit \( u_t \), but also that from its neighboring bandits (as described by the Kronecker product between \( W \) and \( I \)). When \( W \) is an identity matrix, such effect disappears. Clearly, this collaborative confidence interval estimation will help the algorithm quickly reduce estimation uncertainty, and thus leads to the optimal solution more rapidly.

One potential issue with CoLin is its computational complexity: matrix inverse has to be performed on \( A_{u_t} \) at every trial. First, because of the rank one update of matrix \( A_{u_t} \) (8th step in Algorithm 1), quadratic computation complexity is possible via applying the Sherman-Morrison formula. Second, we may compute \( A_{u_t}^{-1} \) in a mini-batch manner to further reduce computation with some extra penalty in regret. We will leave this as our future research.

### 3.3 Regret Analysis

In this section, we provide detailed regret analysis of our proposed CoLin algorithm. We first prove that the estimation error of bandit parameters \( \Theta \) is upper bounded in Lemma 1.

**Lemma 1:** For any \( \delta > 0 \), with probability at least \( 1 - \delta \), the estimation error of bandit parameters in CoLin is bounded by,

\[
    \| \hat{\theta}_i - \theta^* \|_{A_i} \leq \sqrt{dN \ln \left( 1 + \sum_{t=1}^{T} \sum_{j=1}^{N} w_{t,i,j}^2 \right) \lambda dN} - 2 \ln(\delta) + \sqrt{\lambda} \| \theta^* \| 
\]

in which \( \| \hat{\theta}_i - \theta^* \|_{A_i} = \sqrt{(\hat{\theta}_i - \theta^*)^T A_i (\hat{\theta}_i - \theta^*)} \), i.e., the matrix norm induced by the positive semidefinite matrix \( A_i \).

Based on Lemma 1, we have the following theorem about the upper regret bound of the CoLin algorithm.

**Theorem 1:** With probability at least \( 1 - \delta \), the cumulated regret of CoLin algorithm satisfies,

\[
    R(T) \leq 2\alpha_T \sqrt{2dN T \ln \left( 1 + \frac{\sum_{t=1}^{T} \sum_{j=1}^{N} w_{t,i,j}^2}{\lambda dN} \right)} 
\]

in which \( \alpha_T \) is the upper bound of \( \| \hat{\Theta}_T - \Theta^* \|_{A_T} \), and it can be explicitly calculated based on Lemma 1. The detailed proof of this theorem is provided in the Appendix.

As shown in Theorem 1, the graph structure plays an important role in the upper regret bound of our CoLin algorithm. Consider two extreme cases. First, when \( W \) is an identity matrix, i.e., no influence among users, the upper regret bound degenerates to \( O(N \sqrt{T} \ln \frac{T}{\delta}) \). Second, when the graph is fully connected and uniform, i.e., \( \forall i, j, w_{i,j} = \frac{1}{N} \), such that users have homogeneous influence among each other, and the upper regret bound of CoLin decreases to \( O(N \sqrt{T} \ln \frac{T}{N \delta}) \). That means via collaboration, CoLin achieves an \( O(\sqrt{T} \ln N) \) regret reduction for every single user in the graph comparing to the independent case.

Note that the regret analysis in Theorem 1 is in a very general form, in which we did not make any assumption about the order or frequency that each user will be served. To illustrate the relationship between the proposed collaborative bandit algorithm and conventional independent bandit algorithms in a more intuitive way, we can make a very specific assumption about how a sequential learner interacts with a set of users. Assuming all the users are evenly served by CoLin, i.e., each user interacts with the learner \( T = \frac{T}{N} \) times. When \( W \) is an identity matrix, the regret bound of CoLin degenerates to the case of running \( N \) independent LinUCB, whose upper regret bound is \( O(N \sqrt{T} \ln T) \). When \( W \) is uniform, the regret bound reduces to \( O(N \sqrt{T} \ln \frac{T}{N}) \), where we achieve an \( O(\sqrt{T} \ln N) \) regret reduction comparing to running \( N \) independent LinUCBs on each single user. The proof of regret bound in this special case is given in the Appendix.

It is necessary to compare the derived upper regret bound of CoLin with that in the GOB.Lin algorithm [10], which also exploits the relatedness among a set of users. In GOB.Lin, the divergence among every pair of bandits (if connected in the graph) is measured by Euclidean distance between the learned bandit parameters. In its upper regret bound, such divergence is accumulated throughout the iterations. In extreme case where users are all connected but associate with totally distinct bandit parameters, GOB.Lin’s upper regret bound could be much worse than running \( N \) independent bandits, due to this additive pairwise divergence. While in our algorithm, such divergence is controlled by the multiplicative factor \( \sum_{t=1}^{T} \sum_{j=1}^{N} w_{t,j}^2 \leq T \). We can rigorously prove the following inequalities between the upper regret bound of CoLin (\( R_C(T) \)) and GOB.Lin (\( R_G(T) \)) always holds,

\[
    0 \leq R_G(T) - R_C(T) \leq 16 T N \ln(1 + \frac{2T}{dN^2}) \sum_{(i,j) \in E} \| \Theta_i^* - \Theta_j^* \|^2 
\]

It is clear to notice that if there is no influence between the users in the collection, i.e., no edge in \( G \), these two algorithms’ regret bound touches (both degenerate to \( N \) independent contextual bandits). Otherwise, GOB.Lin will always lead to a worse and faster growing regret bound than our CoLin algorithm.

In addition, limited by the use of graph Laplacian, GOB.Lin can only capture the binary connectivity relation among users. CoLin differentiates the strength of connections with a stochastic relational graph. This makes CoLin more general when modeling the relatedness among bandits and provides a tighter upper regret bound. This effect is also empirically verified by our extensive experiments on both synthetic and real-world data sets.

### 4. EXPERIMENTS

We performed empirical evaluations of our CoLin algorithm against several state-of-the-art contextual bandit algorithms, including \( N \) independent LinUCB [21], hybrid LinUCB with user features [21], GOB.Lin [10], and online cluster of Bandits (CLUB) [15]. Among these algorithms, hybrid LinUCB exploits user dependency via a set of hybrid linear models over user features, GOB.Lin encodes the user dependency via graph-based regularization over the learned bandit parameters, and CLUB clusters users during online learning to enable model sharing. In addition, we also compared with several popularly used context-free bandit algorithms, including EXP3 [5], UCB1 [4] and \( \epsilon \)-greedy [4]. But their performance is much worse than the contextual bandits, and thus we do not include their performance in the following discussions.

We tested all the algorithms on a synthetic data set via simulations, a large collection of click stream from Yahoo! Today Module dataset [21], and two real-world dataset extracted from the social...
unveils the nature of each bandit algorithm, e.g., how accurately learned parameters in each contextual bandit algorithms. This validates that with the proposed collaborative learning among users, CoLin better alleviates the cold-start challenge comparing to the baselines.

4.1 Experiments on synthetic dataset

In this experiment, we compare the bandit algorithms based on simulations and use the accumulated regret and bandit parameter estimation accuracy as the performance metrics.

4.1.1 Simulation Setting

In simulation, we generate \( N \) users, each of which is associated with a \( d \)-dimensional parameter vector \( \Theta^* = (\Theta^*_1, \ldots, \Theta^*_K) \). Each dimension of \( \Theta^*_i \) is drawn from a uniform distribution \( U(0, 1) \) and normalized to \( \|\Theta^*_i\| = 1 \). \( \Theta^* \) is treated as the ground-truth bandit parameters for reward generation, and they are unknown to bandit algorithms. We then construct the golden relational stochastic matrix \( W \) for the graph of users by defining \( w_{ij} \propto \langle \Theta^*_i, \Theta^*_j \rangle \), and normalize each column of \( W \) by its L1 norm. The resulting \( W \) is disclosed to the bandit algorithms. In the end, we generate a size-\( K \) action pool \( A \). Each action \( a \in A \) is associated with a \( d \)-dimensional feature vector \( x_a \), each dimension of which is drawn from \( U(0, 1) \). We also normalize \( x_a \) by its L1 norm. To construct user features for hybrid LinUCB algorithm, we perform Principle Component Analysis (PCA) on the relational matrix \( W \), and use the first 5 principle components to construct the user features.

To simulate the collaborative reward generation process among users, we first compute \( \Theta^* \) and then compute the payoff of action \( a \) for user \( i \) at trial \( t \) as \( r_{at_i} = \text{diag}(X_i^j \Theta^*) + \epsilon_t \), where \( \epsilon_t \sim N(0, \sigma^2) \). To increase the learning complexity, at each trial \( t \), our simulator only discloses a subset of actions in \( A \) to the learning algorithms for selection, e.g., randomly select 10 actions from \( A \) without replacement. In simulation, based on the known bandit parameters \( \Theta^* \), the optimal action \( a^*_t \) and the corresponding payoff \( r_{a^*_t} \) for each bandit at trial \( t \) can be explicitly computed.

Under this simulation setting, we compared hybrid LinUCB, \( N \) independent LinUCB, GOB.Lin, CLUB and our CoLin algorithm. In particular, at each trial \( t \), the same set of actions are presented to all the algorithms; and the Gaussian noise \( \epsilon_t \) is fixed for all those actions at trial \( t \). In our experiments, we fixed the feature dimension \( d \) to 5, article pool size \( K \) to 1000, and set the trade-off parameter \( \lambda \) for L2 regularization to 0.2 in all the algorithms. We compared the regret of different bandit algorithms during adaptive learning. Furthermore, since we have the ground-truth bandit parameters available in the simulator, we also compared the quality of learned parameters in each contextual bandit algorithms. This unveils the nature of each bandit algorithm, e.g., how accurately they can recover a user’s true preference.

4.1.2 Results & Analysis

We first set the user size \( N \) to 100 and fix the standard deviation \( \sigma \) to 0.1 in the Gaussian noise for reward generation. All the contextual bandit algorithms are executed up to 300 iterations per user in this experiment. We report the cumulated regret as defined in Eq (1) and the Euclidean distance between the learnt bandit parameters from different algorithms and the ground-truth in Figure 1. To reduce randomness in simulation-based evaluation, we reported the mean and standard deviation of final regret from different algorithms after 30,000 iterations over 5 independent runs for results in all following experiments. To increase visibility, we did not plot error bars in Figure 1 (a) and (b).

As we can find in Figure 1 (a), simply running \( N \) independent LinUCB algorithm gives us the worst regret, which is expected. Hybrid LinUCB, which exploits user dependency via a set of hybrid linear models over user features performed better than LinUCB, but still much worse than CoLin. Although GOB.Lin also exploits the graph structure when estimating the bandit parameters, its assumption about the dependency among bandits is too restrictive to well capture the information embedded in the interaction with users. We should note that in our simulation, by multiplying the relational matrix \( W \) with the ground-truth bandit parameter matrix \( \Theta^* \), the resulting bandit parameters \( \Theta^* \) align with GOB.Lin’s assumption, i.e., neighboring bandits are similar. And \( \Theta^* \) is used in reward generation. Therefore, our simulation does not produce any bias against GOB.Lin. In Figure 1 (a) we did not include CLUB, whose regret grew linearly. After looking into the selected arms from CLUB, we found because of the aggregated decisions from users in the automatically constructed user clusters, CLUB always chose suboptimal arms, which led to a linearly increasing regret.

In Figure 1 (b), we compared accuracy of the learnt bandit parameters from different algorithms. Because of their distinct modeling assumptions, LinUCB, hybrid LinUCB, CLUB and GOBLin cannot directly estimate \( \Theta^* \), i.e., the true bandit parameters for each user. Instead, they can only estimate \( \bar{\Theta}^* \), which directly governs the generation of observed payoffs. Only CoLin can estimate both \( \Theta^* \) and \( \bar{\Theta}^* \). As we can find in the results, CoLin gave the most accurate estimation of \( \bar{\Theta}^* \), which partially explains its superior performance in regret. We also find that LinUCB actually achieved a more accurate estimation of \( \Theta^* \) than GOB.Lin, but its regret is much worse. To understand this, we looked into the actual execution of LinUCB and GOB.Lin, and found that because of the graph Laplacian regularization in GOB.Lin, it better controlled exploration in arm selection and therefore picked the optimal arm more often than LinUCB. Hybrid LinUCB’s estimation of \( \bar{\Theta}^* \) is the worst, but it is expected: hybrid LinUCB uses a shared model and a personalized model to fit the observations. Comparing to CoLin’s estimation quality of \( \bar{\Theta}^* \), its estimation of \( \Theta^* \) is much worse. The main reason is that CoLin has to decipher \( \Theta^* \) from the estimated \( \bar{\Theta}^* \) based on \( W \), where noise is accumulated to prevent...
Table 1: Cumulated regret with different bandit size ($\sigma=0.1$).

<table>
<thead>
<tr>
<th>Bandit Size ($N$)</th>
<th>40</th>
<th>80</th>
<th>100</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinUCB</td>
<td>75.31±5.11</td>
<td>168.42±9.90</td>
<td>191.53±6.18</td>
<td>355.56±7.23</td>
</tr>
<tr>
<td>HybridLinUCB</td>
<td>59.12±2.11</td>
<td>150.09±5.29</td>
<td>164.11±9.19</td>
<td>311.43±11.59</td>
</tr>
<tr>
<td>GOB.Lin</td>
<td>58.49±5.04</td>
<td>143.42±5.28</td>
<td>141.96±6.36</td>
<td>275.32±10.51</td>
</tr>
<tr>
<td>CoLin</td>
<td>21.78±12.84</td>
<td>47.73±4.31</td>
<td>49.83±6.55</td>
<td>77.38±20.59</td>
</tr>
</tbody>
</table>

Table 2: Cumulated regret with different noise level ($N=100$).

<table>
<thead>
<tr>
<th>Noise ($\sigma$)</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinUCB</td>
<td>92.72±2.56</td>
<td>116.53±3.07</td>
<td>191.53±6.18</td>
<td>830.47±69.48</td>
</tr>
<tr>
<td>HybridLinUCB</td>
<td>69.47±2.12</td>
<td>91.48±1.94</td>
<td>164.11±9.19</td>
<td>759.93±39.15</td>
</tr>
<tr>
<td>GOB.Lin</td>
<td>58.85±6.25</td>
<td>82.33±1.53</td>
<td>141.96±6.36</td>
<td>708.13±43.73</td>
</tr>
<tr>
<td>CoLin</td>
<td>41.69±6.95</td>
<td>40.95±4.43</td>
<td>49.83±6.55</td>
<td>83.98±8.57</td>
</tr>
</tbody>
</table>

Table 3: Cumulated regret with different noise level on matrix $W$ ($N=100$).

<table>
<thead>
<tr>
<th>Matrix noise ($\delta$)</th>
<th>0</th>
<th>0.01</th>
<th>0.03</th>
<th>0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>HybridLinUCB</td>
<td>164.11±9.19</td>
<td>171.74±7.67</td>
<td>171.51±13.31</td>
<td>163.93±9.19</td>
</tr>
<tr>
<td>GOB.Lin</td>
<td>141.96±6.36</td>
<td>163.28±5.63</td>
<td>164.36±6.66</td>
<td>169.87±11.89</td>
</tr>
<tr>
<td>CoLin</td>
<td>49.83±6.55</td>
<td>54.42±2.45</td>
<td>101.39±6.01</td>
<td>239.88±13.86</td>
</tr>
</tbody>
</table>

Table 4: Cumulated regret with different matrix sparsity level.

<table>
<thead>
<tr>
<th>Sparsity ($M/N$)</th>
<th>20/100</th>
<th>40/100</th>
<th>60/100</th>
<th>80/100</th>
</tr>
</thead>
<tbody>
<tr>
<td>HybridLinUCB</td>
<td>135.98±5.11</td>
<td>141.15±4.82</td>
<td>150.49±4.58</td>
<td>160.12±7.55</td>
</tr>
<tr>
<td>GOB.Lin</td>
<td>133.30±3.98</td>
<td>126.13±5.59</td>
<td>143.29±6.49</td>
<td>143.42±5.82</td>
</tr>
<tr>
<td>CoLin</td>
<td>39.74±8.80</td>
<td>30.76±3.66</td>
<td>37.29±3.55</td>
<td>49.56±8.88</td>
</tr>
</tbody>
</table>

accurate estimation. Nevertheless, this result demonstrates the possibility of discovering each individual user’s true preference from their compound feedback. This is meaningful for many practical applications, such as user modeling and social influence analysis.

We also notice that although $\text{CLUB}$’s estimated $\hat{\Theta}$ is almost as good as $\text{LinUCB}$’s (as shown in Figure 1 (b)), its regret is the worst. As we described earlier, $\text{CLUB}$’s aggregated decision at user cluster level constantly forced the algorithm to choose suboptimal arms; but the reward generation for each arm in our simulator follows that defined in Eq (2), which still provides validate information for $\text{CLUB}$ to estimate $\hat{\Theta}$ with reasonable accuracy.

In Figure 1 (c), we investigated the effect of exploration parameter $\alpha_1$’s setting in different algorithms. The last column indexed by $\alpha_1$ represented the theoretical values of $\alpha$ computed from the algorithms’ corresponding regret analysis. As shown in the results, the empirically tuned $\alpha$ yields comparable performance to the theoretical values, and makes online computation more efficient. As a result, in all our following experiments we will use a fixed $\alpha$ instead of a computed $\alpha_1$.

To further investigate the convergence property of different bandit algorithms, we examined the following four scenarios: 1) various user sizes $N$, 2) different noise level $\sigma$, 3) a corrupted affinity matrix $W$, and 4) a sparse affinity matrix $W$, in reward generation. We report the results in Table 1 to 4. Because of its poor performance, we did not include $\text{CLUB}$ in those tables.

Firstly, in Table 1, we fixed the noise level $\sigma$ to 0.1 and varied the user size $N$ from 40 to 200. We should note in this experiment the total number of iterations varies as every user will interact with the bandit learner 300 times. The regret in $\text{LinUCB}$ goes linearly with respect to the number of users, since no information is shared across them. Via model sharing, hybrid $\text{LinUCB}$ achieved some regret reduction compared with $\text{LinUCB}$, but its regret still increases linearly with the number of users. Compared with the independent bandits, we can clearly observe the regret reduction in $\text{CoLin}$ with increasing number of users. As we discussed in Section 3.3, although $\text{GOB.Lin}$ exploits the dependency among users, its regret might be even worse than running $N$ independent $\text{LinUCB}$s, especially when the divergence between users is large.

Secondly, in Table 2, we fixed $N$ to 100 and varied the noise level $\sigma$ from 0.01 to 0.3. We can notice that $\text{CoLin}$ is more robust to noise in the feedback: its regret grows much slower than all baselines. Our current regret analysis does not consider the effect of noise in reward, as long as it has a zero mean and finite variance. It would be interesting to incorporate this factor in regret analysis to provide more insight of collaborative bandit algorithms.

Thirdly, in $\text{CoLin}$, we have assumed the adjacency matrix $W$ is known to the algorithm beforehand. However, in reality one might not precisely recover this matrix from noisy observations, e.g., via social network analysis. It is thus important to investigate the robustness of collaborative bandit algorithms to a noisy W. We fixed the user size $N$ to 100 and corrupted the ground-truth adjacency matrix $W$: add Gaussian noise $N(0, \delta)$ to $w_{ij}$ and normalize the resulting matrix. We refer to this noisy adjacency matrix as $W_0$. The simulator still uses the true adjacency matrix $W$ to compute the reward of each action for a given user; while the noisy matrix $W_0$ will be provided to the bandit algorithms, i.e., $\text{CoLin}$ and $\text{GOB.Lin}$. This newly introduced Gaussian noise is different from the noise in generating the rewards as described in Eq (2).

From the cumulated regret shown in Table 3, we can find that under moderate noise level, $\text{CoLin}$ performed much better than $\text{GOB.Lin}$; but $\text{CoLin}$ is more sensitive to noise in $W$ than $\text{GOB.Lin}$. Because $\text{CoLin}$ utilizes a weighted adjacency graph to capture the dependency among users, it becomes more sensitive to the estimation error in $W$. While in $\text{GOB.Lin}$, because only the graph connectivity is used and the random noise is very unlikely to change the graph connectivity, its performance is more stable. Further theoretical analysis of how an inaccurate estimation of $W$ would affect the resulting regret will be an interesting future work yet to explore.

Finally, the regret analysis of $\text{CoLin}$ shows that its upper regret bound is related with the graph structure through the term $\sum_{i=1}^{N} \sum_{j=1}^{N} W_{ij}^2$ and $\text{GOB.Lin}$’s regret bound is related to the graph connectivity [10]. We designed another set of simulation experiments to verify the effect of graph structure on $\text{CoLin}$ and $\text{GOB.Lin}$. In this experiment, we set the user size $N$ to 100 and controlled the graph sparsity as follows: for each user in graph $G$, we only included the edges from his/her top $M$ most influential
neighbors (measured by the edge weight in W) in the adjacency matrix, and normalized the resulting adjacency matrix to a stochastic matrix. No noise is added to W in this experiment (i.e., δ = 0).

As shown in Table 4, the regret of all bandit algorithms increases as W becomes sparser, i.e., less information can be shared across users. We can observe that the regret of CoLin increases faster than that in GOB.Lin, since more information becomes unavailable to CoLin. The results empirically verified that CoLin’s regret bound is directly related to the graph structure described by the term \( \sum_{t=1}^{T} \sum_{m=1}^{N} w_{mij} \) and GOB.Lin’s regret bound is only related to the graph connectivity.

4.2 Experiments on Yahoo! Today Module

In this experiment, we compared our CoLin algorithm with LinUCB, hybrid LinUCB, GOB.Lin and CLUB on a large collection of ten days’ real traffic data from Yahoo! Today Module [21] using the unbiased offline evaluation protocol proposed in [22].

The dataset contains 45,811,883 user visits to the Today Module in a ten-day period in May 2009. For each logged event, both the user and each of the 10 candidate articles are associated with a feature vector of six dimensions (including a constant bias feature), which is constructed by a conjunct analysis with a bilinear model [21]. However, this data set does not contain any user identity. This forbids us to associate the observations with individual users. To address this limitation, we first clustered all users into user groups by applying K-means algorithm on the given user features. Each observation is assigned to its closest user group. The weight in the adjacency matrix W is estimated by the dot product between the centroids from K-means’ output, i.e., \( u_{ij} \propto \langle u_i, u_j \rangle \). The CoLin and GOB.Lin algorithms are then executed over those identified user groups. For the LinUCB baseline, we tested two variants: one is individual LinUCBs running over the identified user groups and it is denoted as M-LinUCB; another one is a uniform LinUCB shared by all the users, i.e., it does not distinguish individual users, and thus it is denoted as Uniform-LinUCB.

In this experiment, click-through-rate (CTR) was used to evaluate the performance of all bandit algorithms. Average CTR is computed in every 2000 observations (not the cumulated CTR) for each algorithm based on the unbiased offline evaluation protocol proposed in [22, 21]. Following the same evaluation principle used in [21], we normalized the resulting CTR from different bandit algorithms by the corresponding logged random strategy’s CTR. We report the normalized CTR results from different contextual bandit algorithms over 160 derived user groups in Figure 2 (a).

CoLin outperformed all baselines on this real-world data set, except CLUB on the first day. Results from other user cluster sizes (40 and 80) showed consistent improvement as demonstrated in Figure 2 (a) with 160 user clusters; but due to space limit, we did not include these results. As we can find CLUB achieved the best CTR on the first day; but as some popular news articles became out-of-date, CLUB cannot correctly recognize their decreased popularity, and thus provided degenerated recommendations. But in CoLin, because of collaborative preference learning, it better controlled the exploration-exploitation trade-off and thus timely recognized the change of items’ popularity. However, one potential limitation of CoLin is its computational complexity: because the dimension of global statistic matrix A, defined in Eq (4) is \( dN \times dN \), the running time of CoLin scales quadratically with the number of users. It makes CoLin less attractive in practical applications where the size of users is large. One potential solution is to enforce sparsity in the estimated W matrix such that distributed model update is possible, i.e., only share information within the connected users. The simulation study in Table 4 confirms the feasibility of this direction and we will explore it in our future work.

4.3 Experiments on LastFM & Delicious

The LastFM dataset is extracted from the music streaming service Last.fm (http://www.last.fm), and the Delicious data set is extracted the social bookmark sharing service website Delicious (https://delicious.com). These two datasets were generated by the Information Retrieval group at Universidad Autonoma de Madrid for the HetRec 2011 workshop with the goal of investigating the usage of heterogeneous information in recommendation system\(^1\). The LastFM dataset contains 1,892 users and 17,632 items (artists). We used the information of “listened artists” of each user to create payoffs for bandit algorithms: if a user listened to an artist at least once, the payoff is 1, otherwise 0. The Delicious dataset contains 1,861 users and 69,226 items (URLs). We generated the payoffs using the information about the bookmarked URLs for each user: the payoff is 1 if the user bookmarked a particular URL, otherwise 0. Both of these two datasets contains the users’ social network graph, which makes them a perfect real-world testbed for collaborative bandits.

Following the same settings in [10], we pre-processed these two datasets in order to fit them into the contextual bandit setting. We first used all tags associated with a single item to create a TF-IDF feature vector, which uniquely represents the item. Then we used PCA to reduce the dimensionality of the feature vectors. In both datasets, we only retained the first 25 principle components to construct the context vectors, i.e., the feature dimension \( d = 25 \). We generated the candidate arm pool as follows: we fixed the size of candidate arm pool to be \( K = 25 \); for a particular user \( u \), we picked one item from those nonzero payoff items for user \( u \) according to the whole observations in the dataset, and randomly picked the other 24 from those zero-payoff items for user \( u \).

User relation graph is extracted from the social network provided by the datasets. In order to make the graph denser and make the algorithms computationally feasible, we performed graph-cut to cluster users into M clusters (following the same setting as in [10]). Users in the same cluster are assumed to share the same bandit parameters. In our experiments, M was set to be 50, 100, and 200.

\(^1\)Datasets and their full description is available at http://grouplens.org/datasets/hetrec-2011
Our reported results are from the setting of $M = 200$, and similar results were achieved in other settings of $M$. After user clustering, a weighted graph can be generated: the nodes are the clusters of nodes in the original graph; and the edges between different clusters are weighted by the number of inter-cluster edges in the original graph. In CoLin, we also need the diagonal elements in $W$, which is undefined in a graph-cut based clustering algorithm. We computed the diagonal elements based on the derived regret bound of CoLin. Specifically, we first set $w_{ij} \propto c(i,j)$, where $c(i,j)$ is the number of edges between cluster $i$ and $j$; then we optimized $\{w_{ij}\}_{i,j}$ which minimizes the term $\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij}^2$.

We included three variants of LinUCB, hybrid LinUCB, GOB.Lin and CLUB as our baselines. The three variants of LinUCB include: 1) LinUCB that runs independently on each user, denoted as N-LinUCB; 2) LinUCB that is shared in each user cluster, denoted as M-LinUCB (M is the number of clusters); 3) LinUCB that is shared by all the users, denoted as Uniform-LinUCB. Following the setting in [10], GOB.Lin also operates at the user cluster level and it takes the connectivity among clusters as input. We normalized the cumulated reward in each algorithm by a random strategy’s cumulated reward, and compute the average accumulated normalized reward in every 50 iterations. Note that user features required by hybrid LinUCB are not given in these two datasets. We applied the same strategy as we used in simulation to generate the user features.

From the results shown in Figure 2 (b) and (c), we can find that CoLin outperformed all the baselines on both Delicious and LastFM datasets. It is worth noting that these two datasets are structurally different, as shown in Figure 3 (a), the popularity of items on these two data sets differs significantly: 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 highly skewed distribution of item popularity makes recommendation on Delicious dataset much more challenging. Because most of items are only bookmarked by a handful of users, exploiting the relatedness among users to propagate feedback become vital. While on LastFM since there are much more popular items that most users would like, most algorithms can easily recognize the quality of items. In order to better understand this difference, we performed detailed item-level analysis to examine the effectiveness of different algorithms on items with varied popularity. Specifically, we first ranked all the items in these two datasets in a descending order of item popularity and then examined the item-based recommendation precision from all the bandit algorithms, e.g., percentage of item recommendations that are accepted by the users. In order to better visualize the results, we grouped the ranked items into different batches and report the average recommendation precision over each batch in Figure 3 (b) and Figure 3 (c).

From the results about the item-based recommendation precision, we can clearly find that on the LastFM dataset, CoLin achieved improved performance against all the baselines in every category of items, given the popularity of items in this data set is more balanced. On Delicious dataset, CoLin achieved better performance on the top-ranked items; however, because of the skewness of item popularity, less popular items are still challenging for all the bandit algorithms to correctly recognize on this dataset.

This analysis motivates us to further analyze the user-level recommendation performance of different bandit algorithms, especially to understand the effectiveness of collaboration among users in alleviating the cold-start problem. To quantitatively evaluate this, we first ranked the user clusters in a descending order with respect to the number of observations in it. We then selected top 50 clusters as group 1. From the bottom 100 user clusters, we select 50 of them who are mostly connected to the users in group 1, and refer to them as group 2. The first group of users is called “learning bucket” and the second group is called “testing bucket.” Based on this separation of user clusters, we performed two experiments: one is warm-start, and another is cold-start. In the warm-start setting, we first run all the algorithms on the learning bucket to estimate parameters for both group of users, such as $A_i$ and $b_i$ in CoLin. However, because users in the second group do not have any observation in the learning bucket, their model parameters can only be updated via the collaboration among bands, i.e., in CoLin and GOB.Lin. Then with the model parameters estimated from the learning bucket, we evaluate different algorithms on the deployment bucket. Correspondingly, in the cold-start setting, we directly run and evaluate the bandit algorithms on the deployment bucket. It is obvious that since LinUCB assumes users are independent and there is no information shared among users, LinUCB’s performance will not change under warm-start and cold start settings. While in CoLin, GOB.Lin and CLUB, because of the collaboration among users, information is propagated among users. In this case, user preference information learned from the learning bucket can be propagated to the deployment bucket.

We reported the performance on the first 10% observations in the deployment bucket instead of the whole observations, in order to better simulate the cold-start situation (i.e., all the algorithms do not have sufficient observations to confidently estimate model parameters). In Figure 4 (a) and (b), we reported the gap of cumulated rewards from CoLin GOB.Lin, and CLUB between warm-start and cold-start, normalized by rewards obtained from LinUCB. From Figure 4(a) we can notice that on Delicious dataset, although at the very beginning of the warm-start setting both GOBLin and CoLin performed worse than the cold-start setting, both algorithms in warm-start quickly improved and outperformed the cold-start setting. One possible explanation is that the algorithms might take the first several iterations to adapt the models propagated from the first user group to the second. In particular, from Figure 4 (a), it is clear that once both algorithms are adapted, the improvement between warm-start and cold-start on CoLin is larger than that on GOB.Lin. This verified CoLin’s effectiveness in address the cold-start challenge. From Figure 4 (b), we can find that warm-start helps both algorithms immediately at the first several iterations on
This paper is based upon work supported by the National Science Foundation under grant IIS-1553568.

5. CONCLUSIONS

In this paper, we developed CoLin algorithm to study contextual bandit problem in a collaborative environment. In CoLin, context and payoffs are shared among the neighboring bandits during online update, and it helps reduce the preference learning complexity (i.e., requires less observations to achieve satisfactory prediction performance) and leads to reduced overall regret. We religiously proved a reduced upper regret bound comparing to independent bandit algorithms. Experimental results based on extensive simulations and three real-world datasets consistently confirmed the improvement of the proposed collaborative bandit algorithm against several state-of-the-art contextual bandit algorithms.

In our current setting, we have assumed the graph structure among users is known to the algorithm beforehand. It is meaningful to study how to dynamically estimate the graph structure during online update with provable regret bound. The computation complexity in CoLin is now expensive. Utilizing the sparse graph structure to decentralize the computation is an important and interesting research direction. Besides, CoLin should not be only limited to linear payoffs; various types of link functions and regret definitions can be explored as our future work.

6. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments. This paper is based upon work supported by the National Science Foundation.

7. REFERENCES

2008.


APPENDIX

Proof of Lemma 1:
Consider the objective function of ridge regression defined in Eq (3). By taking the gradient of $L(\Theta)$ with respect to $\Theta$ and applying our model assumption specified in Eq (2), we have,

$$A_t(\tilde{\Theta}_t - \Theta^*) = \sum_{t'=1}^{t} vec(\mathbf{X}_{a_{t'}, u'} \mathbf{W}^T) \epsilon_{t'} - \lambda \Theta^*$$

in which $\epsilon_{t'}$ is the Gaussian noise at time $t'$ in reward generation.

Define $S_t = \sum_{t'=1}^{t} vec(\mathbf{X}_{a_{t'}, u'} \mathbf{W}^T) \epsilon_{t'}$, we have,

$$\tilde{\Theta}_t - \Theta^* = A_t^{-1} (S_t - \lambda \Theta^*)$$

Because $S_t$ is a martingale, according to Theorem 1 and 2 in [1],

$$\|\tilde{\Theta}_t - \Theta^*\|_{A_t} \leq \sqrt{2 \ln \left( \frac{\text{det}(A_t)}{\delta} \right)} + \sqrt{X} ||\Theta^*||$$

(9) Since $||\mathbf{x}_{u,a_t}|| \leq 1$, trace$(\mathbf{A}_t) \leq \lambda dN + \sum_{i=1}^{t} \sum_{j=1}^{N} w_{u,a_t} \mathbf{x}_{i,j}$, we have

$$\text{det}(\mathbf{A}_t) \leq \left( \text{trace}(\mathbf{A}_t) \right)^{dN} \leq \left( \lambda + \frac{\sum_{i=1}^{t} \sum_{j=1}^{N} w_{u,a_t} \mathbf{x}_{i,j}}{\lambda dN} \right)^{dN}.$$  

Similarly, we have $\text{det}(\mathbf{A}_{dN}) \leq \lambda dN$. Putting all these into Eq (9), we have

$$\|\tilde{\Theta}_t - \Theta^*\|_{A_t} \leq \sqrt{\ln \left( \frac{\text{det}(\mathbf{A}_t)}{\delta} \right)} + \sqrt{dX ||\Theta^*||}$$

□

Proof of Theorem 1:
According to the definition of regret in Eq (1), the regret of CoLin at time $t$ can be written as,

$$R_t = r_{a_t,u} - r_{a_t,u'} = vec(\mathbf{X}_{a_t, u'} W^T)^T \Theta^* - vec(\mathbf{X}_{a_t, u} W^T)^T \Theta^* \leq vec(\mathbf{X}_{a_t, u} W^T)^T \tilde{\Theta}_{t-1} + \alpha_t ||vec(\mathbf{X}_{a_t, u} W^T)||_{A_{t-1}} - vec(\mathbf{X}_{a_t, u} W^T)^T \Theta^* \leq ||vec(\mathbf{X}_{a_t, u} W^T)||_{A_{t-1}} ||\tilde{\Theta}_{t-1} - \Theta^*||_{A_{t-1}} + \alpha_t ||vec(\mathbf{X}_{a_t, u} W^T)||_{A_{t-1}} \leq 2\alpha_t ||vec(\mathbf{X}_{a_t, u} W^T)||_{A_{t-1}}$$

where the first inequality is based on the following two inequalities,

(1) Based on CoLin’s arm selection strategy, if arm $\mathbf{X}_t$ is chosen at time $t$, it must satisfy,

$$vec(\mathbf{X}_{a_t, W^T})^T \tilde{\Theta}_{t-1} + \alpha_t ||vec(\mathbf{X}_{a_t, W^T})||_{A_{t-1}} \geq vec(\mathbf{X}_{a_t, W^T})^T \tilde{\Theta}_{t-1} + \alpha_t ||vec(\mathbf{X}_{a_t, W^T})||_{A_{t-1}}$$

(2) Based on Cauchy-Schwarz inequality, we have,

$$vec(\mathbf{X}_{a_t, W^T})^T \tilde{\Theta}_{t-1} + \alpha_t ||vec(\mathbf{X}_{a_t, W^T})||_{A_{t-1}} \geq -||vec(\mathbf{X}_{a_t, W^T})||_{A_{t-1}} - ||vec(\mathbf{X}_{a_t, W^T})||_{A_{t-1}} \geq -\alpha_t ||vec(\mathbf{X}_{a_t, W^T})||_{A_{t-1}}$$

and therefore, we have

$$vec(\mathbf{X}_{a_t, W^T})^T \tilde{\Theta}_{t-1} + \alpha_t ||vec(\mathbf{X}_{a_t, W^T})||_{A_{t-1}} \geq vec(\mathbf{X}_{a_t, W^T})^T \Theta^*$$

And according to Lemma 11 in [1], we have,

$$\frac{\ln \left( \frac{\text{det}(\mathbf{A}_t)}{\delta} \right)}{dN} \leq \sum_{t=1}^{T} ||vec(\mathbf{X}_{a_t, W^T})||_{A_{t-1}}^2 \leq 2\alpha_t \sqrt{dN \ln \left( \frac{T}{\delta} \right)}$$

Thus the accumulated regret at time $T$ in CoLin can be bounded by,

$$R(T) \leq \sqrt{\ln \left( \frac{\text{det}(\mathbf{A}_T)}{\delta} \right)} \leq \sum_{t=1}^{T} ||vec(\mathbf{X}_{a_t, W^T})||_{A_{t-1}}^2 \leq 2\alpha_t \sqrt{dT/N \ln \left( \frac{T}{\delta} \right)}$$

(10) In the extreme case that $W$ is identity matrix, the $\mathbf{A}$ matrix in CoLin becomes a block diagonal matrix, which consists of $\mathbf{A}^{u,t}$, $u \in 1, \ldots, N$. Using the block diagonal matrix’s property that $\text{det}(\mathbf{A}) = \prod_{u} \text{det}(\mathbf{A}^{u,t})$, the regret can be further simplified as,

$$R(T) \leq \sum_{u=1}^{N} \sqrt{T \ln \left( \frac{\text{det}(\mathbf{A}^{u,t})}{{\lambda}_u \text{det}(\mathbf{A}^{u,t})} \right)} \leq \sqrt{\ln \left( \frac{T}{\delta} \right)} \lambda \text{det}(\mathbf{A}^{u,t}) \text{det}(\mathbf{A})$$

(11) which is $O(N \ln(T) \sqrt{T})$ and is identical with $N$ independent LinUCB.

(2) In the extreme case where $W$ is uniform, we have,

$$\mathbf{A}_t = \mathbf{A}_t + T vec(\mathbf{X} \mathbf{W}^T) vec(\mathbf{X} \mathbf{W}^T)^T$$

such that,

$$\text{det}(\mathbf{A}_t) = \text{det}(\mathbf{A}_t)(1 + T ||vec(\mathbf{X} \mathbf{W}^T)||^2_{A_{t-1}})$$

By applying Eq(12) and according to Eq (10) the regret bound of CoLin can be written as,

$$R(T) \leq N \sqrt{\ln \left( \frac{T}{\delta N} \right) \lambda \text{det}(\mathbf{A})} \text{det}(\mathbf{A})$$

which is $O(N \ln(T) \sqrt{T})$. 

□