Online Clustering of Bandits

Claudio Gentile CLAUDIO.GENTILE@UNINSUBRIA.IT
DiSTA, University of Insubria, Italy

Shuai Li SHUAILI.SLI@GMAIL.COM
DiSTA, University of Insubria, Italy

Giovanni Zappella ZAPPELLA@AMAZON.COM
Amazon Development Center Germany, Germany
(Work done when the author was PhD student at Univeristy of Milan)

Abstract

We introduce a novel algorithmic approach to content recommendation based on adaptive clustering of exploration-exploitation ("bandit") strategies. We provide a sharp regret analysis of this algorithm in a standard stochastic noise setting, demonstrate its scalability properties, and prove its effectiveness on a number of artificial and real-world datasets. Our experiments show a significant increase in prediction performance over state-of-the-art methods for bandit problems.

1. Introduction

Presenting personalized content to users is nowdays a crucial functionality for many online recommendation services. Due to the ever-changing set of available options, these services have to exhibit strong adaptation capabilities when trying to match users' preferences. Coarsely speaking, the underlying systems repeatedly learn a mapping between available content and users, the mapping being based on *context* information (that is, sets of features) which is typically extracted from both users and contents. The need to focus on content that raises the users' interest, combined with the need of exploring new content so as to globally improve users' experience, generates a well-known exploration-exploitation dilemma, which is commonly formalized as a multi-armed bandit problem (e.g., (Lai & Robbins, 1985; Auer et al., 2001; Audibert et al., 2009; Caron et al., 2012)). In particular, the contextual bandit methods (e.g., (Auer, 2002; Langford & Zhang, 2007; Li et al., 2010; Chu et al., 2011; Bogers, 2010; Abbasi-Yadkori et al., 2011; Crammer & Gentile, 2011; Krause & Ong, 2011; Seldin et al., 2011; Yue et al., 2012; Djolonga et al., 2013), and references therein) have rapidly become a reference algorithmic technique for implementing adaptive recommender systems.

Within the above scenarios, the widespread adoption of online social networks, where users are engaged in technology-mediated social interactions (making product endorsement and word-of-mouth advertising a common practice), raises further challenges and opportunities to content recommendation systems: On one hand, because of the mutual influence among friends, acquaintances, business partners, etc., users having strong ties are more likely to exhibit similar interests, and therefore similar behavior. On the other hand, the nature and scale of such interactions calls for adaptive algorithmic solutions which are also computationally affordable.

Incorporating social components into bandit algorithms can lead to a dramatic increase in the quality of recommendations. For instance, we may want to serve content to a group of users by taking advantage of an underlying network of social relationships among them. These social relationships can either be explicitly encoded in a graph, where adjacent nodes/users are deemed similar to one another, or implicitly contained in the data, and given as the outcome of an inference process that recognizes similarities across users based on their past behavior. Examples of the first approach are the recent works (Buccapatnam et al., 2013; Delporte et al., 2013; Cesa-Bianchi et al., 2013), where a social network structure over the users is assumed to be given that reflects actual interest similarities among users – see also (Caron & Bhagat, 2013; Valko et al., 2014) for recent usage of social information to tackle the so-called "cold-start" problem. Examples of the second approach are the more traditional collaborative-filtering (e.g., (Schafer et al., 1999)), content-based filtering, and hybrid approaches (e.g. (Burke, 2005)).

Both approaches have important drawbacks hindering their practical deployment. One obvious drawback of the "ex-

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

plicit network" approach is that the social network information may be misleading (see, e.g., the experimental evidence reported by (Delporte et al., 2013)), or simply unavailable. Moreover, even in the case when this information is indeed available and useful, the algorithmic strategies to implement the needed feedback sharing mechanisms might lead to severe scaling issues (Cesa-Bianchi et al., 2013), especially when the number of targeted users is large. A standard drawback of the "implicit network" approach of traditional recommender systems is that in many practically relevant scenarios (e.g., web-based), content universe and popularity often undergo dramatic changes, making these approaches difficult to apply.

In such settings, most notably in the relevant case when the involved users are many, it is often possible to identify a few subgroups or communities within which users share similar interests (Rashid et al., 2006; Buscher et al., 2012), thereby greatly facilitating the targeting of users by means of *group* recommendations. Hence the system need not learn a different model for each user of the service, but just a single model for each group.

In this paper, we carry out[1] a theoretical and experimental investigation of adaptive clustering algorithms for linear (contextual) bandits under the assumption that we have to serve content to a set of n users organized into $m << n$ groups (or *clusters*) such that users within each group tend to provide similar feedback to content recommendations. We give a $O(\sqrt{T})$ regret analysis holding in a standard stochastically linear setting for payoffs where, importantly, the hidden constants in the big-oh depend on m, rather than n, as well as on the geometry of the user models within the different clusters. The main idea of our algorithm is to use confidence balls of the users' models to both estimate user similarity, and to share feedback across (deemed similar) users. The algorithm adaptively interpolates between the case when we have a single instance of a contextual bandit algorithm making the same predictions for all users and the case when we have n-many instances providing fully personalized recommendations. We show that our algorithm can be implemented efficiently (the large n scenario being of special concern here) by means of off-the-shelf data-structures relying on random graphs. Finally, we test our algorithm on medium-size synthetic and real-world datasets, often reporting a significant increase in prediction performance over known state-of-the-art methods for bandit problems.

2. Learning Model

We assume the user behavior similarity is encoded as an *unknown* clustering of the users. Specifically, let $V = \{1, \ldots, n\}$ represent the set of n users. Then V can be partitioned into a small number m of clusters $V_1, V_2, \ldots, V_m$, with $m << n$, such that users lying in the same cluster share similar behavior and users lying in different clusters have different behavior. The actual partition of V (including the number of clusters m) and the common user behavior within each cluster are unknown to the learner, and have to be inferred on the fly.

Learning proceeds in a sequential fashion: At each round $t = 1, 2, \ldots$, the learner receives a user index $i_t \in V$ together with a set of context vectors $C_{i_t} = \{\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2}, \ldots, \boldsymbol{x}_{t,c_t}\} \subseteq \mathbb{R}^d$. The learner then selects some $\bar{\boldsymbol{x}}_t = \boldsymbol{x}_{t,k_t} \in C_{i_t}$ to recommend to user i_t, and observes some payoff $a_t \in \mathbb{R}$, which is a function of both i_t and the recommended $\bar{\boldsymbol{x}}_t$. The following assumptions are made on how index i_t, set C_{i_t}, and payoff a_t are generated in round t. Index i_t represents the user to be served by the system, and we assume i_t is selected uniformly at random[2] from V. Once i_t is selected, the number of context vectors c_t in C_{i_t} is generated arbitrarily as a function of past indices $i_1, \ldots, i_{t-1}$, payoffs $a_1, \ldots, a_{t-1}$, and sets $C_{i_1}, \ldots, C_{i_{t-1}}$, as well as the current index i_t. Then the sequence $\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2}, \ldots, \boldsymbol{x}_{t,c_t}$ of context vectors within C_{i_t} is generated i.i.d. (conditioned on i_t, c_t and all past indices $i_1, \ldots, i_{t-1}$, payoffs $a_1, \ldots, a_{t-1}$, and sets $C_{i_1}, \ldots, C_{i_{t-1}}$) from a random process on the surface of the unit sphere, whose process matrix $\mathbb{E}[XX^\top]$ is full rank, with minimal eigenvalue $\lambda > 0$. Further assumptions on the process matrix $\mathbb{E}[XX^\top]$ are made later on. Finally, payoffs are generated by noisy versions of unknown linear functions of the context vectors. That is, we assume each cluster V_j, $j = 1, \ldots, m$, hosts an unknown parameter vector $\boldsymbol{u}_j \in \mathbb{R}^d$ which is common to each user $i \in V_j$. Then the payoff value $a_i(\boldsymbol{x})$ associated with user i and context vector $\boldsymbol{x} \in \mathbb{R}^d$ is given by the random variable

$$a_i(\boldsymbol{x}) = \boldsymbol{u}_{j(i)}^\top \boldsymbol{x} + \epsilon_{j(i)}(\boldsymbol{x}) \,,$$

where $j(i) \in \{1, 2, \ldots, m\}$ is the index of the cluster that node i belongs to, and $\epsilon_{j(i)}(\boldsymbol{x})$ is a conditionally zero-mean and bounded variance noise term. Specifically, denoting by $\mathbb{E}_t[\,\cdot\,]$ the conditional expectation $\mathbb{E}\big[\,\cdot\,\big|\,(i_1, C_{i_1}, a_1), \ldots, (i_{t-1}, C_{i_{t-1}}, a_{t-1}), i_t\,\big]$, we assume that for any fixed $j \in \{1, \ldots, m\}$ and $\boldsymbol{x} \in \mathbb{R}^d$, the variable $\epsilon_j(\boldsymbol{x})$ is such that $\mathbb{E}_t[\epsilon_j(\boldsymbol{x}) | \,\boldsymbol{x}\,] = 0$ and $\mathbb{V}_t\big[\epsilon_j(\boldsymbol{x}) | \,\boldsymbol{x}\,\big] \le \sigma^2$, where $\mathbb{V}_t[\,\cdot\,]$ is a shorthand for the conditional variance $\mathbb{V}\big[\,\cdot\,\big|\,(i_1, C_{i_1}, a_1), \ldots, (i_{t-1}, C_{i_{t-1}}, a_{t-1}), i_t\,\big]$ of the variable at argument. So we clearly have $\mathbb{E}_t[a_i(\boldsymbol{x}) | \,\boldsymbol{x}\,] = \boldsymbol{u}_{j(i)}^\top \boldsymbol{x}$ and $\mathbb{V}_t\big[a_i(\boldsymbol{x}) | \,\boldsymbol{x}\,\big] \le \sigma^2$. Therefore, $\boldsymbol{u}_{j(i)}^\top \boldsymbol{x}$ is the expected payoff observed at user i for context vector $\boldsymbol{x}$. In the special case when the noise $\epsilon_{j(i)}(\boldsymbol{x})$ is a bounded random variable taking values in the range $[-1, 1]$, this implies $\sigma^2 \le 1$. We will make throughout the assumption

[1] Due to space limitations, we postpone the discussion of related work to the supplementary material.

[2] Any other distribution that insures a positive probability of visiting each node of V would suffice here.

that $a_i(\boldsymbol{x}) \in [-1, 1]$ for all $i \in V$ and $\boldsymbol{x}$. Notice that this implies $-1 \le \boldsymbol{u}_{j(i)}^\top \boldsymbol{x} \le 1$ for all $i \in V$ and $\boldsymbol{x}$. Finally, we assume well-separatedness among the clusters, in that $||\boldsymbol{u}_j - \boldsymbol{u}_{j'}|| \ge \gamma > 0$ for all $j \ne j'$. We define the regret r_t of the learner at time t as

$$r_t = \left(\max_{\boldsymbol{x} \in C_{i_t}} \ \boldsymbol{u}_{j(i_t)}^\top \boldsymbol{x} \right) - \boldsymbol{u}_{j(i_t)}^\top \bar{\boldsymbol{x}}_t \ .$$

We are aimed at bounding with high probability (over the variables i_t, $\boldsymbol{x}_{t,k}$, $k = 1, \ldots, c_t$, and the noise variables $\epsilon_{j(i_t)}$) the cumulative regret $\sum_{t=1}^T r_t$. The kind of regret bound we would like to obtain (we call it the *reference* bound) is one where the clustering structure of V (i.e., the partition of V into $V_1, \ldots, V_m$) is known to the algorithm ahead of time, and we simply view each one of the m clusters as an independent bandit problem. In this case, a standard contextual bandit analysis (Auer, 2002; Chu et al., 2011; Abbasi-Yadkori et al., 2011) shows that, as T grows large, the cumulative regret $\sum_{t=1}^T r_t$ can be bounded with high probability as[3]

$$\textstyle\sum_{t=1}^T r_t = \widetilde{O}\Big(\sum_{j=1}^m \Big(\sigma\, d + ||\boldsymbol{u}_j||\, \sqrt{d} \Big)\, \sqrt{T} \Big) \ .$$

For simplicity, we shall assume that $||\boldsymbol{u}_j|| = 1$ for all $j = 1, \ldots, m$. Now, a more careful analysis exploiting our assumption about the randomness of i_t (see the supplementary material) reveals that one can replace the $\sqrt{T}$ term contributed by each bandit j by a term of the form $\sqrt{T}\, \left(\frac{1}{m} + \sqrt{\frac{|V_j|}{n}} \right)$, so that under our assumptions the reference bound becomes

$$\sum_{t=1}^T r_t = \widetilde{O}\left(\Big(\sigma\, d + \sqrt{d} \Big)\, \sqrt{T} \Big(1 + \sum_{j=1}^m \sqrt{\frac{|V_j|}{n}} \Big) \right) \ . \quad (1)$$

Observe the dependence of this bound on the size of clusters V_j. The worst-case scenario is when we have m clusters of the same size $\frac{n}{m}$, resulting in the bound

$$\textstyle\sum_{t=1}^T r_t = \widetilde{O}\left(\Big(\sigma\, d + \sqrt{d} \Big)\, \sqrt{m\, T} \right) \ .$$

At the other extreme lies the easy case when we have a single big cluster and many small ones. For instance, $|V_1| = n - m + 1$, and $|V_2| = |V_3| = \ldots |V_m| = 1$, for $m << n$, gives

$$\textstyle\sum_{t=1}^T r_t = \widetilde{O}\Big(\Big(\sigma\, d + \sqrt{d} \Big)\, \sqrt{T}\, \Big(1 + \frac{m}{\sqrt{n}} \Big) \Big) \ .$$

A relevant geometric parameter of the set of $\boldsymbol{u}_j$ is the *sum of distances* $SD(\boldsymbol{u}_j)$ of a given vector $\boldsymbol{u}_j$ w.r.t. the set of vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m$, which we define as $SD(\boldsymbol{u}_j) = \sum_{\ell=1}^m ||\boldsymbol{u}_j - \boldsymbol{u}_\ell||$. If it is known that $SD(\boldsymbol{u}_j)$ is small for all j, one can modify the abovementioned independent bandit algorithm, by letting the bandits share signals, as is done, e.g., in (Cesa-Bianchi et al., 2013). This allows one to exploit the vicinity of the $\boldsymbol{u}_j$ vectors, and roughly replace $1 + \sum_{j=1}^m \sqrt{\frac{|V_j|}{n}}$ in (1) by a quantity also depending on the mutual distances $||\boldsymbol{u}_j - \boldsymbol{u}_{j'}||$ among cluster vectors. However, this improvement is obtained at the cost of a substantial increase of running time (Cesa-Bianchi et al., 2013). In our analysis (Theorem 1 in Section 3), we would like to leverage both the geometry of the clusters, as encoded by vectors $\boldsymbol{u}_j$, and the relative size $|V_j|$ of the clusters, with no prior knowledge of m (or γ), and without too much extra computational burden.

3. The Algorithm

Our algorithm, called Cluster of Bandits (CLUB), is described in Figure 1. In order to describe the algorithm we find it convenient to re-parameterize the problem described in Section 2, and introduce n parameter vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n$, one per node, where nodes within the same cluster V_j share the same vector. An illustrative example is given in Figure 2.

The algorithm maintains at time t an estimate $\boldsymbol{w}_{i,t}$ for vector $\boldsymbol{u}_i$ associated with user $i \in V$. Vectors $\boldsymbol{w}_{i,t}$ are updated based on the payoff signals, similar to a standard linear bandit algorithm (e.g., (Chu et al., 2011)) operating on the context vectors contained in C_{i_t}. Every user i in V hosts a linear bandit algorithm like the one described in (Cesa-Bianchi et al., 2013). One can see that the prototype vector $\boldsymbol{w}_{i,t}$ is the result of a standard linear least-squares approximation to the corresponding unknown parameter vector $\boldsymbol{u}_i$. In particular, $\boldsymbol{w}_{i,t-1}$ is defined through the inverse correlation matrix $M_{i,t-1}^{-1}$, and the additively-updated vector $\boldsymbol{b}_{i,t-1}$. Matrices $M_{i,t}$ are initialized to the $d \times d$ identity matrix, and vectors $\boldsymbol{b}_{i,t}$ are initialized to the d-dimensional zero vector. In addition, the algorithm maintains at time t an undirected graph $G_t = (V, E_t)$ whose nodes are precisely the users in V. The algorithm starts off from the complete graph, and progressively erases edges based on the evolution of vectors $\boldsymbol{w}_{i,t}$. The graph is intended to encode the current partition of V by means of the *connected components* of G_t. We denote by $\hat{V}_{1,t}, \hat{V}_{2,t}, \ldots, \hat{V}_{m_t,t}$ the partition of V induced by the connected components of G_t. Initially, we have $m_1 = 1$ and $\hat{V}_{1,1} = V$. The clusters $\hat{V}_{1,1}, \hat{V}_{2,t}, \ldots, \hat{V}_{m_t,t}$ (henceforth called the *current* clusters) are indeed meant to estimate the underlying true partition $V_1, V_2, \ldots, V_m$, henceforth called the *underlying* or *true* clusters.

At each time $t = 1, 2, \ldots$, the algorithm receives the index i_t of the user to serve, and the associated context vectors $\boldsymbol{x}_{t,1}, \ldots, \boldsymbol{x}_{t,c_t}$ (the set C_{i_t}), and must select one among them. In doing so, the algorithm first determines which cluster (among $\hat{V}_{1,1}, \hat{V}_{2,t}, \ldots, \hat{V}_{m_t,t}$) node i_t belongs to, call this cluster $\hat{V}_{\hat{j}_t,t}$, then builds the aggregate weight vec-

[3] The $\widetilde{O}$-notation hides logarithmic factors.

```
Input: Exploration parameter $\alpha > 0$; edge deletion parameter $\alpha_2 > 0$
Init:
  • $\boldsymbol{b}_{i,0} = \boldsymbol{0} \in \mathbb{R}^d$ and $M_{i,0} = I \in \mathbb{R}^{d\times d},\ i = 1,\dots n$;
  • Clusters $\hat{V}_{1,1} = V$, number of clusters $m_1 = 1$;
  • Graph $G_1 = (V, E_1)$, $G_1$ is connected over $V$.
for $t = 1, 2, \dots, T$ do
  Set $\boldsymbol{w}_{i,t-1} = M_{i,t-1}^{-1}\boldsymbol{b}_{i,t-1}, \quad i = 1,\dots,n$;
  Receive $i_t \in V$, and get context $C_{i_t} = \{\boldsymbol{x}_{t,1},\dots,\boldsymbol{x}_{t,c_t}\}$;
  Determine $\widehat{j}_t \in \{1,\dots,m_t\}$ such that $i_t \in \hat{V}_{\widehat{j}_t,t}$, and set
      $\bar{M}_{\widehat{j}_t,t-1} = I + \sum_{i\in\hat{V}_{\widehat{j}_t,t}} (M_{i,t-1} - I)$,
      $\bar{\boldsymbol{b}}_{\widehat{j}_t,t-1} = \sum_{i\in\hat{V}_{\widehat{j}_t,t}} \boldsymbol{b}_{i,t-1}$,
      $\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1} = \bar{M}_{\widehat{j}_t,t-1}^{-1}\bar{\boldsymbol{b}}_{\widehat{j}_t,t-1}$ ;
    Set $k_t = \operatorname{argmax}_{k=1,\dots,c_t} \left(\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}^\top\boldsymbol{x}_{t,k} + \mathrm{CB}_{\widehat{j}_t,t-1}(\boldsymbol{x}_{t,k})\right)$,
  $\mathrm{CB}_{j,t-1}(\boldsymbol{x}) = \alpha\sqrt{\boldsymbol{x}^\top\bar{M}_{j,t-1}^{-1}\boldsymbol{x}\ \log(t+1)}$,
    $\bar{M}_{j,t-1} = I + \sum_{i\in\hat{V}_{j,t}} (M_{i,t-1} - I), \quad j = 1,\dots,m_t$ .
  Observe payoff $a_t \in [-1, 1]$;
  Update weights:
  •   $M_{i_t,t} = M_{i_t,t-1} + \bar{\boldsymbol{x}}_t\bar{\boldsymbol{x}}_t^\top$,
  •   $\boldsymbol{b}_{i_t,t} = \boldsymbol{b}_{i_t,t-1} + a_t\bar{\boldsymbol{x}}_t$,
  •   Set $M_{i,t} = M_{i,t-1},\ \boldsymbol{b}_{i,t} = \boldsymbol{b}_{i,t-1}$ for all $i \neq i_t$ ;
  Update clusters:
  •   Delete from $E_t$ all $(i_t, \ell)$ such that
          $||\boldsymbol{w}_{i_t,t-1} - \boldsymbol{w}_{\ell,t-1}|| > \widetilde{\mathrm{CB}}_{i_t,t-1} + \widetilde{\mathrm{CB}}_{\ell,t-1}$ ,
          $\widetilde{\mathrm{CB}}_{i,t-1} = \alpha_2\sqrt{\frac{1+\log(1+T_{i,t-1})}{1+T_{i,t-1}}}$,
          $T_{i,t-1} = |\{s \le t-1 \,:\, i_s = i\}|, \qquad i \in V$;
  •   Let $E_{t+1}$ be the resulting set of edges, set $G_{t+1} = (V, E_{t+1})$, and compute associated clusters $\hat{V}_{1,t+1}, \hat{V}_{2,t+1}, \dots, \hat{V}_{m_{t+1},t+1}$ .
end for
```

Figure 1. Pseudocode of the CLUB algorithm. The confidence functions $\mathrm{CB}_{j,t-1}$ and $\widetilde{\mathrm{CB}}_{i,t-1}$ are simplified versions of their "theoretical" counterparts $\mathrm{TCB}_{j,t-1}$ and $\widetilde{\mathrm{TCB}}_{i,t-1}$, defined later on. The factors α and α_2 are used here as tunable parameters that bridge the simplified versions to the theoretical ones.

tor $\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}$ by taking prior $\bar{\boldsymbol{x}}_s$, $s < t$, such that $i_s \in \hat{V}_{\widehat{j}_t,t}$, and computing the least squares approximation as if all nodes $i \in \hat{V}_{\widehat{j}_t,t}$ have been collapsed into one. It is weight vector $\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}$ that the algorithm uses to select k_t. In particular,

$$k_t = \operatorname*{argmax}_{k=1,\dots,c_t} \left(\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}^\top\boldsymbol{x}_{t,k} + \mathrm{CB}_{\widehat{j}_t,t-1}(\boldsymbol{x}_{t,k})\right) .$$

The quantity $\mathrm{CB}_{\widehat{j}_t,t-1}(\boldsymbol{x})$ is a version of the upper confidence bound in the approximation of $\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}$ to a suitable combination of vectors $\boldsymbol{u}_i$, $i \in \hat{V}_{\widehat{j}_t,t}$ – see the supplementary material for details.

Once this selection is done and the associated payoff a_t is observed, the algorithm uses the selected vector $\bar{\boldsymbol{x}}_t$ for updating $M_{i_t,t-1}$ to $M_{i_t,t}$ via a rank-one adjustment, and for turning vector $\boldsymbol{b}_{i_t,t-1}$ to $\boldsymbol{b}_{i_t,t}$ via an additive update whose learning rate is precisely a_t. Notice that the update is only performed at node i_t, since for all other $i \neq i_t$ we have $\boldsymbol{w}_{i,t} = \boldsymbol{w}_{i,t-1}$. However, this update at i_t will also implicitly update the aggregate weight vector $\bar{\boldsymbol{w}}_{\widehat{j}_{t+1},t}$ associated with cluster $\hat{V}_{\widehat{j}_{t+1},t+1}$ that node i_t will happen to belong to in the next round. Finally, the cluster structure is possibly modified. At this point CLUB compares, for all existing edges $(i_t, \ell) \in E_t$, the distance $||\boldsymbol{w}_{i_t,t-1} - \boldsymbol{w}_{\ell,t-1}||$ between vectors $\boldsymbol{w}_{i_t,t-1}$ and $\boldsymbol{w}_{\ell,t-1}$ to the quantity $\widetilde{\mathrm{CB}}_{i_t,t-1} + \widetilde{\mathrm{CB}}_{\ell,t-1}$. If the above distance is significantly large (and $\boldsymbol{w}_{i_t,t-1}$ and $\boldsymbol{w}_{\ell,t-1}$ are good approximations to the respective underlying vectors $\boldsymbol{u}_{i_t}$ and $\boldsymbol{u}_\ell$), then this is a good indication that $\boldsymbol{u}_{i_t} \neq \boldsymbol{u}_\ell$ (i.e., that node i_t and node ℓ cannot belong to the same true cluster), so that edge (i_t, ℓ) gets deleted. The new graph G_{t+1}, and the induced partitioning clusters $\hat{V}_{1,t+1}, \hat{V}_{2,t+1}, \dots, \hat{V}_{m_{t+1},t+1}$, are then computed, and a new round begins.

3.1. Implementation

In implementing the algorithm in Figure 1, the reader should bear in mind that we are expecting n (the number of users) to be quite large, d (the number of features of each item) to be relatively small, and m (the number of true clusters) to be very small compared to n. With this in mind, the algorithm can be implemented by storing a least-squares estimator $\boldsymbol{w}_{i,t-1}$ at each node $i \in V$, an aggregate least squares estimator $\bar{\boldsymbol{w}}_{\widehat{j}_t,t-1}$ for each current cluster $\widehat{j}_t \in \{1,\dots,m_t\}$, and an extra data-structure which is able to perform decremental dynamic connectivity. Fast implementations of such data-structures are those studied by (Thorup, 1997; Kapron et al., 2013) (see also the research thread referenced therein). One can show (see the supplementary material) that in T rounds we have an overall (expected) running time

$$O\Big(T\left(d^2 + \frac{|E_1|}{n}\,d\right) + m\,(n\,d^2 + d^3) + |E_1| + \min\{n^2, |E_1|\,\log n\} + \sqrt{n\,|E_1|}\,\log^{2.5} n\Big) . \qquad (2)$$

Notice that the above is $n \cdot \mathrm{poly}(\log n)$, if so is $|E_1|$. In addition, if T is large compared to n and d, the average running time per round becomes $O(d^2 + d \cdot \mathrm{poly}(\log n))$. As for memory requirements, this implementation takes $O(n\,d^2 + m\,d^2 + |E_1|) = O(n\,d^2 + |E_1|)$. Again, this is $n \cdot \mathrm{poly}(\log n)$ if so is $|E_1|$.

3.2. Regret Analysis

Our analysis relies on the high probability analysis contained in (Abbasi-Yadkori et al., 2011) (Theorems 1 and 2 therein). The analysis (Theorem 1 below) is carried out in the case when the initial graph G_1 is the complete graph. However, if the true clusters are sufficiently large, then we can show (see Remark 4) that a formal statement can be made even if we start off from sparser random graphs, with substantial time and memory savings.

The analysis actually refers to a version of the algorithm where the confidence bound functions $\mathrm{CB}_{j,t-1}(\cdot)$ and $\widetilde{\mathrm{CB}}_{i,t-1}$ in Figure 1 are replaced by their "theoretical" counterparts $\mathrm{TCB}_{j,t-1}(\cdot)$, and $\widetilde{\mathrm{TCB}}_{i,t-1}$, respectively,[4] which are defined as follows. Set for brevity

$$A_\lambda(T,\delta) = \left(\frac{\lambda T}{4} - 8\log\left(\frac{T+3}{\delta}\right) - 2\sqrt{T\log\left(\frac{T+3}{\delta}\right)}\right)_+$$

where $(x)_+ = \max\{x,0\}$, $x \in \mathbb{R}$. Then, for $j = 1,\ldots,m_t$,

$$\mathrm{TCB}_{j,t-1}(\boldsymbol{x}) = \sqrt{\boldsymbol{x}^\top \bar{M}_{j,t-1}^{-1}\boldsymbol{x}}\left(\sigma\sqrt{2\log\frac{|\bar{M}_{j,t-1}|}{\delta/2}} + 1\right), \tag{3}$$

being $|\cdot|$ the determinant of the matrix at argument, and, for $i \in V$,

$$\widetilde{\mathrm{TCB}}_{i,t-1} = \frac{\sigma\sqrt{2d\,\log t + 2\log(2/\delta)} + 1}{\sqrt{1 + A_\lambda(T_{i,t-1}, \delta/(2nd))}}. \tag{4}$$

Recall the difference between *true* clusters $V_1,\ldots,V_m$ and *current* clusters $\hat{V}_{1,t},\ldots,\hat{V}_{m_t,t}$ maintained by the algorithm at time t. Consistent with this difference, we let $G = (V,E)$ be the true underlying graph, made up of the m disjoint cliques over the sets of nodes $V_1,\ldots,V_m \subseteq V$, and $G_t = (V,E_t)$ be the one kept by the algorithm – see again Figure 2 for an illustration of how the algorithm works. The following is the main theoretical result of this paper,[5] where additional conditions are needed on the process X generating the context vectors.

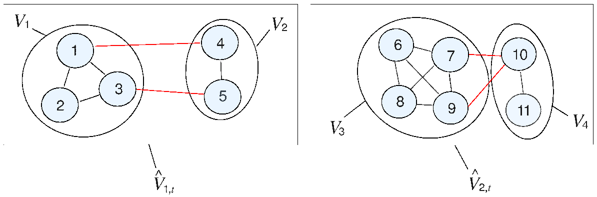

Figure 2. A true underlying graph $G = (V,E)$ made up of $n = |V| = 11$ nodes, and $m = 4$ true clusters $V_1 = \{1,2,3\}$, $V_2 = \{4,5\}$, $V_3 = \{6,7,8,9\}$, and $V_4 = \{10,11\}$. There are $m_t = 2$ current clusters $\hat{V}_{1,t}$ and $\hat{V}_{2,t}$. The black edges are the ones contained in E, while the red edges are those contained in $E_t \setminus E$. The two current clusters also correspond to the two connected components of graph $G_t = (V,E_t)$. Since aggregate vectors $\bar{\boldsymbol{w}}_{j,t}$ are build based on current cluster membership, if for instance, $i_t = 3$, then $\widehat{j}_t = 1$, so $\bar{M}_{1,t-1} = I + \sum_{i=1}^5 (M_{i,t-1} - I)$, $\bar{\boldsymbol{b}}_{1,t-1} = \sum_{i=1}^5 \boldsymbol{b}_{i,t-1}$, and $\bar{\boldsymbol{w}}_{1,t-1} = \bar{M}_{1,t-1}^{-1}\bar{\boldsymbol{b}}_{1,t-1}$.

Theorem 1. *Let the CLUB algorithm of Figure 1 be run on the initial complete graph $G_1 = (V,E_1)$, whose nodes $V = \{1,\ldots,n\}$ can be partitioned into m clusters $V_1,\ldots,V_m$ where, for each $j = 1,\ldots,m$, nodes within cluster V_j host the same vector $\boldsymbol{u}_j$, with $||\boldsymbol{u}_j|| = 1$ for $j = 1,\ldots,m$, and $||\boldsymbol{u}_j - \boldsymbol{u}_{j'}|| \geq \gamma > 0$ for any $j \neq j'$. Denote by $v_j = |V_j|$ the cardinality of cluster V_j. Let the $\mathrm{CB}_{j,t}(\cdot)$ function in Figure 1 be replaced by the $\mathrm{TCB}_{j,t}(\cdot)$ function defined in (3), and $\widetilde{\mathrm{CB}}_{i,t}$ be replaced by $\widetilde{\mathrm{TCB}}_{i,t}$ defined in (4). In both $\mathrm{TCB}_{j,t}$ and $\widetilde{\mathrm{TCB}}_{i,t}$, let δ therein be replaced by $\delta/10.5$. Let, at each round t, context vectors $C_{i_t} = \{\boldsymbol{x}_{t,1},\ldots,\boldsymbol{x}_{t,c_t}\}$ being generated i.i.d. (conditioned on i_t, c_t and all past indices $i_1,\ldots,i_{t-1}$, payoffs $a_1,\ldots,a_{t-1}$, and sets $C_{i_1},\ldots,C_{i_{t-1}}$) from a random process X such that $||X|| = 1$, $\mathbb{E}[XX^\top]$ is full rank, with minimal eigenvalue $\lambda > 0$. Moreover, for any fixed unit vector $\boldsymbol{z} \in \mathbb{R}^d$, let the random variable $(\boldsymbol{z}^\top X)^2$ be (conditionally) sub-Gaussian with variance parameter $\nu^2 = \mathbb{V}_t\big[(\boldsymbol{z}^\top X)^2 \,|\, c_t\big] \leq \frac{\lambda^2}{8\log(4c)}$, with $c_t \leq c$ for all t. Then with probability at least $1-\delta$ the cumulative regret satisfies*

$$\sum_{t=1}^T r_t = \widetilde{O}\Bigg((\sigma\sqrt{d}+1)\sqrt{m}\left(\frac{n}{\lambda^2} + \sqrt{T}\Big(1 + \sum_{j=1}^m\sqrt{\frac{v_j}{\lambda n}}\Big)\right) + \left(\frac{n}{\lambda^2} + \frac{n\,\sigma^2\,d}{\lambda\gamma^2}\right)\mathbb{E}[SD(\boldsymbol{u}_{i_t})] + m\Bigg)$$

$$= \widetilde{O}\left((\sigma\sqrt{d}+1)\sqrt{m\,T}\left(1 + \sum_{j=1}^m\sqrt{\frac{v_j}{\lambda n}}\right)\right), \tag{5}$$

as T grows large. In the above, the $\widetilde{O}$-notation hides $\log(1/\delta)$, $\log m$, $\log n$, and $\log T$ factors.

Remark 1. *A close look at the cumulative regret bound presented in Theorem 1 reveals that this bound is made up of three main terms: The first term is of the form*

$$(\sigma\sqrt{dm} + \sqrt{m})\,\frac{n}{\lambda^2} + m\,.$$

This term is constant with T, and essentially accounts for the transient regime due to the convergence of the minimal eigenvalues of $\bar{M}_{j,t}$ and $M_{i,t}$ to the corresponding minimal eigenvalue λ of $\mathbb{E}[XX^\top]$. The second term is of the form

$$\left(\frac{n}{\lambda^2} + \frac{n\,\sigma^2\,d}{\lambda\gamma^2}\right)\mathbb{E}[SD(\boldsymbol{u}_{i_t})]\,.$$

This term is again constant with T, but it depends through $\mathbb{E}[SD(\boldsymbol{u}_{i_t})]$ on the geometric properties of the set of $\boldsymbol{u}_j$ as well as on the way such $\boldsymbol{u}_j$ interact with the cluster sizes v_j. Specifically,

[4] Notice that, in all our notation, index i always ranges over nodes, while index j always ranges over clusters. Accordingly, the quantities $\widetilde{\mathrm{CB}}_{i,t}$ and $\widetilde{\mathrm{TCB}}_{i,t}$ are always associates with node $i \in V$, while the quantities $\mathrm{CB}_{j,t-1}(\cdot)$ and $\mathrm{TCB}_{j,t-1}(\cdot)$ are always associates with clusters $j \in \{1,\ldots,m_t\}$.

[5] The proof is provided in the supplementary material.

$$\mathbb{E}[SD(\boldsymbol{u}_{i_t})] = \sum_{j=1}^{m} \frac{v_j}{n} \sum_{j'=1}^{m} ||\boldsymbol{u}_j - \boldsymbol{u}_{j'}|| \,.$$

Hence this term is small if, say, among the m clusters, a few of them together cover almost all nodes in V (this is a typical situation in practice) and, in addition, the corresponding $\boldsymbol{u}_j$ are close to one another. This term accounts for the hardness of learning the true underlying clustering through edge pruning. We also have an inverse dependence on γ^2, which is likely due to an artifact of our analysis. Recall that γ is not known to our algorithm. Finally, the third term is the one characterizing the asymptotic behavior of our algorithm as $T \to \infty$, its form being just (5). It is instructive to compare this term to the reference bound (1) obtained by assuming prior knowledge of the cluster structure. Broadly speaking, (5) has an extra $\sqrt{m}$ factor,[6] *and replaces a factor $\sqrt{d}$ in (1) by the larger factor $\sqrt{\frac{1}{\lambda}}$.*

Remark 2. *The reader should observe that a similar algorithm as CLUB can be designed that starts off from the empty graph instead, and progressively draws edges (thereby merging connected components and associated aggregate vectors) as soon as two nodes host individual vectors $\boldsymbol{w}_{i,t}$ which are close enough to one another. This would have the advantage to lean on even faster data-structures for maintaining disjoint sets (e.g., (Cormen et al., 1990)[Ch. 22]), but has also the significant drawback of requiring prior knowledge of the separation parameter γ. In fact, it would not be possible to connect two previously unconnected nodes without knowing something about this parameter. A regret analysis similar to the one in Theorem 1 exists, though our current understanding is that the cumulative regret would depend linearly on $\sqrt{n}$ instead of $\sqrt{m}$. Intuitively, this algorithm is biased towards a large number of true clusters, rather than a small number.*

Remark 3. *A data-dependent variant of the CLUB algorithm can be designed and analyzed which relies on data-dependent clusterability assumptions of the set of users with respect to a set of context vectors. These data-dependent assumptions allow us to work in a fixed design setting for the sequence of context vectors $\boldsymbol{x}_{t,k}$, and remove the sub-Gaussian and full-rank hypotheses regarding $\mathbb{E}[XX^\top]$. On the other hand, they also require that the power of the adversary generating context vectors be suitably restricted. See the supplementary material for details.*

Remark 4. *Last but not least, we would like to stress that the same analysis contained in Theorem 1 extends to the case when we start off from a p-random Erdos-Renyi initial graph $G_1 = (V, E_1)$, where p is the independent probability that two nodes are connected by an edge in G_1. Translated into our context, a classical result on random graphs due to (Karger, 1994) reads as follows.*

Lemma 1. *Given $V = \{1, \ldots, n\}$, let $V_1, \ldots, V_m$ be a partition of V, where $|V_j| \geq s$ for all $j = 1, \ldots, m$. Let $G_1 = (V, E_1)$ be a p-random Erdos-Renyi graph with $p \geq \frac{12 \log(6n^2/\delta)}{s-1}$. Then with probability at least $1-\delta$ (over the random draw of edges), all m subgraphs induced by true clusters $V_1, \ldots, V_m$ on G_1 are connected in G_1.*

For instance, if $|V_j| = \beta \frac{n}{m}$, $j = 1, \ldots, m$, for some constant $\beta \in (0,1)$, then it suffices to have $|E_1| = O\left(\frac{m\, n\, \log(n/\delta)}{\beta}\right)$. Under these assumptions, if the initial graph G_1 is such a random graph, it is easy to show that Theorem 1 still holds. As mentioned in Section 3.1 (Eq. (2) therein), the striking advantage of beginning with a sparser connected graph than the complete graph is computational, since we need not handle anymore a (possibly huge) data-structure having n^2-many items. In our experiments, described next, we set $p = \frac{3 \log n}{n}$, so as to be reasonably confident that G_1 is (at the very least) connected.

4. Experiments

We tested our algorithm on both artificial and freely available real-world datasets against standard bandit baselines.

4.1. Datasets

Artificial datasets. We firstly generated synthetic datasets, so as to have a more controlled experimental setting. We tested the relative performance of the algorithms along different axes: number of underlying clusters, balancedness of cluster sizes, and amount of payoff noise. We set $c_t = 10$ for all $t = 1, \ldots, T$, with time horizon $T = 5,000 + 50,000$, $d = 25$, and $n = 500$. For each cluster V_j of users, we created a random unit norm vector $\boldsymbol{u}_j \in \mathbb{R}^d$. All d-dimensional context vectors $\boldsymbol{x}_{t,k}$ have then been generated uniformly at random on the surface of the Euclidean ball. The payoff value associated with cluster vector $\boldsymbol{u}_j$ and context vector $\boldsymbol{x}_{t,k}$ has been generated by perturbing the inner product $\boldsymbol{u}_j^\top \boldsymbol{x}_{t,k}$ through an additive white noise term ϵ drawn uniformly at random across the interval $[-\sigma, \sigma]$. It is the value of σ that determines the amount of payoff noise. The two remaining parameters are the number of clusters m and the clusters' relative size. We assigned to cluster V_j a number of users $|V_j|$ calculated as[7] $|V_j| = n \frac{j^{-z}}{\sum_{\ell=1}^{m} \ell^{-z}}$, $j = 1, \ldots, m$, with $z \in \{0, 1, 2, 3\}$, so that $z = 0$ corresponds to equally-sized clusters, and $z = 3$ yields highly unbalanced cluster sizes. Finally, the sequence of served users i_t is generated uniformly at random over the n users.

LastFM & Delicious datasets. These datasets are extracted from the music streaming service Last.fm and the social bookmarking web service Delicious. The LastFM dataset contains $n = 1{,}892$ nodes, and 17,632 items

[6] This extra factor could be eliminated at the cost of having a higher second term in the bound, which does not leverage the geometry of the set of $\boldsymbol{u}_j$.

[7] We took the integer part in this formula, and reassigned the remaining fractionary parts of users to the first cluster.

(artists). This dataset contains information about the listened artists, and we used this information to create payoffs: if a user listened to an artist at least once the payoff is 1, otherwise the payoff is 0. Delicious is a dataset with $n = 1{,}861$ users, and 69,226 items (URLs). The payoffs were created using the information about the bookmarked URLs for each user: the payoff is 1 if the user bookmarked the URL, otherwise the payoff is 0.[8] These two datasets are inherently different: on Delicious, payoffs depend on users more strongly than on LastFM, that is, there are more popular artists whom everybody listens to than popular websites which everybody bookmarks. LastFM is a "few hits" scenario, while Delicious is a "many niches" scenario, making a big difference in recommendation practice. Preprocessing was carried out by closely following previous experimental settings, like the one in (Cesa-Bianchi et al., 2013). In particular, we only retained the first 25 principal components of the context vectors resulting from a tf-idf representation of the available items, so that on both datasets $d = 25$. We generated random context sets C_{i_t} of size $c_t = 25$ for all t by selecting index i_t at random over the n users, then picking 24 vectors at random from the available items, and one among those with nonzero payoff for user i_t.[9] We repeated this process $T = 5,000 + 50,000$ times for the two datasets.

Yahoo dataset. We extracted two datasets from the one adopted by the "ICML 2012 Exploration and Exploitation 3 Challenge"[10] for news article recommendation. Each user is represented by a 136-dimensional binary feature vector, and we took this feature vector as a proxy for the identity of the user. We operated on the first week of data. After removing "empty" users,[11] this gave rise to a dataset of $8,362,905$ records, corresponding to $n = 713,862$ distinct users. The overall number of distinct news items turned out to be 323, c_t changing from round to round, with a maximum of 51, and a median of 41. The news items have no features, hence they have been represented as d-dimensional *versors*, with $d = 323$. Payoff values a_t are either 0 or 1 depending on whether the logged web system which these data refer to has observed a positive (click) or negative (no-click) feedback from the user in round t. We then extracted the two datasets "5k users" and "18k users" by filtering out users that have occurred less than 100 times and less than 50 times, respectively. Since the system's recommendation need not coincide with the recommendation issued by the algorithms we tested, we could only retain the records on which the two recommendations were indeed the same. Because records are discarded on the fly, the actual number of retained records changes across algorithms, but it is about $50,000$ for the "5k users" version and about $70,000$ for the "18k users" version.

4.2. Algorithms

We compared CLUB with two main competitors: LinUCB-ONE and LinUCB-IND. Both competitors are members of the LinUCB family of algorithms (Auer, 2002; Chu et al., 2011; Li et al., 2010; Abbasi-Yadkori et al., 2011; Cesa-Bianchi et al., 2013). LinUCB-ONE allocates a single instance of LinUCB across all users (thereby making the same prediction for all users), whereas LinUCB-IND ("LinUCB INDependent") allocates an independent instance of LinUCB to each user, thereby making predictions in a fully personalised fashion. Moreover, on the synthetic experiments, we added two idealized baselines: a GOBLIN-like algorithm (Cesa-Bianchi et al., 2013) fed with a Laplacian matrix encoding the true underlying graph G, and a CLAIRVOYANT algorithm that knows the true clusters a priori, and runs one instance of LinUCB *per cluster*. Notice that an experimental comparison to multitask-like algorithms, like GOBLIN, or to the idealized algorithm that knows all clusters beforehand, can only be done on the artificial datasets, not in the real-world case where no cluster information is available. On the Yahoo dataset, we tested the featureless version of the LinUCB-like algorithm in (Cesa-Bianchi et al., 2013), which is essentially a version of the UCB1 algorithm of (Auer et al., 2001). The corresponding ONE and IND versions are denoted by UCB-ONE and UCB-IND, respectively. On this dataset, we also tried a single instance of UCB-V (Audibert et al., 2009) across all users, the winner of the abovementioned ICML Challenge. Finally, all algorithms have also been compared to the trivial baseline (denoted by RAN) that picks the item within C_{i_t} fully at random.

As for parameter tuning, CLUB was run with $p = \frac{3 \log n}{n}$, so as to be reasonably confident that the initial graph is at least connected. In fact, after each generation of the graph, we checked for its connectedness, and repeated the process until the graph happened to be connected.[12] All algorithms (but RAN) require parameter tuning: an exploration-exploitation tradeoff parameter which is common to all algorithms (in Figure 1, this is the α parameter), and the edge deletion parameter α_2 in CLUB. On the synthetic datasets, as well as on the LastFM and Delicious datasets, we tuned these parameters by picking the best setting (as measured by cumulative regret) after the first $t_0 = 5,000$ rounds, and then sticked to those values

[8] Datasets and their full descriptions are available at www.grouplens.org/node/462.

[9] This is done so as to avoid a meaningless comparison: With high probability, a purely random selection would result in payoffs equal to zero for all the context vectors in C_{i_t}.

[10] https://explochallenge.inria.fr/

[11] Out of the 136 Boolean features, the first feature is always 1 throughout all records. We call "empty" the users whose only nonzero feature is the first feature.

[12] Our results are averaged over 5 random initial graphs, but this randomness turned out to be a minor source of variance.

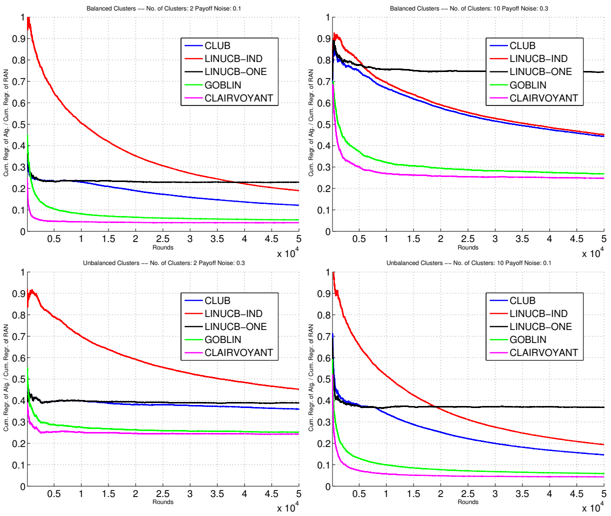

Figure 3. Results on synthetic datasets. Each plot displays the behavior of the ratio of the current cumulative regret of the algorithm ("Alg") to the current cumulative regret of RAN, where "Alg" is either "CLUB" or "LinUCB-IND" or "LinUCB-ONE" or "GOBLIN"or "CLAIRVOYANT". In the top two plots cluster sizes are balanced ($z = 0$), while in the bottom two they are unbalanced ($z = 2$).

for the remaining $T - t_0 = 50,000$ rounds. It is these $50,000$ rounds that our plots refer to. On the Yahoo dataset, this optimal tuning was done within the first $t_0 = 100,000$ records, corresponding to a number of retained records between $4,350$ and $4,450$ across different algorithms.

4.3. Results

Our results are summarized in[13] Figures 3, 4, and 5. On the synthetic datasets (Figure 3) and the LastFM and Delicious datasets (Figure 4) we measured the ratio of the cumulative regret of the algorithm to the cumulative regret of the random predictor RAN (so that the lower the better). On the synthetic datasets, we did so under combinations of number of clusters, payoff noise, and cluster size balancedness. On the Yahoo dataset (Figure 5), because the only available payoffs are those associated with the items recommended in the logs, we instead measured the Clickthrough Rate (CTR), i.e., the fraction of times we get $a_t = 1$ out of the number of retained records so far (so the higher the better). This experimental setting is in line with previous ones (e.g., (Li et al., 2010)) and, by the way data have been prepared, gives rise to a reliable estimation of actual CTR behavior under the tested experimental conditions (Li et al., 2011).

Based on the experimental results, some trends can be spotted: On the **synthetic datasets**, CLUB always outperforms its uninformed competitors LinUCB-IND and LinUCB-ONE, the gap getting larger as we either decrease the number of underlying clusters or we make the clusters sizes more and more unbalanced. Moreover, CLUB can clearly interpolate between these two competitors taking, in a sense, the best of both. On the other hand (and unsurprisingly), the informed competitors GOBLIN and CLEARVOYANT outperform all uninformed ones. On the **"few hits"** scenario of LastFM, CLUB is again outperforming both of its competitors. However, this is not happening in the **"many niches"** case delivered by the Delicious dataset, where CLUB is clearly outperformed by LinUCB-IND. The proposed alternative of CLUB that starts from an empty graph (Remark 2) might be an effective alternative in this case. On the **Yahoo datasets** we extracted, CLUB tends to outperform its competitors, when measured by CTR curves, thereby showing that clustering users solely based on past behavior can be beneficial. In general, CLUB seems to benefit from situations where it is not immediately clear which is the winner between the two extreme solutions (Lin)UCB-ONE and (Lin)UCB-IND, and an adaptive interpolation between these two is needed.

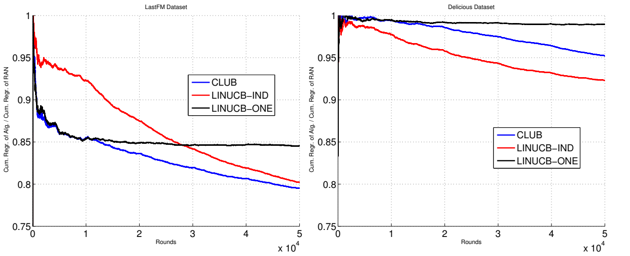

Figure 4. Results on the LastFM (left) and the Delicious (right) datasets. The two plots display the behavior of the ratio of the current cumulative regret of the algorithm ("Alg") to the current cumulative regret of RAN, where "Alg" is either "CLUB" or "LinUCB-IND" or "LinUCB-ONE".

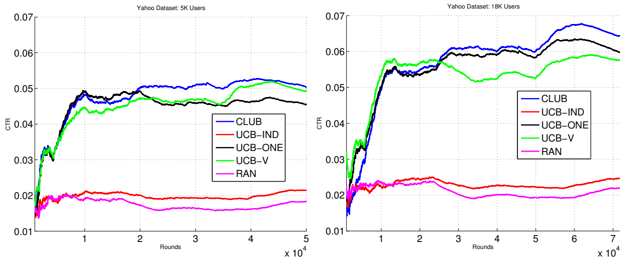

Figure 5. Plots on the Yahoo datasets reporting Clickthrough Rate (CTR) over time, i.e., the fraction of times the algorithm gets payoff one out of the number of retained records so far.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful and constructive comments. Also, the first and the second author acknowledge the support from Amazon AWS Award in Education Machine Learning Research Grant.

[13] Further plots can be found in the supplementary material.

References

Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. Improved algorithms for linear stochastic bandits. *Proc. NIPS*, 2011.

Audibert, J.-Y., Munos, R., and Szepesvári, C. Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

Auer, P. Using confidence bounds for exploration-exploitation trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 2001.

Bogers, T. Movie recommendation using random walks over the contextual graph. In *CARS'10: Proc. 2nd Workshop on Context-Aware Recommender Systems*, 2010.

Buccapatnam, S., Eryilmaz, A., and Shroff, N.B. Multi-armed bandits in the presence of side observations in social networks. In *Proc. 52nd IEEE Conference on Decision and Control*, 2013.

Burke, R. Hybrid systems for personalized recommendations. In *Proc. of the 2003 ITWP*, pp. 133–152, 2005.

Buscher, G., White, R. W., Dumais, S., and Huang, J. Large-scale analysis of individual and task differences in search result page examination strategies. In *Proc. 5th ACM WSDM*, pp. 373–382, 2012.

Caron, S. and Bhagat, S. Mixing bandits: A recipe for improved cold-start recommendations in a social network. In *SNA-KDD, 7th Workshop on Social Network Mining and Analysis*, 2013.

Caron, S., Kveton, B., Lelarge, M., and Bhagat, S. Leveraging side observations in stochastic bandits. In *Proc. UAI*, pp. 142–151, 2012.

Cesa-Bianchi, N., Gentile, C., and Zappella, G. A gang of bandits. In *Proc. NIPS*, 2013.

Chu, W., Li, L., Reyzin, L., and Schapire, R. E. Contextual bandits with linear payoff functions. In *Proc. AISTATS*, 2011.

Cormen, T.H., Leiserson, C.E., and Rivest, R.L. *Introduction to Algorithms*. McGraw Hill, 1990.

Crammer, K. and Gentile, C. Multiclass classification with bandit feedback using adaptive regularization. In *Proc. ICML*, 2011.

Delporte, J., Karatzoglou, A., Matuszczyk, T., and Canu, S. Socially enabled preference learning from implicit feedback data. In *Proc. ECML/PKDD*, pp. 145–160, 2013.

Djolonga, J., Krause, A., and Cevher, V. High-dimensional gaussian process bandits. In *NIPS*, pp. 1025–1033, 2013.

Kapron, Bruce M., King, Valerie, and Mountjoy, Ben. Dynamic graph connectivity in polylogarithmic worst case time. In *Proc. SODA*, pp. 1131–1142, 2013.

Karger, D. R. Random sampling in cut, flow, and network design problems. In *Proc. STOC*, 1994.

Krause, A. and Ong, C.S. Contextual gaussian process bandit optimization. In *Proc. 25th NIPS*, 2011.

Lai, T. and Robbins, H. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6: 4–22, 1985.

Langford, J. and Zhang, T. The epoch-greedy algorithm for contextual multi-armed bandits. In *Proc. NIPS*, 2007.

Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *Proc. WWW*, pp. 661–670, 2010.

Li, L., Chu, W., Langford, J., and Wang, X. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proc. WSDM*, 2011.

Rashid, A. M., Lam, S.K., Karypis, G., and Riedl, J. Clustknn: a highly scalable hybrid model-& memory-based cf algorithm. In *Proc. WebKDD-06, KDD Workshop on Web Mining and Web Usage Analysis*, 2006.

Schafer, J.B., Konstan, J.A., and Riedl, J. Recommender systems in e-commerce. In *Proc. EC*, pp. 158–166, 1999.

Seldin, Y., Auer, P., Laviolette, F., Shawe-Taylor, J., and Ortner, R. Pac-bayesian analysis of contextual bandits. In *NIPS*, pp. 1683–1691, 2011.

Thorup, M. Decremental dynamic connectivity. In *Proc. SODA*, pp. 305–313, 1997.

Valko, M., Munos, R., Kveton, B., and Kocák, T. Spectral Bandits for Smooth Graph Functions. In *31th International Conference on Machine Learning*, 2014.

Yue, Y., Hong, S. A., and Guestrin, C. Hierarchical exploration for accelerating contextual bandits. In *ICML*, 2012.