Bandits Under the Influence

Silviu Maniu
LRI, CNRS
Université Paris-Saclay
Orsay, France
silviu.maniu@lri.fr

Stratis Ioannidis
Electrical and Computer Engineering
Northeastern University
Boston, MA, USA
ioannidis@ece.neu.edu

Bogdan Cautis
LRI, CNRS
Université Paris-Saclay
Orsay, France
bogdan.cautis@lri.fr

***Abstract*—Recommender systems should adapt to user interests as the latter evolve. A prevalent cause for the evolution of user interests is the influence of their social circle. In general, when the interests are not known, online algorithms that explore the recommendation space while also exploiting observed preferences are preferable. We present online recommendation algorithms rooted in the linear multi-armed bandit literature. Our bandit algorithms are tailored precisely to recommendation scenarios where user interests evolve under social influence. In particular, we show that our adaptations of the classic LINREL and THOMPSONSAMPLING algorithms maintain the same asymptotic regret bounds as in the non-social case. We validate our approach experimentally using both synthetic and real datasets.**

I. INTRODUCTION

Recommender systems can benefit significantly from sequential learning techniques, such as multi-armed bandit algorithms, when user interests are a priori unknown, hardly generalizable, or highly dynamic. Such conditions arise in news recommendation scenarios, where the turnover of items is simply too high to enable a reasonable application of traditional recommendation algorithms, or in cold-start scenarios, i.e., when addressing a new or ever-changing user base. Online recommendation algorithms (re)learn preferences over time and continuously, striking a balance between exploiting popular recommendation options and exploring new ones, that may improve overall user satisfaction.

One prevalent reason for the continuous evolution of user interests, calling for such online learning approaches for recommendation, is *social influence*, under which connected users converge to similar interests. While realistic models for social influence remain an only partially understood area, the presence of influence in social media – by either global and local mechanisms – has been extensively studied formally [9], [11] and verified experimentally [4], [6].

Motivated by the above observations, we study an online recommender system *that learns user interests as they evolve under social influence*. We consider a scenario in which the initial interests of users are unknown, but the effects of the social network on their evolution are understood and modelled by a probabilistic social graph – akin to the independent cascade model [12]. More precisely, the recommender follows the combined objective of maximizing rewards (i.e., user ratings) over a finite horizon, while simultaneously discovering user interests. The latter however are subject to a drift caused by social influence.

This setting gives rise to several challenges not present in classic recommender systems. The first is the usual multi-armed bandit (MAB) challenge of exploration, in discovering and tracking user interests, vs. exploitation, via the recommendation of pertinent content. The second (and most crucial) challenge, differentiating us from classic MAB and recommendation literature alike, is that *recommendations become coupled via social influence*. As a result, the status quo of targeting recommendations on individual users separately is suboptimal. Instead, social influence implies that a *global* recommendation strategy needs to be optimized across users. This leads to a combinatorial explosion of the space of possible recommendation strategies, as the latter grow exponentially in the number of users.

Our main contribution is to address these two challenges in a comprehensive fashion. In particular:

- We show that the LINREL [3], [8] and THOMPSONSAMPLING [2] bandit algorithms are a *natural fit to our setting*. We provide *regret* bounds for both methods, taking into account the effect of social influence.
- Crucially, we establish that both algorithms are *tractable*: despite the exponential size of possible global recommendation strategies (i.e., arms) in our setting, we derive polynomial-time algorithms for arm selection under both the LINREL and the THOMPSONSAMPLING algorithms.
- We also consider LINUCB, another popular linear bandit strategy [5], [7], [13], [16]; the corresponding arm selection process turns out to be intractable, but a solution can be approximated within a constant in polynomial time; unfortunately, this implies that the resulting algorithm comes with no regret guarantees.

Importantly, to the best of our knowledge, we are the first to analyze and compare LINREL, THOMPSONSAMPLING, and LINUCB, both from a tractability and a regret perspective, in the presence of social influence. From a technical standpoint, our analysis requires both revisiting regret bounds under a dynamic influence setting, but also tackling the exponential size of the recommendation strategy space. We accomplish the latter through the reduction of the arm selection process to an optimization with a linear objective, which becomes separable across users.

We omit proofs and experimental details from this short paper; both can be found in the extended version [15].

II. Problem Formulation

A. Recommendations

Formally, we consider n users, all receiving suggestions from a recommender at discrete time steps $t \in \mathbb{N}$. Each recommended item is represented by a d-dimensional *item profile* $\mathbf{v} \in \mathbb{R}^d$, capturing this item's features; we denote by $\mathcal{B} \subseteq \mathbb{R}^d$ the set of available items, i.e., the recommender's *catalog*. We denote by $[n] \equiv \{1, 2, \dots, n\}$ the set of all users. At each time step $t \in \mathbb{N}$, the recommender suggests (possibly different) items from $\mathcal{B}$ to each user $i \in [n]$. Each i responds by revealing a rating $r_i(t) \in \mathbb{R}$, indicating her preference towards the item recommended to her.

We assume that ratings are generated according to the following random process. At each $t \in \mathbb{N}$, users $i \in [n]$ have *user profiles* represented by d-dimensional vectors $\mathbf{u}_i(t) \in \mathbb{R}^d$. Then, if $\mathbf{v}_i(t) \in \mathcal{B}$ is the profile of the item recommended to i at time t, ratings satisfy:

$$r_i(t) = \langle \mathbf{u}_i(t), \mathbf{v}_i(t) \rangle + \varepsilon, \tag{1}$$

where ε is zero mean, finite variance noise (i.i.d. across users and timeslots).

B. User interest evolution

Following [14], we assume that profiles *evolve* according to the following dynamics. Each user is associated with an inherent profile, capturing her personal interests. At each timeslot, she chooses with some probability to either use her inherent profile, or she chooses to use a profile that is the result of the influence of her neighborhood. Formally, at each time $t \in \mathbb{N}$, user profiles evolve according to:

$$\mathbf{u}_i(t) = \alpha \mathbf{u}_i^0 + (1-\alpha) \textstyle\sum_{j \in [n]} P_{i,j} \mathbf{u}_j(t-1), \ \ i \in [n], \tag{2}$$

where (a) $\mathbf{u}_i^0 \in \mathbb{R}^d$ is user i's inherent (static) profile, (b) $\alpha \in [0,1]$ captures the probability that users act based on their inherent profiles, and (c) $P_{ij} \in [0,1]$, $i,j \in [n]$, where $\sum_j P_{ij} = 1$, capture the probability user i is influenced by the profile of user j. This setting can allow different values for α for each user; this does not change the analysis in the following.

The probability $P_{ij} \in [0,1]$ captures the influence that user j has on user i. Note that users j for which $P_{ij} = 0$ (i.e., outside i's social circle) have no influence on i. Moreover, the set of pairs (i,j) s.t. $P_{ij} \neq 0$, defines the social network among users. We denote by $P \in [0,1]^{n \times n}$ the stochastic matrix with elements P_{ij}, $i,j \in [n]$; we assume that P is ergodic (i.e., irreducible and aperiodic) [10], a reasonable setting – even if the original social influence graph is not, one can easily make it so by adding a small probability between all pairs in the graph. Then, for $U(t) = [\mathbf{u}_i(t)]_{i \in [n]} \in \mathbb{R}^{n \times d}$ the $n \times d$ matrix consisting of the profiles of all users at time t, (2) can we written in matrix form as:

$$U(t) = \alpha U^0 + (1-\alpha) P U(t-1), \tag{3}$$

where matrix $U^0 = [\mathbf{u}_i^0]_{i \in [n]} \in \mathbb{R}^{n \times d}$ comprises inherent profiles.

C. Bandit setting

We consider a bandit setting, in which the recommender does not know the inherent profiles, but would nevertheless wish to maximize rewards (1) over a finite horizon. In particular, we assume that the recommender (a) knows the probabilities α and P, capturing the dynamics of the interest evolution and (b) observes the history of responses by users in previous timeslots $t = 1, \dots, t-1$. Based on this history, the recommender suggests items $\mathbf{v}_i(t) \in \mathcal{B}$, $i \in [n]$, in order to minimize the *aggregate regret*:

$$R(T) = \textstyle\sum_{t=1}^{T} \sum_{i=1}^{n} \langle \mathbf{u}_i(t), \mathbf{v}_i^*(t) \rangle - \langle \mathbf{u}_i(t), \mathbf{v}_i(t) \rangle, \tag{4}$$

where $\mathbf{v}_i^*(t)$, $i \in [n]$, $t \in [T]$, are recommendations made by an optimal strategy that knows vectors $\mathbf{u}_i^0$, $i \in [n]$ (see Eq. (9) for a closed form characterization).

Recommender suggestions $\mathbf{v}_i(t)$, $i \in [n]$ are selected from a set $\mathcal{B} \subseteq \mathbb{R}^d$. We consider two possibilities:

- $\mathcal{B}$ is a finite subset of $\mathbb{R}^d$, i.e., it is a "catalog" of possible recommendations.
- $\mathcal{B}$ is an arbitrary convex subset of $\mathbb{R}^d$, e.g., the unit ball $\mathbb{B} \equiv \{\mathbf{v} \in \mathbb{R}^d : \|\mathbf{v}\|_2 \leq 1\}$.

D. Relationship to linear bandits

The aggregate expected reward is, at any time t, a linear function of the inherent user profiles U^0. This motivates our exploration of LinREL, Thompson sampling, and LinUCB as candidate online algorithms with bounded regret. To see this, let $V(t) = [\mathbf{v}_i(t)]_{i \in [n]} \in \mathbb{R}^{n \times d}$, be the matrix comprising the recommendations made at time t in each row. Then, the following lemma holds:

Lemma 1. *The total expected reward $\bar{r}(t)$ at time t under recommendations $V(t)$ is given by:*

$$\bar{r}(t) = \langle U^0, A(t)^\top V(t) \rangle, \tag{5}$$

where $\langle A, B \rangle = \texttt{trace}(AB^\top)$, is the Frobenius inner product, and

$$A(t) = \alpha \sum_{k=0}^{t} \left((1-\alpha) P \right)^k \in \mathbb{R}^{n \times n}. \tag{6}$$

Lemma 1 gives a clearer indication that the reward is indeed a linear function of the unknown inherrent profiles U^0. To ease exposition, but also to be consistent with existing bandit literature, we vectorize the representation of recommendations and user profiles. Given a $U^0, V \in \mathbb{R}^{n \times d}$, we define their row-wise vector representations as:

$$\mathbb{u}_0 \equiv \texttt{vec}(U^0) = [(\mathbf{u}_1^0)^\top, (\mathbf{u}_2^0)^\top, \dots, (\mathbf{u}_n^0)^\top] \in \mathbb{R}^{nd}$$
$$\mathbb{v} \equiv \texttt{vec}(V) = [\mathbf{v}_1^\top, \mathbf{v}_2^\top, \dots, \mathbf{v_n}^\top] \in \mathcal{B}^n \subseteq \mathbb{R}^{nd},$$

i.e., $\mathbb{u}_0$, $\mathbb{v}$ are nd-dimensional vectors resulting from representing U^0, V row-wise. We can then directly describe the expected reward at time t as a quadratic form involving $\mathbb{u}_0, \mathbb{v}(t)$:

Algorithm 1 – LinREL

```
Require: matrix $P$, parameter $\alpha$, item set $\mathcal{B}$, users $[n]$
1: Initialization: play $d$ pulls for each $i \in [n]$ and observe rewards $\mathbf{r}_0$
2: $A(0) \leftarrow \alpha I$
3: for $t = 1, \ldots, T$ do
4:     estimate
           $\hat{\mathbb{u}}_0(t) = \arg\min_{\mathbb{u} \in \mathbb{R}^{nd}} \sum_{\tau=1}^{t-1} \|X(V(\tau), A(\tau))\mathbb{u} - \mathbf{r}(\tau)\|_2^2$
5:     Recommend
           $\mathbb{v}_t = \arg\max_{\mathbb{v} \in \mathcal{B}^{(n)}} \max_{\mathbb{u} \in \mathcal{C}_t} \mathbb{u}^\top L(t)\mathbb{v}$
       where $\mathcal{C}_t$ is given by (12a) or (12b)
6:     observe reward vector $\mathbf{r}(t)$
7:     $A(t) \leftarrow A(t-1) + \alpha(1-\alpha)P^t$
8: end for
```

Lemma 2. *The total expected reward at time t under recommendations $V = V(t)$ is then given by:*

$$\bar{r}(t) = \mathbb{u}_0^\top L(t)\mathbb{v} \tag{7}$$

where $\mathbb{u}_0 \equiv \mathtt{vec}(U^0)$, $\mathbb{v} \equiv \mathtt{vec}(V)$, and

$$L(t) \equiv A(t)^\top \otimes I_d \in \mathbb{R}^{nd \times nd} \tag{8}$$

is the Kronecker product of $A(t)^\top$ with the identity matrix $I_d \in \mathbb{R}^{d \times d}$.

Lemma 2 casts our reward (and our problem) in a linear (a.k.a. contextual) bandit setting. The quadratic form (7) suggests that the reward is linear in both the unknown parameters $\mathbb{u}_0$ *and* the recommendations $\mathbb{v}$; in turn, the evolution of interests changes the nature of this relationship via $nd \times nd$ matrix $L(t)$; in particular, the latter fully determines the coupling between recommendations across users. Finally, Lemma 2 allows us to rewrite the regret (4) as follows:

$$R(T) = \sum_{t=1}^{T} \left(\mathbb{u}_0^\top L(t)\mathbb{v}^*(t) - \mathbb{u}_0^\top L(t)\mathbb{v}(t)\right) \tag{9}$$

where $\mathbb{v}^*(t) = \arg\max_{\mathbb{v} \in \mathcal{B}^{(n)}} \mathbb{u}_0^\top L(t)\mathbb{v}$ is the optimal decision at t.

III. LinREL Algorithm

The LinREL algorithm [3], also called "confidence ball" in [8], operates under the following assumptions: arms are selected from a vector space, and the expected reward observed is an (unknown) linear function of the arm selected. To select an arm, the algorithm uses a variation of the Upper Confidence Bound (UCB) principle, in that it considers a confidence bound to an estimator for each arm's reward when selecting it. The unknown linear model is estimated via a least squares fit; the upper confidence bound constrains the next selection over an L_1 or L_2 ellipsoid centred around the current estimate. In our case, the resulting arm selection problem is non-convex; nevertheless, as we show below, it can be solved in polynomial time for a variety of different settings.

A. Algorithm overview

Lemma 2 indicates that the total expected reward in our setting is indeed a linear function of "arms" $\mathbb{v} \in \mathbb{R}^{nd}$, parametrized by unknowns $\mathbb{u}_0 \in \mathbb{R}^{nd}$; nevertheless, the arms are affected by the current state of influence, as captured by $L(t) \in \mathbb{R}^{nd \times nd}$. Applied to our setting, LinREL operates as summarized in Algorithm 1. As an initialization step, for each user in $[n]$, the recommender suggests d arbitrary items that span $\mathbb{B}$. For each round, $\mathbb{u}_0(t)$ is estimated via least squares estimation from past observations, i.e.:

$$\hat{\mathbb{u}}_0(t) = \arg\min_{\mathbb{u} \in \mathbb{R}^{nd}} \sum_{\tau=1}^{t-1} \|X(V(\tau), A(\tau))\mathbb{u} - \mathbf{r}(\tau)\|_2^2 \tag{10}$$

where $X \in \mathbb{R}^{n \times nd}$ is an expanded arm matrix depending on both V and A (see [15]) and $\mathbf{r}(\tau) \in \mathbb{R}^n$ is the vector of rewards collected (i.e., user responses observed) at time τ.

At iteration $t \geqslant 1$, the recommendations $\mathbb{v}(t)$ are selected by LinREL as the solutions of the optimization

$$\mathbb{v}(t) = \arg\max_{\mathbb{v} \in \mathcal{B}^{(n)}} \max_{\mathbb{u} \in \mathcal{C}_t} \mathbb{u}^\top L(t)\mathbb{v} \tag{11}$$

where $\mathcal{C}_t$ is an appropriately selected ellipsoid centered at $\hat{\mathbb{u}}_0$ and capturing the uncertainty of the estimate. Two possible cases considered by [3], [8] are:

$$\mathcal{C}_t^2 = \left\{\mathbb{u} : \|\hat{\mathbb{u}}_0(t) - \mathbb{u}\|_{2,Z(t)} \leq \sqrt{\beta_t}\right\}, \tag{12a}$$

$$\mathcal{C}_t^1 = \left\{\mathbb{u} : \|\hat{\mathbb{u}}_0(t) - \mathbb{u}\|_{1,Z(t)} \leq \sqrt{nd\beta_t}\right\}, \tag{12b}$$

where $\beta_t = \beta(t, n, d, \delta)$ is a closed-form function of t, n, d, and a parameter δ, $Z(t)$ is the inverse of the covariance (i.e., the precision) of estimate $\hat{\mathbb{u}}_0$, given by

$$Z(t) = \sum_{\tau=1}^{t-1} X\big(V(\tau, A(\tau)\big)^\top X\big(V(\tau, A(\tau)\big), \tag{13}$$

and $\|\mathbf{x}\|_{2,A} = \sqrt{\mathbf{x}^\top A\mathbf{x}}$, $\|\mathbf{x}\|_{1,A} = \|A^{1/2}\mathbf{x}\|_1$.

Intuitively, the optimization (11) selects a recommendation that maximizes the expected reward *under a perturbed* $\mathbb{u}$: though $\mathcal{C}_t$ is centered at $\hat{\mathbb{u}}_0$, directions of high variability are favored under optimization (11). A key challenge is that (11) *is not a convex optimization problem*; in fact, when the catalog $\mathcal{B}$ is a finite set, (11) is combinatorial, and the set $\mathcal{B}^{(n)}$ grows exponentially in n. Nevertheless, as we discuss in Sec. III-C below, when $\mathcal{C}_t = \mathcal{C}_t^1$, we can solve (11) efficiently for the different cases of set $\mathcal{B} \subset \mathbb{R}^d$ presented in Sec. II-B, including finite catalogs.

B. Regret

Most importantly, we can show the following bound on the regret:

Theorem 1. *Assume that, for any* $0 < \delta < 1$*:*

$$\beta_t = \max\left\{128nd\ln t \ln\frac{t^2}{\delta}, \left(\frac{8}{3}\ln\frac{t^2}{\delta}\right)^2\right\}, \tag{14}$$

then, for $\mathcal{C}_t = \mathcal{C}_t^2$:

$$\Pr\left(\forall T, R(T) \leqslant n\sqrt{8nd\beta_T T \ln\left(1+\frac{n}{d}T\right)}\right) \geqslant 1-\delta, \quad (15)$$

and, for $\mathcal{C}_t = \mathcal{C}_t^1$:

$$\Pr\left(\forall T, R(T) \leqslant n^2 d\sqrt{8\beta_T T \ln\left(1+\frac{n}{d}T\right)}\right) \geqslant 1-\delta. \quad (16)$$

Thm. 1 implies that LINREL, in both variants, applied to the social bandits case yields the same bound as in [8], i.e. a polylog bound of $\tilde{\mathcal{O}}(\sqrt{T})$, and which is known to be tight [1], [8]. Note that, compared to [8], the bound is worse only by a factor of n, corresponding to the number of users in the network. Though we provide bounds for both $\mathcal{C}^2$, $\mathcal{C}^1$, and the bounds for $\mathcal{C}^2$ are better than the ones for $\mathcal{C}^1$ by a factor of nd, there is a clear advantage for the $\mathcal{C}^1$ version: it makes (11) tractable. We discuss this next.

C. *An efficient solver of* (11) *under* $\mathcal{C}^1$ *constraints*

Consider the case where the constraint set is $\mathcal{C}_t^1$, given by (12b). In this case, we can solve (11) efficiently for several sets $\mathcal{B}$ of interest. In doing so, we exploit the fact that, for any $\mathbb{z} \in \mathbb{R}^{nd}$, the optimization problem:

$$\text{Maximize:} \quad \mathbb{u}^\top \mathbb{z} \quad (17a)$$
$$\text{subject to:} \quad \|\hat{\mathbb{u}}_0 - \mathbb{u}\|_{1,Z(t)} \le c_t \quad (17b)$$

attains its maximum at one of the $2nd$ extreme points of the polytope $\|\mathbb{u} - \hat{\mathbb{u}}_0\|_{1,Z(t)} \le c$. These extreme points can be generated in polynomial time from $Z(t)$, and $\hat{\mathbb{u}}_0$. Hence, given $\mathbb{z}$, solving a problem of the form (17) amounts to finding which of these $2nd$ extreme points yields the maximal inner product with $\mathbb{z}$. This observation leads to the following means of solving (11):

Theorem 2. *Let $\mathcal{E}$ be the set of $2nd$ extreme points of the polytope $\|\mathbb{u} - \hat{\mathbb{u}}_0\|_{1,Z(t)} \le \sqrt{nd\beta_t}$. Then, if $\mathcal{C}_t = \mathcal{C}_t^1$,* (11) *reduces to solving $2n^2d$ problems of the form:*

$$\textit{Maximize:} \quad \mathbf{z}^\top \mathbf{v}_i \quad (18a)$$
$$\textit{subject to:} \quad \mathbf{v}_i \in \mathcal{B}, \quad (18b)$$

for $\mathbf{z} = (\mathbb{u}^T L(t))_i \in \mathbb{R}^d$ the part of $\mathbb{u}^T L(t)$ corresponding to user $i \in [n]$, and $\mathbb{u} \in \mathcal{E}$.

This reduction implies that we can solve (11) for *a broad array of sets $\mathcal{B}$*. In particular:

- (a) When $\mathcal{B}$ is an arbitrary convex set (e.g., the Euclidian ball, a convex polytope, etc.), (18) is a convex optimization problem, so (11) amounts to solving $2n^2d$ convex problems.
- (b) When $\mathcal{B}$ is a finite subset of $\mathbb{R}^d$, the optimal solution to (18) can be found in polynomial time by finding the maximum among all $|\mathcal{B}|$ values (18). Hence, a solution to (11) can be computed after a total of $2n^2d|\mathcal{B}|$ evaluations of a linear function–namely, (18a).

We stress here that the existence of efficient/polytime algorithms in the above cases is remarkable precisely because of the exponential size of the possible combinations in (11): this is most evident in the finite $\mathcal{B}$ case, where there are $|\mathcal{B}|^n$ candidate recommendation combinations across users.

Algorithm 2 – THOMPSONSAMPLING

```
Require: matrix $P$, parameter $\alpha$, item set $\mathcal{B}$, users $[n]$
1: Initialization: play $d$ pulls for each $i \in [n]$ and observe rewards $\mathbf{r}_0$
2: $A(0) \leftarrow \alpha I$
3: Sample $\mathbb{u}_0(0)$ from $\mathcal{N}(\hat{\mathbb{u}}_0(0), \Sigma(0))$
4: for $t = 1, \ldots, T$ do
5:     estimate
           $\hat{\mathbb{u}}_0(t) = \arg\min_{\mathbb{u} \in \mathbb{R}^{nd}} \sum_{\tau=1}^{t-1} \|X(V(\tau), A(\tau))\mathbb{u} - \mathbf{r}(\tau)\|_2^2$
6:     Sample $\mathbb{u}$ from $\mathcal{N}(\hat{\mathbb{u}}_0(t), \Sigma(t))$
7:     Recommend $\mathbb{v}_t = \arg\max_{\mathbb{v} \in \mathcal{B}^{(n)}} \mathbb{u}^\top L(t)\mathbb{v}$
8:     Observe reward $\mathbf{r}(t)$, update $A(t)$ and $\Sigma(t+1)$
9: end for
```

IV. THOMPSON SAMPLING

We turn now to a Bayesian interpretation and its associated algorithm, THOMPSONSAMPLING. Instead of optimizing over a confidence ellipsoid or interval, THOMPSONSAMPLING assumes a prior on the parameter vector $\mathbb{u}_0$ and, at each step, samples this vector from the posterior obtained after the feedback has been observed. This way, the exploration is embedded in the uncertainty of the distribution being sampled.

A. *Algorithm overview*

Algorithm 2 outlines THOMPSONSAMPLING. We assume that our parameter vector, $\mathbb{u}_0$, is distributed as a multi-variate normal having prior $\mathcal{N}(\mathbb{u}(0), \Sigma(0))$ – obtained, for instance, after playing the initialization arms. At each t, the covariance is updated via:

$$\Sigma(t+1) = \left(\Sigma(t)^{-1} + \frac{X(V(t), A(t))^\top X(V(t), A(t))}{\sigma^2}\right)^{-1},$$

where σ^2 is the standard deviation of the reward noise. Then, $\mathbb{u}(t)$ is obtained by sampling from the distribution $\mathcal{N}(\hat{\mathbb{u}}_0(t), \Sigma(t))$. Finally, $\mathbb{v}(t+1)$ is chosen by solving the (simpler than (11)) optimization:

$$\mathbb{v}(t+1) = \arg\min_{\mathbb{v} \in \mathcal{B}^{(n)}} \hat{\mathbb{u}}(t+1)^\top L(t+1)\mathbb{v}. \quad (19)$$

B. *Regret*

The quantity of interest for THOMPSONSAMPLING is the *Bayesian regret*, i.e., the aggregate regret in expectation:

$$\mathrm{BR}(T) = \mathbf{E}\left[\sum_{t=1}^{T}\left(\mathbb{u}_0^\top L(t)\mathbb{v}_*(t) - \mathbb{u}_0^\top L(t)\mathbb{v}(t)\right)\right]. \quad (20)$$

An $\tilde{\mathcal{O}}(\sqrt{T})$ bound applies to THOMPSONSAMPLING in our setting:

Theorem 3. *Assume that, for any $0 < \delta < 1$, β_T is set as: $\beta_T = 1 + \sqrt{2\log\frac{1}{\delta} + nd\log\left(1 + \frac{n}{d}T\right)}$. Then:*

$$\Pr\left(\forall T, \mathrm{BR}(T) \leqslant 2 + 2n\beta_T T\sqrt{2ndT\ln\left(1+\frac{n}{d}T\right)}\right) \geqslant 1 - \delta.$$

V. LinUCB Algorithm

LinUCB [13] is an alternative to LinREL for linear bandits. Unfortunately, in our case, it leads to an intractable problem. Applied to our setting, LinUCB is identical to Alg. 1, with only a change in how recommendations are selected. That is, $\hat{\mathbb{u}}_0(t)$ is again estimated via least squares estimation (10) as in LinREL. For $\Sigma(t) = Z^{-1}(t)$ the covariance of the estimate $\hat{\mathbb{u}}^0$, LinUCB selects

$$\mathbb{v}_t = \arg\max_{\mathbb{v}\in(\mathcal{B})^n}\left(\hat{\mathbb{u}}_0^\top(t)L(t)\mathbb{v} + c\mathbb{v}^\top L(t)^\top\Sigma(t)L(t)\mathbb{v}\right). \quad (21)$$

LinUCB can thus be implemented by replacing line 5 of Algorithm 1 with equation (21).

Unfortunately, (21) is a non-convex optimization, with no obvious solution for different cases of B. In the case where $\mathcal{B}_t = \mathbb{B} = \{v \in \mathbb{R}^d : \|v\| \leq 1\}$, an approximate solution can be constructed via an SDP relaxation, as in [14]. Nevertheless, this solution is only within a constant approximation of the optimal; as such, it cannot be used to bound the regret.

VI. Experiments

To validate our analysis, we evaluated the regret of the LinREL, ThompsonSampling, and LinUCB algorithms for both synthetic and real-life data. We describe experiments on both types of datasets below.

A. Synthetic Data.

Experiment Setup. We generate a complete network (CMP) on which influence probabilities – representing the matrix P – were assigned between the n^2 pairs each having a value of $1/n$. In our experiments on synthetic data, we set α to be 0.05. We consider two cases for the set of recommendations $\mathcal{B}$: (a) a finite catalog of size $M = |\mathcal{B}|$, for M in $\{10, 100\}$, and (b) the unit ball $\mathbb{B}$. In the case of a finite catalog, we generate M random vectors in $[0,1]^d$ by uniform sampling. Finally we generate inherent profiles by sampling n vectors uniformly from $[0,1]^d$. Given a recommendation we assume that ratings by users are generated via (1), where i.i.d. noise ε has standard deviation 1

Algorithms. We implement[1] LinREL with $\mathcal{C}_t^1$ constraints (LinREL 1), ThompsonSampling, and LinUCB. In the case of LinREL, we multiply the theoretically designed radius β_t, given by (14), with a scaling factor β; we set it to $\beta = 10^{-5}$ after experimental validation.

We compare to two baselines: (a) the usual random bandit (Rand), which explores without exploiting by choosing a vector randomly, and (b) a regression baseline (Regression) which exploits without exploring: at each step, it only performs the least squares estimation of profiles and selects optimal recommendations in $\mathcal{B}$, but does not do any exploration. In all cases, we set the number of rounds (the horizon T) to 100.

Regret and Execution Time. We compare the regret attained by different algorithms over topology CMP, in Figure 1. Subfigures (a),(b) show the regret as a function of iterations under a finite set catalog $\mathcal{B}$, and subfigures (d),(e) show the regret as a function of iterations when the catalog is the ℓ_2 ball $\mathbb{B}$. Compared to the baselines, we find that our bandit algorithms significantly outperform Rand and are competitive with "reasonable" baseline Regression. In the finite case, their difference from the latter is less pronounced. However, in the ℓ_2 case, Regression becomes almost as ineffective as the random baseline. ThompsonSampling is always at least competitive with LinREL, and in some cases, especially for low values of n and d, seems to be the best variant. In the only test we could run for LinUCB we found that it is worse than LinREL and ThompsonSampling.

When looking at the execution time (Figure 1c,f), it becomes clear that first, LinUCB is too costly for its regret performance. This is due to the SDP relaxation involved in arm selection. LinREL has a more reasonable running time, but the best trade-off is attained by ThompsonSampling.

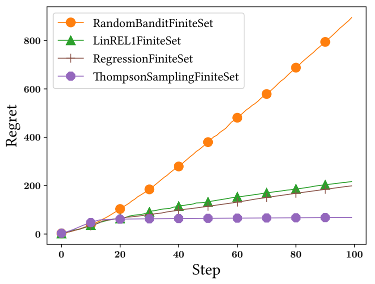
B. Real Data

Finally, to validate the recommendation algorithms on real data, we used data from the now defunct Flixster site, a social network for movie discovery and reviews. The dataset contains 1,049,492 users in a social network of 7,058,819 contact links, 74,240 movies that have generated 8,196,077 reviews, each giving 1 to 5 stars. Each movie description includes one or more categories out of 28 possible; each category is thus a dimension in the profile and item vectors.

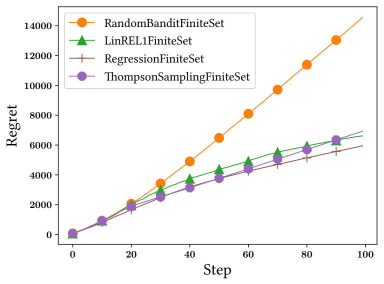
To adapt this dataset to our setting, and to allow reasonable execution time for all methods, we have removed all users not having at least 1,500 reviews. For each resulting user, we have evaluated the $\mathbf{u}_0$ vectors of $d = 28$ by linear regression on their respective reviews, where the feature vector contains the 28 categories and the 1 to 5 star ratings are mapped into the $[-1, 1]$ interval. This resulted in a dataset of $n = 206$ users and $d = 28$. P was generated by extracting the subgraph in the original network corresponding to the 206 remaining users, and setting the influence probability , $1/\deg(v_i)$. The finite set $M = 100$ was generated, as in the synthetic case, by uniformly sampling each from $[0,1]^d$; we have not used real vectors, so as not to bias the recommendation (the u_0 vectors were generated via regression on real items, as described above).

Figure 2 presents the regret of the bandit algorithms. It can be seen that, in line with results on the synthetic data, ThompsonSampling attains the best trade-off. The worse performance of LinREL is most likely due to a confidence ellipsoid that is too large. The good performance of Regression is an artifact of the experiment – as said above, the u_0 vectors are generated via linear regression; in that sense, it can be considered a lower bound on the possible regret.

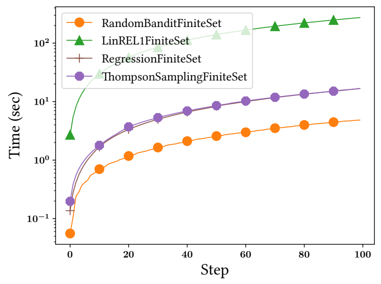
[1]Our code is available at `https://git.io/JUuSE`.



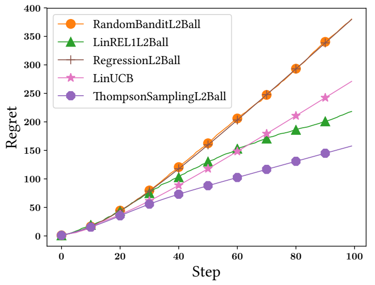
(a) Regret, finite set $n = 10$, $d = 5$, $m = 100$



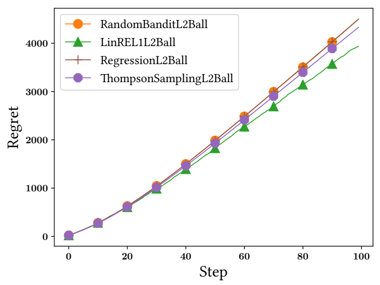
(b) Regret, finite set $n = 100$, $d = 20$, $m = 1000$



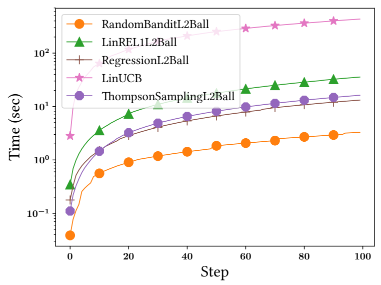
(c) Time, finite set $n = 100$, $d = 20$, $M = 100$ (log y-axis)



(d) Regret, L_2 ball $n = 10$, $d = 5$



(e) Regret, L_2 ball $n = 100$, $d = 20$



(f) Time, L_2 ball $n = 10$, $d = 5$ (log y-axis)

Fig. 1: Regret and execution time analysis. Setup: horizon 100, $\sigma = 1$, 10^{-5} β scale

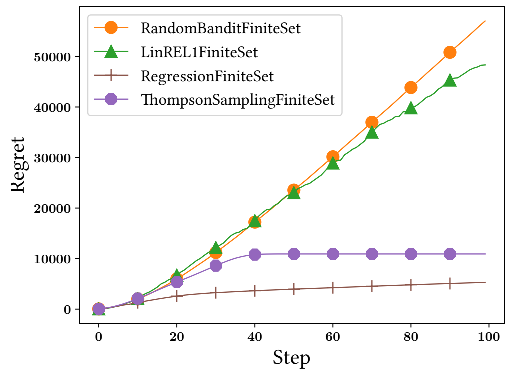


Fig. 2: Flixstr regret $n = 206$, $d = 28$, $M = 100$, $\sigma = 1$

Acknowledgments

Silviu Maniu and Bogdan Cautis gratefully acknowledge support from project PSPC AIDA (2019-PSPC-09), and Stratis Ioannidis gratefully acknowledges support from the NSF (grants CCF-1750539 and CCF-1937500).

References

[1] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári. Improved algorithms for linear stochastic bandits. In *Neural Information Processing Systems*, pages 2312–2320, 2011.
[2] S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. In *JMLR*, 2014.
[3] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3:397–422, 2003.
[4] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts. Everyone's an influencer: Quantifying influence on twitter. In *WSDM*, pages 65–74, 2011.
[5] N. Cesa-Bianchi, C. Gentile, and G. Zappella. A gang of bandits. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 737–745, 2013.
[6] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec. Can cascades be predicted? In *WWW*, pages 925–936, 2014.
[7] W. Chu, L. Li, L. Reyzin, and R. Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214, 2011.
[8] V. Dani, T. P. Hayes, and S. M. Kakade. Stochastic linear optimization under bandit feedback. In *COLT*, 2008.
[9] P. M. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26-29, 2001*, pages 57–66, 2001.
[10] R. G. Gallager. *Discrete Stochastic Processes*. Springer, 1996.
[11] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *SIGKDD*, pages 137–146. ACM, 2003.
[12] S. Lei, S. Maniu, L. Mo, R. Cheng, and P. Senellart. Online influence maximization. In *KDD*, pages 645–654, 2015.
[13] L. Li, Y. Lu, and D. Zhou. Provably optimal algorithms for generalized linear contextual bandits. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, pages 2071–2080, 2017.
[14] W. Lu, S. Ioannidis, S. Bhagat, and L. V. Lakshmanan. Optimal recommendations under attraction, aversion, and social influence. In *KDD*, pages 811–820, 2014.
[15] S. Maniu, S. Ioannidis, and B. Cautis. Bandits under the influence (extended version). *CoRR*, abs/2009.10135, 2020.
[16] Q. Wu, H. Wang, Q. Gu, and H. Wang. Contextual bandits in a collaborative environment. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 529–538, 2016.