Non-stochastic Low Rank Bandit

Author names withheld

Abstract

We study the problem of learning the maximum entry of a low-rank non-negative matrix, from sequential observations. In this setting, the learner chooses a pair of row and column at every timestep and observes the product of their values. The main challenge in this setting is that the learner does not observe the individual latent values of rows and columns as its feedback. Diverging from previous work we assume that the preference matrix is non-stochastic and hence our setting is more general in nature. Existing methods for solving similar problems are based on constructing conservative confidence interval with the strong assumption that underlying distributions are stochastic and i.i.d. We depart from this standard approach and consider the case when the best row and column pair can be learned jointly with help of two separate bandit algorithms working individually on rows and columns. We propose a simple and computationally efficient algorithm that implements this procedure, which we call Low-Rank Bandit (LRB), and prove a sublinear bound on its n-step regret. We evaluate the algorithm empirically on several synthetic and real-world datasets. In all experiments, we outperform existing state-of-the-art algorithms.

1 Introduction

In this work, we study the problem of learning the maximum entry of a low-rank matrix from sequential observations. These type of low-rank structure is observed in many real-world applications and is a standard assumption in recommender systems [Koren *et al.*, 2009; Ricci, 2011]. Our learning model is motivated by a real-world scenario, where a marketer wants to advertise a product and has K population segments and L marketing channels. Now, given a product some population segment prefer some marketing channels more than other. Hence, a successful conversion happens if each population segment is matched to the correct marketing channels which is nothing but the maximum entry of a rank-1 matrix formed by the outer product of the users preference for the product and marketing channels.

An instance of the problem is defined by the tuple (K, L, P_U, P_V), where K is the number of rows, L is the number of columns, P_U and P_V are probability distributions over unit hypercube $[0,1]^{K\times 1}$ and $[0,1]^{L\times 1}$ respectively. Most importantly, note that we do not assume that these probability distributions are stochastic rather, we consider the more general case that they can be fully non-stochastic or even piecewise i.i.d. We formalize our learning problem as the following online learning problem. At time t, the learning agent chooses a pair of row and column arms denoted by $(i_t, j_t) \in [K]\times[L]$ and receives a noisy product of their latent values. This noisy product is denoted by $u_t(i_t)v_t(j_t)$ which is treated as a feedback to the learner. Note, that the learner does not observe the individual latent values but just a noisy realization of their product. The goal of the learning agent is minimize the cumulative regret by quickly identifying the best row and column pair.

Previous works that have studied this setting have either proposed highly conservative algorithms or restricted themselves to a stricter set of assumptions. While Katariya *et al.* [2016] was proposed for a rank-1 bandit model with the assumption that the underlying distributions P_u and P_v are stochastic, Katariya *et al.* [2017] was proposed for the special case when the underlying distributions are Bernoulli. Both these works used different variations of the phase-based UCB-Improved [Auer and Ortner, 2010] algorithm to construct confidence interval to identify and eliminate sub-optimal rows and columns. These naturally results in algorithms that explore conservatively (for the sake of row and column elimination) and cannot work beyond the stochastic distribution assumption. Finally, Kveton *et al.* [2017] can be viewed as a generalization of rank-1 bandits of Katariya *et al.* [2016] to a higher rank of d. However, this work proposes a phase-based algorithm that calculates the determinant of a $d \times d$ sub-matrix to eliminate sub-optimal rows and columns at the end of phases which is impractical for very large low-rank matrices.

Our approach is based on two key insights. First, the Upper Confidence Bound (UCB) based algorithms which are explicitly modeled on the stochastic i.i.d assumption on feedback cannot perform well in non-stochastic settings. Moreover, their theoretical guarantees will also fail in non-stochastic setting. Hence, we need algorithms that can work on more generalized non-stochastic probability distribution settings. Secondly, we can formulate simple and computationally efficient algorithms that learn the best column and best row jointly with two separate non-stochastic bandit algorithm operating on rows and columns individually without needing any sort of row or column eliminations.

We make four major contributions. First, we formulate our online learning problem as a non-stochastic bandit problem on a class of non-negative low-rank matrices. We identify a family of non-negative low-rank matrices where our problem can be solved statistically efficiently, without actually observing the latent values of individual rows and columns. Second, we propose a computationally-efficient algorithm that implements this idea, which we call Low Rank Bandit (LRB) algorithm. The algorithm has two components, column learning and row learning, which learn the pair of optimal columns and rows respectively. Since we are in the non-stochastic setting we use a variation of the Exp3[Auer *et al.*, 2002b] algorithm as our row and column learner. Note, that we do not construct any confidence interval or eliminate rows and columns like the existing works. Infact, we use the well known fact that exponentially weighted algorithm like Exp3are robust and fast learner to construct our algorithm. The Third, we analyze LRB and up to problem-specific factors, we prove a $O\left(\frac{(\sqrt{L}+\sqrt{K})n}{\Delta}\right)$ upper bound on its n-step regret. The regret of a naive solution is $O(\sqrt{KLn})$, and is much worse than that of LRB when all of K, L, and n are large. Finally, we evaluate LRB empirically on several synthetic and real-world problems. Perhaps surprisingly, LRB performs well even when our modeling assumptions are violated.

The paper is organized as follows. We introduce necessary background to understand our work in Section 2 and define our online learning problem in Section 3. We propose our algorithm in Section 4 and bound its regret in Section 5. In Section 6, we evaluate the algorithm empirically. In Section 7, we survey related work. We conclude in Section 8. The detailed proof of our regret bound is presented in **??**.

2 Background

Let $[n] = \{1, \dots, n\}$ be the set of the first n positive integers. For any two sets A and B, we denote by A^B the set of all vectors whose entries take values from A and are indexed by B. Let M be any $m \times n$ matrix. We index the rows and columns of matrices by vectors. For any d and $I \in [m]^d$, $M(I, :)$ denotes a $d \times n$ submatrix of M whose i-th row is $M(I(i), :)$. Similarly, for any d and $J \in [n]^d$, $M(:, J)$ denotes a $m \times d$ submatrix of M whose j-th column is $M(:, J(j))$. Let Π_d be the set of all d-permutations. For any $\pi \in \Pi_d$ and d-dimensional vector v, we denote by $\pi(v)$ the permutation of the entries of v according to π.

We focus on a family of low-rank matrices, which are known as hott topics. We define a *hott-topics matrix* of rank d as $M = UV^\mathsf{T}$, where U is a $K \times d$ non-negative matrix and V is a $L \times d$ non-negative matrix that gives rise to the hott-topics structure. In particular, we assume that there exist d rows J^* in V such that each row of V can be expressed as a convex combination of rows J^* and the zero vector,

$$\forall j \in [L] \; \exists \alpha \in A : V(J^*, :)\alpha = V(j, :) \,, \qquad (1)$$

where $A = \{a \in [0, 1]^{d \times 1} : \|a\|_1 \leq 1\}$.

The matrix M represents preferences of users for items, $M(i, j)$ is the preference of user i for item j. The rank d of M is the number of latent topics. The matrix U are latent preferences of K users over d topics, where $U(i, :)$ are the preferences of user $i \in [K]$. Without loss of generality, we assume that $U \in [0, 1]^{L \times d}$. The matrix V are latent preferences of L items in the space of d topics, where $V(j, :)$ are the coordinates of item $j \in [L]$. We assume that the coordinates are points in a simplex, that is $\|V(j, :)\|_1 \leq 1$ for all $j \in [L]$. Note that our assumptions imply that $M(i, j) \geq 0$ for any $i \in [K]$ and $j \in [L]$.

3 Setting

We study an online learning to rank problem, which we call a *latent ranked bandit*. At time t, the preferences of users are encoded in a $K \times L$ *preference matrix* $M_t = U_t V^\mathsf{T}$, where M, U_t, and V are defined as in Section 2. We assume that user preferences U_t can change with time t. A random user $i_t \in [K]$ arrives to the recommender system at time t and we recommend d items J_t to this user. The *reward* for recommending these items is $r_t(i_t, J_t)$, where

$$r_t(i, J) = \max\left\{\mu(k)\, M_t(i, J(k)) : k \in [d]\right\} \qquad (2)$$

is the reward for recommending items J to user i at time t, $J(k)$ is the k-th item in J, and $\mu(k)$ is the weight of position $k \in [d]$. We assume that higher-ranked positions are more rewarding, $1 \geq \mu(1) \geq \dots \geq \mu(d) \geq 0$. The learning agent *observes* the individual rewards of all recommended items, $M_t(i_t, J_t(k))$ for all $k \in [d]$.

Since U_t can change arbitrarily over time, the reward in (2) is maximized by lists J with highly rewarding items that are diverse, in the sense that they attain high rewards at different times $t \in [n]$. Because the rewards are weighted by μ, more frequent highly-rewarding items should be placed at higher positions. A remarkable property of our user-item preference matrices M_t is that for any user $i \in [K]$ at any time t,

$$\arg\max_{j \in [L]} M_t(i, j) \in J_* \,,$$

where J_* is defined in (1). Therefore, it is possible to learn all potentially most rewarding items statistically efficiently.

Now we are ready to define our notion of optimality and regret. Let J_* be the hott-topics items in (1) and $\pi_{*,i}$ be their permutation that maximizes the reward of user i in hindsight,

$$\pi_{*,i} = \arg\max_{\pi \in \Pi_d} \sum_{t=1}^{n} r_t(i, \pi(J_*)) \,.$$

Let J_t be our recommended items at time t and $\pi_{t,i}$ be their permutation for user i, both of which are learned. Then our goal is to minimize the expected n-step regret,

$$R(n) = \sum_{t=1}^{n} \mathbb{E}\left[r_t(i_t, \pi_{*,i_t}(J_*)) - r_t(i_t, \pi_{t,i_t}(J_t))\right] \,, \qquad (3)$$

where the expectation is with respect to both randomly arriving users and potential randomness in the learning algorithm.

Algorithm 1 Low Rank Bandit (LRB) (Rank-1 case)

Input: Time horizon n ▷ Initialization
Initialize `ColAlg`
Initialize `RowAlg`
for $t = 1, \dots, n$ **do**
  Suggest column j_t by `ColAlg` ▷ Generate response
  Suggest row i_t by `RowAlg`
  Observe feedback $u_t(i_t)v_t(j_t)$ ▷ Update statistics
  Update arm j_t of `ColAlg` with reward $u_t(i_t)v_t(j_t)$
  Update arm i_t of `RowAlg` with reward $u_t(i_t)v_t(j_t)$

4 Algorithm

We propose *Low Rank Bandit (*`LRB`*)* for solving the family of non-stochastic, non-negative and low-rank matrices where the goal is to identify the best pair of rows and columns. The pseudocode of `LRB` is in Algorithm 1. `LRB` has two main components, column learning and row learning algorithm.

At every timestep t, the column learning algorithm recommends a column j_t and a row i_t. It then observes the reward $u_t(i_t)v_t(j_t)$. The `ColAlg` learns the most rewarding column on average, while `RowAlg` learns the most rewarding row on average. Once it observes the reward it updates the statistics of each algorithm with the reward $u_t(i_t)v_t(j_t)$.

4.1 Practical Considerations

Note, that we leave the implementation of the `ColAlg` and `RowAlg` to the users. For theoretical guarantees we use non-stochastic algorithm `Exp3` as `ColAlg` and `RowAlg` which will be explained in detail in section Section 5. For experimental purposes, stochastic algorithms like `UCB1` or thompson sampling can also be used to improve the performance of `LRB`. This has also been explored in Radlinski *et al.* [2008] where `RBA` uses `UCB1` for ranking items.

5 Analysis

Theorem 1. *Let* `ColAlg` *and* `RowAlg` *in* `LRB` *be* `Exp3` *algorithm, respectively. Then the expected n-step regret of* `LRB` *is bounded as*

$$R(n) = O\left(\frac{\left(\sqrt{L}+\sqrt{K}\right)n}{\Delta}\right)$$

where $\Delta = \min_{t\in[n]} \min_{i_t,j_t:\ ,i_t\neq i_t^*\ j_t\neq j_t^*} \mathbb{E}\left[u_t(i_t^*)v_t(j_t^*)\right] - \mathbb{E}\left[u_t(i_t)v_t(j_t)\right]$ *is an instance-specific lower bound on the gap in the expected rewards of the optimal and best suboptimal columns and rows at any time $t \in [n]$, averaged over all users at that time.*

Proof. Let, $(u_t v_t^\intercal)_{t=1}^n$ be a sequence of n non-negative rank-1 matrices such that $u_t \in [0,1]^{K\times 1}, v_t \in [0,1]^{L\times 1}$, and the highest entry is $(1,1)$. Let,

$$((i_t, j_t))_{t=1}^n$$

be a sequence of n row-column pairs chosen by a learning agent. Then the expected n-step regret of the agent is,

$$R(n) = \sum_{t=1}^n \mathbb{E} u_t(1)v_t(1) - u_t(i_t)v_t(j_t)$$

where the expectation is over the randomness of the agent. Now note that for any u, v, i, and j in our problem we can show that,

$$\begin{aligned}
&2(u(1)v(1) - u(i)v(j)) \\
&= 2u(1)v(1) - u(i)v(1) - u(1)v(j) + u(i)v(1) + u(1)v(j) - 2u(i)v(j) \\
&= u(1)(v(1) - v(j)) + v(1)(u(1) - u(i)) + \\
&\qquad u(i)(v(1) - v(j)) + v(j)(u(1) - u(i)) \\
&= (u(1) + u(i))(v(1) - v(j)) + (v(1) + v(j))(u(1) - u(i))
\end{aligned}$$

Therefore, the expected n-step regret can be decomposed as

$$\begin{aligned}
R(n) = &\sum_{t=1}^n \mathbb{E}(v_t(1) + v_t(j_t))(u_t(1) - u_t(i_t)) \\
&+ \sum_{t=1}^n \mathbb{E}(u_t(1) + u_t(i_t))(v_t(1) - v_t(j_t))
\end{aligned}$$

Now suppose that all entries of u_t and v_t for all $t = 1, 2, \dots, n$ are bounded from below by some $\Delta > 0$. Then we get that,

$$\begin{aligned}
R(n) = &\sum_{t=1}^n \mathbb{E}(1 + v_t(1)/v_t(j_t))v_t(j_t)(u_t(1) - u_t(i_t)) + \\
&\sum_{t=1}^n \mathbb{E}(1 + u_t(1)/u_t(i_t))u_t(i_t)(v_t(1) - v_t(j_t)) \\
&\leq (1 + \frac{1}{\Delta})\Bigg[\sum_{t=1}^n \mathbb{E} u_t(1)v_t(j_t) - u_t(i_t)v_t(j_t) + \\
&\sum_{t=1}^n \mathbb{E} u_t(i_t)v_t(1) - u_t(i_t)v_t(j_t)\Bigg]
\end{aligned}$$

Finally, we can show from the result of Auer *et al.* [2002b] that the `ColAlg` using `Exp3` chooses the column j_t at time t and observe reward is $u_t(i_t)v_t(j_t)$. Therefore, the first sum above is bounded by $\sqrt{Ln}$ for any sequence of j_t, and thus also in expectation over the randomness in j_t. Similarly `RowAlg` using `Exp3` chooses the row i_t at time t, and observe reward is $u_t(i_t)v_t(j_t)$. Therefore, the second sum above is bounded by $\sqrt{Kn}$ for any sequence of i_t, and thus also in expectation over the randomness in i_t. Therefore we get the final regret as,

$$R(n) = O\left(\frac{\left(\sqrt{L}+\sqrt{K}\right)n}{\Delta}\right)$$

□

Discussion 1. *The main idea is to decompose the regret of* `LRB` *into two parts, where* `ColAlg` *does not suggest* j_t^* *and the rest. The first part is analyzed as follows.* `ColAlg` *has a sub-linear regret, based on a similar analysis to Auer* et al. *[2002b]. Therefore, our upper bound on the probability that* `ColAlg` *suggests suboptimal columns, which is* $O(d\sqrt{Ln}/\Delta)$*, decreases with time horizon* n*. Similarly, we analyze the regret for the* `RowAlg` *as it also uses the exponentially weighted algorithm* `Exp3`*.*

The regret in Theorem 1 consists of two main parts. The first part, which is $O(\sqrt{Ln}/\Delta)$*, is the regret due to learning the optimal column with a high probability. The second part, which is* $O(\sqrt{Kn}/\Delta)$*, is the regret due to learning the optimal row with a high probability.*

Our regret bound also improves upon a trivial approach where the optimal columns are learned separately for each user. If this problem was solved by RBA [Radlinski et al.*, 2008], the regret would be* $O(\sqrt{KLn})$*.*

Finally, we use non-stochastic algorithms for `ColAlg` *and* `RowAlg` *because our environment is non-stationary. In particular, we assume that user preferences* U_t*, and thus rewards, can change over time* t*.*

6 Experiments

In this section, we compare `LRB` to several bandit algorithms in three experiments. The first two experiments are on synthetic dataset where all modeling assumptions hold. The third experiment is on a real-life dataset where we evaluate `LRB` when our modeling assumptions fail. In all our experiments user come uniform randomly over all time $[n]$. All results are averaged over 10 independent random runs.

6.1 Evaluated Algorithms

Independent User Model Algorithms: In this approach, each user has a separate version of base-bandit algorithm running independent of each other. As base-bandit algorithms we choose two variants of the ranked bandit algorithm (`RBA`) of Radlinski *et al.* [2008]. The two variants of `RBA` uses two types of column learning algorithms, `UCB1` [Auer *et al.*, 2002a] and `Exp3` [Auer *et al.*, 2002b], abbreviated as $\texttt{RBA}-\texttt{UCB1}$ and $\texttt{RBA}-\texttt{EXP3}$ respectively. `Exp3` is a randomized algorithm suited for the adversarial setting while `UCB1` is the standard algorithm used in the stochastic feedback setting. For $\texttt{RBA}-\texttt{UCB1}$, we choose the confidence interval at time t as $c_{i,j}(t) = \sqrt{\frac{2\log t}{N_{i,j}(t)}}$ for user i and item j. Here, $N_{i,j}(t)$ denotes the number of times the j-th item has been observed by the i-th user base-bandit algorithm till timestep t. Note, that running independent vanilla `UCB1` and `Exp3` for every user is not feasible. This is because the vanilla versions are guaranteed to find a single best item for each user at rank 1, while $\texttt{RBA}-\texttt{UCB1}$ and $\texttt{RBA}-\texttt{EXP3}$ will find a diverse list of d best items for each user.

Matrix Completion Algorithms: In the matrix completion approach, the algorithms try to reconstruct the user-item preference matrix M from its noisy realization. We implement the widely used non-negative matrix factorization method to reconstruct partially observed noisy matrices. We term the corresponding algorithm as NMF Bandit ($\texttt{NMF}-\texttt{Ban}$). The objective function of $\texttt{NMF}-\texttt{Ban}$ is:-

$$\text{minimize} \left\| \hat{M} - \hat{U}\hat{V}^\mathsf{T} \right\|_F^2 \text{ with respect to } \hat{U}, \hat{V}$$
$$\text{subject to constraints } \hat{U}, \hat{V} \geq 0$$

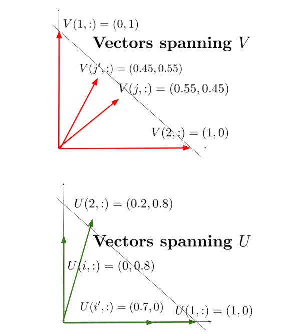
where, $\hat{M}$ is the observed noisy matrix of size $K \times L$ which has a low rank d, $\hat{U} \in [0,1]^{K\times d}$ and $\hat{V} \in [0,1]^{L\times d}$ are estimated non negative matrices which generates $\hat{M}$ such that $\hat{M} \approx \hat{U}\hat{V}^\mathsf{T}$. This objective function is minimized by alternating minimization of $\hat{U}$ and $\hat{V}$ till the loss is very low. $\texttt{NMF}-\texttt{Ban}$ knows the rank of the matrix $\hat{M}$. This algorithm is explore-exploit in implementation whereby it first explores for $cd(K+L)$ rounds by choosing items for incoming users uniform randomly, where c is an exploration parameter which can be tuned depending on the noise in the system. We set $c = 10$ in all experiments. Then it reconstructs $\hat{M}$ using the objective function mentioned above. Then over the reconstructed matrix it behaves greedily and suggest d best items based on decreasing order of their preferences for the i_t-th user at every timestep t.

Personalized Ranking Algorithms: In this approach, we evaluate our proposed algorithm latent ranking bandit (`LRB`) by using three different types of column learning algorithms, `Exp3`, thompson sampling [Thompson, 1933], [Thompson, 1935], [Agrawal and Goyal, 2012] and `UCB1`. We term them as $\texttt{LRA}-\texttt{EXP3}$, $\texttt{LRA}-\texttt{TS}$ and $\texttt{LRA}-\texttt{UCB1}$ respectively. Note, that thompson sampling is a Bayesian algorithm that performs better than `UCB1` in stochastic setting due to its inherent prior assumptions on the distribution of the feedback. The row ranking components for all of these algorithms is the weighted majority algorithm (`WMA`) from Littlestone and Warmuth [1994] which is suited for the full information setting. Note that we only show theoretical guarantees for $\texttt{LRA}-\texttt{EXP3}$. We initialize the k-th column EXP3 with the column exploration parameter $\gamma_k = \sqrt{\frac{L\log L}{n}}$ as stated in Auer *et al.* [2002b]. Similarly, for $\texttt{LRA}-\texttt{UCB1}$ we use a confidence interval of $c_{k,j}(t) = \sqrt{\frac{2\log t}{N_{k,j}(t)}}$ for the k-th column MAB and j-th item. Here, $N_{k,j}(t)$ denotes the number of times the j-th item has been observed by the k-th column `UCB1` algorithm till timestep t.

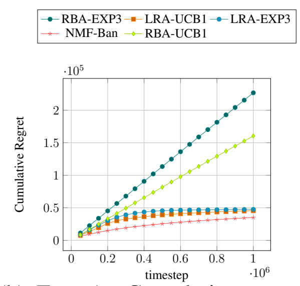
6.2 Synthetic Experiment 1

This experiment is conducted to test the performance of `LRB` over small number of users and items. This simulated testbed consist of 500 users, 50 items, and rank$(M) = 2$. The vectors spanning U and V, generating the user-item preference matrix M, are shown Figure 1(a). The users are evenly distributed into a $50 : 50$ split such that 50% of users prefer item 1 and 50% users prefer item 2. The item hott-topics are $V(1,:) = (0,1)$ and $V(2,:) = (1,0)$ while remaining 70% of items has feature $V(j',:) = (0.45, 0.55)$ and the rest have $V(j,:) = (0.55, 0.45)$. We create the user feature matrix U similarly having a $50 : 50$ split such that $U(1,:) = (0,1)$, $U(2,:) = (0.2, 0.8)$ and the remaining 70% users having $U(i,:) = (0, 0.8)$ and 30% users having $U(i',:) = (0.7, 0)$.

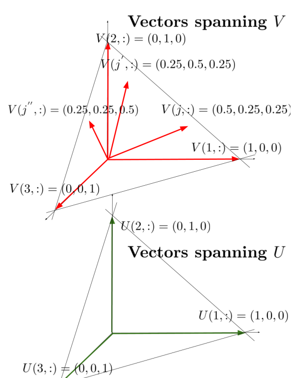
At every timestep t the resulting matrix $M_t = UD_tV^\intercal$ is generated where D_t is a randomly-generated diagonal matrix. So, M_t is such that algorithms that quickly find the easily identifiable hott-topics perform very well. From Figure 1(b) we can clearly see that $\texttt{LRA} - \texttt{EXP3}$, $\texttt{LRA} - \texttt{TS}$ and $\texttt{LRA} - \texttt{UCB1}$ outperforms all the other algorithms. Their regret curve flattens, indicating that they have learned the best items for each user. Independent user model algorithms $\texttt{RBA} - \texttt{UCB1}$ and $\texttt{RBA} - \texttt{EXP3}$ perform poorly as the number of items per user is too large and the independent algorithms are not sharing information between them. $\texttt{NMF} - \texttt{Ban}$ performs better than the independent user model algorithms but is outperformed by $\texttt{LRA} - \texttt{EXP3}$, $\texttt{LRA} - \texttt{TS}$ and $\texttt{LRA} - \texttt{UCB1}$.



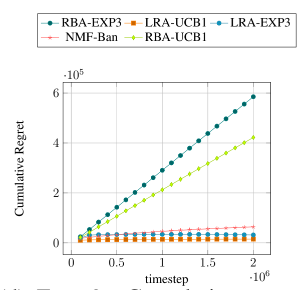
(a) Expt-1: 500 Users, 50 items, Rank 2, User and Item vectors



(b) Expt-1: Cumulative regret of different algorithms



(c) Expt-2: 1500 Users, 100 items, Rank 3, User and Item vectors



(d) Expt-2: Cumulative regret of different algorithms

Figure 1: A comparison of the cumulative regret incurred by the various bandit algorithms.

6.3 Synthetic Experiment 2

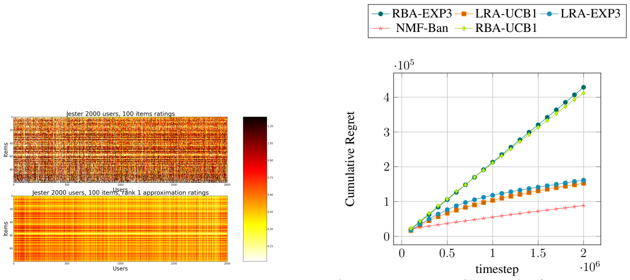
We conduct the second experiment on a larger simulated database of 1500 users, 100 items and rank$(M) = 3$. The vectors spanning U and V, generating the user-item preference matrix M is shown Figure 1(c). The users are divided into an unequal distribution of $60 : 30 : 10$ split such that 60% of the users prefer item item 1, 30% prefer item 2 and 10% prefer item 3. Hence, in this testbed it is difficult to learn item 3 as it is observed for less number of users. Here, hott-topics are $V(1,:) = (1,0,0)$, $V(2,:) = (0,1,0)$ and $V(3,:) = (0,0,1)$. The remaining 60% of items have feature $V(j,:) = (0.5, 0.25, 0.25)$, 30% have $V(j',:) = (0.25, 0.5, 0.25)$ and rest have $V(j'',:) = (0.25, 0.25, 0.5)$. We create the user feature matrix U similarly having a $60 : 30 : 10$ split and the vectors spanning U are only of the type that spans the simplex, i.e $U(i,:) = (1,0,0)$, $U(i',:) = (1,0,0)$ and $U(i'',:) = (1,0,0)$. Again, at every timestep t the resulting matrix $M_t = UD_tV^\intercal$ is generated where D_t is a randomly-generated diagonal matrix. So, M_t is such that algorithms that quickly find the easily identifiable hott-topics perform very well. From Figure 1(d) we can see that $\texttt{LRA} - \texttt{EXP3}$, $\texttt{LRA} - \texttt{TS}$ and $\texttt{LRA} - \texttt{UCB1}$ again outperform all the other algorithms. Their regret curve flattens much before all the other algorithms indicating that they have learned the best items for each user. The matrix completion algorithm $\texttt{NMF} - \texttt{Ban}$ again fails to get a reasonable approximation of M and performs poorly. Also, we see that both the independent user model algorithms $\texttt{RBA} - \texttt{UCB1}$ and $\texttt{RBA} - \texttt{EXP3}$ perform poorly as the number of users and the number of items per user is too large and the independent base-bandits (RBA) are not sharing information between themselves. In both the synthetic datasets, we see that stochastic column learning algorithm (UCB1) is outperforming adversarial column learning algorithm (Exp3) as the user preference over the best item is not changing over time. This has also been observed by Radlinski *et al.* [2008].

6.4 Real World Experiment 3

We conduct the third experiment to test the performance of LRA when our modelling assumptions are violated. We use the Jester dataset [Goldberg *et al.*, 2001] which consist of over 4.1 million continuous ratings of 100 jokes from 73,421 users collected over 5 years. In this dataset there are many users who rated all jokes and we work with these users. Hence the user-item preference matrix is fully observed and we will not have to complete it using matrix completion techniques. Hence, this approach is very real world. We sample randomly 2000 users (who have rated all jokes) from this dataset and use singular value decomposition (SVD) to obtain a rank 4 approximation of this user-joke rating matrix M. In the resultant matrix M, most of the users belong to the four classes preferring jokes 99, 93, 96 and 28, while a very small percentage of users prefer some other jokes. Note, that this condition results from the fact that this real-life dataset does not have the hott-topics structure. The rank 4 approximation of M of is shown in Figure 2(a), where we can clearly see the red stripes spanning the matrix indicating the low-rank structure of M. Furthermore, in this experiment we assume that the noise is independent Bernoulli over the entries of M and hence this experiment deviates from our modeling assumptions. From 2(b) again we see that $\texttt{LRA} - \texttt{EXP3}$, $\texttt{LRA} - \texttt{TS}$ and $\texttt{LRA} - \texttt{UCB1}$ outperform other algorithms. Although the cumulative regret of $\texttt{NMF} - \texttt{Ban}$ is less than our proposed approaches, note that it does not converge and find the d best items.

7 Related Work

Our work lies at the intersection of several existing areas of research, which we survey below.



(a) Expt-3: 2000 Users, 100 items, Rank 1 approximation of Jester Dataset

(b) Expt-3: Cumulative regret of different algorithms

Figure 2: A comparison of the cumulative regret in Jester Dataset

Bandits for Latent Mixtures: The existing algorithms in latent bandit literature can be broadly classified into two groups: the online matrix completion algorithms and the independent user model algorithms. The *online matrix completion algorithms* try to reconstruct the user-item preference matrix M from a noisy realization combining different approaches of online learning algorithms and matrix factorization algorithms. The NMF-Bandit algorithm in Sen *et al.* [2016] is an online matrix completion algorithm which is an ϵ-greedy algorithm that tries to reconstruct the matrix M through non-negative matrix factorization. Note, that this approach requires that all the matrices satisfy a weak statistical Restricted Isometric Property, which is not always feasible in real life applications. Another approach is that of Gopalan *et al.* [2016] where the authors come up with an algorithm which uses the Robust Tensor Power (RTP) method of Anandkumar *et al.* [2014] to reconstruct the matrix M, and then use the OFUL procedure of Abbasi-Yadkori *et al.* [2011] to behave greedily over the reconstructed matrix. But the RTP is a costly operation because the learner needs to construct a matrix of order $L \times L$ and $L \times L \times L$ to calculate the second and third order tensors for the reconstruction. A more simpler setting has also been studied in Maillard and Mannor [2014] where all the users tend to come from only one class and hence this approach is also not quite realistic.

The second type of algorithms are the *independent user model algorithms* where for each user $i \in [K]$ a separate instance of a base-bandit algorithm is implemented to find the best item for the user. These base -bandits run independent of each other without sharing any information. These base-bandits can be any randomized algorithms suited for the adversarial setting or stochastic algorithms which tend to perform better under stochastic feedback assumptions.

Ranked Bandits: Bandits have been used to rank items for online recommendations where the goal is is to present a list of d items out of L that maximizes the satisfaction of the user. A popular approach is to model each of the d rank positions as a Multi Armed Bandit (MAB) problem and use a base-bandit algorithm to solve it. This was first proposed in Radlinski *et al.* [2008] which showed that query abandonment by user can also be successfully used to learn rankings. Later works on ranking such as Slivkins *et al.* [2010] and Slivkins *et al.* [2013] uses additional assumptions to handle exponentially large number of items such that items and user models lie within a metric space and satisfy Lipschitz condition.

Ranking in Click Models: Several algorithms have been proposed to solve the ranking problem in specific click models. Popular click models that have been studied extensively are Document Click Model (DCM), Position Based Click Model (PBM) and Cascade Click Model (CBM). For a survey of existing click models a reader may look into Chuklin *et al.* [2015]. While Katariya *et al.* [2017], Katariya *et al.* [2016] works in PBM, Zoghi *et al.* [2017] works in both PBM and CBM. Finally, Kveton *et al.* [2017] can be viewed as a generalization of rank-1 bandits of Katariya *et al.* [2016] to a higher rank. Note, that the theoretical guarantees of these algorithms does not hold beyond the specific click models.

Online Sub-modular maximization: Maximization of submodular functions has wide applications in machine learning, artificial intelligence and in recommender systems [Nemhauser *et al.*, 1978], [Krause and Golovin, 2014]. Intuitively, a submodular function states that after performing a set A of actions, the marginal gain of another action e does not increase the gain for performing other actions in $B \setminus A$. Online submodular function maximization has been studied in Streeter and Golovin [2009] where the authors propose a general algorithm whereas Radlinski *et al.* [2008] can be considered as special case of it when the payoff is only between $\{0, 1\}$. Also, in the contextual feature based setup online submodular maximization has been studied by Yue and Guestrin [2011]. An interesting property of submodular function is that a greedy algorithm using it is guaranteed to perform atleast $\left(1 - \frac{1}{e}\right)$ of the optimal algorithm and this factor $\left(1 - \frac{1}{e}\right)$ is not improvable by any polynomial time algorithm [Nemhauser *et al.*, 1978]. Note that the $\max$ function is a submodular function which satisfies the condition of monotonicity and submodularity.

8 Conclusions

In this paper, we studied the problem of suggesting a diverse list of items to users, with the best permutation of those items for individual users. The best permutation of items for an user contains its preference for the items in descending order with the best item at rank position 1. We formulated the above problem as a personalized ranking problem and proposed the latent ranker algorithm for this setting. We proved that an instance of algorithm has a regret bound that scales as $O\left(\frac{d\sqrt{Ln}}{\Delta} + K\log n + Kd\log d\right)$ and has the correct order with respect to users, items and rank of the user-item preference matrix M. We also evaluated our proposed algorithm on several simulated and real-life datasets and show that it outperforms the existing state-of-the-art algorithms.

There are several directions where this work can be extended. Note, that observing d items at every timestep is helping LRA to learn more efficiently. Hence, while keeping the hott-topics assumption it is worthwhile to study the personalized ranking setting when only 1 item is allowed to be suggested at every timestep t. Another interesting direction is to look at structures where there are hott-topics assumption on user matrix as well as item matrix or maybe even at structures beyond hott-topics.

References

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 2312–2320, 2011.

Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*, pages 39.1–39.26, 2012.

Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research*, 15(1):2773–2832, 2014.

Peter Auer and Ronald Ortner. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010.

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

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. Click models for web search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 7(3):1–115, 2015.

Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *information retrieval*, 4(2):133–151, 2001.

Aditya Gopalan, Odalric-Ambrym Maillard, and Mohammadi Zaki. Low-rank bandits with latent mixtures. *arXiv preprint arXiv:1609.01508*, 2016.

Sumeet Katariya, Branislav Kveton, Csaba Szepesvari, Claire Vernade, and Zheng Wen. Stochastic rank-1 bandits. *arXiv preprint arXiv:1608.03023*, 2016.

Sumeet Katariya, Branislav Kveton, Csaba Szepesvári, Claire Vernade, and Zheng Wen. Bernoulli rank-1 bandits for click feedback. *arXiv preprint arXiv:1703.06513*, 2017.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

Andreas Krause and Daniel Golovin. Submodular function maximization., 2014.

Branislav Kveton, Csaba Szepesvari, Anup Rao, Zheng Wen, Yasin Abbasi-Yadkori, and S Muthukrishnan. Stochastic low-rank bandits. *arXiv preprint arXiv:1712.04644*, 2017.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, 1994.

Odalric-Ambrym Maillard and Shie Mannor. Latent bandits. In *International Conference on Machine Learning*, pages 136–144, 2014.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*, pages 784–791. ACM, 2008.

Francesco Ricci. Liorrokach, and brachashapira." introduction to recommender systems handbook, 2011.

Rajat Sen, Karthikeyan Shanmugam, Murat Kocaoglu, Alexandros G Dimakis, and Sanjay Shakkottai. Contextual bandits with latent confounders: An nmf approach. *arXiv preprint arXiv:1606.00119*, 2016.

Aleksandrs Slivkins, Filip Radlinski, and Sreenivas Gollapudi. Ranked bandits in metric spaces: learning optimally diverse rankings over large document collections. *arXiv preprint arXiv:1005.5197*, 2010.

Aleksandrs Slivkins, Filip Radlinski, and Sreenivas Gollapudi. Ranked bandits in metric spaces: learning diverse rankings over large document collections. *Journal of Machine Learning Research*, 14(Feb):399–436, 2013.

Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *Advances in Neural Information Processing Systems*, pages 1577–1584, 2009.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

William R Thompson. On the theory of apportionment. *American Journal of Mathematics*, 57(2):450–456, 1935.

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

Masrour Zoghi, Tomas Tunys, Mohammad Ghavamzadeh, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. Online learning to rank in stochastic click models. *arXiv preprint arXiv:1703.02527*, 2017.