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 pairs of rows and columns 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 rely on UCB-type algorithms 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 sub-linear bound on its n-step regret in the rank-1 special case. 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 the matrix formed by the outer product of the users preference for the product and marketing channels.

We formalize our learning problem as the following online learning problem. At time t, the learning agent chooses d-pairs of rows and columns where d is the rank of a non-negative and low-rank matrix M. This user-item preference matrix M is formed by the outer product of user and item latent preference over d topics. Note that the learner does not observe the individual latent values of user or item preference but just a noisy realization of their product. The user-item preference matrix M is low-rank at each time t, can vary substantially over time, and does not have to be stochastic. The goal of our learning agent is to minimize the cumulative regret with respect to a best solution in hindsight by finding the maximum entry in M as fast as possible.

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 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 square of 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 non-negative low-rank matrices. Some other approaches involving non-negative matrix factorization Sen *et al.* [2016] or tensor based methods [Gopalan *et al.*, 2016] to reconstruct the matrix have also been proposed. These works require strong assumptions on the structure of the matrix such as all the matrices satisfy a weak statistical Restricted Isometric Property (RIP) or calculate third order tensors as in [Anandkumar *et al.*, 2014]. A more simpler setting has also been studied in Maillard and Mannor [2014].

Our approach is based on two key insights. First, the earlier methods (like Upper Confidence Bound (UCB) algorithms, NMF-Bandits) are explicitly modeled on the stochas-

tic i.i.d assumption on feedback and 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 set of columns and best set of rows jointly with two separate non-stochastic bandit algorithm operating on rows and columns individually. These does not require any sort of costly matrix inversion or reconstruction operations or even row or column eliminations and hence are faster in implementation.

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 `Exp3` are robust and fast learner to construct our algorithm. The Third, we analyze `LRB` and up to problem-specific factors, we prove a $O\left(\frac{(\sqrt{L}+\sqrt{K})\sqrt{n}}{\Delta}\right)$ upper bound on its n-step regret in the special case when rank is 1. 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 first analyze the rank 1 setting in Section 4 to explain the crucial aspects of our design. We then propose our algorithm for the general rank d in Section 5 and bound the regret of rank 1 algorithm in Section 6. In Section 7, we evaluate the algorithm empirically. We conclude in Section 8.

2 Background

Let $[n] = \{1, \ldots, n\}$ 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.

B: This section seems like an exact copy of the arxiv paper. Please change language. Otherwise we may be accused of plagiarism.

Hott-topics Assumption: 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 $M = UV^\intercal$, 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 exists d rows I^* in U such that each row in U can be represented as a convex combination of rows of I^* and the zero vector. Hence, for an $A = \{a \in [0,1]^{d\times 1} : \|a\|_1 \leq 1\}$ each row of U can be expressed as,

$$\forall i \in [K]\ \exists \alpha \in A : U(I^*,:)\alpha = U(i,:)\,, \tag{1}$$

Similarly, 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,

$$\forall j \in [L]\ \exists \alpha \in A : V(J^*,:)\alpha = V(j,:)\,, \tag{2}$$

where $A = \{a \in [0,1]^{d\times 1} : \|a\|_1 \leq 1\}$. Note that we refer to the rows as users and to the columns as items because this is a standard terminology in recommender systems, where we envision applications of our work. Hence, 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 $i \in [K]$. The matrix V are latent preferences of L items in the space of d topics, where $V(j,:)$ are the coordinates of item $j \in [L]$. Without loss of generality, we assume that $U \in [0,1]^{K\times d}$ and $V \in [0,1]^{L\times d}$. We assume that the coordinates are points in a simplex, that is $\|U(i,:)\|_1 \leq 1$ for all $i \in [K]$ and $\|V(j,:)\|_1 \leq 1$ 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 the online learning problem of finding the maximum entry of a non-stochastic, low-rank and non-negative matrix which we call as a *non-stochastic low-rank bandit problem*. At time t, the preferences of users over items are encoded in a $K \times L$ *preference matrix* $M_t = U_t V_t^\intercal$, where M, U_t, and V_t are defined as in Section 2. We assume that user and item preferences (U_t and V_t respectively) can change with time t. At every timestep t the learner chooses d-pairs of rows and columns from M_t denoted by $(I_t, J_t) \in \Pi_d([K]) \times \Pi_d([L])$. It then observes all the values from the matrix $M_t(I_t, J_t)$ for all $i_t \in I_t$ and $j_t \in J_t$. The *reward* for the agent for choosing arms (I_t, J_t) at time t is denoted by $r_t(i^*(I_t,J_t), j^*(I_t,J_t))$ such that,

$$(i^*(I,J), j^*(I,J)) = \underset{(i,j)\in(I\times J)}{\arg\max}\ M_t(i,j) \tag{3}$$

B: Say how the hott topics assumption simplifies the problem of finding the maximum entry of a matrix. Write it formally.

Since U_t can change arbitrarily over time

B: This obviously cannot be arbitrary. The assumption is that all M_t have the same hott topics rows and columns. Write it formally.

, the reward in (3) is maximized by lists J with highly rewarding items that are diverse, in the sense that they attain high rewards at different times $t \in [n]$. A remarkable property of our user-item preference matrices M_t is that for any user $i \in [K]$ and any item $j \in [L]$ at any time t,

$$\underset{(i,j)\in([K]\times[L])}{\arg\max} \quad M_t(i,j) \in (I^*, J^*),$$

where I^* and J^* is defined in (1) and (2). Therefore, it is possible to learn all potentially most rewarding pairs of rows and columns statistically efficiently.

B: The definition of the regret below is incorrect for $d > 1$. We need to think about this. Actually, do we need the definition of the regret for $d > 1$ if we never bound it? We should use determinants and then intuitively explain what they mean. The rank 1 case is intuitive and easy to explain.

Now we are ready to define our notion of optimality and regret. Our goal is to minimize the expected n-step regret,

$$R(n) = \sum_{t=1}^{n} \mathbb{E}\left[r_t(i_t^*, j_t^*) - r_t(i^*(I,J), j^*(I,J))\right], \quad (4)$$

where the expectation is with respect to both randomly choosing rows (I_t) and columns (J_t) by the learning algorithm and potential randomness in the environment.

4 Rank 1 Problem

We first analyze the simple rank 1 scenario and propose our solution for this setting. Many of the key aspects of our design principle are captured in this rank-1 setting. Note that for $d = 1$, $U_t \in [0,1]^{K\times 1}$ is the user preference over a single topic and $V_t \in [0,1]^{L\times 1}$ is the item preference over a single topic. The user-item preference matrix $M_t = U_t V_t^\intercal$ is non-negative, non-stochastic, and rank 1. Also, the hott-topics assumption in eq (1) and (2) still holds in the rank 1 setting. In this setting, at every timestep $t \in [n]$ the learner selects a pair of rows and columns, denoted by i_t and j_t respectively and observes the feedback $M_t(i_t, j_t)$.

B: Explain why.

We present the simple algorithm Low Rank Bandit LRB in Algorithm 2 for the rank 1 setting. The LRB consist of two key components, a row learning algorithm and a column learning algorithm. At every round t the row algorithm suggest the row $i_t \in [K]$ and the column algorithm suggests the column $j_t \in [L]$. Note, that in this non-stochastic scenario we use Exp3 as the row and column learning algorithm. While the row Exp3 consist of $[K]$ arms, the column Exp3 consist of $[L]$ arms. The main idea is to use the row Exp3 to learn the best row on average while the column Exp3 will learn the best column on average. The learner then observes the reward $M_t(i_t, j_t)$ and updates the row and column Exp3 simultaneously. A key insight to this simple design is that when both the row and column learner are run simultaneously, they will learn the most rewarding row and column on average and converge on the maximum entry of the the matrix M. From the definition of U_t, for any sequence of n columns, the maximum value is in row i^*. This means that the row algorithm learns no matter what the column algorithm does. From the definition of V_t, for any sequence of n rows, the maximum value is in column j^*. This means that the column algorithm learns no matter what the row algorithm does.

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

Input: Time horizon n ▷ Initialization
Initialize RowAlg
Initialize ColAlg
for $t = 1, \ldots, n$ **do** ▷ Generate response
  Row i_t suggested by RowAlg
  Column j_t suggested by ColAlg
  Observe $M_t(i_t, j_t)$ ▷ Update statistics
  Update arm i_t of RowAlg with reward $M_t(i_t, j_t)$.
  Update arm j_t of ColAlg with reward $M_t(i_t, j_t)$.

B: We need to clearly explain what is going on here. From the definition of U_t, for any sequence of n columns, the maximum value is in row i^*. This means that the row algorithm learns no matter what the column algorithm does. From the definition of V_t, for any sequence of n rows, the maximum value is in column j^*. This means that the column algorithm learns no matter what the row algorithm does.

5 Algorithm for Rank d Setting

Now, we propose the general *Low Rank Bandit (*LRB*)* algorithm for solving the family of non-stochastic, non-negative and low-rank matrices of rank d. Again, the goal is to identify the maximum entry of the matrix by quickly identifying the d-best rows or 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 row learning algorithm recommends a list of d rows and is the same as the Ranked Bandit Algorithm (RBA) in Radlinski *et al.* [2008]. But we exploit an additional structure in our problem to show that we learn the optimal rows I^*. The row learning algorithm are d instances of multi-armed bandit algorithms, which we denote by RowAlg(k) for algorithm $k \in [d]$. RowAlg(1) learns the most rewarding row on average, RowAlg(2) learns the second most rewarding row on average conditioned on the first learned column, and so on.

Similarly, column learning algorithm recommends a a list of d columns by exploiting the same structure in our rewards. Again the goal of the column learning algorithm ColAlg is to learn the optimal set of columns J^*. Hence, ColAlg$(1), \ldots,$ ColAlg(d) learns the most rewarding columns $\{j_1, j_2, \ldots, j_d\}$ on average. Note, that this sequence of learning the rows or columns first does not matter because the *hott-topics* structure is defined on both U and V matrix generating $M_t = U_t V_t^\intercal$, and so we will be learning the d-best rows or columns in average. Another way of looking at this is to first realize that if we fix the column selection strategy, which is simply some distribution over d-tuples of chosen columns then for any such distribution, the d hott-topic rows are the optimal solution to the row selection problem. By symmetry, the same is true for the column selection problem. If we run both in parallel, and the distributions in the other dimensions do not change too fast (this is true by our design), then $i_1, \ldots, i_d$ and $j_1, \ldots, j_d$ would slowly converge to the d hott-topic rows and columns.

Finally, LRB observes the individual rewards of $M(i,j)$ for all $(i,j) \in (I_t, J_t)$. Then we update both column and row

Algorithm 2 Low Rank Bandit (LRB)

Input: Time horizon n, Rank d
for $k = 1, \ldots, d$ **do** ▷ Initialization
  Initialize RowAlg(k)
  Initialize ColAlg(k)
for $t = 1, \ldots, n$ **do**
  for $k = 1, \ldots, d$ **do** ▷ Generate response
    $\hat{i}_k \leftarrow$ Suggested row i_t by RowAlg(k)
    if $\hat{i}_k \in \{i_1, \ldots, i_{k-1}\}$ **then**
      $i_k \leftarrow$ Random row not in $\{i_1, \ldots, i_{k-1}\}$
    else
      $i_k \leftarrow \hat{i}_k$
    $\hat{j}_k \leftarrow$ Suggested column j_t by ColAlg(k)
    if $\hat{j}_k \in \{j_1, \ldots, j_{k-1}\}$ **then**
      $j_k \leftarrow$ Random column not in $\{j_1, \ldots, j_{k-1}\}$
    else
      $j_k \leftarrow \hat{j}_k$
  $I_t \leftarrow (i_1, \ldots, i_d)$
  $J_t \leftarrow (j_1, \ldots, j_d)$
  Observe $M_t(I_t(k), J_t(k))$ for all $k \in [d]$
  for $k_1 = 1, \ldots, d$ **do**
    for $k_2 = 1, \ldots, d$ **do** ▷ Update statistics
      if $i_k = \hat{i}_k$ **then**
        Update arm i_k of RowAlg(k) with reward

$$\max_{k \le k_1} M_t(i_k, j_{k_2}) - \max_{k < k_1} M_t(i_k, j_{k_2})$$

      else
        Update $\hat{i}_k$ of RowAlg(k) with reward 0
      if $j_k = \hat{j}_k$ **then**
        Update arm j_k of ColAlg(k) with reward

$$\max_{k \le k_2} M_t(i_{k_1}, j_k) - \max_{k < k_2} M_t(i_{k_1}, j_k)$$

      else
        Update $\hat{j}_k$ of ColAlg(k) with reward 0

learning algorithms. The reward of the arm in RowAlg(k), which selects the k-th row in I_t, is updated as follows. If the k-th arm was not previously suggested row its reward is updated d times such that, $\max_{k \le k_1} M_t(i_k, j_{k_2}) - \max_{k < k_1} M_t(i_k, j_{k_2})$ for all $k_1, k_2 \in [d]$. By the choice of our design and previous argument a similar update is performed on the k-th column learning algorithm ColAlg such that its reward is also updated d times such that, $\max_{k \le k_2} M_t(i_{k_1}, j_k) - \max_{k < k_2} M_t(i_{k_1}, j_k)$ for all $k_1, k_2 \in [d]$. Otherwise, if any of the row or column has been previously selected by the corresponding RowAlg or ColAlg algorithm then we update it with reward 0.

5.1 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 6. 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. The proposed LRB algorithm only has to update/look through $(K + L)d$ items for each of the d ColAlg and the RowAlg at every timestep t. This is in stark contrast to some of the existing matrix completion algorithms which has to reconstruct a $K \times L$ matrix [Sen *et al.*, 2016] or calculate second or third order tensors [Gopalan *et al.*, 2016].

6 Analysis of Rank 1 Setting

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

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

where $\Delta = \min_{t \in [n]} \min_{i_t, j_t:\, , i_t \neq i_t^* \; j_t \neq j_t^*} \mathbb{E}\left[u_t(i_t^*) v_t(j_t^*)\right] - \mathbb{E}\left[u_t(i_t) v_t(j_t)\right]$

B: I do not think that this definition of Δ is correct. If you look into the proof, you will note that Δ there has nothing to do with the gap. We should not call this quantity Δ because this will confuse reviewers.

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,

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

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 $t = 1, 2, \ldots, n$ are bounded from below by some $\Delta > 0$. Then we get that,

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

B: The argument below is just bla bla bla and needs to be written properly. In particular, it needs to be clear what we condition on and what we take the expectation over.

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,

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

□

Discussion: The main idea is to decompose the regret of LRB into two parts, where ColAlg does not suggest j_t^* and the RowAlg does not suggest i_t^*. 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 expected regret 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 for any sequence of rows. The second part, which is $O(\sqrt{Kn}/\Delta)$, is the regret due to learning the optimal row, for any sequence of columns. Our regret bound also improves upon a trivial approach where the optimal columns are learned separately for each user. In that case, 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.

7 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. All results are averaged over 10 independent random runs. We test in both rank 1 and rank 2 settings to clearly illustrate the failures of the current Rank-1 algorithms and show the efficiency of our proposed method. We use the term rows/users and columns/items interchangeably.

7.1 Evaluated Algorithms

Rank 1 **Algorithms:** We compare against several state-of-the-art rank 1 algorithms. Note, that all the rank 1 algorithms suggest a single row and column at every round. The UCB1 algorithm from Auer *et al.* [2002a] builds a confidence set at every round t over all the entries of M_t as $c_{i,j}(t) = \sqrt{\frac{2 \log t}{N_{i,j}(t)}}$ where $N_{i,j}(t)$ denotes the number of times the $M(i,j)$-th entry has been observed. It the suggests the best row-column pair based on the term $\hat{M}_t(i,j) + c_{i,j}(t)$ where $\hat{M}_t(i,j)$ denotes the empirical mean of all the observed rewards for $M(i,j)$. The UCB1 $-$ Elim [Auer and Ortner, 2010] is similar to UCB1 but it eliminates sub-optimal rows and columns based on a similar confidence set $c_{i,j}(t)$ till it finally converges on the best pair of row and column. The algorithm LinUCB was first proposed in Li *et al.* [2010] for the contextual bandit setting. Note, that for a set of features $\theta \in 0, 1^{K+L}$, rank-1 bandit generalizes to the stochastic linear bandit setting and can be solved by LinUCB. Similarly, GLM-UCB from Filippi *et al.* [2010] which computes the maximum-likelihood estimates of the parameter vector $\theta \in 0, 1^{K+L}$ (using Expectation-Maximization algorithm) can also be used solve the rank 1 bandit problem. Finally, we compare against the algorithm Rank1$-$Elim from Katariya *et al.* [2016] which is an improved version of UCB1 $-$ Elim and employs row and column elimination and aggressive exploration to converge on the best row and column pair. For LRB we use the Algorithm 1 from Section 4.

Rank 2 **Algorithms:** We similarly design the Rank 2 algorithms by modifying the rank 1 algorithms. Again, note that all the rank 2 algorithms suggest two pairs of rows and columns at every round t. For all of the algorithms UCB1, UCB1 $-$ Elim, LinUCB, GLM $-$ UCB, and Rank1 $-$ Elim we modify these algorithms so that they suggest 2 pairs of rows and columns based on their respective confidence interval set $c_{i,j}(t)$. The row and column pair with the highest and the second highest $\hat{M}_t(i,j) + c_{i,j}(t)$ are suggested for each round t and consequently after observing all the entries of $M_t(i,j)$ all of the algorithms update their estimates of $\hat{M}_t(i,j)$ for each $i, j \in [d]$. For LRB we use the Algorithm 2 from Section 5. Note, that there are two RowAlg and ColAlg, each running an Exp3 algorithm with the exploration parameters as discussed before. For LRB we use the Algorithm 2 from Section 5.

7.2 Synthetic Experiment 1

This experiment is conducted to test the performance of LRB over a small number of users and items and to show how

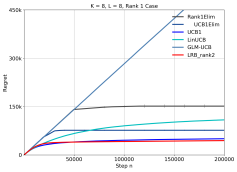
LRB scales with increasing number of users and items. Note, that in this experiment all our modeling assumptions hold. This simulated testbed consist of two scenarios: (1) 8 users and 8 items and (2) 16 users and 16 items. In this setting, $U = \{0.7, 0.9\}^{K\times 1}$ and similarly $V = \{0.7, 0.9\}^{L\times 1}$ with the entry $U_t(K/2, 1) = V_t(L/2, 1) = 0.9$. Hence, the matrix $M = UV^\intercal$ is rank 1 and the hott-topics structure is maintained. At every round t, $u_t(i)$ is an independent Bernoulli variable with mean $U(i, 1)$ and $v_t(j)$ is an independent Bernoulli variable with mean $v_t(j) = V(j, 1)$. The learner observes the entry $u_t(i)v_t(j)$ when it selects the i-th user and j-th item. A similar environment has been discussed as B_{spike} in [Katariya *et al.*, 2016]. From Figure 1(b) we can clearly see that LRB outperforms all the other algorithms. The regret curve of LRB flattens, indicating that it has learned the best user-item pair. As we scale the number of users and items we see that LRB performs even better than other algorithms. The key realization is that LRB takes advantage of the hott-topics structure and quickly identifies them. Note, that for any rank d scenario the best user-item pair must be one of the hott-topics in (I^*, J^*).

7.3 Synthetic Experiment 2

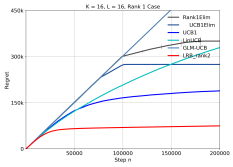
This experiment is conducted to test the performance of LRB over a large number of users and items. This simulated testbed consist of 64 users, 64 items, and rank$(M) = 2$. The vectors spanning U and V, generating the user-item preference matrix M, are shown Figure 1(c). The users and items are evenly 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 $V(1, :) = (1, 0)$ and $V(2, :) = (0, 0.6)$ while 50% remaining items has feature $V(j', :) = (0.45, 0.5)$ and the rest have $V(j, :) = (0.5, 0.45)$. Similarly, we create the user feature matrix U having a $50 : 50$ split such that $U(1, :) = (1, 0)$, $U(2, :) = (0, 0.6)$ and the remaining 50% users having $U(i, :) = (0.5, 0.4)$ and the rest having $U(i', :) = (0.4, 0.5)$. 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 LRB outperforms all the other algorithms. It's regret curve flattens, indicating that it has learned the best user-item pair.

7.4 Real World Experiment 3

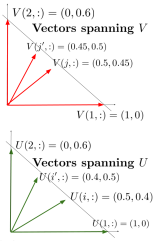
We conduct the third experiment to test the performance of LRBwhen our modeling assumptions are violated. We use the Jester dataset [Goldberg *et al.*, 2001] which consist of over 4.1 million continuous ratings of 100 jokes from 73,421 users collected over 5 years. In this dataset there are many users 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 10 users (who have rated all jokes) from this dataset and use singular value decomposition (SVD) to obtain a rank 2 approximation of this user-joke rating matrix M. In the resultant matrix M, most of the users belong to the two classes preferring jokes 98, and 28, while



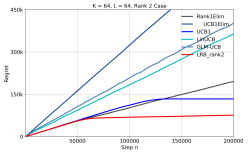
(a) Expt-1: 8 Users, 8 items, Rank 1, Algorithm Performance



(b) Expt-1: 16 Users, 16 items, Rank 1, Algorithm Performance



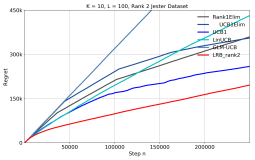
(c) Expt-2: 64 Users, 64 items, Rank 2, User and Item vectors



(d) Expt-2: 64 Users, 64 items, Rank 2, Algorithm Performance

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

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. 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(a) again we see that LRB outperform other algorithms.



(a) Expt-1: 10 Users, 100 items, Rank 2

8 Conclusions

In this paper, we studied the problem of finding the highest entry of a non-stochastic, non-negative low-rank matrix. We formulated the above problem as an online-learning problem and proposed the LRB algorithm for this setting. We proved that an instance of algorithm has a regret bound in the special case of rank-1 setting that scales as $O\left(\frac{(\sqrt{L}+\sqrt{K})\sqrt{n}}{\Delta}\right)$ 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 we only proved our theoretical results for the rank 1 setting. Proving theoretical guarantees for LRB algorithm will require additional assumptions on the structure of rewards and the matrix M. Another interesting direction is to look at structures beyond hott-topics assumption on user and item matrix.

References

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.

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

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.

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

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 661–670, 2010.

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

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.