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 each round, the learner chooses pairs of rows and columns, and observes the product of their values

B: "values" should be "latent vectors". The same issue in the next sentence.

. The main challenge is that the learner does not observe the individual latent values of rows and columns as its feedback. Diverging from previous works, we assume that the preference matrix

B: It is not clear what the preference matrix is.

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

B: Check for typos. "interval" should be "intervals".

with the strong assumption that underlying distributions are stochastic and i.i.d. We depart from this design and learn the best row and column pair jointly with two separate bandit algorithms, 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

B: Write "sublinear" instead of "sub-linear". No need for the hyphen.

bound on its n-step regret in rank 1. We evaluate the algorithm empirically on several synthetic and real-world problems. 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. The low-rank structure is observed in many real-world applications and is a standard assumption in recommender systems [**??**]. 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.

B: Give a plain English example of what we mean by this. Potentially cite.

Hence, a successful conversion happens if each population segment is matched to the correct marketing channel which is nothing but the maximum entry of the matrix formed by the outer product of the users preference and marketing channels over some number of common topics.

A: This is misleading. We don't match each population segment with a marketing channel, but just identify the pair with maximum reward.

We formalize our learning problem as the following online learning problem. At time t, the learning agent chooses d-pairs

B: A pair means 2. If you have more than 2, it is a tuple.

of rows and columns where d is the rank of a non-negative and low-rank matrix M. We use the terminology row/user and column/item interchangeably, keeping in sync with our proposed application area. This user-item preference

B: Again, the term preference appears out of nowhere.

matrix M is formed by the outer product of user and item latent preferences over d topics. Note that the learner does not observe the individual latent values

B: latent vectors

of user or item preferences but just their product. The user-item preference matrix M is low-rank at each round 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 quickly as possible.

B: The two paragraphs below should be moved to "Related Work". They contain too many details that are not necessary to understand our design and contributions. Put "Related Work" right before "Conclusions". Also, the current comparison to prior work is a laundry list and completely inefficient. Better structure how we differ. Paragraph 1: Some people do only rank 1. We do rank d. Paragraph 2: Most papers do stochastic. We do adversarial (with restrictions). Paragraph 3: Our main selling point is a simple algorithm that can be easily generalized beyond rank 1. Focus on this difference.

Previous works that have studied this setting have either proposed highly conservative algorithms or restricted themselves to a stricter set of assumptions. While **?** was proposed for a rank 1 bandit model with the assumption that the underlying distributions are stochastic, **?** was proposed for the special case when the underlying distributions are Bernoulli. Both these works used different variations of the phase-based UCB-Improved [**?**] algorithm to construct a confidence inter-

val set over row-column pairs 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, **?** can be viewed as a generalization of rank-1 bandits of **?** 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 **?** or tensor based methods [**?**] 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 **?**. A more simpler setting has also been studied in **?**.

Our approach is based on two key insights.

A: I wouldn't call these key insights. These are points for how we are different from earlier papers.

First, the earlier methods (like Upper Confidence Bound (UCB) algorithms, NMF-Bandits [**?**]) are explicitly modeled on the stochastic 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 do not require any sort of costly matrix inversion or reconstruction operations or even row or column eliminations and hence are faster to implementation.

A: Do you want to say the algorithms run faster or that they easier to implement?

We make four major contributions.

A: I don't think we should call these as four major contributions. For example, if we propose an algorithm in a paper, it better work. We don't say we make two major contributions in that case, proposing an algorithm and showing it is true.

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.

B: You keep calling our algorithm both "Low Rank Bandit" and LRB in the same sentence. This is super confusing. The point of the abbreviation is that you do not need to repeat the full name. Stick to LRB and explain here that this stands for low rank bandit. Note that I did not use CAPS.

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

A: 'variant'

of the Exp3 [**?**] 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. In fact, we use the well known fact that exponentially weighted algorithm like Exp3 are robust and fast learners to design our algorithm. The Third, we analyze LRB and up to problem-specific factors, we prove a $O\left(\frac{(\sqrt{L}+\sqrt{K})\sqrt{n}}{\alpha}\right)$ upper bound on its n-step regret in the special case when rank is 1 and for some $\alpha > 0$. The regret of a naive solution is $O(\sqrt{KLn})$, and is much worse than that of LRB when $K \approx L$. 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. In Section 2, we introduce the rank-1 setting. In Sections 3 and 4, we propose our rank-1 algorithm and derive a sublinear upper bound on its regret. In Sections 5 and 6, we introduce the rank-d setting and propose an algorithm for it. In Section 7, we evaluate our algorithms empirically. We conclude in Section 8.

2 Rank-1 Setting

Notations: We denote $[n] = \{1, \ldots, n\}$ as the set of the first n positive integers. Let M denote any arbitrary set of size $m \times n$ matrix.

B: The previous sentence does not make any sense. What is the set of size $m \times n$?

Let the rank of the matrix be denoted by d. We denote by A^B the set of all vectors whose entries take values from set A and are indexed by set B, where A and B can be any arbitrary sets. We index the rows and columns

A: We don't index the rows and columns by vectors, but define what it means to use vector indexing. You can just remove this sentence.

of the 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 $M(:, J(j))$. Finally, we denote Π_d as the set of all d-permutations. For an element $\pi \in \Pi_d$ and d-dimensional vector v, we denote by $\pi(v)$ the permutation of the entries of v according to π.

Rank-1 Setting: We study the online learning problem of finding the maximum entry of a non-stochastic, low-rank and non-negative matrix

B: You make it sound like there is only one matrix. This is false.

which we call as a *non-stochastic low-rank bandit 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. Let $U_t \in [0, 1]^{K \times 1}$ be the user preference over a single topic and $V_t \in [0, 1]^{L \times 1}$ be the item preference over a single topic. 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. The user-item *preference matrix* $M_t = U_t V_t^\intercal$ is non-negative, non-stochastic, and rank 1. We assume that user and item preferences (U_t and V_t respectively) can change with time t.

B: This is obviously false. They can change arbitrarily, as long the best row and column do not change. Explain what does this mean from the modeling point of view. What kind of noise can this model?

At every timestep

B: round

t, the learner chooses one pair of row and column indexed by i_t and j_t respectively and observes their product $U_t(i_t)V_t(i_t)$ as its feedback.

Regret Definition (Rank-1): The goal of the learner is to minimize the expected n-step regret with respect to the optimal solution in the hindsight as follows,

$$R(n) = \sum_{t=1}^{n} \mathbb{E}\left[r_t(i_t^*, j_t^*) - r_t(i_t, j_t)\right], \tag{1}$$

where, the expectation is over any random choice of rows and columns and $i_t^* = \arg\max_{\{i\in[K], t\in[n]\}} U_t(i,1)$ and $j_t^* = \arg\max_{\{j\in[L], t\in[n]\}} V_t(j,1)$.

B: Please do not use non-standard notation. {} is a set!!! You want to write either $\arg\max_{i\in[K],\, t\in[n]} U_t(i,1)$ or $\arg\max_{(i,t)\in[K]\times[n]} U_t(i,1)$. You also need to move this earlier. You want to say that the best row and column does not change over time.

3 Rank-1 Algorithm

We present the simple

B: Avoid repetitive use of "simple". You make it sound like we do only easy things.

algorithm Low Rank Bandit LRB

B: Confusing. A name followed by its abbreviation.

in Algorithm 1 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. The row Exp3 has K arms and the column Exp3 has L arms. The main idea is to use the row Exp3 to learn the best row on average while the column Exp3 learns 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^*.

B: This property of U_t and V_t needs to stated formally. They cannot be arbitrary, as you claimed earlier.

This means that the row algorithm learns irrespective of what the column algorithm does. From the definition of V_t, for any sequence of n rows, the maximum value is in column j^*.

B: Again, this property of U_t and V_t needs to stated formally. They cannot be arbitrary, as you claimed earlier.

This means that the column algorithm learns irrespective of what the row algorithm does.

4 Rank-1Analysis

B: Add a sentence or two that say what this section is about.

B: I did not double check the analysis. Let me know when I can.

Theorem 1. ***(Rank-1 Case)***

B: No need to rub it in that this is rank 1. This is the only theorem in the paper, right?

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 $M_t(i_t, j_t)$.

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

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

for any $\alpha > 0$ such that $U_t(i,1) > \alpha$ and $V_t(j,1) > \alpha$

A: Both should be $\geq$

for all $i \in [K]$, $j \in [L]$, and $t \in [n]$.

Proof. Let, $(u_t v_t^\intercal)_{t=1}^n$

B: All u and v should be capitalized, no? At least consistent with the notation in the setting.

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,

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

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

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

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

we get that,

$$R(n) = \sum_{t=1}^{n} \mathbb{E}[(1 + v_t(1)/v_t(j_t))v_t(j_t)(u_t(1) - u_t(i_t))] + \sum_{t=1}^{n} \mathbb{E}[(1 + u_t(1)/u_t(i_t))u_t(i_t)(v_t(1) - v_t(j_t))]$$
$$\leq (1 + \frac{1}{\alpha})\Big[\sum_{t=1}^{n} \mathbb{E}[u_t(1)v_t(j_t) - u_t(i_t)v_t(j_t)] + \sum_{t=1}^{n} \mathbb{E}[u_t(i_t)v_t(1) - u_t(i_t)v_t(j_t)]\Big]$$

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 **?** 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,

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

□

Discussion: The main idea in Theorem 1 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^*.

A: The optimal row,columns should be i^*, j^*, shouldn't have a time subscript.

The first part is analyzed as follows. ColAlg has a sublinear regret, based on a similar analysis to **?**.

B: Why do you cite this paper? Our analysis has nothing to do with what Auer did in 2002.

We use non-stochastic algorithm Exp3 for ColAlg and RowAlg because our environment is non-stationary. Thus, the upper bound on the expected regret that ColAlg suggests sub-optimal columns for *any sequence* of rows is bounded. This regret scales as $O(\sqrt{Ln})$ and decreases

B: Decreases with n? $\sqrt{n}$ does not decrease with n.

with time horizon n. Similarly, we analyze the regret for the RowAlg as it also uses the exponentially weighted algorithm Exp3. The RowAlg suffers a regret of $O(\sqrt{Kn})$ for suggesting sub-optimal rows for *any sequence* of columns . Hence, the regret in Theorem 1 consists of two main parts.

B: We keep repeating ourselves. You already said that the regret decomposes in two parts and that each part can be bounded independently.

The first part, which is $O(\sqrt{Ln})$, is the regret due to learning the optimal column and the second part, which is $O(\sqrt{Kn})$, is the regret due to learning the optimal row. Finally, we can consider any matrix M of dimension $K \times L$ as a K-bandit problem with each bandit having L arms and the optimal columns are learned separately for each bandit. Such a trivial setting gives rise to the regret bound of order $O(\sqrt{KLn})$. Our regret bound also improves upon this trivial approach.

5 Rank-d Setting

In this section, we study the online learning problem of finding the maximum entry of a non-stochastic, low-rank and non-negative matrix for the general rank-d setting.

B: You make it sound like there is only one matrix. This is false.

Hott-topics Assumption: We focus on a family of low-rank matrices, which are known as hott topics.

B: Cite. Why do these matrices matter?

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 $A = \{a \in [0,1]^{d\times 1} : \|a\|_1 \leq 1\}$ each row of U can be expressed as,

$$\forall i \in [K]\ \exists \alpha \in A : U(I^*, :)\alpha = U(i, :)\,, \tag{2}$$

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,

$$\forall j \in [L]\ \exists \alpha \in A : V(J^*, :)\alpha = V(j, :)\,, \tag{3}$$

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

B: This is a strange transition. You start talking about users, items, and topics like this is synonymous to hott topics. This is false. This is just an illustration of what hott topics can mean in practice, right?

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 $j \in [L]$.

Rank-d Setting: Again, note that 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 U_t and V_t are defined as in (2) and (3).

B: You need to watch what you write. U and V are defined in (2) and (3). Not U_t and V_t. Now you need to say what the properties of U_t and V_t are. You need to state that the best rows and columns in M_t remain the same.

We assume that user and item preferences (U_t and V_t respectively) can change with time t. At every round 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,

$$(i^*(I, J), j^*(I, J)) = \underset{(i,j)\in(I\times J)}{\arg\max}\ M_t(i, j) \tag{4}$$

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,

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

where I^* and J^* is defined in (2) and (3). Hence, the hott-topics assumption makes it possible to learn the maximum entry of M_t statistically efficiently as at any time $t \in [n]$ the maximum entry $M_t(i_t^*, j_t^*)$ will be in $M_t(I^*, J^*)$. Note, that even though different entries of U_t and V_t can attain high rewards at different times but the hott-topics assumption makes sure that I^* and J^* remain fixed for all time $t \in [n]$.

A: It is not hott topics structure that ensures this. This is an extra assumption we make.

Regret Definition (Rank-d): Now we are ready to define our notion of optimality and regret for the general rank-d scenario. Our goal is to minimize the expected n-step regret,

$$R(n) = \sum_{t=1}^{n} \mathbb{E}\left[r_t(i_t^*, j_t^*) - r_t(i^*(I, J), j^*(I, J))\right], \tag{5}$$

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.

6 Rank-d Algorithm

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.

B: There is no single matrix.

The pseudocode of `LRB` is in Algorithm 2. `LRB` has two main components, column learning and row learning algorithm.

At every timestep

B: round

t, the row learning algorithm recommends a list of d rows and is the same as the Ranked Bandit Algorithm (RBA) in **?**.

B: Leave talking about related work for "Related Work". It is distracting if you do it here.

But we exploit an additional structure in our problem to show

A: We don't 'show' anything for rank-d case.

that we learn the optimal rows I^*. The row learning algorithm are d instances of multi-armed bandit algorithms, which we denote by $\texttt{RowAlg}(k)$ for algorithm $k \in [d]$. $\texttt{RowAlg}(1)$ learns the most rewarding row on average, $\texttt{RowAlg}(2)$ learns the second most rewarding row on average conditioned on the first learned column, and so on.

Similarly, column learning algorithm recommends a 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, $\texttt{ColAlg}(1), \ldots, \texttt{ColAlg}(d)$ learns the most rewarding columns $j_1, \ldots, j_d$

B: This is not the notation for the best columns.

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.

B: This needs to be stated formally. What does it mean mathematically?

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 learning algorithms. The reward of the arm in $\texttt{RowAlg}(k)$, 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.

B: Explain why this is needed.

6.1 Practical Considerations

We leave the implementation of the `ColAlg` and `RowAlg` to the users. For theoretical guarantees

B: What guarantees? We have none. Say that this is motivated by rank 1.

we use non-stochastic algorithm `Exp3` as `ColAlg` and `RowAlg` and showed how the regret scales for the rank-1 setting. 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 **?** where `RBA` uses `UCB1` for ranking items. The proposed `LRB` algorithm only has to update/look

B: I am pretty sure that you can use just one of these words. Why would you say update/look?

through $(K+L)d$ entries for the d `ColAlg` and the `RowAlg` respectively at every round t. This is in stark contrast to some of the existing matrix completion algorithms which has to reconstruct a $K \times L$ matrix [**?**] or calculate second or third order tensors [**?**].

7 Experiments

In this section, we compare `LRB` to several bandit algorithms in three experiments. The first two experiments are on synthetic problems where all modeling assumptions hold. The

Algorithm 2 Low Rank Bandit (LRB) (Rank-d)

Input: Time horizon n, Rank d
for $k = 1, \ldots, d$ **do** ▷ Initialization
  Initialize $\texttt{RowAlg}(k)$
  Initialize $\texttt{ColAlg}(k)$
for $t = 1, \ldots, n$ **do**
  for $k = 1, \ldots, d$ **do** ▷ Generate response
    $\hat{i}_k \leftarrow$ Suggested row i_t by $\texttt{RowAlg}(k)$
    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 $\texttt{ColAlg}(k)$
    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 $\texttt{RowAlg}(k)$ 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 $\texttt{RowAlg}(k)$ with reward 0
      if $j_k = \hat{j}_k$ **then**
        Update arm j_k of $\texttt{ColAlg}(k)$ 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 $\texttt{ColAlg}(k)$ with reward 0

third experiment is on a real-world 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.

B: You already said this before multiple times. What are users and items really buying us? We should call rows "rows" and columns "columns". In our motivating example, the columns are marketing channels. Item does not sound like the right analogue.

7.1 Evaluated Algorithms

Rank 1 **Algorithms:** We compare against several state-of-the-art rank 1 algorithms. Note, that

A: I don't think you need a ',' between 'Note' and 'that'. This repeats in at least a couple of places.

all the rank 1 algorithms suggest a single row and column at every round. The UCB1 algorithm from **?** 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 $\texttt{UCB1} - \texttt{Elim}$

B: Weird hyphen. Use mhyphen instead.

[**?**] 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 **?** 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.

B: I do not think that this is true.

Similarly, GLM-UCB from **?** 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 $\texttt{Rank1} - \texttt{Elim}$ from **?** which is an improved version of $\texttt{UCB1} - \texttt{Elim}$ 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 3. We use eq (1) definition to calculate regret in rank-1 setting.

B: Restructure this section as follows. Baselines for rank 1. Rank 1 results. Algorithms for rank 2. Rank 2 results. It is too early to talk about rank 2 algorithms before you should rank 1 results.

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, $\texttt{UCB1} - \texttt{Elim}$, LinUCB, $\texttt{GLM} - \texttt{UCB}$, and $\texttt{Rank1} - \texttt{Elim}$ 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 6. Note, that there are two RowAlg and ColAlg, each running an Exp3 algorithm with the exploration parameters as discussed before. For LRB-rank2 we use the Algorithm 2 from Section 6. We also modify the rank-1 LRB (Algorithm 1) so that the algorithm works in rank-2 setting. After LRB-rank1 has sampled one pair of rows and columns from $[K]$ and $[L]$, it then samples again another choice that does not clash with the first pair and then updates all the pairs with the feedback observed. We use eq (5) definition to calculate regret in rank-2 setting.

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

only the entry $U(K/2,1) = V(L/2,1) = 0.9$. 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 $V(j,1)$. 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 **?**. From Figure 1(a) and 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.

A: Why do our assumptions hold in this experiment? You sample a Bernoulli outcome, so the optimal row/column can change across time steps.

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

A: This sentence doesn't make sense and not formal. Please delete it.

find the easily identifiable hott-topics perform very well. From Figure 1(b) we can clearly see that LRB-rank2 outperforms all the other algorithms. It's regret curve flattens, indicating that it has learned the best user-item pair. The key realization is that LRB takes advantage of the hott-topics structure and quickly identifies them. Note, that for any rank d scenario

A: 'scenario'? Please use a different word.

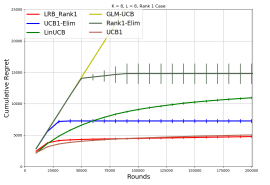
the best user-item pair must be one of the hott-topics in (I^*, J^*). Also note the failure of LRB-rank1 in this setting which clearly shows why a general rank-d algorithm with our specific type of update is required.

7.4 Real World Experiment 3

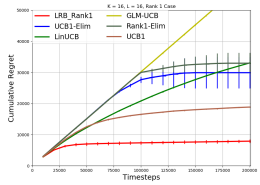
We conduct the third experiment to test the performance of LRBwhen our modeling assumptions are violated. We use the Jester dataset [**?**] which consist of over 4.1 million continuous ratings of 100 jokes from 73,421 users collected over 5 years. In this dataset there are many users who rated all jokes and we work with these users. Hence the user-item preference matrix is fully observed and we will not have to complete it using

A: You don't have to mention matrix completion here

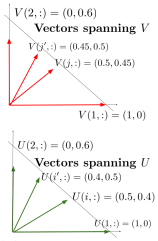
matrix completion techniques. Hence, this approach is very real world. We sample randomly 10 users (who have



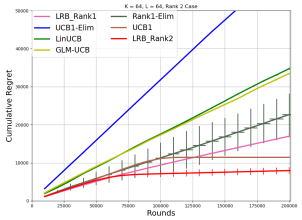
(a) Expt-1: 8 Users, 8 items, Rank 1 Setting



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



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



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

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

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 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. In Figure 2 we see that LRB-rank2 outperform other algorithms. Finally, LRB-rank1 again fails to perform well in this setting.

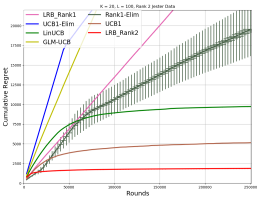


Figure 2: Expt-3: 10 Users, 100 items, Rank 2, Jester Dataset

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 $O\left(\frac{(\sqrt{L}+\sqrt{K})\sqrt{n}}{\alpha}\right)$ 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.