Latent Ranked Bandit

Author names withheld

Abstract

We study the problem of learning personalized ranked lists of diverse items for multiple users, from sequential observations of user preferences. The user-item preference matrix is non-negative and low-rank. Existing methods for solving similar problems are based on reconstructing the preference matrix from its noisy observations using matrix factorization techniques, and typically require strong assumptions on the reconstructed matrix. We depart from this standard approach and consider a family of low-rank matrices, where the set of most preferred items of all users is small and can be learned efficiently. Then we learn to present this set to each user in a personalized manner, in the order of the descending preferences of the user. We propose a computationally efficient algorithm that implements this procedure, which we call latent ranker (LRA), 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 personalized ranked lists of diverse items for multiple users, from sequential observations of user preferences. We are interested in utilizing latent similarities among users and items to learn these lists much faster than learning a separate ranked list for each user. The key structure in our problem is that the user-item preference matrix is low rank, which is a standard assumption in recommender systems (Koren, Bell, and Volinsky, 2009; Ricci, 2011). The learning agent has access to noisy observations of the user-item matrix. It does not have access to either user or item latent factors.

We formalize our learning problem as the following online learning problem. At time t, a random user i_t from a pool of K users arrives to the recommender system. The learning agent observes the identity of the user i_t, recommends a list of d diverse items J_t from a pool of L items as a response, and observes the preferences of user i_t for all recommended items J_t. The user-item preference matrix is low-rank at each time t, can vary substantially over time, and does not have to be stochastic. The reward of the recommended list is high when highly preferred items of the user are recommended at higher positions. The goal of our learning agent is to compete with the most rewarding diverse list for each user in hindsight.

Our learning model is motivated by a real-world scenario, where the learning agent suggests movies to users and each movie belongs to different movie genres. The agent typically does not observe instantaneous preferences of the user, and therefore suggests multiple movies that may be of interest to the user under different circumstances. A similar model has also been studied in Carbonell and Goldstein (1998) where the goal is to suggest a diversified list to each incoming user that combines relevance to the query as well as novelty. The authors suggest an approach where each item in the list is relevant to the query but also has *"marginal relevance"* or less similarity with previously selected documents and this improves the quality of recommendation.

We make four major contributions. First, we formulate our online learning problem as a latent ranked bandit on low-rank matrices. We identify a family of non-negative low-rank matrices where our problem can be solved statistically efficiently, without estimating the latent factors of the user-item preference matrix. The key structure of our matrix is that the set of optimal items of all users is small and can be learned jointly for all users. Given these items, the problem of learning the optimal order for each user can be solved in the full-information setting and thus is easy. Second, we propose a computationally-efficient algorithm that implements this idea, which we call latent ranker algorithm (`LRA`). The algorithm has two components, column learning and row ranking, which learn the set of optimal items of all users and then sort them, respectively. The column learning algorithm is similar to ranked bandits. In particular, we learn the k-th most diverse item using a multi-armed bandit, whose rewards are conditioned on the rewards of $k-1$ previously chosen items. The row learning problem is solved separately for each user. Because it is in the full-information setting, as we observe the individual rewards of all recommended items, we solve it using the weighted majority algorithm. Third, we analyze `LRA` and up to problem-specific factors, we prove a $O\left(d\sqrt{Ln} + K\log n + Kd\log d\right)$ upper bound on its n-step regret. The regret of a naive solution is $O(\sqrt{KLn})$, and is much worse than that of `LRA` when all of K, L, and n are large. Finally, we evaluate `LRA` empiri-

cally on several synthetic and real-world problems. Perhaps surprisingly, LRA 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 Appendix A.

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.

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 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,

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

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* $M_t = U_t V^\intercal$, 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

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

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, $1 \geq \mu(1) \geq \cdots \geq \mu(d) \geq 0$. 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,

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

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

4 Algorithm

We propose *latent ranker algorithm (*LRA*)* for solving the personalized ranking problem. The pseudocode of LRA is in Algorithm 1. LRA has two main components, column learning and row ranking.

The column learning algorithm recommends a list of d columns and is the same as in Radlinski, Kleinberg, and Joachims (2008). But we exploit an additional structure in our problem to show that we learn the optimal columns J_*. The column learning algorithm are d instances of multi-armed bandit algorithms, which we denote by $\texttt{ColAlg}(k)$ for algorithm $k \in [d]$. $\texttt{ColAlg}(1)$ learns the most rewarding column on average, $\texttt{ColAlg}(2)$ learns the second most rewarding column on average conditioned on the first learned column, and so on.

The row ranking algorithm permutes columns suggested by the column learning algorithm. It consists of multiple instances of full-information algorithms. More precisely, for each user $i \in [K]$ and set of d columns J, we have algorithm $\texttt{RowAlg}(i, J)$ with $d!$ arms, which correspond to all possible permutations of J. The objective of $\texttt{RowAlg}(i, J)$ is to learn a permutation of J with the highest reward, as measured by (2).

Algorithm 1 Latent Ranker Algorithm (LRA)

Input: Rank d, horizon n

for $k = 1, \ldots, d$ **do** ▷ Initialization
  Initialize $\texttt{ColAlg}(k)$
for all $i \in [K]$, $J \subset [L]$ such that $|J| = d$ **do**
  Initialize $\texttt{RowAlg}(i, J)$

for $t = 1, \ldots, n$ **do**
  User i_t is revealed
  for $k = 1, \ldots, d$ **do** ▷ Generate response
    $\hat{\ell}_k \leftarrow$ Suggested item by $\texttt{ColAlg}(k)$
    if $\hat{\ell}_k \in \{\ell_1, \ldots, \ell_{k-1}\}$ **then**
      $\ell_k \leftarrow$ Random item not in $\{\ell_1, \ldots, \ell_{k-1}\}$
    else
      $\ell_k \leftarrow \hat{\ell}_k$
  $J_t \leftarrow (\ell_1, \ldots, \ell_d)$
  $\pi_{t,i_t} \leftarrow$ Suggested permutation by $\texttt{RowAlg}(i_t, J_t)$
  Recommend $\pi_{t,i_t}(J_t)$
  Observe $M_t(i_t, J_t(k))$ for all $k \in [d]$
  for $k = 1, \ldots, d$ **do** ▷ Update statistics
    if $\ell_k = \hat{\ell}_k$ **then**
      Update arm ℓ_k of $\texttt{ColAlg}(k)$ with reward
      $$\max\{M_t(i_t, J_t(a)) : a \in [k]\} - \max\{M_t(i_t, J_t(a)) : a \in [k-1]\}$$
    else
      Update arm $\hat{\ell}_k$ of $\texttt{ColAlg}(k)$ with reward 0
  for all arms π in $\texttt{RowAlg}(i_t, J_t)$ **do**
    Update arm π with reward $r_t(i_t, \pi(J_t))$ in (2)

LRA interacts with the environment as follows. At time t, a random user i_t is revealed to LRA. Then, in the ascending order of $k \in [d]$, $\texttt{ColAlg}(k)$ suggests column ℓ_k. If $\texttt{ColAlg}(k)$ suggests one of the previously suggested columns $\ell_1, \ldots, \ell_{k-1}$, then ℓ_k is chosen uniformly at random from the remaining columns. We denote the vector of d suggested columns by J_t. Then $\texttt{RowAlg}(i_t, J_t)$, the row learning algorithm for user i_t and columns J_t, selects permutation π_{t,i_t} of J_t.

The user is recommended a permuted list $\pi_{t,i_t}(J_t)$ and LRA observes the individual rewards of all recommended items. Then we update both column and row learning algorithms. The reward of the arm in $\texttt{ColAlg}(k)$, which selects the k-th column in J_t, is updated as follows. If the arm was not one of the previously suggested columns, its reward is $\max\{M_t(i_t, J_t(a)) : a \in [k]\} - \max\{M_t(i_t, J_t(a)) : a \in [k-1]\}$. Otherwise, we update the initially suggested arm with reward 0. Since LRA observes the individual rewards of all recommended items, we can compute the reward of any permutation of J_t in row i_t. These rewards are then used to update $\texttt{RowAlg}(i_t, J_t)$.

4.1 Practical Considerations

The proposed LRA algorithm only has to update/look through $(Kd+d)$ items for each of the d ColAlg and the i-th 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, Maillard, and Zaki, 2016).

We do not have one row Exp3 for each user and any combination of columns, right?

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 WMA as 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 LRA. This has also been explored in Radlinski, Kleinberg, and Joachims (2008) where RBA uses UCB1 for ranking items.

5 Analysis

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

$$R(n) = O\left(\frac{d\sqrt{Ln}}{\Delta} + K \log n + Kd \log d\right),$$

where $\Delta = \min_{t \in [n]} \min_{J:\, J \neq J_*} \mathbb{E}\left[\tilde{r}_t(i, J_*)\right] - \mathbb{E}\left[\tilde{r}_t(i, J)\right]$ is an instance-specific lower bound on the gap in the unweighted expected rewards of the optimal and best suboptimal columns at any time $t \in [n]$, averaged over all users at that time.

Proof. The complete proof is in Appendix A. We sketch it below. The main idea is to decompose the regret of LRA into two parts, where ColAlg does not suggest J_* and the rest.

The first part is analyzed as follows. ColAlg has a sublinear regret, based on a similar analysis to Radlinski, Kleinberg, and Joachims (2008). Therefore, our upper bound on the probability that ColAlg suggests suboptimal columns, which is $O(d\sqrt{Ln}/\Delta)$, decreases with time horizon n.

The second part is analyzed as follows. Conditioned on $J_t = J_*$, the only remaining regret at time t is due to the fact that columns J_t are not ordered optimally for user i_t. Since this is a full-information problem where RowAlg is the weighted majority algorithm (Littlestone and Warmuth, 1994), the regret due to learning to order d columns for K users is K times that of the weighted majority algorithm with $d!$ arms. □

The regret in Theorem 1 consists of two main parts. The first part, which is $O(d\sqrt{Ln}/\Delta)$, is the regret due to learning the d optimal hott-topics columns with a high probability. The second part, which is $O(K \log n + Kd \log d)$, is due to learning the most rewarding permutations of the d optimal columns for all users.

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, Kleinberg, and Joachims, 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. In addition, the rewards in ColAlg(k) are non-stationary due to chosen columns at higher positions $1, \ldots, k-1$.

6 Experiments

In this section, we compare LRA 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 LRA 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, Kleinberg, and Joachims (2008). The two variants of RBA uses two types of column learning algorithms, UCB1 (Auer, Cesa-Bianchi, and Fischer, 2002) and Exp3 (Auer et al., 2002), abbreviated as RBA $-$ UCB1 and RBA $-$ 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 RBA $-$ 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 RBA $-$ UCB1 and RBA $-$ 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 (NMF $-$ Ban). The objective function of NMF $-$ Ban is:-

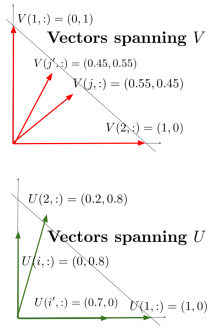
$$\text{minimize} \left\| \hat{M} - \hat{U}\hat{V}^\intercal \right\|_F^2 \text{ with respect to } \hat{U}, \hat{V}$$
$$\text{subject to constraints } \hat{U}, \hat{V} \geq 0$$

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 $\hat{M} \approx \hat{U}\hat{V}^\intercal$. This objective function is minimized by alternating minimization of $\hat{U}$ and $\hat{V}$ till the loss is very low. NMF $-$ 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.

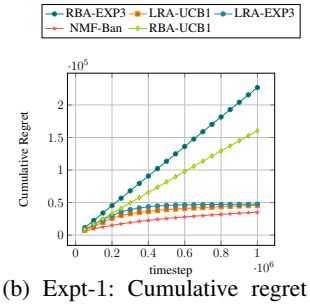
Personalized Ranking Algorithms: In this approach, we evaluate our proposed algorithm latent ranking bandit (LRA) by using three different types of column learning algorithms, Exp3, thompson sampling (Thompson, 1933), (Thompson, 1935), (Agrawal and Goyal, 2012) and UCB1. We term them as LRA $-$ EXP3, LRA $-$ TS and LRA $-$ 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 LRA $-$ 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. (2002). Similarly, for LRA $-$ 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

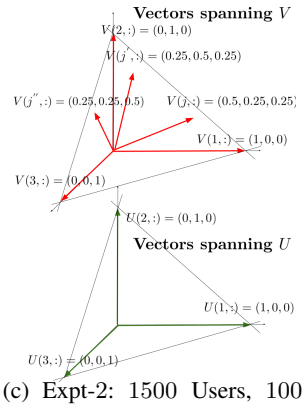
This experiment is conducted to test the performance of LRA over small number of users and items. This simulated testbed consist of 500 users, 50 items, and rank$(M) = 2$. The vectors spanning U and V, generating the user-item preference matrix M, are shown Figure 1(a). The users are evenly distributed 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 LRA $-$ EXP3, LRA $-$ TS and LRA $-$ 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 RBA $-$ UCB1 and RBA $-$ EXP3 perform poorly as the number of items per user is too large and the independent algorithms are not sharing information between them. NMF $-$ Ban performs better than the independent user model algorithms but is outperformed by LRA $-$ EXP3, LRA $-$ TS and LRA $-$ UCB1.



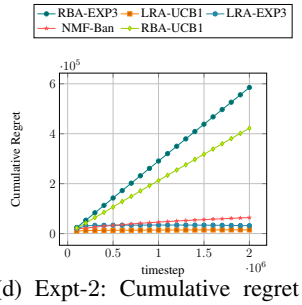
(a) Expt-1: 500 Users, 50 items, Rank 2, User and Item vectors



(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

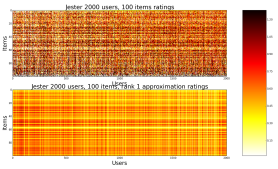
Figure 1: A comparison of the cumulative regret incurred by the various bandit algorithms.

6.3 Synthetic Experiment 2

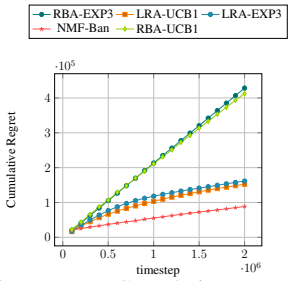
We conduct the second experiment on a larger simulated database of 1500 users, 100 items and rank$(M) = 3$. The vectors spanning U and V, generating the user-item preference matrix M is shown Figure 1(c). The users are divided into an unequal distribution of $60 : 30 : 10$ split such that 60% of the 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 LRA $-$ EXP3, LRA $-$ TS and LRA $-$ 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 NMF $-$ Ban again fails to get a reasonable approximation of M and performs poorly. Also, we see that both the independent user model algorithms RBA $-$ UCB1 and RBA $-$ 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, Kleinberg, and Joachims (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 LRA $-$ EXP3, LRA $-$ TS and LRA $-$ UCB1 outperform other algorithms. Although the cumulative regret of NMF $-$ Ban is less than our proposed approaches, note that it does not converge and find the d best items.



(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

7 Related Work

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

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, Maillard, and Zaki (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, Pál, and Szepesvári (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, Kleinberg, and Joachims (2008) which showed that query abandonment by user can also be successfully used to learn rankings. Later works on ranking such as Slivkins, Radlinski, and Gollapudi (2010) and Slivkins, Radlinski, and Gollapudi (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, Markov, and Rijke (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, Wolsey, and Fisher, 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, Kleinberg, and Joachims (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, Wolsey, and Fisher, 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

Abbasi-Yadkori, Y.; Pál, D.; and Szepesvári, C. 2011. Improved algorithms for linear stochastic bandits. *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Sys-*

tems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain. 2312–2320.

Agrawal, S., and Goyal, N. 2012. 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* 39.1–39.26.

Anandkumar, A.; Ge, R.; Hsu, D.; Kakade, S. M.; and Telgarsky, M. 2014. Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research* 15(1):2773–2832.

Auer, P.; Cesa-Bianchi, N.; Freund, Y.; and Schapire, R. E. 2002. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing* 32(1):48–77.

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

Carbonell, J., and Goldstein, J. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 335–336. ACM.

Chuklin, A.; Markov, I.; and Rijke, M. d. 2015. Click models for web search. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 7(3):1–115.

Goldberg, K.; Roeder, T.; Gupta, D.; and Perkins, C. 2001. Eigentaste: A constant time collaborative filtering algorithm. *information retrieval* 4(2):133–151.

Gopalan, A.; Maillard, O.; and Zaki, M. 2016. Low-rank bandits with latent mixtures. *arXiv preprint arXiv:1609.01508*.

Katariya, S.; Kveton, B.; Szepesvari, C.; Vernade, C.; and Wen, Z. 2016. Stochastic rank-1 bandits. *arXiv preprint arXiv:1608.03023*.

Katariya, S.; Kveton, B.; Szepesvári, C.; Vernade, C.; and Wen, Z. 2017. Bernoulli rank-1 bandits for click feedback. *arXiv preprint arXiv:1703.06513*.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* (8):30–37.

Krause, A., and Golovin, D. 2014. Submodular function maximization.

Kveton, B.; Szepesvari, C.; Rao, A.; Wen, Z.; Abbasi-Yadkori, Y.; and Muthukrishnan, S. 2017. Stochastic low-rank bandits. *arXiv preprint arXiv:1712.04644*.

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

Maillard, O.-A., and Mannor, S. 2014. Latent bandits. In *International Conference on Machine Learning*, 136–144.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functionsi. *Mathematical programming* 14(1):265–294.

Radlinski, F.; Kleinberg, R.; and Joachims, T. 2008. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*, 784–791. ACM.

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

Sen, R.; Shanmugam, K.; Kocaoglu, M.; Dimakis, A. G.; and Shakkottai, S. 2016. Contextual bandits with latent confounders: An nmf approach. *arXiv preprint arXiv:1606.00119*.

Slivkins, A.; Radlinski, F.; and Gollapudi, S. 2010. Ranked bandits in metric spaces: learning optimally diverse rankings over large document collections. *arXiv preprint arXiv:1005.5197*.

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

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

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

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

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

Zoghi, M.; Tunys, T.; Ghavamzadeh, M.; Kveton, B.; Szepesvari, C.; and Wen, Z. 2017. Online learning to rank in stochastic click models. *arXiv preprint arXiv:1703.02527*.

A Proof

The reward for recommending d columns J to user i is

$$r_t(i, J) = \max\{\mu(k)\, r_t(i, J(k)) : k \in [d]\}$$

for weights $\mu(1) \geq \cdots \geq \mu(d) > 0$. We also define the corresponding unweighted reward as

$$\tilde{r}_t(i, J) = \max\{r_t(i, J(k)) : k \in [\text{length}(J)]\}\,.$$

Let J_* be the indices of hott topics and $\pi_{*,i}$ be their highest-reward permutation for user i. Let J_t be our recommended columns at time t and $\pi_{t,i}$ be their permutation for user i, which is computed by some later-defined row algorithm. Then the expected n-step regret of our learning agent is

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

To decompose the regret into its row and column components, we introduce event $1\{J_t \neq J_*\}$,

$$\begin{aligned}
R(n) &= \sum_{t=1}^{n} \mathbb{E}\left[1\{J_t \neq J_*\}(r_t(i_t, \pi_{*,i_t}(J_*)) - r_t(i_t, \pi_{t,i_t}(J_t)))\right] + \sum_{t=1}^{n} \mathbb{E}\left[1\{J_t = J_*\}(r_t(i_t, \pi_{*,i_t}(J_*)) - r_t(i_t, \pi_{t,i_t}(J_*)))\right] \\
&\leq \sum_{t=1}^{n} \mathbb{E}\left[1\{J_t \neq J_*\}\right] + \sum_{i=1}^{K} \mathbb{E}\left[\sum_{t=1}^{n} 1\{i_t = i, J_t = J_*\}(r_t(i, \pi_{*,i}(J_*)) - r_t(i, \pi_{t,i}(J_*)))\right]\,,
\end{aligned}$$

where the inequality follows from the fact that the maximum instantaneous regret is 1.

The first term in the regret decomposition can be bounded as follows. Let

$$C_{t,i}(J, k) = \max\{r_t(i, J(\ell)) : \ell \in [k]\}\,, \quad g_{t,i}(J, k) = C_{t,i}(J, k) - C_{t,i}(J, k-1)\,.$$

Let J_* be defined greedily as

$$J_*(k) = \operatorname*{arg\,max}_{j \in [L]} \sum_{t=1}^{n} \mathbb{E}\left[g_{t,i_t}(J_*(:k-1) \oplus j, k)\right]$$

for $k \in [d]$. Then from the design of our column learning algorithm,

$$\max_{j \in [L]} \sum_{t=1}^{n} \mathbb{E}\left[g_{t,i_t}(J_t(:k-1) \oplus j, k)\right] - \sum_{t=1}^{n} \mathbb{E}\left[g_{t,i_t}(J_t, k)\right] \leq \sqrt{Ln}$$

for any $k \in [d]$. This implies that

$$\begin{aligned}
\sum_{t=1}^{n} \mathbb{E}\left[g_{t,i_t}(J_t, k)\right] &\geq \max_{j \in [L]} \sum_{t=1}^{n} \mathbb{E}\left[g_{t,i_t}(J_t(:k-1) \oplus j, k)\right] - \sqrt{Ln} \\
&\geq \frac{1}{d} \sum_{j \in J_*} \sum_{t=1}^{n} \mathbb{E}\left[g_{t,i_t}(J_t(:k-1) \oplus j, k)\right] - \sqrt{Ln} \\
&= \frac{1}{d} \sum_{t=1}^{n} \mathbb{E}\left[\sum_{j \in J_*} g_{t,i_t}(J_t(:k-1) \oplus j, k)\right] - \sqrt{Ln} \\
&\geq \frac{1}{d} \sum_{t=1}^{n} \mathbb{E}\left[C_{t,i_t}(J^*, d)\right] - \sqrt{Ln}\,.
\end{aligned}$$

Now we prove for any $k \in [d]$,

$$\sum_{t=1}^{n} \mathbb{E}\left[C_{t,i_t}(J^*, d)\right] - \sum_{t=1}^{n} \mathbb{E}\left[C_{t,i_t}(J_t, k)\right] \leq \frac{d-k}{d} \sum_{t=1}^{n} \mathbb{E}\left[C_{t,i_t}(J^*, d)\right] + k\sqrt{Ln}\,.$$

The above claim clearly holds for $k = 0$. Now for $k > 0$,

$$\begin{aligned}\sum_{t=1}^{n} \mathbb{E}\left[C_{t,i_t}(J^*, d)\right] - \sum_{t=1}^{n} \mathbb{E}\left[C_{t,i_t}(J_t, k)\right] &= \sum_{t=1}^{n} \mathbb{E}\left[C_{t,i_t}(J^*, d)\right] - \sum_{t=1}^{n} \mathbb{E}\left[C_{t,i_t}(J_t, k-1)\right] - \sum_{t=1}^{n} \mathbb{E}\left[g_{t,i_t}(J_t, k)\right] \\ &\leq \frac{d-k+1}{d} \sum_{t=1}^{n} \mathbb{E}\left[C_{t,i_t}(J^*, d)\right] + (k-1)\sqrt{Ln} - \sum_{t=1}^{n} \mathbb{E}\left[g_{t,i_t}(J_t, k)\right] \\ &\leq \frac{d-k}{d} \sum_{t=1}^{n} \mathbb{E}\left[C_{t,i_t}(J^*, d)\right] + k\sqrt{Ln}\,,\end{aligned}$$

where the first inequality is from our induction hypothesis and the last inequality is from our lower bound on $C_{t,i_t}(J_*, d)$. For $k = d$, the claim we have that

$$\sum_{t=1}^{n} \mathbb{E}\left[C_{t,i_t}(J^*, d)\right] - \sum_{t=1}^{n} \mathbb{E}\left[C_{t,i_t}(J_t, k)\right] \leq d\sqrt{Ln}\,,$$

This is what we needed.

Let

$$\Delta = \min_{t \in [n]} \min_{J:\, J \neq J_*} \mathbb{E}\left[\tilde{r}_t(i, J^*)\right] - \mathbb{E}\left[\tilde{r}_t(i, J)\right]$$

be the minimum gap between the optimal and best suboptimal columns, averaged over users. Then by Lemma **??**, we have

$$\sum_{t} \mathbb{E}\tilde{r}_t(i_t, J^*) - \mathbb{E}\tilde{r}_t(i_t, J_t) \leq O(d\sqrt{nL}).$$

Combining with with the definition of Δ, we get

$$\sum_{t=1}^{n} \mathbb{E}\left[1\{J_t \neq J_*\}\right] \leq O\left(\frac{d\sqrt{Ln}}{\Delta}\right).$$

The second term in the regret decomposition can be bounded as

$$\sum_{i=1}^{K} \mathbb{E}\left[\sum_{t=1}^{n} 1\{i_t = i, J_t = J_*\}(r_t(i, \pi_{*,i}(J_*)) - r_t(i, \pi_{t,i}(J_*)))\right] \leq \sum_{i=1}^{K} R_i(n)\,,$$

where $R_i(n)$ is the expected n-step regret of the row algorithm in row i, conditioned on the event that the column algorithm chooses J_*. One suitable row algorithm is the weighted majority algorithm, which learns the optimal permutation for each J. Conditioned on J, this is a full-information setting with $d!$ arms. Therefore, $R_i(n) = O(\log n + \log d!) = O(\log n + d \log d)$. Now we chain all above inequalities and get that

$$R(n) = O\left(\frac{d\sqrt{Ln}}{\Delta} + K \log n + Kd \log d\right).$$