Latent Ranked Bandit

Author names withheld

Abstract

In this paper, we study the problem of suggesting a diverse list of items that contains the best item for users who are observed sequentially. The learner suggests some items to the user and receives feedback on all the recommended items. The user-item preference matrix generating the feedback for user-item interaction consists of unobserved latent factors and is low rank. While previous methods have tried to reconstruct the preference matrix using various statistical methods from its noisy observations, these are prone to failure and require a broad set of assumptions for the theoretical guarantees to hold. We propose the Latent Ranked Bandit algorithm which does not try to reconstruct the preference matrix but instead races to find the best item for each user by leveraging the low-rank structure of the preference matrix. To do this, we borrow ideas from learning to rank. Under our modeling assumptions, we prove that our algorithm performs optimally. We also evaluate the performance of our algorithm in several simulated and real-world datasets. In all the experiments it outperforms the existing state-of-the-art algorithms.

1 Introduction

In this paper, we study the problem of recommending the best items to users who are coming sequentially. The learner has access to very less prior information about the users (cold start) and it has to adapt quickly to the user preferences and suggest the best item to each user. We assume that the user-item preferences depends on latent factors. The learner only has access to noisy observations from the user-item preference matrix and the latent user or item features are not accessible. We further assume that the user-item preference matrix has low-rank, which is a very common assumption in recommender systems (Koren, Bell, and Volinsky, 2009), (Ricci, 2011).

This complex problem can be conceptualized as a low rank online learning problem where there are K users and L items. The user-item preference matrix, denoted by $M \in [0,1]^{K \times L}$, generating the feedback for user, item interaction has a low rank structure. The online learning game proceeds as follows, at every timestep t, nature reveals one user (or row index) from M where user is denoted by i_t. Then the learner selects d items (or columns) from $[L]$, where an item is denoted by $\ell_j \in [L]$. Let this d items be denoted by $\ell_1, \ell_2, \ldots, \ell_d$ and belong to the set J_t . The learner then receives feedback $r_t(i_t, \ell_j)$ for all the d items in J_t from a noisy realization of M. This noisy realization of M at time t is denoted by M_t and $r_t(i_t, \ell_j) \sim M_t(i_t, \ell_j)$ where $\ell_j \in J_t$. Let, $\pi_{t,i}$ be the permutation of items in J_t for user i_t by the learner. Also, let $\pi_{*,i}$ be a permutation such that the best item is displayed to the user i at rank position 1. Then the goal of the learner is to minimize the cumulative regret, that is to find the best permutation $\pi_{*,i}$ for each user $i \in [K]$. Hence, the learner needs to quickly identify the best item ℓ_{j^*} for each $i \in [K]$ where $M(i, \ell_{j^*}) = \arg\max_{\ell_j \in [L]} M(i, \ell_j)$ and show it in rank position 1.

This learning model is motivated from the real-world scenario where the learner has to suggest movies to users and each movie belongs to a different genre (say thriller, romance, comedy, etc). So, the learner can suggest d movies belonging to different genres to each incoming user on a webpage, and the user can click one, or all, or none of the recommended movies.

1.1 Contributions

Our contributions are mainly three fold. First, we formulate the Latent Ranked Bandit problem as an online learning problem on a class of low-rank non-negative matrices that can be solved efficiently without estimating the latent factors that generate the matrices. We borrow ideas from the ranking literature to solve this problem efficiently and term this setting as personalized ranking setting. This is because an efficient algorithm can find a global ranking of best items and then permute that list to find a personalized ranking for individual users, sorted by their preference from high to low. Secondly, we propose the Latent Ranker Algorithm (LRA) for this personalized ranking setting. LRB has two components, the column MABs and the row MABs that work simultaneously to find a diverse list of items that will contain the best item at rank 1 position with high probability. The column MABs leverage the low rank structure of the user-preference matrix M and quickly finds out the d best items. Simultaneously the row MABs permute the suggested list of d items to find the best item at rank 1 position for the individual users. Finally, we show that an instance of LRB which uses exponential weighting algorithm EXP3 as column MABs and weighted majority algorithm (WMA) as row

MABs suffer a regret of atmost $O\left(d^2\sqrt{Ln} + K\log n\right)$. On diverse experimental settings we test our proposed algorithm and show improved performance even when our modeling assumptions does not hold.

The rest of the paper is organized as follows. Then we state our setting, assumptions and notations in Section 2. We propose the algorithm LRB in Section 3 and in Section 4 we analyze LRB. Finally, we show experiments in Section 5 and give a brief survey of the existing literature in Section 6. We conclude in Section 7 while the proof is contained in the Appendix A.

2 Setting

Let $[n] := \{1, 2, \ldots, n\}$. For any two sets A and B, A^B denotes the set of all vectors indexed by B and whose coordinates are in A. Let $M \in [0,1]^{K\times L}$ be a matrix and $I \subset [K]$, then $M(I,:)$ denotes the $|I| \times L$ dimensional submatrix of M corresponding to the rows whose indices are given by I. Similarly, we use $M(:,J)$ to denote the submatrix of M whose columns are given by J.

Let $M \in [0,1]^{K\times L}$ reward matrix.. Also, let us assume that the rank of M is $d \ll \min\{L, K\}$. Let $U \in [\mathbb{R}^+]^{K\times d}$, $V \in [0,1]^{L\times d}$ be matrices representing the user and item latent factors,

$$M = UV^\intercal.$$

Furthermore, without loss of generality, we put a constraint on V such that, $\forall j \in [L]$, $\|V(j,:)\|_1 \leq 1$. Note that this assumption only affects the magnitudes of rewards.

Assumption 1. ***(Hott-Topics)*** *We assume that there exists d rows J^*, such that every row of V can be written as a convex combination of the rows in $V(J^*,:)$ and the zero vector.*

We denote the row factors by $V^* = V(J^*,:)$. Therefore, for every $i \in [L]$, there exists a column vector $a \in [0,1]^d$ and $\|a\|_1 \leq 1$ such that

$$V(i,:) = V(J^*,:)a.$$

B: The above wording is very confusing because we have hott topics columns. I know that these are the rows of V.

Observation Model: For each user i_t revealed by the nature at round t, the learner is allowed to suggest atmost d-items, where d is the rank of the matrix M. The user can click one, or all, or none of the recommendations and the learner observes all the d clicks. We assume the feedback to be a non-negative number from the preference matrix M. This can be thought of as a preference factor of the user towards the item.

Discussion 1. *Our observation model is different from the document click model (DCM) from Craswell et al. (2008). In DCM every user-item pair has a single parameter called the user-item attraction factor which determines the click probability of the user when the item is shown. This click probability in DCM is explicitly modeled as a Bernoulli trial with clicks being $\{0,1\}$. But in our setting the clicks can be any positive real number in $[0,1]$.*

Noise Model: At every timestep t, we generate a noisy matrix $M_t = UD_tV^\intercal$, where D_t is a diagonal matrix such that $D_t(i,i) \in [0,1]$. Thus, for every such realization of $\tilde{M}_t$ the hott-topics structure of M is preserved.

Discussion 2. *Our noise model is quite different from the existing stochastic noise model assumptions of various click models. The usual i.i.d Bernoulli reward assumption on the entries of the user-item preference matrix M is not feasible because the hott-topics assumption is required for every realization of the matrix M. Note, that in our noise model the d-hott topics do not change over time.*

Let, the reward for recommending set J_t consisting of d columns to user i_t is

$$r_t(i_t, J_t) = \max\left\{\mu(k)\, r_t(i_t, J_t(k)) : k \in [d]\right\}$$

for weights $\mu(1) \geq \cdots \geq \mu(d) > 0$. Then the main goal of the learning agent is to minimize the cumulative regret $R(n)$ over n time steps as

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

where, J_t is the set of d suggested items, $\pi_{*,i_t}(J_*)$ is the best possible permutation of suggested items to user i_t on hindsight where the best item $\ell_{j^*} \in J^*$ is at rank position 1 and $\pi_{t,i_t}(J_t)$ is the actual suggested items.

3 Proposed Algorithms

We propose the algorithm latent ranker algorithm, abbreviated as LRA (see Algorithm 1) for solving the personalized ranking problem. This algorithm is motivated by the ranked bandit algorithm (RBA) from Radlinski, Kleinberg, and Joachims (2008). The RBA is suited for finding a global ranking of items amongst all the users without being personalized for individual user and so it will fail to suggest the best ranking for each user sorted by their preference from high to low. LRA is divided into two main components, the d column MABs denoted by $\text{MAB}_1(n)$, $\text{MAB}_2(n), \ldots,$ $\text{MAB}_d(n)$ and the K row weighted majority algorithms (WMA) for each user $[K]$. The WMA from Littlestone and Warmuth (1994) is suited for the total information setting. Note, that in our setting all the clicks by the user is seen by the system and so it is a total information setting. Each WMA consist of $d!$ arms for each of the permutations of rank d. Its goal is to suggest a permutation $\pi_{i_t}(J_t)$ such that the best item for the user i_t is in rank 1 of the permutation $\pi_{i_t}(J_t)$.

LRB proceeds as follows. At every timestep $t \in [n]$, a user i_t is revealed by nature, then the d column MABs suggests columns $J_t = \{\ell_1, \ell_2, \ldots, \ell_d\}$ which it deems to be the d best columns and by virtue of our setting, the d hott-topics. If, there is any overlap in the suggestion, an arbitrary column is suggested which has not been selected before. Then, the row WMA for the i_t-th user selects a permutation $\Pi_{i_t}(J_t)$ by sampling through its distribution over the $d!$ arms and suggests a permutation $\pi_{i_t}(J_t) = \{\tilde{\ell}_1, \tilde{\ell}_2, \ldots, \tilde{\ell}_d\}$ such that the item in rank 1 is the best item for the i_t-th user with a

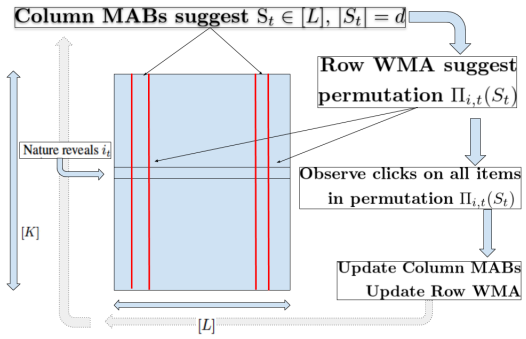


Figure 1: Latent Ranked Bandit in rank $d = 2$ scenario.

high probability. Note, that we leave the implementation of the column MAB to the user which can be any stochastic or adversarial base-bandit algorithm discussed in Section 6.

Finally, after all the user clicks are recorded by the system both the column MABs and row $\text{WMA}_{i_t}(n)$ is updated. The k-th column $\text{MAB}_k(n)$ is updated with feedback $f_{k,t} = \max_{j\in[k]} r_t(i_t, \ell_{j,t}) - \max_{j\in[k-1]} r_t(i_t, \ell_{j,t})$. The intuition behind this update is that first the rank of the k-th column will be fixed and then based on that the rank of the $(k-1)$-th column will be determined. This type of update follows from the fact that the $\max$ function is monotonic and submodular (see section 6) and by updating $f_{k,t}$ like this the column MABs are guaranteed to find the d best columns with high probability. Note that RBA is also a special case of submodular bandits such that $f_{k,t} \in \{0, 1\}$ (Streeter and Golovin, 2009).

The row $\text{WMA}_{i_t}(n)$ update for the i_t-th user is quite straightforward as all the d clicks are observed (total information). Hence, LRA can easily calculate the feedback for all the $d!$ permutation of $\pi_{i_t}(J_t)$ and update its $d!$ arms representing each of those permutations. LRA calculates a weighted sum of feedback for each of the $d!$ permutation of $\pi_{i_t}(J_t)$ and update the weights and its probabilities of the corresponding permutation. An illustrative diagram of the entire process is shown in Figure 1.

4 Analysis

Theorem 1. *The worst case cumulative regret upper bound for latent ranker algorithm is,*

$$R(n) \leq c^2 d\sqrt{Ln} + K \log n$$

when $n > L$ and c is a constant such that $c > 2\sqrt{d}$.

Proof. **(Outline)** The proof of Theorem 1 is given in Appendix A. We divide the proof into two parts. The first part deals with the regret incurred due to the column MABs and the minimum number of pulls required before the column MABs starts suggesting the d-best columns (and by design the d hott-topics) more often. This analysis follows from the proof of regret bound from Radlinski, Kleinberg, and Joachims (2008) with modifications made to suit our setting. The second part of the proof deals with finding the best permutation amongst the suggested d best items by using

Algorithm 1 Latent Ranker Algorithm

Input: Rank d, horizon n.
Initialize $\text{MAB}_1(n), \text{MAB}_2(n), \ldots, \text{MAB}_d(n)$
Initialize $\text{WMA}_1(n), \text{WMA}_2(n), \ldots, \text{WMA}_K(n)$
for $t = 1, \ldots, n$ **do**
  User i_t comes to the system
  for $k = 1, \ldots, d$ **do**
    $\ell_{k,t} \leftarrow$ suggest item $MAB_k(n)$
    if $\ell_{k,t} \in \ell_{1,t}, \ldots, \ell_{k-1,t}$ **then**
      $\ell_{k,t} \leftarrow$ Select arbitrary unselected item from $[L] \setminus \ell_{1,t}, \ldots, \ell_{k-1,t}$
  $\tilde{\ell}_{1,t}, \tilde{\ell}_{2,t}, \ldots, \tilde{\ell}_{d,t} \leftarrow$ Permutation by $\text{WMA}_{i_t}(\ell_{1,t}, \ell_{2,t}, \ldots, \ell_{d,t})$ by sampling according to $p_{i_t,1}, p_{i_t,2}, \ldots, p_{i_t,d!}$.
  Present $\tilde{\ell}_{1,t}, \tilde{\ell}_{2,t}, \ldots, \tilde{\ell}_{d,t}$ to user i_t and record feedback $r_t(\tilde{\ell}_{1,t}), r_t(\tilde{\ell}_{2,t}), \ldots, r_t(\tilde{\ell}_{d,t})$.
  Call Procedure UpdateColumnMAB
  Call Procedure UpdateRowWMA(i_t)
procedure UPDATECOLUMNMAB
  for $k = 1, \ldots, d$ **do**
    Update $\text{MAB}_k(n)$ with feedback $f_{k,t} = \max_{j\in[k]} r_t(i_t, \ell_{j,t}) - \max_{j\in[k-1]} r_t(i_t, \ell_{j,t})$
procedure UPDATEROWWMA(i_t)
  for $k = 1, 2, \ldots, d!$ **do**
    $r_k = 0$ ▷ Calculate weighted sum of rewards
    for $j = 1, \ldots, d$ **do**
      $r_k = r_k + \frac{1}{j} r_t(\tilde{\ell}_{j,t})$
    $w_{i_t,k} = w_{i_t,k} + r_k$ ▷ Update weights
  for $k = 1, \ldots, d!$ **do**
    $p_{i_t,k} = \dfrac{\exp(w_{i_t,k})}{\sum_{b=1}^{d!} \exp(w_{i_t,b})}$

WMA. Finally, we combine both these steps to derive the regret upper bound. □

Discussion 3. *From the result in Theorem 1 we see that the regret consist of two parts of unequal order. The first part of order* $O\left(d^2\sqrt{Ln}\right)$ *for* $c = 2\sqrt{d}$ *is incurred for finding the* d *best items (hott-topics) with high probability. The second part of order* $O(K \log n)$ *is incurred by WMA for finding the best permutation once the column MABs starts suggesting the* d*-best items. Note, that the result has the correct order as it does not scale with* $O\left(\sqrt{KLn}\right)$ *like the independent user model algorithms.*

Discussion 4. *Note, that for proving the regret bound we need an instance of LRA which uses EXP3 as column MAB. This is because the feedbacks are no longer independent of each other. The feedback* $f_{k,t}$ *to the* k*-th column MAB is dependent on the feedback of the* $k-1$*-th column MAB. Hence, adversarial MABs which can work with any sequence of feedbacks are required for giving theoretical guarantees in this setting.*

5 Experiments

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

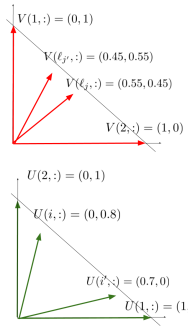
Independent User Model Algorithms: In this approach, each user has a separate version of base-bandit algorithm running independent of each other. As base-bandit algorithms we choose two versions of stochastic MAB, UCB1 and Thompson Sampling (TS), abbreviated as Contextual UCB1 (CUCB1) and Contextual TS (CTS) respectively. For UCB1, we choose the confidence interval at timestep t as $c_{i_t,j}(t) = \sqrt{\frac{1.5 \log t}{N_{i_t,j}(t)}}$ for the i_t-th user and j-th item. Note, that both the vanilla UCB1 and TS is used to find the best item for each user at rank 1, while for the remaining positions $k = 2, \ldots, d$ it suggest previously unselected items by sampling uniform randomly at every timestep t.

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 method to reconstruct partially observed noisy matrices, the non-negative matrix factorization. We term the corresponding algorithm as NMF Bandit (NMF-Ban). This algorithm is ϵ-greedy in implementation whereby it reconstructs M with ϵ probability and with $1-\epsilon$ probability it behaves greedily over the reconstructed matrix and suggest d best item for the i_t-th user at every timestep t.

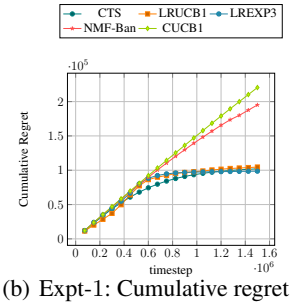
Personalized Ranking Algorithms: In this approach, we evaluate our proposed algorithm Latent Ranking Bandit (LRA) by using two different types of Column MABs, EXP3 and UCB1. We term them as LREXP3 and LRUCB1 respectively. The row MABs for both of these algorithms is still Weighted Majority Algorithm (WMA). Note that we only show theoretical guarantees for LREXP3. For LREXP3 we set the column exploration parameter $\gamma_k = \sqrt{\frac{L \log L}{n}}, \forall\ \text{MAB}_k(n), k = 1, \ldots, d$ and for LRUCB1 we use a confidence interval of $c_{k,j}(t) = \sqrt{\frac{1.5 \log t}{N_{k,j}(t)}}$ for the k-th column MAB and j-th item.

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 $\text{rank}(M) = 2$. The vectors spanning U and V, generating the user-item preference matrix M is shown Figure 2(a). The users are divided into a $70:30$ split such that 70% of users prefer item j_1^* and 30% users prefer item j_2^*. The item hott-topics are $V(1,:) = (0,1)$ and $V(2,:) = (1,0)$ while remaining 70% of items has feature $V(\ell_{j'},:) = (0.45, 0.55)$ and the rest have $V(\ell_j,:) = (0.55, 0.45)$. We create the user feature matrix U similarly having a $70:30$ split such that $U(1,:) = (0,1)$, $U(2,:) = (1,0)$ and the remaining 70% users having $U(i,:) = (0, 0.8)$ and 30% users having $U(i',:) = (0.7, 0)$. The resultant matrix $M = UV^\intercal$ is such that algorithms that quickly find the easily identifiable hott-topics perform very well. From Figure 2(b) we can clearly see that both LREXP3 and LRUCB1 outperforms all the other algorithms. Their regret curve flattens, indicating that they have learned the best items for each user. NMF-Ban performs poorly as it fails to get a reasonable approximation of M although it performs better than CUCB1. Contextual algorithm CUCB1 performs poorly as the number of items per user is too large and the gaps are also small. Although CTS performs well in this small testbed, its performance eventually degrades for larger environments.

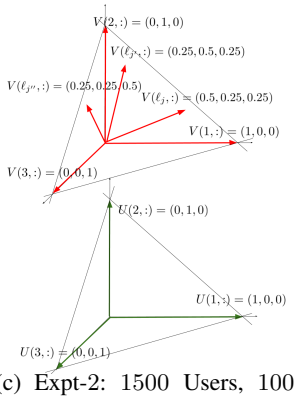
Experiment 2: We conduct the second experiment on a larger simulated database of 1500 users, 100 items and $\text{rank}(M) = 3$. The vectors spanning U and V, generating the user-item preference matrix M is shown Figure 2(c). The users are divided into a $60:30:10$ split such that 60% of the users prefer item j_1^*, 30% prefer j_2^* and 10% prefer j_3^*. 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 $V(\ell_j,:) = (0.5, 0.25, 0.25)$, 30% have $V(\ell_{j'},:) = (0.25, 0.5, 0.25)$ and rest have $V(\ell_{j''},:) = (0.25, 0.25, 0.5)$. 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, the resultant matrix $M = UV^\intercal$ is such that algorithms that quickly spot the easily identifiable hott-topics outperform others. From Figure 2(d) we can see that both LREXP3 and LRUCB1 again outperforms 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 contextual algorithms CUCB1 and CTS perform poorly as the number of users and the number of items per user is too large and the independent base-bandits are not sharing information between themselves. In



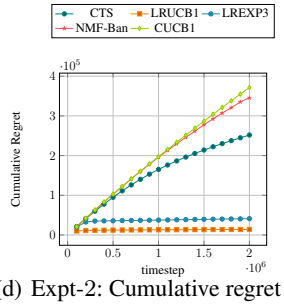
(a) Expt-1: 500 Users, 50 items, Rank 2, User and Item vectors



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



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

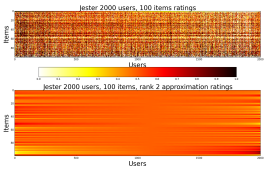


(d) Expt-2: Cumulative regret of different algorithms

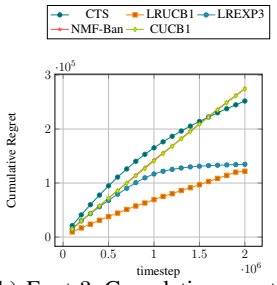
Figure 2: A comparison of the cumulative regret incurred by the various bandit algorithms.

both the simulated datasets, we see that stochastic column MAB (UCB1) is outperforming adversarial column MAB (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).

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. We sample randomly around 2000 users from this dataset and use singular value decomposition (SVD) to obtain a rank 2 approximation of this user-joke rating matrix M. The rank 2 approximation of M of is shown in Figure 3(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 3(b) again we see that LREXP3 outperforms other algorithms. The regret curve of LRUCB1 does not flatten out which we attribute to the fact that LRUCB1 uses too large a confidence interval. The contextual and matrix completion algorithms perform significantly worse in this large testbed.



(a) Expt-3: 2000 Users, 100 items, Rank 2 approximation of Jester Dataset



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

Figure 3: A comparison of the cumulative regret in Jester Dataset

6 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 can be randomized algorithms suited for the adversarial setting like EXP3 (Auer et al., 2002) or UCB type algorithms suited for the stochastic setting llike UCB1 (Auer, Cesa-Bianchi, and Fischer, 2002), MOSS (Audibert and Bubeck, 2009), OCUCB (Lattimore, 2015), KL-UCB (Cappé et al., 2013), (Garivier and Cappé, 2011) or even Bayesian algorithms like Thompson Sampling (Thompson, 1933), (Thompson, 1935), (Agrawal and Goyal, 2012).

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). A submodular function $f : 2^V \to \mathbb{R}$ for a finite ground set V is a set function that assign each subset $S \subseteq V$ a value $f(S)$. We define the gain of the function f as $G_f(e|S) = f(S \cup \{e\}) - f(S)$ where the element $\{e\} \in V \setminus S$ and $S \subseteq V$. Also, f satisfies the following two criteria:-

1. Monotonicity: A set function $f : 2^V \to \mathbb{R}$ is monotone if for every $A \subseteq B \subseteq V, f(A) \leq f(B)$.
2. Submodularity: A set function $f : 2^V \to \mathbb{R}$ is submodular if for every $A \subseteq B \subseteq V$ and $\{e\} \in V \setminus B$ it holds that $G_f(e|A) \geq G_f(e|B)$.

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).

7 Conclusions and Future Directions

In this paper, we studied the problem of suggesting a diverse list of items to users, with the best item of individual users at rank position 1. We formulated the above problem as a personalized ranking problem and proposed the Latent Ranked Bandit algorithm for this setting. We proved that an instance of algorithm has a regret bound that scales as $O\left(d^2\sqrt{Ln} + K \log n\right)$ and has the correct order with respect to user, 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 future 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 Systems 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.

Audibert, J., and Bubeck, S. 2009. Minimax policies for adversarial and stochastic bandits. *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009* 217–226.

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.

Cappé, O.; Garivier, A.; Maillard, O.-A.; Munos, R.; Stoltz, G.; et al. 2013. Kullback–leibler upper confidence bounds for optimal sequential allocation. *The Annals of Statistics* 41(3):1516–1541.

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.

Craswell, N.; Zoeter, O.; Taylor, M.; and Ramsey, B. 2008. An experimental comparison of click position-bias models. In *Proceedings of the 2008 international conference on web search and data mining*, 87–94. ACM.

Garivier, A., and Cappé, O. 2011. The KL-UCB algorithm for bounded stochastic bandits and beyond. *COLT 2011 - The 24th Annual Conference on Learning Theory, June 9-11, 2011, Budapest, Hungary* 19:359–376.

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*.

Lattimore, T. 2015. Optimally confident UCB : Improved regret for finite-armed bandits. *CoRR* abs/1507.07880.

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

Our learning agent operates in the following setting. Let $i_1, \dots, i_n$ be a fixed sequences of users in n steps, which is unknown to the agent. Let $r_t = UD_tV^\intercal$ be the reward matrix at time t, where U is a non-negative matrix, D_t is a non-negative diagonal matrix that may change with t, and V is a non-negative hott-topics matrix. We assume that $r_t \in [0,1]^{K \times L}$ at all times $t \in [n]$. The only randomness in our problem is due to the learning agent.

The reward for recommending d columns J to user i is

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

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

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

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. The expected n-step regret, where the only randomness is due to the learning agent, is

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

The regret of the column learning algorithm in n_0 steps is bounded as

$$\mathbb{E}\left[\sum_{t=1}^{n_0} \tilde{r}_t(i_t, J_*)\right] - \mathbb{E}\left[\sum_{t=1}^{n_0} \tilde{r}_t(i_t, J_t)\right] \leq d\sqrt{Ln_0}$$

for any n_0, based on a similar analysis to ranked bandits. Let

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

be the minimum gap.

B: The above definition of the gap needs to be adjusted. It is zero whenever J contains the optimal column for user i. Our algorithm is sound and learns J^*. So this is just a technicality.

Then, based on the above inequalities, the probability that the column learning algorithm chooses J_* at any time $t \geq n_0$ is bounded from below by

$$1 - \frac{d\sqrt{Ln_0}}{\Delta n_0} = \frac{\Delta\sqrt{n_0} - d\sqrt{L}}{\Delta\sqrt{n_0}} \tag{1}$$

for any $\Delta \geq d\sqrt{L/n_0}$.

Let p_t be the probability that the column learning algorithm chooses J_* at time t and let $\pi_{t,i}(J_*)$ be its permutation for user i at time t, according to our row algorithm. Then we can bound the regret from time n_0 as

$$\begin{aligned}
R(n) &= \sum_{i=1}^{K} \mathbb{E}\left[\sum_{t=n_0}^{n} 1\{i_t = i\}(r_t(i, \pi_{*,i}(J_*)) - r_t(i, \pi_{t,i}(J_t))\right] \\
&= \sum_{i=1}^{K} \mathbb{E}\left[\sum_{t=n_0}^{n} \frac{1}{p_t} 1\{i_t = i, J_t = J_*\}(r_t(i, \pi_{*,i}(J_*)) - r_t(i, \pi_{t,i}(J_*))\right] \\
&\leq \left(1 + \frac{d\sqrt{L}}{\Delta\sqrt{n_0} - d\sqrt{L}}\right) \sum_{i=1}^{K} R_i(n)\,,
\end{aligned}$$

where $R_i(n)$ is the expected n-step regret of the row algorithm in row i, conditioned on the fact that the column learning algorithm chooses J_*. One suitable row algorithm would be the weighted majority algorithm, which learns the optimal permutation for each J. Then $R_i(n) = O(\log n + \log d!) \approx O(\log n + d \log d)$.

In the first n_0 steps, we bound the regret trivially by n_0. Then the expected n-step regret is bounded up to log factors as

$$R(n) \leq n_0 + \left(1 + \frac{d\sqrt{L}}{\Delta\sqrt{n_0} - d\sqrt{L}}\right) K\,.$$

The bound can be interpreted as follows. Choose some reasonable n_0 that makes the regret comparable to ranked bandits, such as $n_0 = c^2 d\sqrt{Ln}$ for some $c > 0$. Then the multiplier at K becomes smaller than 2, for instance, when $\Delta c d^{\frac{1}{2}} L^{\frac{1}{4}} n^{\frac{1}{4}} \geq 2dL^{\frac{1}{2}}$. Under the assumption that $n \geq L$, this happens when $\Delta \geq 2\sqrt{d}/c$, which makes sense for $c > 2\sqrt{d}$.