Submodular Bandits for Online Personalized Ranking

Author names withheld

Abstract

To be written

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 very common in recommender systems (**?**), (**?**).

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) from M where user is denoted by i_t. The learner selects d items (or columns) from $[L]$, where an item is denoted by $j_t \in [L]$. Then the learner receives feedback $r_t(i_t, j_t)$ for all the d items suggested from one noisy realization of M. This noisy realization of M at time t is denoted by $\tilde{M}_t(i_t, j_t)$ and $r_t(i_t, j_t)] \sim \tilde{M}_t(i_t, j_t)$. Then the goal of the learner is to minimize the cumulative regret, that is to minimize the total number of wrong items displayed over time to the user. Hence, the learner needs to quickly identify the best item j^* for each $i \in [K]$ where $M(i, j^*) = \arg\max_{j \in [L]} M(i, j)$.

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 (query abandonement).

1.1 Related Works

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

Latent Bandits: The existing algorithms in latent bandit literature can be broadly classified into two groups: the online matrix completion algorithms and the online contextual bandit 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 **?** 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 **?** where the authors come up with an algorithm which uses the Robust Tensor Power (RTP) method of **?** to reconstruct the matrix M, and then use the OFUL procedure of **?** 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 **?** 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 *online contextual bandit 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 can be randomized algorithms suited for the adversarial setting like EXP3 (**?**) or UCB type algorithms suited for the stochastic setting llike UCB1 (**?**), MOSS (**?**), OCUCB (**?**), KL-UCB (**?**), (**?**) or even Bayesian algorithms like Thompson Sampling (**?**), (**?**), (**?**).

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 **?** which showed that query abandonment by user can also be successfully used to learn rankings. Later works on ranking such as **?** and **?** uses additional assumptions to handle ex-

ponentially 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 **?**. While **?**, **?** works in PBM, **?** works in both PBM and CBM. Finally, **?** can be viewed as a generalization of rank-1 bandits of **?** 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 (**?**), (**?**). A submodular function $f : 2^V \rightarrow \mathbb{R}$ 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 $f : 2^V \rightarrow \mathbb{R}$ is monotone if for every $A \subseteq B \subseteq V, f(A) \leq f(B)$.
2. Submodularity: A set function $f : 2^V \rightarrow \mathbb{R}$ 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 **?** where the authors propose a general algorithm whereas **?** 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 **?**. 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 (**?**).

1.2 Notations, Problem Formulation and Assumptions

We define $[n] = \{1, 2, \ldots, n\}$ and for any two sets A and B, A^B denotes the set of all vectors who take values from A and are indexed by B. Let, $M \in [0, 1]^{K \times L}$ denote any matrix, then $M(I, :)$ denote any submatrix of k rows such that $I \in [K]^k$ and similarly $M(:, J)$ denote any submatrix of j columns such that $J \in [L]^j$.

Let M be reward matrix of dimension $K \times L$ where K is the number of user or rows and L is the number of arms or columns. Also, let us assume that this matrix M has a low rank structure of rank $d << \min\{L, K\}$. Let U and V denote the latent matrices for the users and items, which are not visible to the learner such that,

$$M = UV^\intercal \quad \textbf{s.t.} \quad U \in [\mathbb{R}^+]^{K \times d}, V \in [0, 1]^{L \times d}.$$

Furthermore, we put a constraint on V such that, $\forall j \in [L]$, $\|V(j, :)\|_1 \leq 1$.

Assumption 1. ***(Hott-Topics)*** *We assume that there exists d-column base factors, denoted by $V(J^*, :)$, such that all rows of V can be written as a convex combination of $V(J^*, :)$ and the zero vector and $J^* = [d]$. We denote the column factors by $V^* = V(J^*, :)$. Therefore, for any $i \in [L]$, it can be represented by*

$$V(i, :) = a_i V(J^*, :),$$

where $\exists a_i \in [0, 1]^d$ and $\|a_i\|_1 \leq 1$.

Assumption 2. ***(Click 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 a Document Click Model (DCM) such that 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.*

Discussion 1. *The above Assumption 2 is an instance of the Document Click Model (DCM) first studied in **?**. Since, this click model depends on learning only one attraction factor for each user-item pair, it often leads to overfitting of model parameters. DCM is independent on the position of the item (PBM) and does not model the decreasing interest of the user (CBM) while surveying an ordered set of items.*

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

The main goal of the learning agent is to minimize the cumulative regret until the end of horizon n. We define the cumulative regret, denoted by $\mathcal{R}_n$ as,

$$\mathcal{R}_n = \sum_{t=1}^{n} \left\{ \sum_{z=1}^{d} \left(r_t\left(i_t, j^*\right) - r_t\left(i_t, j_{t,z}\right) \right) \right\}$$

where, $j^* = \arg\max_{j \in [L]} \{M(i_t, j)\}$ and $j_{t,z}$ be the suggestion of the learner for the i_t -th user for $z = 1, 2, \ldots, d$. Note that $r_t\left(i_t, j^*\right) \sim \tilde{M}_t\left(i_t, j^*\right)$ and $r_t\left(i_t, j_{t,z}\right) \sim \tilde{M}_t\left(i_t, j_{t,z}\right)$. Taking expectation over both sides, we can show that,

$$\begin{aligned}\mathbb{E}[\mathcal{R}_n] &= \mathbb{E}\left[\sum_{t=1}^{n} \left\{ \sum_{z=1}^{d} \left(r_t\left(i_t, j^*\right) - r_t\left(i_t, j_{t,z}\right) \right) \right\} \right] \\ &= \mathbb{E}\left[\sum_{t=1}^{n} \sum_{z=1}^{d} \left(N_{i_t, j_{z,t}}(t) \right) \right] \Delta_{i_t, j_{t,z}}\end{aligned}$$

where, $\Delta_{i_t,j_{z,t}} = M(i_t, j^*) - M(i_t, j_{z,t})$ and $N_{i_t,j_{t,z}}(t)$ is the number of times the learner has observed the $j_{t,z}$-th item for the i_t-th user. Let, $\Delta = \min_{i\in[K],j\in[L]}\{\Delta_{i,j}\}$ be the minimum gap over all the user, item pair in M.

2 Contributions

To be written.

3 Proposed Algorithms

We propose the algorithm Latent Ranked Bandit, abbreviated as LRB (see Algorithm 1) for solving the personalized ranking problem. This algorithm is motivated by the Ranked Bandit Algorithm (RBA) from **?** which is suited for finding a global ranking amongst all the users in the CBM click model. LRB 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 is motivated from **?** which is suited for the total information setting. Note, that in our DCM setting all the clicks by the user is seen by the system and so it is a variation of total information setting. Each WMA consist of $d!$ arms for each of the permutations of rank d. Its main goal is to suggest a permutation $\Pi_{i_t}(S_t), \exists S_t \subseteq [L]$ such that the best item for the user i_t is in rank 1 of the permutation $\Pi_{i_t}(S_t)$. 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 $S_t = \{\ell_1, \ell_2, \ldots, \ell_d\}$ 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 $\Pi_{i_t}(S_t)$ by sampling through its distribution over the $d!$ arms and suggests a permutation $\tilde{S}_t = \{\tilde{\ell}_1, \tilde{\ell}_2, \ldots, \tilde{\ell}_d\}$ such that the item in rank 1 is the best item for the i_t-th user with a 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 1.1.

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 bandit $\text{MAB}_k(n)$ is updated with feedback $f_{k,t} = \max_{j\in[k]} r_t(i_t, \ell_{j,t}) - \max_{j\in[k-1]} r_t(i_t, \ell_{j,t})$ such the monotonicity and submodularity properties discussed in section 1.1 are maintained. Note that RBA is also a special case of submodular bandits such that $f_{k,t} \in \{0, 1\}$ (**?**).

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, LRB can easily calculate the feedback for all the $d!$ permutation of $\Pi_{i_t}(S_t)$ and update its $d!$ arms representing each of those permutations. LRB calculates a weighted sum of feedback for each of the $d!$ permutation of $\Pi_{i_t}(S_t)$ and update the weights and its probabilities of the corresponding permutation. An illustrative diagram of the entire process is shown in Figure 1.

4 Experiments

In this section, we conduct three experiments and evaluate the performance of LRB against several bandit algorithms.

Algorithm 1 Latent Ranked Bandit

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**
    $p_{i_t,k} = \dfrac{\exp(w_{i_t,k})}{\sum_{b=1}^{d!} \exp(w_{i_t,b})}$

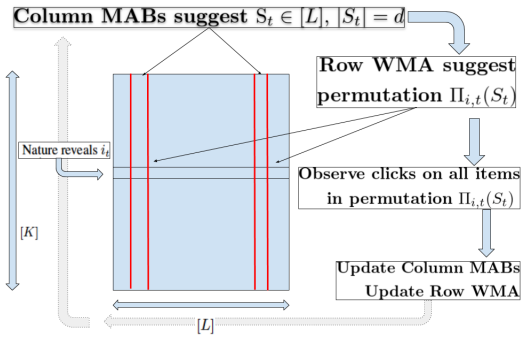


Figure 1: Latent Ranked Bandit in rank $d = 2$ scenario.

Note, that at every timestep t, nature reveals an user i_t and each algorithm suggests d items to it and records all the d feedbacks. The first two experiments are on simulated dataset where all our modelling assumptions hold. The third experiment is on a real-life dataset where we evaluate LRB when our modelling assumptions fail. In all our experiments users come in a Round Robin fashion over all time $[n]$. All the algorithms are averaged over 10 independent runs.

Contextual Algorithms: In the contextual 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.

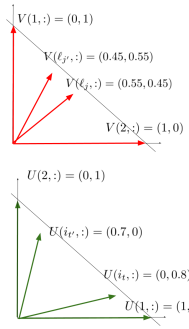
Matrix Completion Algorithms: In the matrix completion approach, the algorithms try to reconstruct the user-item preference matrix M from its noisy realization. We use two widely used method to reconstruct partially observed noisy matrices, linear ridge regression and non-negative matrix factorization. We term the corresponding algorithms as Linear Bandit (LinBan) and NMF Bandit (NMF-Ban) respectively. Both of these algorithms are ϵ-greedy in implementation whereby they reconstruct M with ϵ probability and with $1 - \epsilon$ probability they behave greedily over the reconstructed matrix and suggest d best item for the i_t-th user at every timestep t. LinBan uses ridge regression to reconstruct M from its estimated d-best columns while NMF-Ban uses matrix factorization to estimate the U and V matrix and reconstruct M from its noisy realization.

Personalized Ranking Algorithms: In this approach, we evaluate our proposed algorithm Latent Ranking Bandit (LRB) by using two different types of base-bandits, EXP3 and UCB1 as column MABs. We term them as LREXP3 and LRUCB1 respectively. 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.

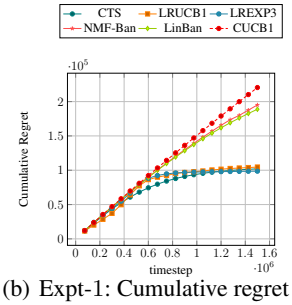
Experiment 1: This experiment is conducted to test the performance of LRB over small number of users and items. This simulated testbed consist of 500 users, 50 items and rank$(M) = 2$. The vectors spanning U and V, generating the user-item preference matrix M is shown Figure 2(a). The users are divided 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. 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 and LinBan has nearly similar performance as both of these algorithms fail to get a reasonable approximation of M although they perform 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 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 $\{1, 0, 0\}$, $\{0, 1, 0\}$ and $\{0, 0, 1\}$. 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 algorithms NMF-Ban and LinBan fail to get a reasonable approximation of M and perform 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 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 **?**.

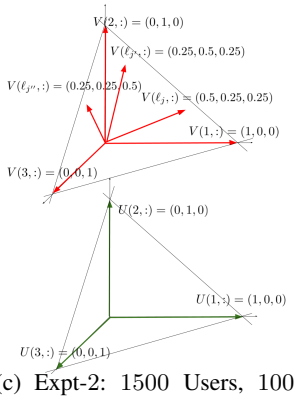
Experiment 3: We conduct the third experiment to test the performance of LRB when our modelling assumptions are violated. We use the Jester dataset (**?**) which consist of



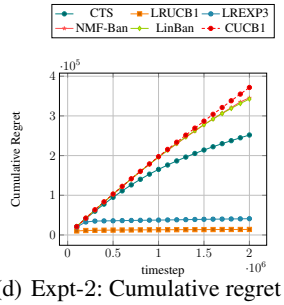
(a) Expt-1: 500 Users, 50 items, Rank 2, User and Item vectors



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

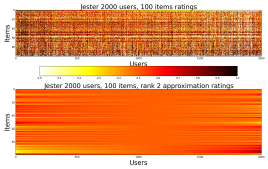


(c) Expt-2: 1500 Users, 100 items, Rank 3, User and Item vectors

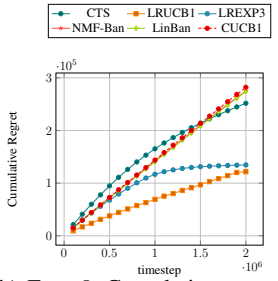


(d) Expt-2: Cumulative regret of different algorithms

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



(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

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.

5 Conclusions and Future Directions

To be written.