Efficient Online Recommendation via Low-Rank Ensemble Sampling

Xiuyuan Lu
Stanford University
Stanford, CA
lxy@stanford.edu

Zheng Wen
Adobe Research
San Jose, CA
zwen@adobe.com

Branislav Kveton
Google Research
Mountain View, CA
bkveton@google.com

ABSTRACT

The low-rank structure is one of the most prominent features in modern recommendation problems. In this paper, we consider an online learning problem with a low-rank expected reward matrix where both row features and column features are unknown a priori, and the agent aims to learn to choose the best row-column pair (i.e. the maximum entry) in the matrix. We develop a novel online recommendation algorithm based on ensemble sampling, a recently developed computationally efficient approximation of Thompson sampling. Our computational results show that our algorithm consistently achieves order-of-magnitude improvements over the baselines in both synthetic and real-world experiments.

KEYWORDS

Online Recommendation; Multi-Armed Bandit; Low Rank; Thompson Sampling; Ensemble Sampling

1 INTRODUCTION

The low-rank structure is one of the most prominent features in modern recommendation problems, and is arguably the key assumption behind many modern recommender systems [9]. In most recommendation problems, there is an expected reward/rating matrix, where rows represent users and columns represent items. While the number of users and the number of items can be very large in practice, many such matrices exhibit a low-rank structure, which makes efficient learning and inference possible.

Online learning with low-rank matrices remains an important challenge for many recommender systems. Online learning is crucial to the cold start problem, as well as to the quick adaptation to any behavioral or popularity changes of existing users and items. Depending on the specific context, there are different settings for online learning with low-rank matrices. One common setting is that, at each time step, the nature chooses a user, and then the agent (recommender) chooses an item [12, 15, 23, 24]. In another setting, the agent chooses both the user and the item at each time step [10, 11, 14]. The latter setting occurs if the recommender is interested in learning the best user-item pair, i.e. the maximum entry in the expected reward matrix, rather than the best item for each user group. One such example is an online streaming platform that aims to increase engagement with an unpopular movie genre by showing the most representative movie in that genre to the most interested user group. Variants of the second setting also occur in online learning in the position-based click model and e-mail marketing [11]. In both settings, after choosing an action (an item in the first setting, and a user-item pair in the second setting), the agent observes and receives a *reward* of that action, which might be a rating, a click, or a conversion. The agent's objective is to maximize its expected cumulative reward for a given time horizon.

This paper focuses on the second setting mentioned above. While various algorithms have been proposed, either they have restrictive assumptions such as requiring the matrix rank to be one [10, 11], or the proposed algorithms are computationally intractable in practice [14]. In this paper, we introduce a practical algorithm for online learning with low-rank matrices, referred to as the *low-rank ensemble sampling* algorithm. In particular, we assume that both row features and column features are unknown *a priori*. Our approach is based on Thompson sampling (TS) [19], a popular learning algorithm for a wide range of online learning problems. To make TS computationally efficient in our problem, we apply the recently developed ensemble sampling approach [16] to approximate TS. The computational results in this paper show that our algorithm consistently achieves order-of-magnitude improvements over the baselines in both synthetic and real-world experiments. Further, our algorithm offers a general exploration framework for online learning with low-rank matrices, which may be combined with many existing offline recommendation algorithms to allow for continual learning beyond the initial training. Finally, we would like to clarify that our algorithm can be easily adapted to the first setting mentioned above, which we will discuss more in detail in Section 5.

The remainder of this paper is organized as follows. In Section 2, we formulate the considered online learning problem as a *low-rank bandit* and briefly review Thompson sampling. We then propose the low-rank ensemble sampling algorithm in Section 3. Experiment results are shown in Section 4, and we briefly review the relevant literature and conclude the paper in Section 5.

2 LOW-RANK BANDIT & THOMPSON SAMPLING

We formulate the considered online recommendation problem as a multi-armed bandit (MAB) with low-rank structure, referred to as the *low-rank bandit*. Specifically, there is an expected reward matrix $R \in \mathfrak{R}^{K \times L}$, which is unknown to the agent. Depending on the specific context, rows of R may represent user and columns may represent items. At each time $t = 1, 2, \ldots, T$, the agent adaptively chooses a pair of indices $(i_t, j_t) \in \{1, \ldots, K\} \times \{1, \ldots, L\}$ based

RecSys '18, October 2–7, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5901-6/18/10…$15.00
https://doi.org/10.1145/3240323.3240408

on past observations. It then receives a reward r_t, which equals $R_{i_t j_t}$ plus some independent zero-mean noise. The agent's objective is to maximize its expected cumulative reward $\mathbb{E}\left[\sum_{t=1}^{T} r_t\right]$, or equivalently, to minimize its *expected cumulative regret*, defined as

$$\text{Regret}(T) = \mathbb{E}\left[\sum_{t=1}^{T} (r^* - r_t)\right],$$

where $r^* = \max_{i,j} R_{ij}$ is the optimal expected reward. As is commonly observed in recommendation problems, the reward matrix can usually be approximated well by a low-rank matrix. Thus, in this paper, we assume that R has low rank $d \ll K, L$. Consequently, we can write $R = UV^\top$ where $U \in \mathfrak{R}^{K \times d}$ and $V \in \mathfrak{R}^{L \times d}$.

Thompson sampling (TS) [19] is a widely used learning algorithm for variants of MAB. Recent literature [4] shows that it typically outperforms other types of MAB algorithms, such as UCB-based algorithms, in practice. A typical TS algorithm proceeds as follows: it is initialized with a prior distribution over the plausible models. At each time step, it samples a model from the posterior distribution, chooses an action that optimizes the sampled model, and then updates the posterior distribution based on its action and observation.

We illustrate a special case of the low-rank bandit where Thompson sampling can be efficiently applied. Consider the positive rank-1 case in which $d = 1$ and $R = uv^\top$, where $u \in \mathfrak{R}_+^K$ and $v \in \mathfrak{R}_+^L$. We assume that (u, v) is drawn from a jointly log-normal prior, and the observation noise is multiplicative and i.i.d. log-normal. More specifically, at each time t, the agent selects a pair (i_t, j_t) and observes $r_t = u_{i_t} v_{j_t} w_t$, where $w_t \sim \text{LogNormal}(-\sigma_w^2/2, \sigma_w^2)$. Note that by taking the logarithm of r_t, this problem can be transformed to the standard Gaussian linear bandits [1, 20, 21]. Hence, TS can be efficiently implemented through Kalman filtering. It is also easy to derive performance guarantees from previous analyses of TS on the linear bandit [1, 20, 21].

To derive a TS-based learning algorithm for general low-rank bandits, we assume that the prior over (U, V) follows the probabilistic matrix factorization (PMF) model [17]. Let U_i denote the i^{th} row of U and V_j denote the j^{th} row of V. Both are d-dimensional column vectors. We assume that each U_i and each V_j are drawn independently from some normal distribution,

$$U_i \sim N(\mu_i, \sigma_U^2 I), \quad V_j \sim N(\nu_j, \sigma_V^2 I)$$

for $i = 1, \ldots, K$ and $j = 1, \ldots, L$, where μ_i's and ν_j's are d-dimensional column vectors and $\sigma_U, \sigma_V \in \mathfrak{R}$. Further, we assume that $r_t = U_{i_t}^T V_{j_t} + w_t$, where w_t is i.i.d. $N(0, \sigma_w^2)$. Under this model, the posterior distribution over (U, V) is complicated and difficult to sample from. While it is possible to use Markov chain Monte Carlo [22] or particle filtering [12] to approximately sample from the posterior, such methods are typically computationally cumbersome and impractical if the system needs to make fast recommendations. In the following section, we introduce a computationally efficient algorithm for the low-rank bandit by applying a recently developed approximate TS framework.

Algorithm 1 Low-Rank Ensemble Sampling

Input: d, σ_w^2, $\{\mu_i\}_{i=1}^K$, $\{\nu_j\}_{j=1}^L$, σ_U^2, σ_V^2, M, and $\sigma_{\tilde{w}}^2$

Initialize: sample $\tilde{U}_{0,m,i} \sim N(\mu_i, \sigma_U^2 I)$, $\tilde{V}_{0,m,j} \sim N(\nu_j, \sigma_V^2 I)$ for $m = 1, \ldots, M$, $i = 1, \ldots, K$, $j = 1, \ldots, L$
for $t = 0, 1, 2, \ldots$ **do**
  Sample: $\hat{m} \sim \text{Unif}\{1, \ldots, M\}$
  Act: $(i_t, j_t) = \arg\max_{i,j} \tilde{U}_{t,\hat{m},i}^\top \tilde{V}_{t,\hat{m},j}$
  Observe: r_t
  for $m = 1, \ldots, M$ **do**
    Sample $\tilde{w}_{t,m} \sim N(0, \sigma_{\tilde{w}}^2)$
    $\tilde{r}_{t,m} = r_t + \tilde{w}_{t,m}$
    Compute $(\tilde{U}_{t+1,m}, \tilde{V}_{t+1,m})$ by solving (3)
  end for
end for

3 LOW-RANK ENSEMBLE SAMPLING

Since exact TS is computationally inefficient for general low-rank bandits, we resort to approximate TS methods. In particular, ensemble sampling [16] is a flexible and incremental approximate TS method that is well-suited for our problem.

We note that although the posterior distribution of (U, V) is complicated under the considered PMF model, it is not hard to compute an approximation of the maximum a posteriori (MAP) estimate. Let $\mathcal{H}_t = \{(i_\tau, j_\tau, r_\tau)\}_{\tau=1}^{t-1}$ denote the history of actions and observed rewards up to time t. The MAP estimate of (U, V) given $\mathcal{H}_t$ is

$$(\hat{U}_t, \hat{V}_t) = \arg\min_{U,V} \left(\frac{1}{\sigma_w^2} \sum_{\tau=1}^{t-1} (r_\tau - U_{i_\tau}^\top V_{j_\tau})^2 + \frac{1}{\sigma_U^2} \sum_{i=1}^{K} \|U_i - \mu_i\|_2^2 + \frac{1}{\sigma_V^2} \sum_{j=1}^{L} \|V_j - \nu_j\|_2^2 \right). \quad (1)$$

While this is a non-convex optimization problem, there are various approximation algorithms [2, 5, 8, 18] that come with certain guarantees. For example, one widely used approach is alternating least squares (ALS) [8], which alternatively optimizes U and V by solving least squares problems. More concretely, the updates of U for a given V are

$$U_i = \left(\frac{1}{\sigma_w^2} \sum_{\tau=1}^{t-1} \mathbb{1}_{\{i_\tau = i\}} V_{j_\tau} V_{j_\tau}^\top + \frac{1}{\sigma_U^2} I \right)^{-1} \cdot \left(\frac{1}{\sigma_w^2} \sum_{\tau=1}^{t-1} \mathbb{1}_{\{i_\tau = i\}} r_\tau V_{j_\tau} + \frac{1}{\sigma_U^2} \mu_i \right) \quad (2)$$

for $i = 1, \ldots, K$. The updates of V for a given U are similar and hence omitted here.

The idea of ensemble sampling is to approximate the posterior distribution with an ensemble of point estimates, each of which is fitted to a perturbed prior and perturbed observations. Intuitively, each point estimate is a statistically plausible model, and by fitting to a perturbed prior and perturbed observations, we diversify the ensemble in the aim of capturing model uncertainty and approximating the posterior.

The algorithm is presented in Algorithm 1. Its input includes the agent's estimated[1] rank d and noise variance σ_w^2, the number of

[1]Notice that the input d and σ_w^2 to Algorithm 1 are the agent's estimated values, which might not equal the true values.

models M, the perturbation magnitude $\sigma_{\tilde{w}}^2$, as well as parameters to specify the Gaussian prior. Let $(\tilde{U}_{t,m}, \tilde{V}_{t,m})$, $m = 1, \ldots, M$ denote the ensemble of M estimates at time t. Our algorithm proceeds as follows. First, we initialize $(\tilde{U}_{0,m}, \tilde{V}_{0,m})$, $m = 1, \ldots, M$, independently from the prior distribution. During each time period t, we sample a model uniformly from the M models and apply the greedy action according to the sampled model (line 3-4). After we observe the reward r_t, we perturb it independently for each model (line 7-8), and we update each model m by approximately solving the following optimization problem:

$$\min_{U,V} \frac{1}{\sigma_w^2} \sum_{\tau=0}^{t} (\tilde{r}_{\tau,m} - U_{i_\tau}^\top V_{j_\tau})^2 + \frac{1}{\sigma_U^2} \sum_{i=1}^{K} \|U_i - \tilde{U}_{0,m,i}\|_2^2 + \frac{1}{\sigma_V^2} \sum_{j=1}^{L} \|V_j - \tilde{V}_{0,m,j}\|_2^2 \quad (3)$$

We observe that (3) is very similar to (1), except that we use a perturbed dataset and regularize towards the prior sample $(\tilde{U}_{0,m}, \tilde{V}_{0,m})$.

Algorithm 1 does not rely on the specific choice of optimization methods for solving (3). We use ALS in our numerical experiments in Section 4, but other optimization methods are also applicable. The computational complexity of this algorithm depends primarily on how we update the ensemble. Note that in our algorithm, all the models can be updated in parallel. Further, for applications that require fast decision making, we may solve (3) incrementally. For example, if ALS is used, we may choose only to update a few rows of U or V.

4 COMPUTATIONAL RESULTS

We compare Algorithm 1 with a baseline algorithm that first purely explores and then purely exploits. The baseline algorithm is motivated by the matrix completion literature [3, 5, 6, 13], which offers various guarantees under suitable conditions that we are able to recover the unknown matrix with high accuracy if the sample size is approximately on the order of $O((K+L)d)$. Thus, a natural baseline algorithm to consider is one where the agent first samples some multiple of $(K+L)d$ entries uniformly at random (exploration), and then performs matrix completion. Then, it always selects the largest entry of the completed matrix (exploitation). We call this algorithm the explore-then-exploit algorithm.

In the following sections, we present computational results on synthetic datasets, a digital marketing dataset, and a MovieLens dataset.

4.1 Synthetic datasets

We generate the expected reward matrix $R = UV^\top$ by sampling $U_i \sim N(\mathbf{1}, I_d)$ and $V_j \sim N(\mathbf{1}, I_d)$ independently for $i = 1, \ldots, K$, $j = 1, \ldots L$. The observation noise w_t is sampled i.i.d. from $N(0, 1)$. All results are averaged over 500 to 5000 trials. In our implementation of Algorithm 1, the minimization problem (3) is solved using ALS in an incremental fashion as described in Section 3. The perturbation noise is sampled from the same distribution as w_t, except for the scaling experiment in Figure 2(d).

In Figure 1, we compare the cumulative regret of Algorithm 1 with the baseline algorithm on matrices of size 10×20 and rank 2. Algorithm 1 is run with different ensemble sizes, and the baseline is run with different sample sizes. We see that Algorithm 1, which acquires data more intelligently than uniform sampling, outperforms

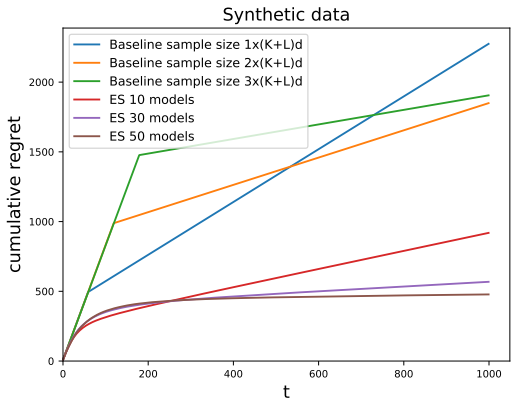


Figure 1: Experiment results on synthetic datasets.

the baseline by a large margin. In Figure 2, we show simulation results on how the cumulative regret of Algorithm 1 scales with the matrix dimension, rank, ensemble size, and perturbation magnitude. In Figure 2(a), we fix the rank $d = 2$ and ensemble size $M = 50$, and plot the cumulative regret of the algorithm at $T = 1000$ across different matrix dimensions. Figure 2(a) shows that the cumulative regret seems to scale linearly rather than quadratically with the matrix dimension, which implies that it effectively exploits the low-rank structure. In Figure 2(b), we plot the algorithm's cumulative regret on 10×20 matrices with $M = 30$ at $T = 1000$ across different values of matrix rank. The plot shows an approximate linear scaling as expected. Figure 2(c) and (d) illustrate the scaling of cumulative regret with respect to two key algorithm parameters, the number of models M and the perturbation magnitude $\sigma_{\tilde{w}}^2$. As is expected, increasing M will improve the performance of Algorithm 1. On the other hand, an overly small $\sigma_{\tilde{w}}^2$ will lead to insufficient exploration, while an overly large $\sigma_{\tilde{w}}^2$ will lead to over-exploration.

4.2 Digital Marketing datasets

We use the open-to-send datasets from a major digital marketing company, which contain data from email campaigns for 28 countries (rows) and 36 products (columns) over the years 2015 and 2016. For each year, we construct an expected reward matrix $R \in \Re^{28\times 36}$ where the entries are the email open rate for each country-product pair. We assume that the agent (marketer) will receive Bernoulli rewards based on the open rate. For both years 2015 and 2016, R can be well-approximated by a rank 1 matrix.

Figure 3 compares Algorithm 1 with the explore-then-exploit baseline, as well as the rank-1 elimination algorithm introduced in [11]. It is worth noting that the rewards in this setting are Bernoulli and do not follow the assumption used to derive Algorithm 1 that the observation noise is Gaussian. Yet, our algorithm still performs considerably better than the baseline algorithms.

4.3 MovieLens dataset

Based on the MovieLens dataset [7], we construct an expected reward matrix $R \in \Re^{128\times 128}$, whose rows represent user groups and columns represent movies. Each entry (i, j) represents the average

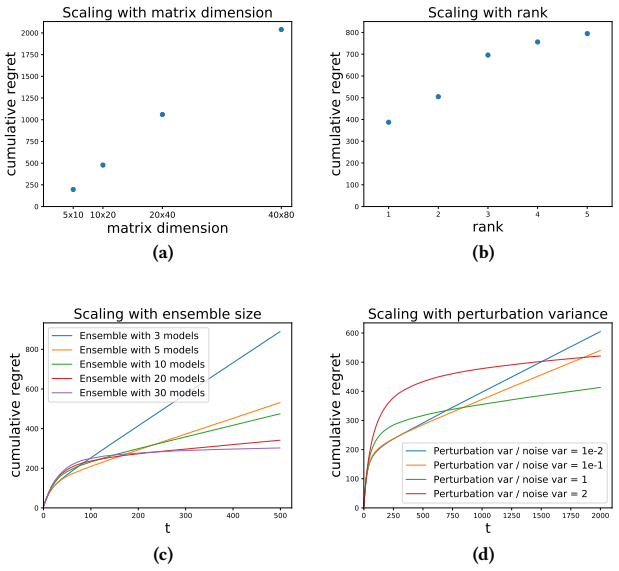


Figure 2: Scaling of cumulative regret of Algorithm 1 with the matrix dimension (a), rank (b), ensemble size (c), and perturbation magnitude (d).

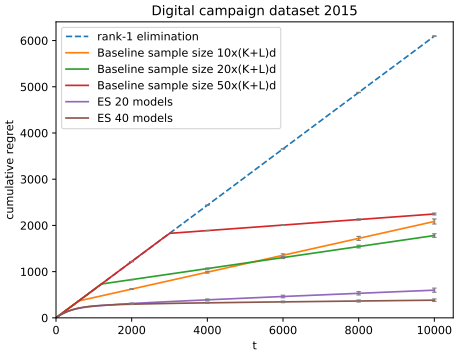
rating of movie j by user group i. The user and item biases have been removed from R and the resulting matrix has approximately rank 5. The agent will receive rewards (movie ratings) taking values in $\{1, 2, 3, 4, 5\}$.

In Figure 4, we compare the cumulative regret of Algorithm 1 and the explore-then-exploit baseline. Both algorithms are run across several different internal representations, where different rank approximations are used. In addition to significantly outperforming the baseline, Algorithm 1 also seems more robust to rank misspecification in this experiment.

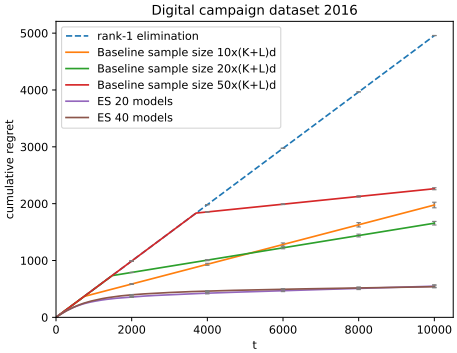
5 LITERATURE REVIEW AND CONCLUSION

Low-rank bandits have been studied in [10–12, 14, 15]. Similar to our problem, [10, 11, 14] also consider the setting where the agent chooses both the row and column. However, [10, 11] focus on the special rank-1 case, and [14] is restricted to the case with *hott topics* R. In contrast, we do not impose any additional assumptions on R other than the low-rank assumption. Moreover, the algorithms proposed in [10, 11, 14] are elimination-based algorithms, and the algorithm proposed in [14] has a computational complexity exponential in d. Our proposed algorithm is a computationally efficient approximate Thompson sampling algorithm, and experiment results show that it achieves order-of-magnitude improvement over the baselines, including the rank-1 elimination algorithm proposed in [11].

In this paper we assume that the agent chooses both the row i_t and column j_t. Some other literature [12, 15, 23, 24] study a variant of the low-rank bandit in which nature chooses the row i_t, and the agent chooses the column j_t. Note that our algorithm can be easily adapted to this setting. Specifically, we only need to change



(a) 2015 Digital Marketing Dataset



(b) 2016 Digital Marketing Dataset

Figure 3: Experiment results on Digital marketing open-to-send datasets

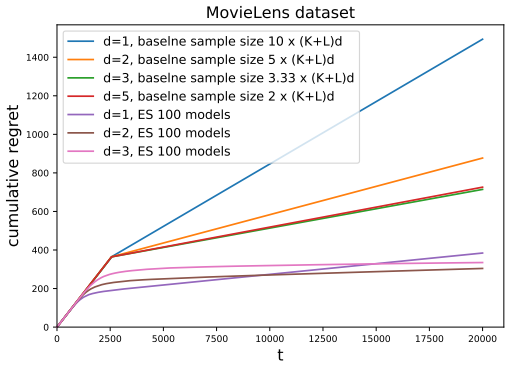


Figure 4: Experiment results on MovieLens dataset.

Line 4 of Algorithm 1 to $j_t = \arg\max_j \tilde{U}_{t,\hat{m},i_t}^\top \tilde{V}_{t,\hat{m},j}$. We leave the comparison with existing algorithms in this setting to future work. Another possible extension of this paper is to generalize our algorithm from the low-rank matrix case to the low-rank tensor case.

REFERENCES

[1] S. Agrawal and N. Goyal. 2013. Thompson Sampling for Contextual Bandits with Linear Payoffs. In *Proceedings of The 30th International Conference on Machine Learning*. 127–135.

[2] Samuel Burer and Renato D.C. Monteiro. 2005. Local Minima and Convergence in Low-Rank Semidefinite Programming. *Mathematical Programming* 103, 3 (01 Jul 2005), 427–444.

[3] Emmanuel J. Candès and Benjamin Recht. 2009. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics* 9, 6 (03 Apr 2009), 717.

[4] Olivier Chapelle and Lihong Li. 2011. An Empirical Evaluation of Thompson Sampling. In *Advances in Neural Information Processing Systems 24*. 2249–2257.

[5] M. A. Davenport and J. Romberg. 2016. An Overview of Low-Rank Matrix Recovery From Incomplete Observations. *IEEE Journal of Selected Topics in Signal Processing* 10, 4 (June 2016), 608–622.

[6] D. Gross. 2011. Recovering Low-Rank Matrices From Few Coefficients in Any Basis. *IEEE Trans. Inf. Theor.* 57, 3 (March 2011), 1548–1566.

[7] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages.

[8] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. 2013. Low-rank Matrix Completion Using Alternating Minimization. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing (STOC '13)*. ACM, New York, NY, USA, 665–674.

[9] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. 2010. *Recommender Systems: An Introduction* (1st ed.). Cambridge University Press, New York, NY, USA.

[10] Sumeet Katariya, Branislav Kveton, Csaba Szepesvári, Claire Vernade, and Zheng Wen. 2017. Bernoulli Rank-1 Bandits for Click Feedback. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. 2001–2007.

[11] Sumeet Katariya, Branislav Kveton, Csaba Szepesvari, Claire Vernade, and Zheng Wen. 2017. Stochastic Rank-1 Bandits. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, Fort Lauderdale, FL, USA.

[12] Jaya Kawale, Hung H Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla. 2015. Efficient Thompson Sampling for Online Matrix-Factorization Recommendation. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 1297–1305.

[13] R. H. Keshavan, A. Montanari, and S. Oh. 2010. Matrix Completion From a Few Entries. *IEEE Transactions on Information Theory* 56, 6 (June 2010), 2980–2998.

[14] Branislav Kveton, Csaba Szepesvári, Anup Rao, Zheng Wen, Yasin Abbasi-Yadkori, and S. Muthukrishnan. 2017. Stochastic Low-Rank Bandits. *CoRR* abs/1712.04644 (2017). http://arxiv.org/abs/1712.04644

[15] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. 2016. Collaborative Filtering Bandits. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, New York, NY, USA, 539–548.

[16] Xiuyuan Lu and Benjamin Van Roy. 2017. Ensemble Sampling. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 3258–3266.

[17] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (Eds.). Curran Associates, Inc., 1257–1264.

[18] Benjamin Recht and Christopher Ré. 2013. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation* 5, 2 (01 Jun 2013), 201–226.

[19] Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. 2018. A Tutorial on Thompson Sampling. *CoRR* (2018).

[20] D. Russo and B. Van Roy. 2014. Learning to optimize via posterior sampling. *Mathematics of Operations Research* 39, 4 (2014), 1221–1243.

[21] Daniel Russo and Benjamin Van Roy. 2016. An Information-theoretic Analysis of Thompson Sampling. *J. Mach. Learn. Res.* 17, 1 (Jan. 2016), 2442–2471.

[22] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian Probabilistic Matrix Factorization Using Markov Chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*. ACM, New York, NY, USA, 880–887.

[23] Rajat Sen, Karthikeyan Shanmugam, Murat Kocaoglu, Alexandros G. Dimakis, and Sanjay Shakkottai. 2016. Latent Contextual Bandits: A Non-Negative Matrix Factorization Approach. *CoRR* abs/1606.00119 (2016).

[24] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. 2013. Interactive Collaborative Filtering. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management (CIKM '13)*. 1411–1420.