Learning to Interact with Users: A Collaborative-Bandit Approach*

Konstantina Christakopoulou and Arindam Banerjee†

Abstract

Learning to interact with users and discover their preferences is central in most web applications, with recommender systems being a notable example. From such a perspective, merging interactive learning algorithms with recommendation models is natural. While recent literature has explored the idea of combining collaborative filtering approaches with bandit techniques, there exist two limitations: (1) they usually consider Gaussian rewards, which are not suitable for implicit feedback data powering most recommender systems, and (2) they are restricted to the one-item recommendation setting while typically a list of recommendations is given. In this paper, to address these limitations, apart from Gaussian rewards we also consider Bernoulli rewards, the latter being suitable for dyadic data. Also, we consider two user click models: the one-item click/no-click model, and the cascade click model which is suitable for top-K recommendations. For these settings, we propose novel machine learning algorithms that learn to interact with users by learning the underlying parameters collaboratively across users and items. We provide an extensive empirical study, which is the first to illustrate all pairwise empirical comparisons across different interactive learning algorithms for recommendation. Our experiments demonstrate that when the number of users and items is large, propagating the feedback across users and items while learning latent features is the most effective approach for systems to learn to interact with the users.

1 Introduction

Learning to interact with users is at the core of many web applications, ranging from search engines, to recommender systems, and more [15]. The reason is that these systems exhibit the user-system interaction loop: every time a user uses the system, the system has decided to show a list of items to the user, from which the user selects zero, one, or more items; then, the system gets the user's feedback (in terms of clicks, time spent, and other signals) to update its model of the user's preferences – which might affect the system's future decisions on this and other users.

This inherently interactive nature of web systems creates the need for machine learning models that learn how to interact with users over time; instead of static models trained based on offline data, that are periodically retrained to incorporate newly acquired information. Such interactive learning models are largely composed of two parts: which technique is used to balance the need for exploring user preferences vs. exploiting what has been learned so far, and what are the underlying model assumptions for the user reward?

In this work, inspired by recent works which pose recommendation as an interactive learning problem [17,20,22,36], we build on two good ideas, corresponding to each of these two parts. First, given that the recommender system sequentially learns about its users and items from repeated interactions, we make use of the construction of multi-armed bandits, i.e., a well-known tool used to accumulate rewards from an unknown reward distribution, whose parameters are to be sequentially learned [27] – we particularly focus on Thompson Sampling thanks to its good empirical guarantees [7]. Second, the collaborative filtering principle, i.e., similar users tend to like different items similarly, is a powerful tool which can be used to inform the underlying interactive learning models. By merging the two ideas, one can develop algorithms which learn to effectively interact with users, by propagating the user feedback across users and items. The construction, although illustrated in the context of recommender systems, can be used in other web systems where personalization plays a key factor, ranging from medical interventions, to personalized search engines, and more.

Particularly, our contributions are three-fold:

1. As most user interaction data are implicit and are an important signal for learning recommendations [14, 16], we develop new algorithms well-suited for Bernoulli rewards (Sections 4.1, 4.2).
2. Establishing the connection among one-item and cascade list recommendation algorithms (Section 3), we offer the novel construction of *collaborative cascade ranking bandits* (Section 4.2).
3. Empirical evaluation carefully considers both the

*The research was supported by NSF grants IIS-1563950, IIS-1447566, IIS-1447574, IIS-1422557, CCF-1451986, CNS-1314560, IIS-0953274, IIS-1029711, NASA grant NNX12AQ39A, and gifts from Adobe, IBM, and Yahoo.

†Department of Computer Science & Engineering, University of Minnesota, Twin Cities. {christa, banerjee}@cs.umn.edu

one-item and the cascade list setup, under both Gaussian and Bernoulli rewards; as far as we know, this is the first study making all such pairwise comparisons (Section 5).

Our extensive experiments show that for both the one-item and cascade list setup, when the number of users and items in the system is large, (1) collaborative bandits outperform running individual bandits per user; and (2) collaboratively learning across users indeed outperforms the contextual linear and clustering principle in the interactive learning setting.

2 Key Concepts & Overview

We start with a discussion of some useful concepts.

2.1 Collaborative Filtering. A widely used principle in recommender systems is *collaborative filtering*, which relies on the observation that similar users tend to have similar preferences over the items [26]. Latent factor-based collaborative filtering achieves state-of-the-art performance in recommender systems, by learning the latent preferences of users and items in a low dimensional space [25, 26]. If M is the number of users and N the number of items, the rating matrix $R \in \mathbf{R}^{M \times N}$ of users' ratings on items can be approximated by the inner product of two low dimensional latent factor matrices: $U \in \mathbf{R}^{M \times d}$ and $V \in \mathbf{R}^{N \times d}$, which represent the latent features of the users and items respectively.

2.2 Recommendation as Learning to Interact. Recommender systems sequentially learn the users' preferences based on repeated interactions with them. Let's assume that at each interaction round, a user, randomly drawn from the user population, comes to the system. Then, the system-user interaction model is characterized by two steps:

- **step 1:** the decision step, where the system decides to show an item (or list) to the user, on which the user gives feedback (reward) explicitly or implicitly; and
- **step 2:** the model update step, where the system updates its parameters using the user-provided feedback.

The system's goal is to select items so that maximum cumulative reward (or minimum cumulative regret) over the total T interaction rounds is achieved.

Given the sequential nature of the interactions, every user can be seen as a *multi-armed bandit* [27] – that is, for every user who interacts with the system the parameters underlying the user's reward model have to be sequentially learned.The *arms* or *actions* of the user-bandit are the candidate items for recommendation. In the rest of the paper, we will use the terms users/bandits and items/arms interchangeably. For a user $i \in \mathcal{U}$, let $\mathcal{L}_i$ be the set of active arms, which are the candidate items for recommendation (noting that the cardinality of $\mathcal{L}_i$ can be less than N). The reward of item j for a user i is denoted by r_{ij} and captures the feedback (e.g., click/no-click, buy/not-buy, rating) user i gives on item j. Rather than the reward matrix $R = [r_{ij}]$ being arbitrary, we will assume that R has a parametric form, and specific rewards r_{ij} depend on the user i, the item j, and the unknown parameters $\boldsymbol{\theta}^*$ (such an assumption will be empirically evaluated in Section 5).

2.3 User Reward Models. We assume that the rewards of the arms per bandit are stochastic [10]. This allows for a flexible setting where user feedback is noisy, as is the case in various recommendation scenarios, including news and content recommendation [2]. We consider the following two types of rewards:

- **Gaussian Rewards:** the reward of item j for user i is modeled with a Gaussian distribution $r_{ij} \sim \mathcal{N}(\mu^*_{ij}, \sigma^2_{ij})$, $r_{ij} = \mu^*_{ij} + \epsilon_{ij}$, where $\epsilon_{ij} \sim \mathcal{N}(0, \sigma^2_{ij})$ is the noise and μ^*_{ij} is the average rating of user i on j.
- **Bernoulli Rewards:** suited for settings where a user gives binary feedback (explicit, e.g., thumbs up/down on a song, or implicit e.g., click/skip on a news article). Let $\pi^*_{ij} \in [0, 1]$ be the probability that user i likes an item j. Then, we model the reward of user i on j with a Bernoulli distribution $r_{ij} = \text{Ber}(\pi^*_{ij})$, i.e., $r_{ij} = 1$ with probability π^*_{ij} and $r_{ij} = 0$ with probability $1 - \pi^*_{ij}$.

In both reward models, the reward of item $j \in \mathcal{L}_i$ for every user i is modeled based on a true parameter θ^*_{ij} (π^*_{ij} for Bernoulli rewards or μ^*_{ij} for Gaussian rewards). If we knew the true value of θ^*_{ij}, at every round the system would pick for incoming user i the item that maximizes the expected reward, $\max_j \mathbf{E}(r_{ij}|i, j, \theta^*_{ij})$.

2.4 Random Probability Matching Principle. Since we do not know the true value of the underlying parameters $\boldsymbol{\theta}^*$, we maintain a distribution over the parameters $P(\boldsymbol{\theta})$. If we wanted to maximize the immediate reward in a pure exploit setting, one would choose for user i the item that maximizes $\mathbf{E}[r_{ij}|i, j] = \int \mathbf{E}[r_{ij}|i, j, \theta_{ij}] P(\theta_{ij}|\cdot) d\theta_{ij}$. However, to balance the explore/exploit (EE) tradeoff, Thompson Sampling (TS) [30] uses the probability matching heuristic, and chooses for bandit i an action j according to its probability of being optimal [7], i.e., with probability:

$$\int \mathbb{1}\left[\mathbf{E}[r_{ij}|i, j, \theta_{ij}] = \max_{j' \in \mathcal{L}_i} \mathbf{E}[r_{ij'}|i, j', \theta_{ij'}]\right] P(\theta_{ij}|\cdot) d\theta_{ij},$$

where $\mathbb{1}[\cdot]$ is the indicator function, the $\mathbf{E}$ inside the integral denotes expectation over the stochastic nature of the rewards, and the integral denotes expectation over the parameter distribution. The integral can be approximated by drawing for every arm j a sample from

Algorithm 1 TS-based One-Item Recommendation

Require: hyper-parameters of the prior $P^0(\theta)$.
for round $t = 1, \ldots, T$ **do**
  User $i \sim \text{Uniform}(1, M)$ comes to the system.
  for $j = 1, \ldots, |\mathcal{L}_i|$ **do**
    Draw $\tilde{\theta}_{ij} \sim$ Posterior distribution $P(\theta_{ij}|\mathcal{D}, \cdot)$.
  end for
  $\forall j$: estimate reward as a function of the sampled parameters $\tilde{r}_{ij} = f(\tilde{\theta}_{ij}, \cdot)$.
  Show the item $\hat{j} = \arg\max_j \tilde{r}_{ij}$.
  Observe reward $r_{i\hat{j}}$: (i) for Bernoulli: $\sim \text{Ber}(\theta^*_{i\hat{j}})$, (ii) for Gaussian: $\sim \mathcal{N}(\theta^*_{i\hat{j}}, \sigma^2_{ij})$.
  Update parameters θ.
end for

Algorithm 2 TS-based Cascade List Recommendation

Require: hyper-parameters of the prior $P^0(\theta)$.
for round $t = 1, \ldots, T$ **do**
  User $i \sim \text{Uniform}(1, M)$ comes to the system.
  for $j = 1, \ldots, |\mathcal{L}_i|$ **do**
    Draw $\tilde{\theta}_{ij} \sim$ Posterior distribution $P(\theta_{ij}|\mathcal{D}, \cdot)$.
  end for
  $\forall j$: estimate $\tilde{r}_{ij} = f(\tilde{\theta}_{ij}, \cdot)$, sort items based on $\tilde{r}_{ij}$, and show the top K items.
  Observe feedback on position C_t (at most 1 click).
  for $l = 0, \ldots, \min(C_t, K)$ **do**
    Update parameters for item $\hat{j}$ at position l.
  end for
end for

the corresponding posterior $P(\theta_{ij}|\cdot)$. TS chooses the arm with the largest drawn posterior sample.

Various EE strategies such as UCB [3], ϵ-greedy or TS have been used in the different works posing recommendation as a learning to interact problem [17, 20,22,36]. Throughout this work, we use TS as it can be efficiently implemented and it has been shown to have competitive empirical performance [7].[1]

2.5 Click Models.

- **One/no-click:** a single item is presented to the user and the user decides to click or skip (or to explicitly like or dislike). We interchangeably refer to this as *one-item*.
- **Cascade Model:** originally introduced to study web search behavior, it models the interaction of the user with a top-K list [9]. It assumes that the user examines the list top to bottom, and clicks one item as soon as they find an interesting one. The user decides whether to click on each item before moving to the next, and the click probability on each item is independent from the rest. Thus, if a user clicks on item j at position k (j_k), they dislike the items on the above $k-1$ positions, and ignore the items from $k+1$ onward. The probability of clicking item j_k is $c_{j_k} = r_{j_k} \prod_{l=1}^{k-1}(1 - r_{j_l})$, where r_j is the probability of clicking item j, and $1 - r_j$ is the skipping probability.

3 Learning to Interact with Users

Recommender systems that *learn* to interact with users balance the need to learn new information about the user (*explore*) while also focusing on what has already been learned about their preferences (*exploit*). They achieve this for a user i in the following way. They start with a prior distribution over the latent parameters of the reward distribution $p^0(\theta_{ij})$ for every item j. Typically, a conjugate prior of the likelihood $p(\mathcal{D}|\theta_{ij}, \cdot)$ is used, where $\mathcal{D}$ is the collection of the thus far observed rewards, so that the posterior distribution $p(\theta_{ij}|\mathcal{D}, \cdot) \propto p(\mathcal{D}|\theta_{ij}, \cdot)p^0(\theta_{ij})$ is of the same form as the prior. Next, for every item j the system gets a sample from the posterior $\tilde{\theta}_{ij} \sim P(\theta_{ij}|\mathcal{D}, \cdot)$, and shows the item that results in the largest reward $\tilde{r}_{ij}$, estimated as a function of $\tilde{\theta}_{ij}$. The user gives feedback on the shown item $\hat{j}$, and based on this feedback the system updates its belief about the underlying parameters of the reward distribution. In the next interaction round the updated posterior distribution becomes the new prior. The wider the posterior distribution, the more the system explores. As more feedback is collected, the posterior distribution $P(\theta_{ij}|\mathcal{D}, \cdot)$ becomes more peaked and shifts to better capture the mean of the true underlying parameters θ^*_{ij} – thus, the system exploits more.

One/no-click vs. Cascade list. Algorithms 1 and 2 show the discussed procedure for the one/no-click model and the cascade list model respectively. Comparing the two algorithms, it is easy to establish the connection among learning to interact algorithms for one-item and cascade list recommendation. The cascade list recommendation algorithms differ from the click/no-click ones in two aspects: (i) they select the top K items instead of just the top one, and (ii) the model updates its parameters based on skips/clicks for all items up until and including the clicked one; the model performs no update for the items after the click. Later in the paper, we will exploit this connection to create novel learning to interact algorithms for the cascade model.

3.1 Parametric Assumptions for User Rewards.

The various learning to interact recommendation algorithms differ only in their parametric assumptions for the reward distribution; specifically in the input prior

[1] Our goal is *not* to compare TS with other EE strategies, or to compare alternative ways of posterior sampling; rather, keeping TS as the EE strategy of our choice, we aim to compare the interactive learning recommendation algorithms.

distribution (line 0), the posterior distribution (line 4), the definition of reward as a function of the parameters θ (line 6), and the way they update the parameters (line 9). Some parametric assumptions are:

1. *Independent* [7]: The reward parameters of the items j, j' of a given user i are uncorrelated, i.e., $\theta^*_{ij} \neq \theta^*_{ij'}$. Also, the parameters for every user i are independent from the parameters of every other user $i' \in \mathcal{U}$.
2. *Contextual Linear* [20,21]: Every arm j is represented by a feature vector $\mathbf{x}_j \in \mathbf{R}^d$ representing the context. For a user i the rewards of the different arms are correlated via a common parameter vector $\boldsymbol{\theta}^*_i \in \mathbf{R}^d$ capturing how important each dimension of the context is: $\mathbf{E}[r_{ij}] = h(\boldsymbol{\theta}^{*T}_i \mathbf{x}_j)$, where for Gaussian rewards h is the identity function, while for Bernoulli rewards it is the sigmoid function $\sigma(\cdot) = \frac{1}{1+\exp(-\cdot)}$.
3. *Contextual Clustering* [12, 13, 18, 24]: This case assumes that there exist clusters of the user population which have similar rating behaviors over the items, and contextual features $\mathbf{x}_j \in \mathbf{R}^d$ are available for each item j. All users who belong to the same cluster c will share the same reward parameter vector $\boldsymbol{\theta}^*_c \in \mathbf{R}^d$: $\mathbf{E}[r_{ij}] = h(\boldsymbol{\theta}^{*T}_c \mathbf{x}_j)$. The parameters of cluster c are independent from those of any other cluster c'.
4. *Low-Rank CF* [8, 17, 36]: Using the CF view and particularly the low-rank assumption (Section 2.1), both users and items can be represented by latent feature vectors. Concretely, considering $\forall i \in \mathcal{U}$ $\mathbf{r}_i \in \mathbb{R}^N$ the vector of true underlying rewards, we assume the reward matrix $R \in \mathbb{R}^{M \times N}$ is drawn from a parameter matrix Θ^* which is a function of a low-rank matrix. Thus, if $\mathbf{u}^*_i$ denotes the i-th row of the true latent user matrix U^* and $\mathbf{v}^*_j$ the j-th row of the true underlying latent item matrix V^*, $\mathbf{E}[r_{ij}] = h(\mathbf{u}^{*T}_i \mathbf{v}^*_j)$.

Note that with the independent assumption, if $\forall i \in \mathcal{U}, |\mathcal{L}_i| = N$, $M \times N$ parameters need to be learned. In contrast, the other parametric models do parameter sharing, coupling the items (and the users/clusters). Contextual linear and clustering models assume that contextual features are available and informative, which is not always true. Also, while both clustering and CF models share parameters among users with similar rating patterns, the former capture coarser user preferences represented by the clusters, whereas the latter can collaboratively learn finer grained latent preferences.

3.2 Existing Algorithms & Outline. For *One-Item Recommendation:* As far as we know, the independent assumption has been employed only within the context of a *single* multi-armed bandit [7], not as a baseline in multiple user-bandits for recommender systems – we have derived `Gauss/Bernoulli Independent TS`. Using the contextual linear principle has led to the pioneering work for applying bandits on news recommendation, resulting in `LinTS` and `LogTS` for Gaussian and Bernoulli rewards respectively [20, 21]. The contextual clustering principle has been applied for Gaussian rewards, resulting in `CluLinTS` (in Section 5 we experiment with the variant of [24], but other sophisticated variants exist [12, 13, 18]) – we have also devised `CluLogTS` for Bernoulli rewards. The CF low-rank principle has been applied on interactive recommenders for Gaussian rewards, giving rise to `Gauss Low Rank TS` [17] (also, the CF principle has been applied using co-clustering instead of matrix factorization in [22]). In Section 4.1, we will offer collaborative low-rank bandits for Bernoulli rewards as well. Its predecessor was `Gauss/ Bernoulli ICF`, standing for Interactive Collaborative Filtering, where the items' latent factors V are pre-learned and used as contextual features, allowing to pose the problem in a contextual linear setting [36].

For the *Cascade setup*, the independent assumption has been applied, resulting to `Gauss/Bernoulli Independent Cascade TS` [19]. Also, the contextual linear principle has been used, leading to `CascadeLinTS` [37] for Gaussian feedback – we have also devised `CascadeLogTS` for Bernoulli rewards. On the one hand, these methods are suitable for large-scale recommendation data, since the independent assumption does not scale well when the number of items candidate for recommendation is large (as is usually the case) [37]. On the other hand, they suffer from the following drawbacks: they rely on contextual features, and they do not propagate feedback among users, thus neglecting that similar users tend to like the various items similarly. In Section 4.2, we will apply the low-rank CF assumption, giving rise to the novel construction of *collaborative low-rank cascade bandits*. Also, we have devised `Gauss/ Bernoulli ICF Cascade TS`; specifically we have formulated the problem as a separate `CascadeLinTS` or `CascadeLogTS` per user, and used as contextual features the pre-learned latent attributes of the items.[2]

The parameter updates of the existing and novel interactive recommendation algorithms are in the supplementary material.[3]

4 Learning to *Collaboratively* Interact with Users: Proposed Algorithms

In this section we proceed with our novel contributions on learning to interact with users by *collaboratively*

[2] We omit the development of clustering cascade bandits, as early experiments showed the value of low-rank.

[3] `https://www-users.cs.umn.edu/~chri3275/suppl-230-sdm-2018.pdf`

learning parameters across them.

In our formulations, we build on two good ideas: (1) collaborative filtering and (2) interactive learning. The merge of these two ideas is key for efficiently propagating feedback among users, while at the same time learning the low dimensional embeddings of users and items on the same underlying latent space. The construction is suited for scenarios where contextual features are not explicitly given, and the fine-grained similar rating patterns are to be discovered collaboratively.

Although the combination of the two ideas has been explored in [8,17,22], we propose two technical developments: (1) We develop collaborative *bernoulli* bandits referred to as `Bernoulli Low Rank TS`, considering the popular logistic matrix factorization model [16] which is successful for modeling click/no-click data, which are widely available in recommender systems. This is useful as most recommendation algorithms for top-K recommendation are powered from implicit type data [14]. (2) Having realized the connection among the one-item and cascade setup (Section 3), we develop collaborative interactive learning algorithms for the cascade setting both for Gaussian and Bernoulli user reward models.

4.1 One-Item Bernoulli Collaborative Bandits.

In this setting, we use the low-rank CF assumption to couple the probabilities $\{\pi_{ij}\}$ across users $\{i\}_{i=1}^{M}$ and items $\{j\}_{j=1}^{N}$. We assume that every entry π_{ij} of the matrix $\Pi \in \mathbb{R}^{M\times N}$ (i.e., the probabilities of how much each user likes every item) will be estimated by the sigmoid function of the inner product of $\mathbf{u}_i$ and $\mathbf{v}_j$. The sigmoid function is used to map the entries to the range of [0, 1] as they represent probabilities. Particularly, `Bernoulli Low Rank TS` assumes that $\mathbf{E}[r_{ij}] = \sigma(\mathbf{u}_i^T\mathbf{v}_j) = 1/(1+\exp(-\mathbf{u}_i^T\mathbf{v}_j))$. The model considered for parameter estimation follows the modeling assumptions of probabilistic logistic matrix factorization [16] and is also related to [11]. At step t:

$$\forall i: \quad \mathbf{u}_i^t \sim \mathcal{N}(\hat{\mathbf{u}}_i^{(t-1)}, S_{\mathbf{u}_i}^{t-1}) \quad \forall j: \quad \mathbf{v}_j^t \sim \mathcal{N}(\hat{\mathbf{v}}_j^{(t-1)}, S_{\mathbf{v}_j}^{t-1})$$
$$\forall ij: \ \pi_{ij}^t = \sigma({\mathbf{u}_i^t}^T\mathbf{v}_j^t), \ r_{ij}^t \sim \text{Bernoulli}(\pi_{ij}^t)$$

The log of the posterior distribution over the user and the item latent features U, V is:

$$\begin{aligned}
&\sum_i \sum_{\hat{j}} r_{i\hat{j}}^t \ln \pi_{i\hat{j}}^t + (1-r_{i\hat{j}}^t)\ln(1-\pi_{i\hat{j}}^t) \\
&- \frac{1}{2}\sum_{i=1}^{M}(\mathbf{u}_i^t - \hat{\mathbf{u}}_i^{(t-1)})^T {S_{\mathbf{u}_i}^{(t-1)}}^{-1}(\mathbf{u}_i^t - \hat{\mathbf{u}}_i^{(t-1)}) \\
&- \frac{1}{2}\sum_{j=1}^{N}(\mathbf{v}_j^t - \hat{\mathbf{v}}_j^{(t-1)})^T {S_{\mathbf{v}_j}^{(t-1)}}^{-1}(\mathbf{v}_j^t - \hat{\mathbf{v}}_j^{(t-1)}).
\end{aligned} \tag{4.1}$$

If we fix the latent item features V and consider a single user i, Equation (4.1) reduces to a logistic regression problem with parameter vector $\mathbf{u}_i$. Given that exact Bayesian inference for logistic regression is intractable, we use the Laplace approximation to approximate the conditional posterior of $\mathbf{u}_i$ with a Gaussian distribution $q(\mathbf{u_i}) = \mathcal{N}(\mathbf{u}_i|\hat{\mathbf{u}}_i, S_{\mathbf{u}_i}, \cdot)$, where $\hat{\mathbf{u}}_i = \mathbf{u}_i^{\text{MAP}}$ is the mode of the posterior and $S_{\mathbf{u}_i}$ is the Hessian matrix of second derivatives of the negative log posterior [4]. We compute the mode of the posterior by minimizing the negative log posterior with online gradient descent $\hat{\mathbf{u}}_i^{t+1} \leftarrow \hat{\mathbf{u}}_i^t - \eta_t \nabla_{\mathbf{u}_i}^t$:

$$\nabla_{\mathbf{u}_i}^t = {S_{\mathbf{u}_i}^{(t-1)}}^{-1}(\hat{\mathbf{u}}_i^t - \hat{\mathbf{u}}_i^{t-1}) + (\pi_{i\hat{j}}^t - r_{i\hat{j}}^t)\mathbf{v}_{\hat{j}}^t,$$

where the step size follows the Adagrad rule $\eta_t = 1/\sqrt{\sum_{\tau=0}^{t-1}\nabla \mathbf{u}_{i,\tau}^2}$. The inverse of $S_{\mathbf{u}_i}$, denoted as $S_{\mathbf{u}_i}^{-1}$, is constructed online as:

$$S_{\mathbf{u}_i}^{t+1^{-1}} = S_{\mathbf{u}_i}^{t^{-1}} + \pi_{i\hat{j}}^t(1-\pi_{i\hat{j}}^t)\mathbf{v}_{\hat{j}}^t{\mathbf{v}_{\hat{j}}^t}^T \ . \tag{4.2}$$

using moment matching [4]. Similarly, we approximate the posterior distribution of the latent feature of items $q(\mathbf{v}_j) = \mathcal{N}(\mathbf{v}_j|\hat{\mathbf{v}}_j, S_{\mathbf{v}_j}, \cdot)$. In practice, to efficiently compute the inverse of $S_{\mathbf{u}_i}, S_{\mathbf{v}_j}$, we used the Woodburry matrix identity [34].

Concretely, `Bernoulli Low Rank TS` proceeds as follows: At every round that user i interacts with the recommender system, we sample $\tilde{\mathbf{u}}_i$ from the posterior $\mathcal{N}(\mathbf{u}_i|\hat{\mathbf{u}}_i, S_{\mathbf{u}_i}, \cdot)$, and for every arm $j \in \mathcal{L}_i$, we sample $\tilde{\mathbf{v}}_j$ from the posterior $\mathcal{N}(\mathbf{v}_j|\hat{\mathbf{v}}_j, S_{\mathbf{v}_j}, \cdot)$. We show the item $\hat{j} = \arg\max_j \tilde{\mathbf{u}}_i^T\tilde{\mathbf{v}}_j$ (as sigmoid is a monotonic function). The user provides feedback $r_{i\hat{j}}$. Based on this feedback, we update $\hat{\mathbf{u}}_i, S_{\mathbf{u}_i}, \hat{\mathbf{v}}_{\hat{j}}, S_{\mathbf{v}_{\hat{j}}}$, thus updating the posteriors. At the next round, the posteriors become the new priors.

4.2 Collaborative Cascade Ranking Bandits.

Here, we propose a novel algorithm assuming the low-rank CF principle for the cascade list recommendation setting. Recall that the cascade list setting is suitable for systems learning to interact with users in a top-K list recommendation setup (Section 2.5). We model the reward of an item j for user i with $\mathbf{E}[r_{ij}] = \mathbf{u}_i^T\mathbf{v}_j$ for Gaussian rewards, and $\mathbf{E}[r_{ij}] = \text{sigmoid}(\mathbf{u}_i^T\mathbf{v}_j)$ for Bernoulli rewards similarly to Section 4.1.

`Gauss/Bernoulli Low Rank Cascade TS` proceeds: At every user i-system interaction round we sample $\tilde{\mathbf{u}}_i \sim \mathcal{N}(\mathbf{u}_i|\hat{\mathbf{u}}_i, S_{\mathbf{u}_i}, \cdot)$, and $\tilde{\mathbf{v}}_j \sim \mathcal{N}(\mathbf{v}_j|\hat{\mathbf{v}}_j, S_{\mathbf{v}_j}, \cdot)$ $\forall$ candidate item j. Sorting the estimated rewards $\tilde{r}_{ij} = \tilde{\mathbf{u}}_i^T\tilde{\mathbf{v}}_j$, we present the list of items with the top K rewards. We observe the user's click on an item from the top-K list (if any), and update $\hat{\mathbf{u}}_i, S_{\mathbf{u}_i}, \hat{\mathbf{v}}_{\hat{j}}, S_{\mathbf{v}_{\hat{j}}}$ *for every item* $\hat{j}$ *at position* $\leq$ *the position of the click*

using as feedback $r_{i\hat{j}} = 0$ for the skipped items, and $r_{i\hat{j}} = 1$ for the clicked item.

This new collaborative cascade ranking formulation does not need contextual features, and can learn the underlying implicit structure among users and items on the fly. This is what allows for effective propagation of the feedback across items and users via the learned embedding; and can lead to improved regret performance.

5 Experimental Results

We present our experiments with the goal of addressing the following questions:

RQ1 For one-item recommendation, which TS-based interactive recommendation algorithm is the most effective?

RQ2 For top-K cascade list recommendation, how collaborative cascade ranking bandits compare to the other learning to interact algorithms?

5.1 Experimental Procedure. For each question we consider both Gaussian rewards and Bernoulli rewards, using the corresponding set of algorithms.[4]

Methods. We compare ten algorithms for the click/no-click setup – five for Gaussian and five for Bernoulli; and eight algorithms for the cascade click setup – four for Gaussian and four for Bernoulli rewards. Additionally, we included the baseline strategies of: (i) `Random`, an explore-only strategy, which at every round, selects for the incoming user i a random item from the set of active arms $\mathcal{L}_i$, and (ii) `Low Rank (LR) Greedy`, an exploit-only low-rank CF strategy, which at every round, instead of sampling from the distributions $P(\mathbf{u}_i|\hat{\mathbf{u}}_i, S_{\mathbf{u}_i}, \cdot)$, $P(\mathbf{v}_{\hat{j}}|\hat{\mathbf{v}}_{\hat{j}}, S_{\mathbf{v}_{\hat{j}}}, \cdot)$, operates according to the distribution means $\hat{\mathbf{u}}_i, \hat{\mathbf{v}}_{\hat{j}}$, following precisely [23].

Evaluation Metric. We evaluate the algorithms in terms of *cumulative regret*, as learning to interact algorithms aim to minimize the difference between their actual performance and the optimal performance [27]. Formally, the cumulative regret over T rounds is $\mathcal{R}_T = \mathbf{E}[\sum_{t=1}^{T} r_{\mathbf{A}^*,\theta^*} - r_{\mathbf{A}_t,\theta_t})]$, where $\boldsymbol{A}^*$ is the *optimal* item/list maximizing the reward at any time t, $\boldsymbol{A}_t$ what the system actually showed at round t, and r denotes the reward. Smaller values of $\mathcal{R}_T$ are better, with 0 indicating optimal system performance.

Datasets. We used real recommendation data, whose statistics are shown in Table 1. Particularly, we considered three movie datasets: Movielens 100K, Movielens 1M, Netflix and a musical artist dataset from Yahoo!. For the Yahoo! dataset we first selected the 1,000 most popular items and then the 6,000 most heavy users, re-scaling the ratings to the scale of 1-6.

[4] A comparison among the two sets though is not applicable, due to the separate reward generation procedures.

Dataset	# Users	# Items	# Ratings
Movielens 100K	943	1,682	100,000
Movielens 1M	6,040	3,706	1,000,209
Netflix	5,000	500	1,922,815
Yahoo!	6,000	1,000	3,522,232

Table 1: Real Data Statistics

Similarly for the Netflix dataset, we considered a dense sub-matrix containing the 500 most popular items and the 5,000 most heavy users.

Offline to Interactive Setup. Evaluating learning to interact algorithms on offline datasets, is a known issue and an active research topic [21]. Most public recommendation data, including the ones considered in Table 1, contain ratings on user-item interactions without giving access to the counter-factual response, i.e., how every user would have interacted with other items, had they been shown to them. Although rejection sampling and importance sampling techniques have been proposed, they typically require large-scale exploration data [20,21]; however, the collected recommendation data are far from being purely exploratory. Thus, adopting the setup used in other interactive recommendation works (e.g., [17]), we incrementally showed entries of the observed rating matrix as follows: At the 0-th interaction round, no entry of the reward matrix R has been revealed. At every round t, we randomly sample one of the users present in the data i_t to interact with the system; this can be a user the system has already interacted with in previous rounds (warm-start), or a new one (cold-start). We consider as the active set of arms for user i, i.e., $\mathcal{L}_i$, only the items he has rated in the original dataset. The under evaluation algorithm decides which arm(s)/item(s) to recommend to the user.

Feedback Simulation. We simulate the user's feedback on the shown item $\hat{j}^t$ using the two reward models discussed in Section 2.3. For this, we need access to (i) the true underlying parameters $\boldsymbol{\theta}^*$ and (ii) for Gaussian rewards, the noise level. For (i), we performed a transformation of the rating values of the original datasets. The original Movielens 100K and 1M contain ratings in a scale of 1 to 5 stars, while Netflix and Yahoo! have ratings in a 1 to 6 scale. To set the μ^*_{ij} of Gaussian rewards, we converted all ratings ≥ 4 to 1 (user likes the item), and < 4 to 0.01 (user dislikes the item). For Bernoulli rewards, we set the true user probability of liking an item π^*_{ij} by transforming the original ratings of scale $[1, 5$ or $6]$ to the scale $[0, 1)$ so as to represent probabilities. For this, we used the mapping $[4.5, 6] \to 0.9$, $[4, 4.5) \to 0.85$, $[3.5, 4) \to 0.8$, $[0.5, 3.5) \to 0.05$, $[0, 0.5) \to 0$ which indicates that the higher the rating, the more likely the user will click on the item. For (ii), for the one/no-click model, we set noise σ_{ij} to 0.5 $\forall i, j$; in contrast, for the cascade click

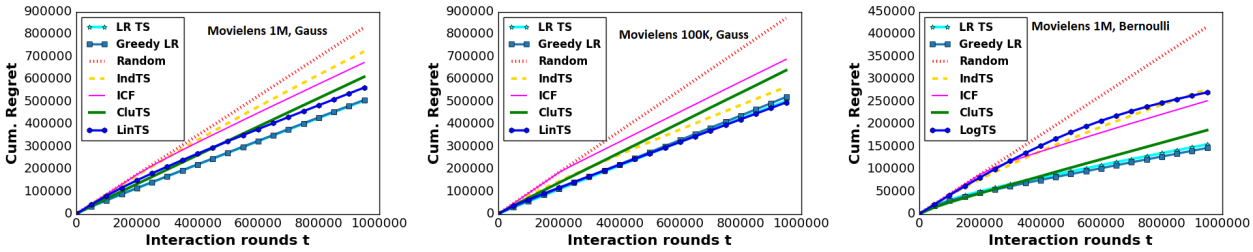

Figure 1: Comparison of learning to interact algorithms for the *One Item* setup w.r.t cumulative regret. The low-rank hypothesis achieves the best performance, except for Movielens 100K Gauss, where `LinTS` is the best.

model we set noise to 0 (1 is a click, and 0.01 a skip).

Parameter Setting. We set the number of interaction rounds T to 1,000,000. We initialize the parameters of the prior distributions as follows: For `Independent (Cascade) TS`, we set the prior to a broad one, i.e., $\mathcal{N}(0, 1)$. For `Bernoulli Low Rank (Cascade) TS`, we set the variance of the user and item latent features to 0.1, i.e., $\sigma_u^2 = \sigma_v^2 = 0.1$. For `Gauss Low Rank (Cascade) TS`, we set σ_u^2, σ_v^2 to 0.001. These choices were made as we found that larger variance values can lead to deteriorated performance. In any method assuming low-rank structure, a rank of 2 is used, similarly to [17]. For clustered bandits, the number of clusters is set to 10. In practice, one should do parameter tuning. In the supplementary material, we explore `Low Rank TS`'s sensitivity to the various parameters.

5.2 RQ1: Which interactive recommender is the best for one/no-click recommendation?

Figure 1 shows the performance of the learning to interact recommendation algorithms for the one/no-click setup. Results only for Movielens 1M (Gauss and Bernoulli) and Movielens 100K (Gauss) are shown for brevity.

Low rank vs. Independent. In all four datasets, both for Bernoulli and Gaussian rewards, `Low Rank TS` is (among) the best strategies. Particularly, for datasets with a larger number of observations (Movielens 1M, Netflix and Yahoo!), `Low Rank TS` seems to couple users and items in the low dimensional embedding effectively; thus making the value of the low rank representation more prominent. In these cases, `Low Rank TS` clearly outperforms `Independent TS` (`IndTS`), whose performance is close to `Random`. In contrast, for datasets with fewer observations (Movielens 100K), the gap between `Low Rank TS` and `Independent TS` closes.

Greedy vs. TS. For Gaussian rewards, for Movielens 100K and Movielens 1M `LR Greedy` performs closely to `Low Rank TS` – this is due to the small value of noise; while for larger scale datasets `Low Rank TS` is the best. For Bernoulli rewards, `LR Greedy` surpasses `Low Rank TS`, indicating that the low-rank part is more important than the exploration part. This happens because we set π_{ij}^* very close to the extremes of 0 and 1.

Latent context vs. Explicit context. Recall that `CluLinTS`, `LinTS`, `CluLogTS` and `LogTS` are the only algorithms which use the contextual information. We evaluate these algorithms' performance only for Movielens 100K and Movielens 1M, as these are the only datasets which contain explicit context. We used as contextual features the items' genre features, since it has been found that genre correlates well with the users' ratings for these datasets [29]. Every such feature is represented as an one-hot encoding of the genres characterizing the movie, and has dimension the total number of genres (19 for Movielens 100K and 18 for Movielens 1M). We can see that for Movielens 100K, the genre feature is informative enough, as after about 400,000 interaction rounds `LinTS` outperforms `Low Rank TS`. However, for the larger dataset of Movielens 1M, even though `Low Rank TS` does not take advantage of the explicit context information, it is able to surpass `LinTS` by learning both user and item features. For Bernoulli rewards, for both datasets, treating recommendation as a low-rank bandit problem instead of a contextual linear bandit is better.

Comparing the contextual algorithms, we see that for Bernoulli rewards, sharing feedback among the clusters outperforms having independent `LogTS` per user for Movielens 1M, whereas the opposite trend is found in Movielens 100K. For Gaussian rewards, in both datasets `LinTS` is better than `CluLinTS`, but the performance gap closes for the larger Movielens 1M.

Fixing V vs. learning V. Recall that `ICF` uses an independent `LinTS` per user using as contextual features pre-learned item latent features $\boldsymbol{V}$. Following [17], for `ICF` we learn $\boldsymbol{V}$ by performing online Gaussian/Logistic matrix factorization in the first 20%T of the interaction rounds, during which we randomly show an item while updating $\mathbf{v}_j$ and the user covariance matrix $S_{\mathbf{u}_i}$. After 20%T rounds, we fix $\mathbf{v}_j$ and we show the item $\hat{j} =$

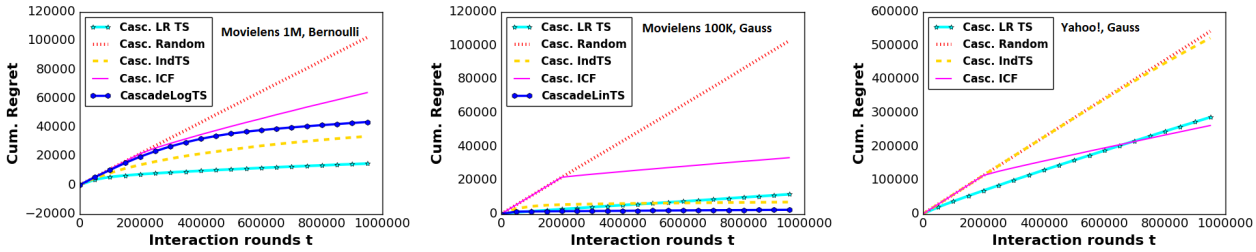

Figure 2: Comparison of learning to interact algorithms for the *Cascade List* setup in terms of cumulative regret. For larger datasets, the low rank principle for the cascade ranking setup achieves the smallest regret.

$\arg\max_j \tilde{\mathbf{u}}_i^T \mathbf{v}_j$. In all the datasets, learning the latent features of both users and items via `Low Rank TS` surpasses `ICF` both for Gaussian and Bernoulli rewards.

5.3 RQ2: Which interactive recommender is the best for cascade list recommendation? Figure 2 shows how collaborative cascade bandits compare to the rest cascade learning to interact algorithms. Based on results from all datasets (although only the performance for Movielens 1M Bernoulli, Movielens 100K Gauss, and Yahoo! Gauss are depicted due to space constraints), we observe the following:

Independent vs. Low Rank. When enough observations are available (Movielens 1M, Netflix, Yahoo!), `Low Rank Cascade TS` outperforms `Independent Cascade TS`. But when the number of these observations is small (Movielens 100K), `Independent Cascade TS` is better.

Latent context vs. Explicit context. For Movielens 100K, using the genre feature as context is enough to achieve quite good performance; particularly, for Gaussian rewards, `CascadeLinTS` is the best performing strategy. In contrast, in the larger dataset of Movielens 1M, learning on the fly the latent attributes of users and items via `Low Rank Cascade TS` results in a smaller regret compared to `CascadeLinTS`/`CascadeLogTS` for either reward setting.

We conclude that for both Bernoulli and Gaussian rewards, `Low Rank Cascade TS` is the best performing strategy for large-scale datasets. The only exception is the Yahoo! dataset for Gaussian rewards, where `ICF Cascade TS` surpasses `Low Rank Cascade TS`; which however again is a *low-rank* cascade bandit strategy.

6 Related Work

Here we highlight some pairwise comparisons among the various learning to interact for recommendation algorithms which have happened in the literature: (1) Li et al. devised contextual linear bandits and showed that their proposed algorithm `LinUCB` outperforms several baselines (e.g., popular, random, user segment-based algorithms) [20]. (2) The authors of [24] using the clustering parametric assumptions on top of `LinUCB`, devised `CluLinUCB` and showed that it outperforms having an independent `LinUCB` for each user, and performs similarly or better than the closely related baseline proposed in [13]. (3) Zhao et al. proposed `ICF`, posing recommendation as a contextual linear bandit with context the learned item features, and showed that it outperforms interview-based and active learning strategies with a few user-system interactions [36]. (4) In [17], Kawale et al. showed that combining TS with online matrix factorization, while maintaining particle filters to sample from the posterior, outperforms `ICF` as well as alternative ways of posterior sampling. A similar model was proposed for conversational recommender systems [8]. (5) In [22], Li et al.'s collaborative filtering bandits, which implement the CF view via co-clustering, were shown to surpass `LinUCB` and the clustering bandits in [6, 13]. Other relevant works are [12, 18, 32, 35], which all combine the clustering or CF principle with contextual features on the Gaussian rewards, one-item setup.

Other works balancing the EE tradeoff in recommendation are: [1] which gives bayesian schemes for serving content, [31] which combines Gaussian processes with EE, [5] which proposes an ϵ-greedy user-user neighbor-based CF method, [28] which gives an ϵ-greedy online matrix factorization method, and [33] which combines CF with TS and topic models.

7 Conclusions

In this paper, we tackled the problem of learning to interact with users, which is central in recommender systems, and made the following contributions:

- **Algorithms:** We (a) developed interactive learning recommendation algorithms suitable for *dyadic data*, and (b) offered the novel *collaborative cascade bandits*.
- **Experiments:** (a) We provided an extensive study making all pairwise comparisons among various learning to interact algorithms for recommendation: 10 al-

gorithms for one-item (5 for Bernoulli, 5 for Gaussian), and 8 algorithms for cascade list (4 for Bernoulli, 4 for Gaussian). (b) We showed that *for large-scale data, collaboratively learning to interact algorithms are the best performing ones.*

References

[1] D. Agarwal, B.-C. Chen, and P. Elango. Explore/exploit schemes for web content optimization. In *ICDM*, pages 1–10, 2009.

[2] D. Agarwal, B.-C. Chen, P. Elango, and R. Ramakrishnan. Content recommendation on web portals. *Communications of the ACM*, 56(6):92–101, 2013.

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

[4] A. Banerjee. On bayesian bounds. In *ICML*, pages 81–88, 2006.

[5] G. Bresler, G. H. Chen, and D. Shah. A latent source model for online collaborative filtering. In *NIPS*, pages 3347–3355, 2014.

[6] N. Cesa-Bianchi, C. Gentile, and G. Zappella. A gang of bandits. In *NIPS*, pages 737–745, 2013.

[7] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *NIPS*, pages 2249–2257, 2011.

[8] K. Christakopoulou, F. Radlinski, and K. Hofmann. Towards conversational recommender systems. In *KDD*, pages 815–824, 2016.

[9] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM*, pages 87–94, 2008.

[10] V. Dani, T. P. Hayes, and S. M. Kakade. Stochastic linear optimization under bandit feedback. In *COLT*, pages 355–366, 2008.

[11] M. A. Davenport, Y. Plan, E. van den Berg, and M. Wootters. 1-bit matrix completion. *Information and Inference*, 3(3):189–223, 2014.

[12] C. Gentile, S. Li, P. Kar, A. Karatzoglou, G. Zappella, and E. Etrue. On context-dependent clustering of bandits. In *ICML*, pages 1253–1262, 2017.

[13] C. Gentile, S. Li, and G. Zappella. Online clustering of bandits. In *ICML*, pages 757–765, 2014.

[14] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[15] D. Jannach, P. Resnick, A. Tuzhilin, and M. Zanker. Recommender systems—: beyond matrix completion. *Communications of the ACM*, 59(11):94–102, 2016.

[16] C. C. Johnson. Logistic matrix factorization for implicit feedback data. In *NIPS 2014 Workshop on Distributed Machine Learning and Matrix Computations*, 2014.

[17] J. Kawale, H. H. Bui, B. Kveton, L. Tran-Thanh, and S. Chawla. Efficient thompson sampling for online? matrix-factorization recommendation. In *NIPS*, pages 1297–1305, 2015.

[18] N. Korda, B. Szörényi, and L. Shuai. Distributed clustering of linear bandits in peer to peer networks. In *ICML*, pages 1301–1309, 2016.

[19] B. Kveton, C. Szepesvari, Z. Wen, and A. Ashkan. Cascading bandits: Learning to rank in the cascade model. In *ICML*, pages 767–776, 2015.

[20] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, pages 661–670, 2010.

[21] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*, pages 297–306, 2011.

[22] S. Li, A. Karatzoglou, and C. Gentile. Collaborative filtering bandits. In *SIGIR*, pages 539–548, 2016.

[23] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *JMLR*, 11(Jan):19–60, 2010.

[24] T. T. Nguyen and H. W. Lauw. Dynamic clustering of contextual multi-armed bandits. In *CIKM*, pages 1959–1962, 2014.

[25] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, pages 880–887, 2008.

[26] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, pages 1–8, 2011.

[27] S. L. Scott. A modern bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

[28] R. Sen, K. Shanmugam, M. Kocaoglu, A. G. Dimakis, and S. Shakkottai. Latent contextual bandits. *arXiv preprint arXiv:1606.00119*, 2016.

[29] H. Shan and A. Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *ICDM*, pages 1025–1030, 2010.

[30] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[31] H. P. Vanchinathan, I. Nikolic, F. De Bona, and A. Krause. Explore-exploit in top-n recommender systems via gaussian processes. In *RecSys*, pages 225–232, 2014.

[32] H. Wang, Q. Wu, and H. Wang. Factorization bandits for interactive recommendation. In *AAAI*, pages 2695–2702, 2017.

[33] Q. Wang, C. Zeng, W. Zhou, T. Li, L. Shwartz, and G. Y. Grabarnik. Online interactive collaborative filtering using multi-armed bandit with dependent arms. *arXiv preprint arXiv:1708.03058*, 2017.

[34] M. A. Woodbury. The stability of out-input matrices. *Chicago, IL*, 93, 1949.

[35] Q. Wu, H. Wang, Q. Gu, and H. Wang. Contextual bandits in a collaborative environment. In *SIGIR*, pages 529–538, 2016.

[36] X. Zhao, W. Zhang, and J. Wang. Interactive collaborative filtering. In *CIKM*, pages 1411–1420, 2013.

[37] S. Zong, H. Ni, K. Sung, N. R. Ke, Z. Wen, and B. Kveton. Cascading bandits for large-scale recommendation problems. *arXiv preprint arXiv:1603.05359*, 2016.