Learning User Preferences in Non-Stationary Environments

Wasim Huleihel
Tel-Aviv University

Soumyabrata Pal
University of Massachusetts Amherst

Ofer Shayevitz
Tel-Aviv University

Abstract

Recommendation systems often use online collaborative filtering (CF) algorithms to identify items a given user likes over time, based on ratings that this user and a large number of other users have provided in the past. This problem has been studied extensively when users' preferences do not change over time (static case); an assumption that is often violated in practical settings. In this paper, we introduce a novel model for online non-stationary recommendation systems which allows for temporal uncertainties in the users' preferences. For this model, we propose a user-based CF algorithm, and provide a theoretical analysis of its achievable reward. Compared to related non-stationary multi-armed bandit literature, the main fundamental difficulty in our model lies in the fact that variations in the preferences of a certain user may affect the recommendations for other users severely. We also test our algorithm over real-world datasets, showing its effectiveness in real-world applications. One of the main surprising observations in our experiments is the fact our algorithm outperforms other static algorithms even when preferences do not change over time. This hints toward the general conclusion that in practice, dynamic algorithms, such as the one we propose, might be beneficial even in stationary environments.

1 Introduction

Recommendation systems provide users with appropriate items based on their revealed preferences such as ratings. Due to their wide applicability, recommendation systems have received significant attention in machine learning and data mining societies [Si and Jin, 2003, Rennie and Srebro, 2005, Salakhutdinov and Mnih, 2007, Bell et al., 2007, Koren, 2008, Salakhutdinov and Mnih, 2008, Jahrer et al., 2010]. In practice, recommendation systems often employ *collaborative filtering* (CF) [Ekstrand et al., 2011], for recommending (potentially) liked items to a given user by considering items that other "similar" users liked. There are two main categories of CF algorithms: *user-based*, e.g., [Resnick et al., 1994, Herlocker et al., 1999, Bresler et al., 2014, Bellogin and Parapar, 2012, Heckel and Ramchandran, 2017], and *item-based*, e.g., [Gregory et al., 1998, Sarwar et al., 2001, Linden et al., 2003, Bresler et al., 2016], and many references therein. User-based algorithms exploit similarity in the user space by recommending a particular user u those items which were liked by other similar users. Item-based algorithms, in contrast, exploit similarity in the item space by recommending items similar to those which were liked in the past by a particular user.

Prevalent recommendation systems typically operate in an online fashion where items are recommended to users over time, and the obtained ratings are used for future recommendations. Typically, the goal in such a problem is to maximize the number of likes revealed at any time. This problem has been studied extensively, e.g., [Bresler et al., 2014, Bresler et al., 2016, Heckel and Ramchandran, 2017, Bresler and Karzand, 2019], always under the assumption that user's preferences do not vary over time. In practice, however, temporal changes in the structure of the user's preferences are an intrinsic characteristic of the problem, since users change their taste occasionally [Hariri et al., 2015, Liu et al., 2010, Moore et al., 2013, Hariri et al., 2014, Karahodza et al., 2014, Shi et al., 2015, Basile et al., 2015, Liu, 2015, Mukherjee et al., 2017, Eskandanian and Mobasher, 2018]. Ignoring these changes results in recommendation algorithms which are doomed to failure in practice. This sets the goal of this paper: *we aim to model and understand the effect of non-stationary environment on online recommendation systems.*

Proceedings of the 24[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

To that end, we introduce a novel latent probabilistic model for online non-stationary recommendation systems, and analyze the reward of an online user-based algorithm. Our model and certain elements of the algorithm are inspired by related static models and algorithms studied in [Bresler et al., 2014, Bresler et al., 2016, Heckel and Ramchandran, 2017]. In a nutshell, each user in our model has a latent probability preference vector which describes the extent to which he/she likes or dislikes each item. Similar users have similar preference vectors (this will be defined rigorously in the following section). At a given time step $t = 1, 2, \ldots$, the algorithm recommends a single item to each user, typically different for each user, and with probability specified by the corresponding preference vector, the user likes or dislikes the recommended item. Following [Bresler et al., 2014, Bresler et al., 2016, Heckel and Ramchandran, 2017] we impose the constraint that an item that has been rated by a user, cannot be recommended to the same user again. This is due to the fact that in many applications, such as, recommendation of movies or books, a rating often corresponds to consuming an item, and there is little point in, e.g., recommending a product that has been previously purchased in the past for a second time, at least not immediately. While in certain applications, this constraint might be unnecessary/irrelevant, it *forces* our algorithm to exploit the user-item structure for collaboration. In any case, repeating the same recommendations only makes the problem easier algorithmically, and the results of this paper can be generalized to account for this case as well. Finally, to model the non-stationarity in the users' preferences, we allow users to change their user-type over time.

Main Contributions. Despite the success of CF, there has been no theoretical development to justify its effectiveness in non-stationary environment. The main contributions of this paper are two-fold. First, we introduce a novel model for general online non-stationary recommendation systems where we generalize the stationary model introduced in [Bresler et al., 2014] by allowing arbitrary shifts in user preferences over time. Our second main contribution is a theoretical analysis of a user-based CF algorithm that maximize the number of recommendations that users like. As time evolves, our CF algorithm randomly explores batches of items, one batch at a time, in order to learn users' preferences of new items in each batch. By splitting the space into optimal number of batches, our algorithms can start exploiting without having learned the preferences of the users regarding at all items. Furthermore, in each batch, our algorithm tests for variations, and once a change in the preference of a certain user is detected, that user is removed from the exploitation steps. Our results allow us to quantify the "price of non-stationarity", which mathematically captures the added complexity embedded in changing users' preferences versus stationary ones. The proposed algorithm achieve near-optimal performance relative to an oracle that recommends all likable items first. Our findings, such as the scaling of the cold-start time on the various parameters, and the effect of non-stationary environment on recommendation, can inform the design of recommendation algorithms, and refine our understanding of the associated benefits and costs while designing a practical recommendation system.

Related Work. While to the best of our knowledge, this is the first work that *analytically* studies temporal changes in the users' preferences, theoretical results have been established for the stationary setting where there are no changes in the user preferences over time. We next briefly survey these prior works. One of the first initial asymptotic theoretical results concerning user-based CF were established in [Biau et al., 2010]. Most related to our approach are the setups and algorithms studied in [Bresler et al., 2014, Bresler et al., 2016, Heckel and Ramchandran, 2017, Bresler and Karzand, 2019]. In particular, [Bresler et al., 2014] analyzed a user-based algorithm for online two-class CF problem in a similar setting to ours, while a corresponding item-based algorithm was analyzed in [Bresler et al., 2016]. A probabilistic model in an online setup was studied in [Dabeer, 2013], and [Barman and Dabeer, 2012] study a probabilistic model in an offline setup, and derived asymptotic optimal performance guarantees for two-class CF problem. Theoretical guarantees for a one-class models were derived in [Heckel and Ramchandran, 2017]. Another related work is [Deshpande and Montanari, 2012], who considered online recommendation systems in the context of linear multi-armed bandit (MAB) problem [Bubeck and Cesa-Bianchi, 2012].

Our setup relates to some variants of the MAB problem. An inherent conceptual difference between our setup and standard MAB formulation [Thompson, 1933] is that in our case an item can be recommended to a user just once, while in MAB an item (or arm) is allowed to be recommended ceaselessly. In fact, the solution principle for MAB is to explore for the best item, and once found, keep exploiting (i.e., recommending) it [Auer et al., 2002, Bubeck and Cesa-Bianchi, 2012]. This observation applies also to clustered bandits [Bui et al., 2012], or, bandits with dependent arms [Pandey et al., 2007]. Specifically, in these formulations the arms are grouped into clusters, and within each cluster arms are correlated. It is assumed, however, that the assignment of arms to clusters is known, while in our setup this information is not available. Another re-

lated formulation of MAB is "sleeping bandits" [Kleinberg et al., 2008], where the availability of arms vary over time in an adversarial manner, while in our setup, the sequence of items that are available is not adversarial. Finally, a more recent related version is the problem of MAB with non-stationary rewards, e.g., [Hartland et al., 2011, Garivier and Moulines, 2008, Yu and Mannor, 2009, Besbes Omer and Gur Yonatan and Zeevi Assaf, 2014, Besbes et al., 2014, Karnin and Anava, 2016, Luo et al., 2017, Cao et al., 2019, Chen et al., 2019, Auer et al., 2019, Jun et al., 2019, Sen et al., 2017, Kveton et al., 2017, Besbes et al., 2015, Allesiardo et al., 2017, Zhao et al., 2020]. This formulation allows for a broad range of temporal uncertainties in the rewards. While the motivation of this setup is similar to ours, due to the same reasons as above, the results and methods in these papers are quite different from ours. In particular, the main fundamental difficulty in our model compared to MAB literature lies in the fact that variations in the preferences of a certain user may affect the recommendations for other users severely.

2 Model and Learning Problem

In this section we introduce the model and the learning problem considered in this paper. We start with the static setting where users' preferences do not change over time, and then generalize to the dynamic setting.

Static Model. Consider a system with N users and M items. A user may "like" $(+1)$ or "dislike" (-1) an item. At each time step, each user is recommended an item that he has not consumed yet. Each user $u \in [\mathsf{N}] \triangleq \{1, 2, \ldots, \mathsf{N}\}$ is associated with a *latent* (unknown) preference vector $\mathbf{p}_u \in [0,1]^{\mathsf{M}}$, whose entries p_{ui} are the probabilities of user u liking item $i \in [\mathsf{M}]$, independently across items. We assume that an item i is either *likable* for user u, i.e., $p_{ui} > 1/2$, or *unlikable*, i.e., $p_{ui} < 1/2$. The reward earned by the recommendation system up to any time step is the total number of liked items that have been recommended so far across all users (a precise definition will be given in the sequel). Accordingly, to maximize this reward, clearly likable items for the user should be recommended before unlikable ones. For a particular item i, recommended to a user u, the observed rating is a random variable $\mathbf{R}_{ui}$, such that $\mathbf{R}_{ui} = 1$ with probability p_{ui}, and $\mathbf{R}_{ui} = -1$ with probability $1 - p_{ui}$. The ratings are assumed random in order to model the fact that users are not fully consistent in their ratings. A CF algorithm operates as follows: at each time step $t = 0, 1, \ldots$, the algorithm recommends a single item $i = \pi_{u,t} \in [\mathsf{M}]$ to each user $u \in [\mathsf{N}]$, and obtains a realization of the binary random variable $\mathbf{R}_{ui}$ in response. Thus, if $\mathbf{R}_{ui} = 1$, user u likes item i, and $\mathbf{R}_{ui} = -1$ means that user u does not like item i. Either way, as we explain and motivate in the Introduction, once rated by user u, item i *will not* be recommended to that user again.

To learn the preference of some user for an item, we need this user to rate that item. However, since we cannot recommend that item to the same user again, the only way to estimate the preferences of a user is through *collaboration* (e.g., by making inferences from ratings made by other users). In order to make collaborative recommendations based on users' preferences we must assume some structure/relation between users and/or items. To that end, we study a latent model, in which users are clustered. Specifically, following [Barman and Dabeer, 2012, Dabeer, 2013, Bresler et al., 2014, Bresler et al., 2016, Heckel and Ramchandran, 2017, Bresler and Karzand, 2019] (and many references therein), we assume that each user belongs to one of $\mathsf{K} < \mathsf{N}$ *user-types*, where users of the same type have "similar" item preference vectors. The number of types K represents the heterogeneity in the population. This assumption is common and implicitly invoked by contemporary user-based CF algorithms [Sarwar et al., 2000], which perform well in practice. Several empirical justifications for the clustering behavior in the user-item space can be found in, e.g., [Sutskever et al., 2009, Bresler et al., 2014, Heckel and Ramchandran, 2017].

There are many ways to define similarity between users. For example, in [Das et al., 2007, Barman and Dabeer, 2012, Dabeer, 2013, Bresler and Karzand, 2019] two users are considered of the same type if they have the same exact ratings, and these rating vectors are generated at random. While this model is perhaps unrealistic and does not capture challenges in real-world datasets, it allows for a neat theoretical analysis. A slightly more general and flexible model was proposed in [Bresler et al., 2014]. Here, two users u and v belong to the same type if their corresponding preference vectors $\mathbf{p}_u$ and $\mathbf{p}_v$ are the same, nonetheless, their actual ratings might be different. In [Heckel and Ramchandran, 2017] this assumption was significantly relaxed by assuming instead that the preference vectors belonging to the same type are more similar (in terms of the magnitude of their inner product) than those belonging to other types. Roughly speaking, in this paper we follow this latter assumption, but we relax it even further. The precise statement of our assumptions will be given in the following section. This concludes the presentation of the static model. We next incorporate the non-stationary aspect.

User Variations. As explained in the introduction, our paper initializes the theoretical investigation of the situation where the preferences of users do not remain

static but vary over time. To model this, we allow users to change their user-type over time. To wit, denote the type of user $u \in [\mathsf{N}]$ at time $t \in [\mathsf{T}]$ by $\mathcal{T}_u(t) \in [\mathsf{K}]$. In the sequel, $\mathcal{T}_u(1)$, for any $u \in [\mathsf{N}]$, designates the initial clustering of users into types. We assume that each user can change his type an *arbitrary* number of times, but bound the total number of such variations. Specifically, we define two notions for variations:

$$\mathsf{V}_1 \triangleq \max_{u\in[\mathsf{N}]} \sum_{t\in[\mathsf{T}-1]} \mathbb{1}[\mathcal{T}_u(t) \neq \mathcal{T}_u(t+1)], \tag{1}$$

$$\mathsf{V}_2 \triangleq \mathsf{N}^{-1} \sum_{t\in[\mathsf{T}-1]} \sum_{u\in[\mathsf{N}]} \mathbb{1}[\mathcal{T}_u(t) \neq \mathcal{T}_u(t+1)], \tag{2}$$

where $\mathbb{1}[\cdot]$ is the indicator function. These definitions capture the constraint imposed on the non-stationary environment faced by the CF algorithm. In words, V_1 is the maximum number of variations over T steps, while $\mathsf{N}\mathsf{V}_2$ is the total number of variations. In general, V_1 and V_2 are designed to depend on the time horizon T. It turns out that in order to obtain the tightest bound on the reward, both definitions are needed. It is clear that $\mathsf{V}_1 \leq \mathsf{N} \cdot \mathsf{V}_2$.

In this paper, we consider the already non-trivial case where the values of (or, at least, upper bounds on) V_1 and V_2 are *given* to the learner/algorithm in advance. This is inline with the various settings of classical results in the non-stationary MAB literature, e.g., [Besbes Omer and Gur Yonatan and Zeevi Assaf, 2014,Besbes et al., 2014,Luo et al., 2017]. Nonetheless, in a similar fashion to recent advances in the MAB literature [Karnin and Anava, 2016,Auer et al., 2019,Chen et al., 2019], it is an important, challenging, and of practical importance to propose and analyze algorithms which are oblivious to the number of variations (see Appendix D).

Learning Problem and Reward. Generally speaking, a reasonable objective for a CF algorithm is to maximize the expected reward up to time T, i.e.,

$$\overline{\mathsf{reward}}(\mathsf{T}) \triangleq \mathbb{E} \sum_{t\in[\mathsf{T}]} \frac{1}{\mathsf{N}} \sum_{u\in[\mathsf{N}]} \mathbf{R}_{u\pi_{u,t}},$$

where we note that the recommended item $\pi_{u,t}$ is a random variable because it is chosen by the CF algorithm as a function of previous responses to recommendations made at previous times. In this paper, however, we focus on recommending *likable* items. Following [Bresler et al., 2014, Bresler et al., 2016, Heckel and Ramchandran, 2017, Bresler and Karzand, 2019], we consider the accumulated reward defined as the expected total number of liked items that are recommended by an algorithm up to time T, i.e.,

$$\mathsf{reward}(\mathsf{T}) \triangleq \mathbb{E} \sum_{t\in[\mathsf{T}]} \frac{1}{\mathsf{N}} \sum_{u\in[\mathsf{N}]} \mathbb{1}\left[p_{u\pi_{u,t}} > 1/2\right]. \tag{3}$$

Note that the main difference between these two objectives is that the former prioritize items according to their probability of being liked, while the latter asks to recommend likable items for a user in an arbitrary order. This is sufficient for many real recommendation systems such as for movies and music (see, [Bresler et al., 2014, Sec. 2] for a detailed discussion). We would like to emphasize that depending on the intended application, other metrics can be considered. For example, one may be interested in the number of actual "clicks" rather then the number of "likable" recommendations. Indeed, in some applications, the former is the measurable quantity. Nonetheless, we believe that our algorithms and techniques can handle such a criterion as well.

3 Algorithms and Theoretical Guarantees

In this section, we present our algorithm Collaborative which recommends items to users over time, followed by a formal theoretical statement for its performance. We start with a non-formal description of our algorithm. The pseudocodes are given in Algorithms 1–3. The algorithm takes as input the parameters $(\alpha, \lambda, \mathsf{T}_{\mathsf{static}}, \Delta_{\mathsf{T}})$, which we shall discuss below.

We now describe the proposed algorithm. Below, we use "t" to denote the time index at any step of the algorithm. Also, in the sequel, we use $\mathcal{P}$ to denote an estimated partitioning of the users into clusters, i.e., $\mathcal{P}$ is a collection of clusters, where each cluster refers to a set of users who have same estimated user type.

Batches. In Algorithm 1, the time horizon T is partitioned into $\lceil \mathsf{T}/\Delta_{\mathsf{T}} \rceil$ batches of size Δ_{T} each, and denoted by $\{\mathcal{B}_b\}_{b\geq 1}$. We use τ to denote the time index at which each batch starts, namely, for the b^{th} batch $\tau = (b-1)\Delta_{\mathsf{T}}$, for $b \in [1, \lceil 1/\Delta_{\mathsf{T}} \rceil]$. At the beginning of each batch, we *restart* Algorithm Recommend, and run it for the entire batch. Also, at the beginning of each batch, we estimate an initial partition $\mathcal{P}_0$ for the set of all users $[\mathsf{N}]$ using Algorithm 3, which we describe bellow. At each subsequent time index t, the Algorithm Recommend either runs a *test for variations* with probability (w.p.) $\mathsf{p}_{\mathsf{T}} \propto \sqrt{\mathsf{V}_1/\mathsf{T}}$, or, explores-exploits w.p. $1 - \mathsf{p}_{\mathsf{T}}$.

User partitioning in the batch. The goal of Algorithm 3 is to create a partition of the users into types. To that end, routine Test$(\mathsf{T}_{\mathsf{static}}, \lambda, \mathcal{S}_{t-1}, \mathcal{P}_0)$ recommends $\mathsf{T}_{\mathsf{static}} \in \mathbb{N}$ random items $\mathcal{T}_{\mathsf{test}}$ ($|\mathcal{T}_{\mathsf{test}}| = \mathsf{T}_{\mathsf{static}}$)

Algorithm 1 COLLABORATIVE Algorithm for recommending items.

Require: Parameters $(\alpha, \lambda, \mathsf{T}_{\mathsf{static}}, \Delta_{\mathsf{T}})$.

1: Set index batch $b = 1$.
2: **while** $b \leq \lceil \mathsf{T}/\Delta_{\mathsf{T}} \rceil$ **do**
3: Set $\tau \leftarrow (b-1)\Delta_{\mathsf{T}}$.
4: Call RECOMMEND$(\tau, \alpha, \lambda, \mathsf{T}_{\mathsf{static}}, \Delta_{\mathsf{T}})$.
5: Set $b \leftarrow b+1$ and return to the beginning of Step 2.

to all users in $\mathcal{S}_{t-1} \subseteq [\mathsf{N}]$, initialized in each batch to be the set of all users $[\mathsf{N}]$. Then, using the obtained responses $\{\mathbf{R}_{ui}\}_{u\in[\mathcal{S}], i\in\mathcal{T}_{\mathsf{test}}}$, in the second and third steps of this algorithm a partition of the users is created. Specifically, for any pair of distinct users $u, v \in \mathcal{S}_{t-1}$, we let $\mathsf{X}_{u,v}$ be the number of items for which u and v had the same responses. Let $\mathsf{E}_{u,v} = \mathbb{1}\left[\mathsf{X}_{u,v} \geq \lambda \cdot \mathsf{T}_{\mathsf{static}}\right]$, for some $\lambda > 0$. Accordingly, users u and v are inferred to have the same type if $\mathsf{E}_{u,v} = 1$. Subsequently, if there exists a valid partitioning $\mathcal{P}$ over the set of users in $\mathcal{S}_{t-1}$, which is consistent with the variables $\mathsf{E}_{u,v}$, then we declare that $\mathcal{P}$ is our estimated user-partition, otherwise, we place all users in the same group. This is true precisely when the graph with edge set $\mathsf{E}_{u,v}$ is a disjoint union of cliques. Note that this partitioning procedure is equivalent to the cosine similarity test, declaring that two users u and v as being neighbors if their cosine similarity is at least as large as some threshold. The values of $\mathsf{T}_{\mathsf{static}}$ and $\lambda \in (0,1)$ are specified in Theorem 1.

Variations detection in the batch. Given the partitions $\mathcal{P}_0$ and $\mathcal{P}$, in step 14 of Algorithm 2 we compare those partitions in order to detect any variations using routine VARIATION. We show that if the user variations is not "too large" in the corresponding batch, then it is possible to draw a one-to-one correspondence between the clusters in $\mathcal{P}$ and $\mathcal{P}_0$, and therefore, it is possible to identify the users who have changed their clusters, i.e., they are in a different cluster in $\mathcal{P}$ than the one in $\mathcal{P}_0$. All such users are declared as "bad" users and are included in the set $\mathcal{V}_t$. We update $\mathcal{S}_t \leftarrow \mathcal{S}_{t-1} \setminus \mathcal{V}_t$. The users in $\mathcal{V}_t$ will be excluded from future exploitation rounds.

Exploration-Exploitation. Since we restart the main algorithm in each batch, we focus on a particular batch b in the explanation below. For ease of notation, we omit the batch index from all definitions. As mentioned above, w.p. $1 - \mathsf{p}_{\mathsf{T}}$ our algorithm performs an exploration-exploitation routine. In such a case, with probability $\mathsf{p}_{\mathsf{R}} = \mathsf{N}^{-\alpha}$ the algorithm randomly explores the space of items, and with complementary probability, $1 - \mathsf{p}_{\mathsf{R}}$, the algorithm exploits by recommending those items that maximize a certain score. With some

Algorithm 2 RECOMMEND$(\tau, \alpha, \lambda, \mathsf{T}_{\mathsf{static}}, \Delta_{\mathsf{T}})$

1: Select a random ordering σ of the items $[\mathsf{M}]$.
2: Define $\mathsf{p}_{\mathsf{R}} = \mathsf{N}^{-\alpha}$ and $\mathsf{p}_{\mathsf{T}} = \sqrt{\mathsf{V}_1/(\mathsf{T}\cdot\mathsf{T}_{\mathsf{static}})}$.
3: Let t to be the time index at any step of the algorithm.
4: $\mathcal{P}_0 \leftarrow$TEST$(\mathsf{T}_{\mathsf{static}}, \lambda, [\mathsf{N}])$.
5: Initialize $\mathcal{S}_{\tau+\mathsf{T}_{\mathsf{static}}} \leftarrow [\mathsf{N}]$.
6: **while** $\tau + \mathsf{T}_{\mathsf{static}} < t \leq \min(\mathsf{T}, \tau + \Delta_{\mathsf{T}})$ **do**
7: Draw $\Theta \sim \mathsf{Bern}(\mathsf{p}_{\mathsf{T}})$.
8: **if** $\Theta = 0$ **then**
9: $\mathcal{S}_t \leftarrow \mathcal{S}_{t-1}$.
10: • **With probability** p_{R}: $\pi_{u,t} \leftarrow$ random item for all $u \in [\mathsf{N}]$ that has not rated.
11: • **With probability** $1 - \mathsf{p}_{\mathsf{R}}$: $\pi_{u,t} \leftarrow$ item i that user $u \in \mathcal{S}_t$ has not rated and that maximizes score $\hat{p}_{ui}^{(t)}$ in (4).
12: **else**
13: $\mathcal{P} \leftarrow$TEST$(\mathsf{T}_{\mathsf{static}}, \lambda, \mathcal{S}_{t-1})$.
14: $\mathcal{V}_t \leftarrow$VARIATION$(\mathcal{P}, \mathcal{P}_0)$.
15: $\mathcal{S}_t \leftarrow \mathcal{S}_{t-1} \setminus \mathcal{V}_t$.

Algorithm 3 TEST$(\mathsf{T}_{\mathsf{static}}, \lambda, \mathcal{S}_{t-1})$ Algorithm for partitioning users.

1: Recommend $\mathsf{T}_{\mathsf{static}}$ random items $\mathcal{T}_{\mathsf{test}}$ to all users in $\mathcal{S}_{t-1}$.
2: For any $u \neq v \in \mathcal{S}_{t-1}$, let $\mathsf{X}_{u,v}$ be the number of items in $\mathcal{T}_{\mathsf{test}}$ for which u and v had the same responses, and let $\mathsf{E}_{u,v} = \mathbb{1}\left[\mathsf{X}_{u,v} \geq \lambda \cdot \mathsf{T}_{\mathsf{static}}\right]$.
3: Let $\mathcal{P}$ be the valid partitioning over users consistent with the variables $\mathsf{E}_{u,v}$. If such a partitioning does not exist, let $\mathcal{P} \equiv \mathcal{S}_{t-1}$.
4: Return $\mathcal{P}$.

abuse of notation let $\mathbf{R}_{ui}^{(t)} \in \{-1, 0, 1\}$ be the observed rating of user u to item i up to time t in the b^{th} batch, where "0" means that no rating has been given yet (in the b^{th} batch). When exploiting, the algorithm evaluates empirical probabilities $\hat{p}_{ui}^{(t)}$, at time t, for user $u \in \mathcal{S}_t$, and item i. These empirical probabilities are defined as follows,

$$\hat{p}_{ui}^{(t)} \triangleq \begin{cases} \frac{\sum_{v\in\mathsf{neigh}(u,t)} \mathbb{1}\left[\mathbf{R}_{vi}^{(t)}=1\right]}{\mathsf{C}_{ut}}, & \text{if } \mathsf{C}_{ut} > 0 \\ 1/2, & \text{otherwise,} \end{cases} \tag{4}$$

where $\mathsf{C}_{ut} \triangleq \sum_{v\in\mathsf{neigh}(u,t)} \mathbb{1}[\mathbf{R}_{vi}^{(t)} \neq 0]$, and the "neighborhood" of user $u \in \mathcal{S}_t$ at time t in the b^{th} batch is $\mathsf{neigh}(u,t) \triangleq \mathcal{P}_0(u) \cap \mathcal{S}_t$, where $\mathcal{P}_0(u)$ is the set of users in the same cluster as user u in the initial partition $\mathcal{P}_0$ created in the beginning of the batch. Note that the exploitation step at any time index t is done only for those users which are present in $\mathcal{S}_t$. Finally, we emphasize here that the empirical probabilities $\hat{p}_{ui}^{(t)}$ as well as the neighborhoods $\mathsf{neigh}(u,t)$, are all refreshed at the

Algorithm 4 VARIATION($\mathcal{P}, \mathcal{P}_0$) Algorithm for testing variations.

1: For each cluster $\mathcal{C}$ in $\mathcal{P}$, find a cluster $\mathcal{C}'$ in $\mathcal{P}_0$ that shares at least half the users in $\mathcal{C}$ i.e., $|\mathcal{C} \cap \mathcal{C}'| \geq \frac{|\mathcal{C}|}{2}$.
2: Establish a one-to-one mapping from clusters in $\mathcal{P}$ to clusters in $\mathcal{P}_0$ in this manner. If such a one-to-one mapping is not possible, return $\emptyset$.
3: Identify the set of users $\mathcal{V}$ who belong to different clusters in $\mathcal{P}$ and $\mathcal{P}_0$.
4: Return $\mathcal{V}$.

beginning of each batch; ratings from previous batches are ignored in the evaluation of these quantities.

Remark 1. *In practice, we can continue recommending items to any user u in $\mathcal{V}_t$ (bad users) based on the items liked by other users who belong to $\mathcal{P}_0(u)$.*

Theoretical performance guarantee. In the following, we state our main theoretical result, which is a lower bound on the reward in (3) achieved by Algorithm COLLABORATIVE. To that end, we introduce three natural and prima facie necessary assumptions, which will be used in order to establish our result.

A1 No Δ-ambiguous items. For every user $u \in [\mathsf{N}]$ and item $i \in [\mathsf{M}]$, there exists a constant $\Delta \in (0, 1/2]$ such that $|p_{ui} - 1/2| \geq \Delta$.

A2 Minimum portion of likeable items. There exists a constant $\mu \in [0, 1]$, such that for every user $u \in [\mathsf{N}]$, it holds $\sum_{i=1}^{\mathsf{M}} \mathbb{1}[p_{ui} > 1/2] \geq \mu \mathsf{M}$.

A3 (In)coherence. There exist constants $\gamma_2 \geq \gamma_1 \in [0, 1)$ such that if two users u and v are of *different* types, then $\langle 2\mathbf{p}_u - \mathbf{1}, 2\mathbf{p}_v - \mathbf{1} \rangle \leq 4\gamma_1 \Delta^2 \mathsf{M}$, and if they are of the *same* type, then $\langle 2\mathbf{p}_u - \mathbf{1}, 2\mathbf{p}_v - \mathbf{1} \rangle \geq 4\gamma_2 \Delta^2 \mathsf{M}$. Here $\mathbf{1}$ is the all ones vector.

In a nutshell, Assumption **A1** is necessary to assure that one can infer whether an item is either likable or unlikable with a finite number of samples. The parameter Δ quantifies the inconsistency (or, noise), where $\Delta = 0$ ($\Delta = 1/2$) is the completely noisy (noiseless) case. The second condition states that each user likes at least a fraction μ of the total items. This assumption is made to avoid degenerate situations were a user u does not like any item. Note that one can always ignore those users whose activity is insignificant since their contribution to the reward will be insignificant as well. Evidently, from a practical point of view, any real-world recommendation engine must prioritize users whose activity is significant. Notice that relevant literature, such as [Bresler et al., 2014,Heckel and Ramchandran, 2017], makes the same assumptions as well.

The more interesting assumption is **A3**. The incoherence part of assumption **A3** requires that the preference vectors for any two users u and v belonging to different user-types are not too similar, so that the cosine similarity test can separate users of different types over time. The parameter γ_1 controls this incoherence; the smaller γ_1 is, the less similar are users of different types. This incoherence assumption appears in [Bresler et al., 2014] as well. The coherence part of assumption **A3** asks that any two users of the same user-type share a large fraction of the items that they find likable, and this fraction is controlled by the parameter γ_2. This coherence assumption should be contrasted with [Bresler et al., 2014] where it was assumed that the preference vectors $\mathbf{p}_u$ and $\mathbf{p}_v$ of two users u and v from the same user-type to be exactly the same, which is evidently a stronger assumption, and accordingly, our coherence assumption relaxes that significantly.

We would like to clarify that the above assumptions are only required for the analysis; Our proposed algorithm can be implemented regardless of these assumptions. As is shown in Section 4, in real-world applications, our algorithm works well even if these assumptions do not hold. In Appendix A we provide two examples where the typical values of the various parameters in Assumptions **A1**–**A3** are derived. Below, we summarize our findings for the noiseless case.

Example 1. *Consider the noiseless case where $\Delta = 1/2$. We generate K d-dimensional binary vectors $\{\mathbf{b}_i\}_{i=1}^{\mathsf{K}}$ by randomly drawing d statistically independent $\mathsf{Bernoulli}(1/2)$ random variables, for each user-type. Here $\mathsf{d} \leq \mathsf{M}$ is some parameter. We show in Appendix A that the incoherence condition holds with high probability if $\gamma_1 > \Theta(\sqrt{\frac{\log \mathsf{M}}{\mathsf{M}}})$, while the coherence condition holds with high probability if $\gamma_2 \leq \frac{\mathsf{d}}{\mathsf{M}} - \Theta(\sqrt{\frac{(\mathsf{M}-\mathsf{d}) \log \mathsf{M}}{\mathsf{M}^2}})$. Finally, the typical value of μ is clearly around $1/2$ with high probability.*

Let $a \vee b \triangleq \max(a, b)$ and $a \wedge b \triangleq \min(a, b)$, for $a, b \in \mathbb{R}$. We are now in position to state our main result, whose proof can be found in the supplementary material.

Theorem 1. *Let $\delta \in (0, 1)$ and $\nu \in (0, 1)$ be some pre-specified tolerances. Take as input to* COLLABORATIVE *$\alpha \in (0, 4/7]$, any $\lambda \in (\lambda_-, \lambda_+)$, and $\Delta_{\mathsf{T}} = \mathsf{T} \wedge \sqrt{\frac{2\nu\mathsf{T}}{3\mathsf{V}_2}}\kappa$, where $\lambda_\pm \triangleq 2\gamma_1\Delta^2 + \frac{1}{2} \pm \Delta^2(\gamma_2 - \gamma_1)$ and $\kappa \triangleq \mathsf{T}_{\mathsf{static}}(1 - \delta - \mu)$. Define $\mathsf{T}_{\mathsf{static}} \triangleq \frac{2\log(3\mathsf{N}^2/\delta)}{\Delta^4(\gamma_2 - \gamma_1)^2}$ and $\mathsf{T}_{\mathsf{learn}} \triangleq \left[1 \vee \frac{3\mathsf{V}_2}{2\nu(1-\delta-\mu)}\right] \mathsf{T}_{\mathsf{static}}$. Consider the latent source model and assumptions **A1**–**A3**. If at every time-point, the portion of users belonging to any user-type is at least ν, then, for any $\mathsf{T}_{\mathsf{learn}} \leq \mathsf{T} \leq \mu \cdot \mathsf{M}$, the expected proportion of liked items recommended by*

COLLABORATIVE *up until time* T *satisfies*

$$\mathsf{reward}(\mathsf{T}) \geq (1-\delta)\cdot\mathsf{T} - \kappa \vee \sqrt{\frac{3\mathsf{V}_2\mathsf{T}\kappa}{2\nu}} - 2\mathsf{V}_1\mathsf{T}_{\mathsf{static}} - 2\sqrt{\mathsf{V}_1\mathsf{T}\mathsf{T}_{\mathsf{static}}} - \frac{3\mathsf{V}_2\mathsf{T}}{2\nu} \wedge \sqrt{\frac{3\mathsf{V}_2\mathsf{T}\kappa}{2\nu}}. \quad (5)$$

For $\mathsf{T} < \mathsf{T}_{\mathsf{learn}}$*, the reward satisfies* $\mathsf{reward}(\mathsf{T}) \geq \mu\cdot\mathsf{T}$.

We now explain the implications of Theorem 1. For $\mathsf{T} < \mathsf{T}_{\mathsf{learn}}$, the algorithm may give poor recommendations. This is reasonable since in the first $\mathsf{T}_{\mathsf{learn}}$ rounds mostly random items are recommended, independently of the users responses, and thus the reward is at least $\mu\cdot\mathsf{T}$. This is the initial phase for which our CF algorithm gives poor recommendations. Then, for $\mathsf{T}_{\mathsf{learn}} < \mathsf{T} < \mu\cdot\mathsf{M}$, the algorithm becomes efficient. Specifically, when $\mathsf{V}_1 = \mathsf{V}_2 = 0$, we get that $\mathsf{reward}(\mathsf{T})/\mathsf{T} \geq (1-\mathsf{T}_{\mathsf{learn}}/\mathsf{T})\cdot(1-\delta-\eta)$. Therefore, the proposed algorithm becomes near-optimal, as the achieved reward is $(1-\varepsilon')$–close to an oracle that recommends only likeable items and thus achieves a reward of T. Note that contrary to multi-armed bandit literature, linear reward is common in collaborative filtering frameworks (see, for example, [Bresler et al., 2014, Heckel and Ramchandran, 2017, Bresler and Karzand, 2019]). For $\mathsf{T} > \mu\cdot\mathsf{M}$, on the other hand, one cannot guarantee that likable items remain, and the learning time (cold-start time) is $\mathsf{T}_{\mathsf{learn}} = \mathsf{T}_{\mathsf{static}}$.

Clearly, when $\mathsf{V}_1, \mathsf{V}_2 > 0$ the reward decreases. Specifically, if both of these parameters scale as $O(\mathsf{T}^c)$, for some constant $c \in [0,1]$, then the payoff for non-stationarity compared to the static case is of order $O(\sqrt{\mathsf{T}^{1+c}})$, a sub-linear cost in T. In particular, ignoring the exact dependency of the reward on the various parameters, the scaling of (5) with T is $\mathsf{reward}(\mathsf{T})/\mathsf{T} \geq 1-\delta-O(\sqrt{\mathsf{T}^{c-1}})$. This result provides a spectrum of orders of the payoff ranging between order $O(\sqrt{\mathsf{T}})$ (constant number of variations), and of order $O(\mathsf{T})$ (number of variations is $O(\mathsf{T})$). The sub-linearity growth implies that our user-based CF algorithm has long run average performance that converges to the performance that would have been achieved in the static environment, where users' preferences do not vary. Finally, in terms of the learning time, it can be checked that $\mathsf{T}_{\mathsf{learn}} = \Theta(\mathsf{T}^c\log(\mathsf{N}^2/\delta))$, and thus the condition $\mathsf{T} > \mathsf{T}_{\mathsf{learn}}$ boils down to $\mathsf{T} > \Theta([\log(\mathsf{N}^2/\delta)]^{\frac{1}{1-c}})$. Therefore, when there are variations, the cold-start time grows, and the scaling of the variations with T dictates the poly-log order of this learning phase.

Next, we study the dependency of the learning time in Theorem 1 on γ_1 and γ_2, for Example 1 above. It can be seen that the learning time depend on these parameters via the term $(\gamma_2-\gamma_1)^{-2}$. Accordingly, when $\mathsf{d} = \mathsf{M}$ in Example 1 we get that $(\gamma_2-\gamma_1)^{-2}$ is of order constant by taking $\gamma_1 = \Theta(\sqrt{\frac{\log\mathsf{M}}{\mathsf{M}}})$ and $\gamma_2 = 1$. In fact, it is evident that this is true when $\mathsf{d} = \Theta(\mathsf{M})$ as well, and thus if any two users from the same user-type share a constant fraction of the total number of items that they find likeable, then this has a multiplicative constant effect on the learning time. If however, $\mathsf{d} = o(\mathsf{M})$, say, $\mathsf{d} = O(\mathsf{M}^q)$, for $q \in (0, 1/2)$, then we can set $\gamma_2 = \Theta(\mathsf{M}^{-c})$, and accordingly, $\mathsf{T}_{\mathsf{static}}$ will scale as $\mathsf{M}^{2q}\cdot\log(\mathsf{N}^2/\delta)$, for $\alpha \to 0$. Accordingly, we see that when negligible amount of common items are likeable by users of the same type, then the learning time is significantly larger, as expected.

Below we mention briefly some of the technical challenges encountered in the proof of Theorem 1. First, we establish a connection between the static reward and the reward of a dynamic oracle in the non-stationary setting. This connection is general and can be used in other possible static recommendation system models that incorporate non-stationary environment. One of the main difficulties in the proof of Theorem 1 is the analysis of how would a variation in the preference of some user affects other users in its estimated neighborhood. Unless this user is detected by Algorithm 3, the algorithm does not know the change of this user, and it will keep using this user's feedback to make recommendations for other users. This is one source of added regret incurred by the unsuccessful and incorrect detections of the change-points. Other sources of regret are the cost associated with the detection/testing delay, and a regret incurred by variations happening when testing. These costs are captured by the third and fourth terms at the r.h.s. of (5).

4 Experiments

We provide only a summary of our experimental results here, deferring full details and further experiments to Appendix C. We follow a similar vein as in [Bresler et al., 2014, Heckel and Ramchandran, 2017], and simulate an online recommender system using real-world data. Specifically, we look at movie ratings from the popular Movielens25m dataset, which provides 5-star rating from Movielens, a movie recommendation service. We quantize movie ratings ≥ 4 as $+1$ (likable), movie ratings < 3 as -1 (unlikable), and missing ratings as 0. We parsed the first 7 million ratings for our experiment, and consider only those users who have rated at least 225 movies, ending up with a total number of $\mathsf{N} = 247$ users. Also, we focused on two genres: `Action` and `Romance`. For each user $u \in [\mathsf{N}]$, we recover piece-wise stationary preferences by the following three steps: 1) We sort the movies rated by user u in ascending order according to the

time-stamp. 2) We partition the movies rated by user u into 15 bins so that each bin contains equal number of movies. We will consider each bin to be a window of time. 3) For each bin, we find the number $\mathsf{a}_u \in \mathbb{N}$ of `Action` movies rated by user u, as well as $\mathsf{r}_u \in \mathbb{N}$, the number of `Romance` movies rated by the same user.

Accordingly, note that in each bin, the probability of user u: liking a movie tagged `Action` but not `Romance` is $\mathsf{a}_u/(\mathsf{a}_u + \mathsf{r}_u)$; liking a movie tagged `Romance` but not `Action` is $\mathsf{r}_u/(\mathsf{a}_u + \mathsf{r}_u)$; liking a movie tagged both `Action` and `Romance` is `1`, and finally, a movie which does not have any of these tags is `0`. We want to point out that we consider the number of `Action` and `Romance` movies that were *rated* by the user, rather than just *liked*, since any user is biased towards rating the movies he will like (see, [Heckel and Ramchandran, 2017]), and therefore the number of movies rated by the user is a better indicator of his preference towards the genre. Fig. 1 shows the probability of 5 randomly chosen users liking `Action` movies across 10 different bins. It is clear that the preferences exhibit a piece-wise stationary behaviour, and that the variations are significant.

We now assume for simplicity that the number of rounds in each bin is 100 (this value is unknown to the algorithm), and we took the total number of rounds to be $\mathsf{T} = 600$. In lieu of creating the initial disjoint clusters at the beginning of each batch (i.e., $\mathcal{P}_0$), we recommend $\mathsf{T}_{\mathsf{static}}$ randomly chosen items to all users. For each user $u \in [\mathsf{N}]$, we take the neighbors of u to be the top 10 users whose feedback vector has the highest cosine similarity with that of user u, over the $\mathsf{T}_{\mathsf{static}}$ recommended items. Further, since $\mathsf{T} = 600$ is quite small, we do not test for bad users in each batch (namely, we skip lines $13-15$ in Algorithm 2). Nevertheless, as we will show, our experiment clearly demonstrates the dependence on Δ_{T} and $\mathsf{T}_{\mathsf{static}}$ in the non-stationary setting. We run Algorithm 1 with $\mathsf{T}_{\mathsf{static}} = 10$ and $p_{\mathsf{R}} = 0.1$, for several different values of the batch-size Δ_{T}, each for 5 different iterations. The performance of the algorithms is measured in terms of the average cumulative reward up to time T, namely, $\mathsf{acc\text{-}reward}(\mathsf{T}) \triangleq \sum_{t\in[\mathsf{T}]} \frac{1}{\mathsf{N}} \sum_{u\in[\mathsf{N}]} \mathbf{R}_{u\pi_{u,t}}$, where $\pi_{u,t}$ is the item recommended by the algorithm to user u at time t. The average cumulative reward up to time T is listed in the form of tuples $(\Delta_{\mathsf{T}}, \mathsf{acc\text{-}reward}(\mathsf{T}))$: $(50, 316.71), (\mathbf{100}, \mathbf{325.71}), (150, 306.54), (200, 278.21)$, $(300, 278.64), (350, 224.9), (400, 239.4)(450, 204.12)$, $(500, 162.96), (550, 169.97), (600, 137.40)$. From this list, it is clear that the highest average cumulative reward is obtained when the batch-size is $\Delta_{\mathsf{T}} = 100$, and decreases gradually as the batch-size increases.

Next, we illustrate the benefit of our algorithm compared to the static algorithm even in a stationary environment. To that end, we run Algorithm 1 with $\Delta_{\mathsf{T}} \in \{100, 600\}$ and $\mathsf{T}_{\mathsf{static}} \in \{10, 30, 60, 80, 100\}$, and assume a single bin of size $\mathsf{T} = 600$. Our results are presented in Fig. 2, and perhaps surprisingly, Algorithm 1 with $\Delta_{\mathsf{T}} = 100$ achieves a better accumulated reward compared to $\Delta_{\mathsf{T}} = 600$ (static algorithm), for small values of $\mathsf{T}_{\mathsf{static}}$. The main reason for this phenomenon is that for Algorithm 1 with $\Delta_{\mathsf{T}} = 600$, the neighbors of any user might not be well chosen for small values of $\mathsf{T}_{\mathsf{static}}$, for which users will receive poor recommendations throughout the entire time frame. On the other hand, running Algorithm 1 with $\Delta_{\mathsf{T}} = 100$ restarts Algorithm 2 at periodic intervals. As a result, the users have a good set of neighbors in some batches and a bad set in others, but the cumulative reward concentrate because the neighbors are independent across the batches. However, the performance of the algorithm with $\Delta_{\mathsf{T}} = 600$ improves as $\mathsf{T}_{\mathsf{static}}$ gets larger since the quality of the estimated neighborhood improves. This experiment hints towards the conclusion that in general it might be better to restart the algorithm periodically, i.e., follow Algorithm 1 (with $\Delta_{\mathsf{T}} < \mathsf{T}$) even in stationary environments.

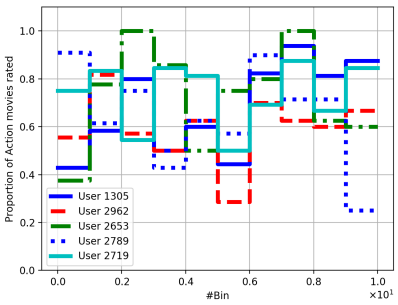

Figure 1: The probability $\mathsf{a}_u/(\mathsf{a}_u + \mathsf{r}_u)$ of user u liking a movie with `Action` tag but not `Romance` tag, for five different users, across 10 different bins.

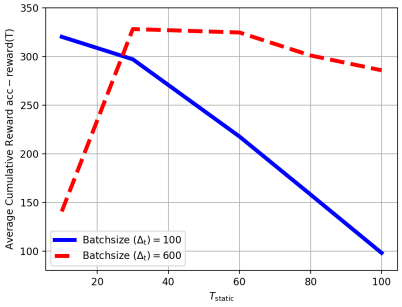

Figure 2: Comparison of Average Cumulative Reward for batchsize $\Delta_{\mathsf{T}} \in \{100, 600\}$, $\mathsf{T}_{\mathsf{static}} \in \{10, 30, 60, 80, 100\}$, $T = 600$ and no non-stationarity.

Acknowledgements

This research was supported by ISF grant no. 1495/18.

References

[Allesiardo et al., 2017] Allesiardo, R., Féraud, R., and Maillard, O. (2017). The non-stationary stochastic multi-armed bandit problem. *Int J Data Sci Anal*, 3:267–283.

[Auer et al., 2002] Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47:235–256.

[Auer et al., 2019] Auer, P., Gajane, P., and Ortner, R. (2019). Adaptively tracking the best bandit arm with an unknown number of distribution changes. In *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99, pages 138–158. PMLR.

[Barman and Dabeer, 2012] Barman, K. and Dabeer, O. (2012). Analysis of a collaborative filter based on popularity amongst neighbors. *IEEE Transactions on Information Theory*, 58(12):7110–7134.

[Basile et al., 2015] Basile, P., Caputo, A., Degemmis, M., Lops, P., and Semeraro, G. (2015). Modeling short-term preferences in time-aware recommender systems. In *UMAP Workshops*.

[Bell et al., 2007] Bell, R., Koren, Y., and Volinsky, C. (2007). Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 95–104, New York, NY, USA. ACM.

[Bellogin and Parapar, 2012] Bellogin, A. and Parapar, J. (2012). Using graph partitioning techniques for neighbour selection in user-based collaborative filtering. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 213–216. ACM.

[Besbes et al., 2014] Besbes, O., Gur, Y., and Zeevi, A. (2014). Stochastic multi-armed-bandit problem with non-stationary rewards. In *Advances in Neural Information Processing Systems 27*, pages 199–207.

[Besbes et al., 2015] Besbes, O., Gur, Y., and Zeevi, A. (2015). Non-stationary stochastic optimization. *Operations Research*, 63(5):1227–1244.

[Besbes Omer and Gur Yonatan and Zeevi Assaf, 2014] Besbes Omer and Gur Yonatan and Zeevi Assaf (2014). Stochastic multi-armed-bandit problem with non-stationary reward. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 199–207.

[Biau et al., 2010] Biau, G., Cadre, B., and Rouvière, L. (2010). Statistical analysis of k -nearest neighbor collaborative recommendation. *Ann. Statist.*, 38(3):1568–1592.

[Bresler et al., 2014] Bresler, G., Chen, G. H., and Shah, D. (2014). A latent source model for online collaborative filtering. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3347–3355.

[Bresler and Karzand, 2019] Bresler, G. and Karzand, M. (2019). Regret bounds and regimes of optimality for user-user and item-item collaborative filtering. *arXiv:1711.02198*.

[Bresler et al., 2016] Bresler, G., Shah, D., and Voloch, L. F. (2016). Collaborative filtering with low regret. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, SIGMETRICS '16, pages 207–220, New York, NY, USA. ACM.

[Bubeck and Cesa-Bianchi, 2012] Bubeck, S. and Cesa-Bianchi, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122.

[Bui et al., 2012] Bui, L., Johari, R., and Mannor, S. (2012). Clustered bandits. *CoRR*.

[Cao et al., 2019] Cao, Y., Zheng, W., Kveton, B., and Xie, Y. (2019). Nearly optimal adaptive procedure for piecewise-stationary bandit: a change-point detection approach. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[Chen et al., 2019] Chen, Y., Lee, C.-W., Luo, H., and Wei, C.-Y. (2019). A new algorithm for non-stationary contextual bandits: Efficient, optimal and parameter-free. In *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99, pages 696–726. PMLR.

[Dabeer, 2013] Dabeer, O. (2013). Adaptive collaborating filtering: The low noise regime. In *2013 IEEE International Symposium on Information Theory*, pages 1197–1201.

[Das et al., 2007] Das, A. S., Datar, M., Garg, A., and Rajaram, S. (2007). Google news personalization:

Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA. ACM.

[Deshpande and Montanari, 2012] Deshpande, Y. and Montanari, A. (2012). Linear bandits in high dimension and recommendation systems. In *Allerton Conference on Communication, Control, and Computation*, pages 1750–1754.

[Ekstrand et al., 2011] Ekstrand, M. D., Riedl, J. T., and Konstan, J. A. (2011). Collaborative filtering recommender systems. *Found. Trends Hum.-Comput. Interact.*, 4(2):81–173.

[Eskandanian and Mobasher, 2018] Eskandanian, F. and Mobasher, B. (2018). Detecting changes in user preferences using hidden markov models for sequential recommendation tasks. *CoRR*, abs/1810.00272.

[Garivier and Moulines, 2008] Garivier, A. and Moulines, E. (2008). On upper-confidence bound policies for non-stationary bandit problems.

[Gregory et al., 1998] Gregory, D. L., Jennifer, A. J., and Eric, A. B. (1998). Collaborative recommendations using item-to-item similarity mappings. *U.S. Patent 6266649B1*.

[Hariri et al., 2014] Hariri, N., Mobasher, B., and Burke, R. (2014). Context adaptation in interactive recommender systems. In *RecSys 2014 - Proceedings of the 8th ACM Conference on Recommender Systems*, pages 41–48.

[Hariri et al., 2015] Hariri, N., Mobasher, B., and Burke, R. (2015). Adapting to user preference changes in interactive recommendation. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 4268–4274.

[Hartland et al., 2011] Hartland, C., Baskiotis, N., Gelly, S., Sebag, M., and Teytaud, O. (2011). Change point detection and meta-bandits for online learning in dynamic environments.

[Heckel and Ramchandran, 2017] Heckel, R. and Ramchandran, K. (2017). The sample complexity of online one-class collaborative filtering. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1452–1460. JMLR.org.

[Herlocker et al., 1999] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 230–237, New York, NY, USA. ACM.

[Jahrer et al., 2010] Jahrer, M., Töscher, A., and Legenstein, R. (2010). Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 693–702, New York, NY, USA. ACM.

[Jun et al., 2019] Jun, K.-S., Willett, R., Wright, S., and Nowak, R. (2019). Bilinear bandits with low-rank structure. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3163–3172. PMLR.

[Karahodza et al., 2014] Karahodza, B., Supic, H., and Donko, D. (2014). An approach to design of time-aware recommender system based on changes in group user's preferences. In *2014 X International Symposium on Telecommunications (BIHTEL)*, pages 1–4.

[Karnin and Anava, 2016] Karnin, Z. S. and Anava, O. (2016). Multi-armed bandits: Competing with optimal sequences. In *Advances in Neural Information Processing Systems 29*, pages 199–207.

[Kleinberg et al., 2008] Kleinberg, R., Niculescu-Mizil, A., and Sharma, Y. (2008). Regret bounds for sleeping experts and bandits. volume 80, pages 425–436.

[Koren, 2008] Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 426–434, New York, NY, USA. ACM.

[Kveton et al., 2017] Kveton, B., Szepesvari, C., Rao, A., Wen, Z., Abbasi-Yadkori, Y., and Muthukrishnan, S. (2017). Stochastic low-rank bandits.

[Linden et al., 2003] Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80.

[Liu et al., 2010] Liu, J., Pedersen, E., and Dolan, P. (2010). Personalized news recommendation based on click behavior. In *2010 International Conference on Intelligent User Interfaces*.

[Liu, 2015] Liu, X. (2015). Modeling users' dynamic preference for personalized recommendation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1785–1791.

[Luo et al., 2017] Luo, H., Agarwal, A., and Langford, J. (2017). Efficient contextual bandits in non-stationary worlds. In *COLT*.

[Moore et al., 2013] Moore, J. L., Chen, S., Joachims, T., and Turnbull, D. (2013). Taste over time: the temporal dynamics of user preferences. In *ISMIR*.

[Mukherjee et al., 2017] Mukherjee, S., Lamba, H., and Weikum, G. (2017). Item recommendation with evolving user preferences and experience. *CoRR*, abs/1705.02519.

[Pandey et al., 2007] Pandey, S., Chakrabarti, D., and Agarwal, D. (2007). Multi-armed bandit problems with dependent arms. pages 721–728.

[Rennie and Srebro, 2005] Rennie, J. D. M. and Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 713–719, New York, NY, USA. ACM.

[Resnick et al., 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA. ACM.

[Salakhutdinov and Mnih, 2007] Salakhutdinov, R. and Mnih, A. (2007). Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1257–1264, USA.

[Salakhutdinov and Mnih, 2008] Salakhutdinov, R. and Mnih, A. (2008). Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 880–887, New York, NY, USA. ACM.

[Sarwar et al., 2000] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2Nd ACM Conference on Electronic Commerce*, EC '00, pages 158–167, New York, NY, USA. ACM.

[Sarwar et al., 2001] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA. ACM.

[Sen et al., 2017] Sen, R., Shanmugam, K., Kocaoglu, M., Dimakis, A., and Shakkottai, S. (2017). Contextual Bandits with Latent Confounders: An NMF Approach. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 518–527. PMLR.

[Shi et al., 2015] Shi, F., Ghedira, C., and Marini, J. (2015). Context adaptation for smart recommender systems. *IT Professional*, 17(6):18–26.

[Si and Jin, 2003] Si, L. and Jin, R. (2003). Flexible mixture model for collaborative filtering. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 704–711. AAAI Press.

[Sutskever et al., 2009] Sutskever, I., Tenenbaum, J. B., and Salakhutdinov, R. R. (2009). Modelling relational data using bayesian clustered tensor factorization. In *Advances in Neural Information Processing Systems 22*, pages 1821–1828.

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

[Yu and Mannor, 2009] Yu, J. Y. and Mannor, S. (2009). Piecewise-stationary bandit problems with side observations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML'09, page 1177–1184.

[Zhao et al., 2020] Zhao, P., Zhang, L., Jiang, Y., and Zhou, Z.-H. (2020). A simple approach for non-stationary linear bandits. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 746–755. PMLR.