The Sample Complexity of Online One-Class Collaborative Filtering

Reinhard Heckel [1] **Kannan Ramchandran** [1]

Abstract

We consider the online one-class collaborative filtering (CF) problem that consists of recommending items to users over time in an online fashion based on *positive* ratings only. This problem arises when users respond only occasionally to a recommendation with a positive rating, and never with a negative one. We study the impact of the probability of a user responding to a recommendation, p_{f}, on the sample complexity, i.e., the number of ratings required to make 'good' recommendations, and ask whether receiving positive and negative ratings, instead of positive ratings only, improves the sample complexity. Both questions arise in the design of recommender systems. We introduce a simple probabilistic user model, and analyze the performance of an online user-based CF algorithm. We prove that after an initial *cold start phase*, where recommendations are invested in exploring the user's preferences, this algorithm makes—up to a fraction of the recommendations required for updating the user's preferences—perfect recommendations. The number of ratings required for the cold start phase is nearly proportional to $1/p_{\mathrm{f}}$, and that for updating the user's preferences is essentially independent of p_{f}. As a consequence we find that, receiving positive and negative ratings instead of only positive ones improves the number of ratings required for initial exploration by a factor of $1/p_{\mathrm{f}}$, which can be significant.

1. Introduction

Recommender systems seek to identify the subset of a large collection of items that a user likes (Aggarwal, 2016). In practice, recommender systems often use collaborative filtering (CF) (Ekstrand et al., 2011) to identify items a given user likes, based on ratings that this user and a large number of other users have provided in the past. To this end, a user-based CF algorithm first identifies similar users, and then predicts the ratings of a given user from the ratings provided by similar users. In practice, recommender systems typically operate in an online fashion, i.e., items are recommended to users over time, and the ratings obtained in response to recommendations are used to improve future recommendations.

However, in many application areas of recommender systems, users only occasionally rate what they 'like', and never what they 'dislike'. E.g., in e-commerce, such as Amazon's recommender system, an item (or a set of items) is recommended to a user, and the user either purchases the item, which indicates a 'like', or the user does not purchase the item. Not purchasing the item, however, does not necessarily indicate a 'dislike', since the user might not even have considered the recommendation. Other examples of such feedback include implicit ratings such as viewing a webpage and listening to a song (Hu et al., 2008), and business-to-business recommender systems (Heckel et al., 2017). The problem of generating recommendations based on positive ratings only is known as one-class CF (Pan et al., 2008). The lack of negative ratings is often considered to make this problem challenging (Pan et al., 2008). However, it is unclear whether it is fundamentally more difficult in the absence of negative ratings to identify the user's preferences, in the sense that the sample complexity (i.e., the number of ratings required to make good recommendations) is fundamentally larger. Additionally, there is little theoretical understanding on how the probability of a user responding to a recommendation, p_{f}, affects the sample complexity of a one-class CF algorithm and in particular its *cold start time*, i.e., the number of recommendations the algorithm needs to invest in learning the user's preferences before being able to make good recommendations. In this paper, we address those two questions, that turn out to be closely related.

To this end, we introduce a probabilistic model for a one-class online recommender system and a corresponding online user-based CF algorithm, termed User-CF, and analyze its performance. Our model, and elements of our algorithm, are inspired by a related model and algorithm by Bresler et al. (2014) for the *two-class CF* problem, i.e.,

[1]University of California, Berkeley, California, USA. Correspondence to: Reinhard Heckel <heckel@berkeley.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

for a setup where positive *and* negative ratings are available. In a nutshell, each user in our model has a latent probability preference vector which describes the extent to which she likes or dislikes each item. Similar users have similar preference vectors. At a given time step $t = 0, 1, \ldots$, the User-CF algorithm recommends a single item to each user, typically different for each user. With probability specified by a corresponding preference vector, the user likes or dislikes the recommended item. If the user likes the item, the user rates it with probability p_{f}, and if the user does not like the item, no rating is given. An item that has been rated cannot be recommended again, since 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. While in practice the probability p_{f} could be different for each user, for ease of presentation, we assume that p_{f} is constant over all users. The goal of the User-CF algorithm is to maximize the number of recommendations that a users likes.

The User-CF algorithm consists of an exploitation step that recommends items that similar users have rated positively, and two kinds of exploration steps; one to learn the preferences of the users and the other to explore similarity between users.

Our main result, stated in Section 4, guarantees that after a certain cold start time in which the User-CF algorithm recommends the order of $(\log(N)/p_{\mathrm{f}}^2)^{\frac{1}{1-\alpha}}$ items to each user, a fraction $1 - c/p_{\mathrm{f}}$ of the remaining recommendations given by the User-CF algorithm are optimal. Here, α is a learning rate that can be chosen very close to zero, N is the number of users, and c a numerical constant. The cold start time is required to identify similar users and learn their preferences regarding a few items. We also show that any algorithm has to make on the order of $1/p_{\mathrm{f}}^2$ recommendations before it can make good recommendations, therefore the User-CF algorithm is near optimal. The fraction c/p_{f} of the remaining time steps is associated with learning the preferences of the users. This 'cost' of c/p_{f} does not have to be paid upfront, but is paid continuously: After the cold start time, the User-CF algorithm starts exploiting successfully. Again, a fraction of the recommendations proportional to $1/p_{\mathrm{f}}$ is necessary to learn the preference of the users. Our numerical results in Section 5 show that even if our data is not generated from the probabilistic model, but is based on real data, the cold start time and the fraction of time steps required to learn the preferences of the users are nearly proportional to $1/p_{\mathrm{f}}^2$ and $1/p_{\mathrm{f}}$, respectively.

As a consequence of this result, we find that obtaining positive and negative ratings instead of only positive ones, improves the number of ratings required for the initial cold start period by a factor of p_{f}. To see this, note that the expected number of ratings obtained by a user in a given number of time steps or equivalently after a given number of recommendations is proportional to p_{f}. Thus, the number of ratings required for the initial cold start time is inversely proportional to p_{f}, and the number of ratings required for continuously learning the preferences is independent of p_{f}. Since $p_{\mathrm{f}} = 1$ corresponds to users giving positive and negative feedback (no positive feedback implies dislike when $p_{\mathrm{f}} = 1$), the number of ratings required for the cold-start time is by a factor of $1/p_{\mathrm{f}}$ larger than the number of ratings required by a user-based CF algorithm that obtains positive and negative ratings.

Those findings are relevant for the design of recommender systems, since both p_{f} and whether positive, or negative and positive ratings are obtained can often be incorporated in the design of a recommender system. Therefore an understanding of the associated benefits and costs in terms of sample complexity, as provided in this paper, is important. We finally note that the goal of this paper is not to improve upon state-of-the art algorithms, but rather to inform the design of algorithms and what to expect in terms of sample complexity as a function of the various parameters involved.

Related literature: While to the best of our knowledge, this is the first work that *analytically* studies one-class CF in an online setting, theoretical results have been established for the two or multiple class CF problems. One of the first analytical results on user-based CF algorithms, an asymptotic performance guarantee under a probabilistic model, was established by Biau et al. (2010). Most related to our approach is the Collaborative-Greedy algorithm studied by Bresler et al. (2014) for the online two-class CF problem. The Collaborative-Greedy algorithm differs from our User-CF algorithm in selecting the nearest neighbors based on thresholding similarity, instead of selecting the k most similar users, and in the way preferences of the users are explored. This difference in the exploration steps is crucial for establishing that after the cold start period, our User-CF algorithm makes optimal recommendations in a fraction $1 - c/p_{\mathrm{f}}$ of the remaining time steps. Dabeer (2013) studies a probabilistic model in an online setup, and Barman & Dabeer (2012) study a probabilistic model in an offline setup, and state performance guarantees for a two-class user-based CF algorithm. Closely related to user-based CF is item-based CF. Item-based CF exploits similarity in item space by recommending items similar to those a given user has rated positively in the past. Our results do not extend trivially to item-based CF, since a corresponding analysis requires assumptions on the similarity in item space, and additionally the exploration strategies of item-based CF algorithms are considerably different. We do not discuss item based CF algorithms here, but refer to (Bresler et al., 2015) for a recent analysis of an item

based CF algorithm for the two-class CF problem. Next, we note that Deshpande & Montanari (2012) study recommender systems in the context of multi-armed bandits (Bubeck & Cesa-Bianchi, 2012). Specifically, Deshpande & Montanari (2012) consider a model where the (continuous) ratings are described by the inner product of a user and item feature vector, and assume the item feature vectors to be given.

A conceptually related online learning problem are multi-armed bandits with dependent arms (Pandey et al., 2007). Specifically, in this variant of the multi-armed bandit problem, the arms are grouped into clusters, and the arms within each cluster are dependent. The assignments of arms to clusters are assumed known. In our paper, we assume that users cluster in user types that have similar distributions. Therefore, the learning problem in our paper can be viewed as an multi-armed bandit problem with dependent arms, but the assignment of the arms to clusters is unknown.

Finally, we note that a class of learning problems reminiscent to that considered here is partial monitoring (Bartók et al., 2014). While partial monitoring has been studied in the context of recommender systems (Kveton et al., 2015), we are not aware of papers on partial monitoring in collaborative filtering.

Outline: In Section 2, we formally specify our model, motivate it, and state the CF problem. Sections 3 and 4 contain the User-CF algorithm and corresponding performance guarantee, respectively. In Section 5 we provide numerical results on real data. The proof of our main result can be found in the supplementary material.

2. Model and learning problem

In this section we introduce the probabilistic model and learning problem considered in this paper. As mentioned previously, this model is inspired by that in (Bresler et al., 2014) for the two-class CF problem.

Model: Consider N users and M items. A user may like an item ($+1$), or dislike an item (-1). Associated with each user is an (unknown) latent preference vector $\mathbf{p}_u \in [0,1]^M$ whose entries p_{ui} are the probabilities of user u liking item i. We assume that an item i is either "likable" for user u, i.e., $p_{ui} > 1/2 + \Delta$, for some $\Delta \in (0, 1/2]$, or "not likable", i.e., $p_{ui} < 1/2 - \Delta$. The hidden ranking R_{ui}^{hidden} is obtained at random as $R_{ui}^{\text{hidden}} = 1$ (like) with probability p_{ui}, and $R_{ui}^{\text{hidden}} = -1$ (dislike) with probability $1 - p_{ui}$. The ratings are stochastic to model that users are not fully consistent in their rating; the parameter Δ quantifies the inconsistency (or uncertainty or noise). The one-class aspect is incorporated in our model by assuming that users *never* reveal that they *dislike* an item.

Specifically, an CF algorithm operates on the model as follows. At each time step $t = 0, 1, \ldots$ the algorithm recommends a single item $i = i(t, u)$ to each user u—typically this item is different for each user—and obtains an realization of the binary random variable

$$R_{ui} = \begin{cases} Z_{ui} \sim \text{Bernoulli}(p_{\text{f}}), & \text{if } R_{ui}^{\text{hidden}} = 1, \\ 0, & \text{if } R_{ui}^{\text{hidden}} = -1 \end{cases}$$

in response, independently across u and i. It follows that $\mathbb{P}[R_{ui} = 1] = p_{ui} p_{\text{f}}$ and $\mathbb{P}[R_{ui} = 0] = 1 - p_{ui} p_{\text{f}},$. Here, p_{f} corresponds to the probability of a user reporting a positive rating. As mentioned before, while one might treat the slightly more general case of the probability p_{f} being different for each user, for ease of presentation, we assume that it is constant over the users. If $R_{ui} = 1$, user u consumes item i, and i will not be recommended to u in subsequent time steps. Note that $R_{ui} = 0$ means that either user u does not like item i ($R_{ui}^{\text{hidden}} = -1$), or user u did not respond to the recommendation. Therefore, if $R_{ui} = 0$, i may be recommended to u again in subsequent time steps. Finally, observe that if $p_{\text{f}} = 1$, the user provides positive *and* negative ratings, since $R_{ui}^{\text{hidden}} = 0$ implies $R_{ui} = -1$ if $p_{\text{f}} = 1$.

In order to make recommendations based on the user's preferences, we must assume some relation between the users. Following (Bresler et al., 2014), we assume that each user belongs to one of $K < N$ user types. Two users u and v belong to the same type if they find the same items likable, i.e., if $\mathbb{1}\{p_{ui} > 1/2 + \Delta\} = \mathbb{1}\{p_{vi} > 1/2 + \Delta\}$, for all items i. This does not require the preference vectors $\mathbf{p}_u$ and $\mathbf{p}_v$ of two users corresponding to the same user type to be equivalent. We note that this assumption could be relaxed by only assuming that users of the same type share a large fraction of the items that they find likable. We assume that the preference vectors belonging to the same type are more similar than those belonging to other types. Specifically, assume that for all $u \in [N]$, $[N] \coloneqq \{0, 1, \ldots, N-1\}$, and for some $\gamma \in [0, 1)$,

$$\gamma \min_{v \in \mathcal{T}_u} \langle \mathbf{p}_u, \mathbf{p}_v \rangle \geq \max_{v \notin \mathcal{T}_u} \langle \mathbf{p}_u, \mathbf{p}_v \rangle , \tag{1}$$

where $\mathcal{T}_u \subset [N]$ is the subset of all users that are of the same type as u. The smaller γ, the more distinct users of the same type are from users of another type. We further assume that each user likes at least a fraction ν of the items. This assumption is made to avoid degenerate situations were a user u does not like any item. Assuming that users cluster in the user-item space in different user types is common and is implicitly used by user-based CF algorithms (Sarwar et al., 2000), which perform well in practice. To further justify this assumption empirically, we plot in Figure 1 the clustering of user ratings of the Movielens 10 Million dataset (Harper & Konstan, 2015). Figure 1 shows that the user's ratings cluster both in user and in item space.

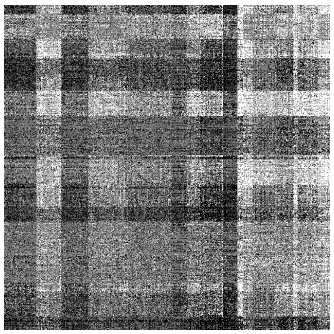

Figure 1. The user-items rating matrix consisting of a subset of the Movielens 10 Million dataset corresponding to the 1000 most rated movies (columns) and the 1000 users (rows) that rated most movies. The Movielens dataset consists of 10 Million movie ratings in $\{1, 2, 3, 4, 5\}$; we took the ratings ≥ 4 as 1, ratings ≤ 3 as -1, and missing ratings as 0; depicted in black, white, and gray.

Learning problem and reward: The goal of a CF algorithm is to maximize reward. A reasonable reward for the online CF problem is the expected number of recommendations that a user rates positively, i.e., the pseudo-reward

$$\sum_{t=0}^{T-1} \sum_{u=0}^{N-1} \mathbb{E}\left[R_{ui(u,t)}\right].$$

Here, $i(u,t)$ is the item recommended to u at time t. In an e-commerce setting this corresponds to the number of recommended products that a user buys. Note that due to the uncertainty of a user liking an item (the random rating R_{ui}^{hidden} might be -1 even when i is likable by u), we cannot expect to do better than maximizing the pseudo-reward.

In this paper our focus is on recommending likable items. Following (Bresler et al., 2014) we therefore consider the closely related accumulated reward defined as the expected total number of likable items ($p_{ui} > 1/2$) that are recommended by an algorithm up to time T:

$$\mathbb{E}\left[\text{reward}(T)\right] \coloneqq \sum_{t=0}^{T-1} \sum_{u=0}^{N-1} \mathbb{E}\left[X_{ui(u,t)}\right]. \qquad (2)$$

Here, $X_{ui(u,t)} = \mathbb{1}\{p_{ui} > 1/2\}$ is the indicator random variable that is equal to one if item $i(u,t)$ recommended to user u at time t is likable and zero otherwise (note that item i is chosen by the CF algorithm as a function of the responses $R_{u'i'}$ to recommendations (i', u') made at previous time steps, and is therefore a random variable).

3. User-CF algorithm

In this section we present our user-based CF algorithm (User-CF). In order to maximize reward the User-CF algorithm balances exploring, i.e., learning about the users, and exploiting, i.e., recommending items predicted to be likable based on previous ratings. To this end, the User-CF algorithm, formally introduced below, performs at time $t = 0, 1, \ldots$, either an *preference exploration*, *similarity exploration*, or *exploitation step*.

An *exploitation step* first identifies the k most similar users in terms of their *rating vectors* $\mathbf{r}_v \in \{0,1\}^M$, for a given user u. The rating vectors consist of the responses R_{ui} of users to recommendations (u, i) made by the User-CF algorithm at previous time steps. The exploitation step proceeds by recommending the item that has received the largest number of positive ratings from the nearest neighbors of u in previous time steps. For an exploitation step to be successful, it is crucial to find similar users and learn their preferences effectively. This is accomplished with *similarity exploration* steps, that recommend the same items to all users, and *preference exploration* steps that recommend random items to certain subsets of the users. Before formally stating the User-CF algorithm, we illustrate its main steps using a toy example.

Example 1 *Consider $N = 6$ users and $M = 5$ items, with preference vectors $\mathbf{p}_u$ and rating vectors $\mathbf{r}_u$ at time $t = 2$ given by*

$$\begin{bmatrix} \mathbf{p}_0^T \\ \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \mathbf{p}_3^T \\ \mathbf{p}_4^T \\ \mathbf{p}_5^T \end{bmatrix} = \begin{bmatrix} .9 & .8 & .9 & .1 & .1 \\ .9 & .8 & .9 & .2 & .3 \\ .9 & .8 & .9 & .1 & .2 \\ .1 & .3 & .1 & .8 & .7 \\ .2 & .2 & .1 & .9 & .9 \\ .1 & .1 & .3 & .7 & .8 \end{bmatrix},$$

$$\begin{bmatrix} \mathbf{r}_0^T \\ \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \\ \mathbf{r}_4^T \\ \mathbf{r}_5^T \end{bmatrix} = \begin{bmatrix} \boxed{1} & 0 & 0 & 0 & \textcircled{0} \\ \boxed{1} & \textcircled{1} & 0 & 0 & 0 \\ \boxed{1} & \textcircled{1} & 0 & 0 & 0 \\ \boxed{0} & 0 & \textcircled{1} & 0 & 0 \\ \boxed{0} & 0 & 0 & \textcircled{1} & 0 \\ \boxed{0} & 0 & 0 & \textcircled{0} & 0 \end{bmatrix}.$$

Users $0, 1, 2$ are of the same type as they find items $0, 1, 2$ likable, and users $3, 4, 5$ belong to a second type as they find items $3, 4$ likable. The preference vectors are obtained by executing at time step 0 a preference exploration step, that recommends the randomly chosen items $4, 1, 1, 2, 3, 3$ to users $0, 1, \ldots, 5$, respectively, and a similarity exploration step that recommends item 0 to all users. The responses R_{ui} obtained from the similarity and preference exploration step are marked with rectangles and circles, respectively. Consider recommending an item to user 0 with an exploitation step at time 2. The $k = 2$ nearest neighbors of $u = 0$ are $\mathcal{N}_u = \{1, 2\}$ ($\langle \mathbf{r}_0, \mathbf{r}_v \rangle = 1$ for $v = 1, 2$ and $\langle \mathbf{r}_0, \mathbf{r}_v \rangle = 0$ for $v = 3, 4, 5$). Since $\sum_{v \in \mathcal{N}_u} \mathbf{r}_{vi}$ is maximized for $i = 1$ ($\mathbf{r}_{vi}$ is the i-th entry of $\mathbf{r}_u$), item 1 is recommended to user u, which happens to be a likable

item, as desired.

We next formally describe the User-CF algorithm, and explain the intuition behind the specific steps. Input parameters of the User-CF algorithm are learning rates α and $\eta \in (0,1)$ (e.g., $\eta = 1/2$) relevant for similarity and preference exploration steps, respectively, a batch size Q relevant for preference exploration steps, and finally the number of nearest neighbors k, relevant for exploitation steps. Our results guarantee that for a range of input parameters that depends on properties of the model such as the number of user types and p_{f}, the User-CF algorithm performs essentially optimally. While we may or may not have prior knowledge of those model parameters, in practice, we can optimize for the hyper-parameters of the User-CF algorithm by using cross validation.

At initialization, the User-CF algorithm generates a random permutation π of the items $[M]$ required by the similarity exploration step. Furthermore, it splits the item space into M/Q random, and equally sized[1] subsets of cardinality Q (batches), denoted by $\mathcal{Q}_q \subset [M]$, $q = 0, \ldots, M/Q - 1$.

At time steps $t = \lfloor \eta Q q \rfloor, q = 0, \ldots, M/Q - 1$, the User-CF algorithm performs preference exploration steps. At all other time steps, with probabilities $p_J = \frac{1}{(t-q)^\alpha}, q = \lfloor t/(\eta Q) \rfloor$, and $p_E = 1 - p_J$, the algorithm performs similarity exploration and exploitation steps, respectively.

Similarity exploration step: For each user u, recommend the first item i in the permutation π that has not been recommended to user u in previous steps of the algorithm. This step explores the item space and its important for selecting 'good' neighborhoods. Performing a sufficient number of similarity exploration steps allows to guarantee that the nearest neighbors of a given user u are of the same type.

Preference exploration step: At time $t = \lfloor \eta Q q \rfloor$, recommend to each user u an item, chosen independently and uniformly at random from $\mathcal{Q}_q$, that has not been rated by u in previous time steps. This step is important to learn the preferences of users.

Exploitation step: For all users u, estimate the probability of u liking a given item i as

$$\hat{p}_{ui} = \begin{cases} \frac{1}{n_{ui}} \sum_{v \in \mathcal{N}_u} R_{vi}, & \text{if } n_{ui} > 0, \\ 0, & \text{if } n_{ui} = 0. \end{cases} \tag{3}$$

Here, R_{vi} is the rating of user v for item i obtained in previous time steps (we use the convention $R_{vi} = 0$ if no rating was obtained at a previous time step). Next, $\mathcal{N}_u$ is the set of users corresponding to the k largest values of $\left\langle \mathbf{r}_u^{\mathrm{sim}}, \mathbf{r}_v^{\mathrm{sim}} \right\rangle, v \in [N]$. Here, $\mathbf{r}_u^{\mathrm{sim}} \in \{0,1\}^M$ is the vector containing only the responses R_{ui} of user u to recommendations i given in previous *similarity* exploration steps up to time t, and is zero otherwise. Moreover, n_{ui} is the number of users in $\mathcal{N}_u$ that received recommendation i. Finally, for each user u, recommend an item i that maximizes $\hat{p}_{ui'}$ over all items i' that have not been rated yet.

The idea behind the User-CF algorithm is as follows. An exploitation step recommends likable items to u if $a)$ most of the neighbors of u are of the same user type as u, and if $b)$ the items are sufficiently well explored so that $\hat{p}_{ui}$ indicates whether i is likable by u (i.e., $p_{ui} > 1/2$) or not, for all i. A large portion of the first few steps is likely to be spent on similarity exploration. This is sensible, as we need to ensure that $a)$ is satisfied in order to make good recommendations. As time evolves, the User-CF algorithm randomly explores batches of items, one batch at a time, in order to estimate the preferences of the users regarding the items in the corresponding batches. Note that if the User-CF algorithm would explore the entire item space at once, e.g., by recommending an item chosen at random from all items, the time required for the algorithm to make 'good' recommendations would grow linearly in M, and M might be very large. By splitting the item space into batches $\mathcal{Q}_q$, the User-CF algorithm can start exploiting without having learned the preferences of the users regarding *all* items. Finally note that the User-CF algorithm may recommend the same item at several time steps (unless the item is rated by the user); this is sensible in particular if p_{f} is small.

4. Main result

Our main result, stated below, shows that after a certain cold start time, the User-CF algorithm produces essentially optimal recommendations.

Theorem 1 *Suppose that there are at least $\frac{N}{2K}$ users of the same type, for all user types, and that condition (1) holds for some $\gamma \in [0,1]$, which ensures that user types are distinct. Moreover, assume that at least a fraction ν of all items is likable to a given user, for all users. Pick $\delta > 0$ and suppose that there are sufficiently many users per user type:*

$$\frac{N}{K} \geq \frac{c}{\nu p_{\mathrm{f}} \Delta^2} \log(M/\delta) \log(4/\delta). \tag{4}$$

Set

$$T_{\mathit{start}} := \left(\frac{\tilde{c} \log(N/\delta)}{p_{\mathrm{f}}^2 (1-\gamma)^2 \nu} \right)^{\frac{1}{1-\alpha}} \left(1 - \max\left(\frac{1}{T}, \frac{K}{N} \frac{c \log(M/\delta)}{p_{\mathrm{f}} \Delta^2} \right) \right),$$

where $\tilde{c}$ is a numerical constant. Then, for appropriate choices[2] of the parameters η, k, and Q, the expected reward accumulated by the User-CF algorithm up to time

[1] We assume for simplicity that M is divisible by Q, if this is not the case, batch $\lceil M/Q \rceil - 1$ may simply contain less than Q items.

[2] Specifically, $\eta = c_1 \nu$, $k = c_2 \frac{N}{K}$, and $Q =$

$T \in [T_{start}, \frac{4}{5}\nu M p_{\mathrm{f}}]$ *satisfies*

$$\frac{\mathbb{E}\left[\mathrm{reward}(T)\right]}{NT} \geq \left(1 - \frac{T_{start}+1}{T} - 2^{\alpha}\frac{(T-T_{start})^{1-\alpha}}{T(1-\alpha)} - \frac{K}{N}\frac{c\log(M/\delta)}{p_{\mathrm{f}}\Delta^2}\right)(1-\delta). \quad (5)$$

Theorem 1 states that after an initial cold start time on the order of T_{start}, the User-CF algorithm recommends only likable items up to a fraction $\frac{K}{N}\frac{c\log(M/\delta)}{p_{\mathrm{f}}\Delta^2}$ of the time steps. This follows since a oracle that only recommends likable items obtains an reward of $\mathbb{E}\left[\mathrm{reward}(T)\right] = NT$. This yields the claim from the introduction, that after the cold start time, a fraction $1 - c/p_{\mathrm{f}}$ of all recommendations made by the User-CF algorithm are likable. Note that condition (4) allows the number of user types, K, to be near linear in the number of users, N.

We note that the particular choice of the parameters of the User-CF algorithm in Theorem 1 is mainly out of expositional convenience; the supplementary material contains a more general statement.

Theorem 1 is proven by showing that after the initial cold start time, exploration steps recommend likable items with very high probability. An exploration step recommends a likable item to user u provided that

a) most of the nearest neighbors of u are of the same user type, and

b) the items are sufficiently well explored so that the maximum of $\hat{p}_{ui}$ over items not rated yet corresponds to a likable item.

For a), we use that the user types are sufficiently distinct and that most of the nearest neighbors of u are of the same user type. The former is ensured by condition (1), and the latter holds after the initial cold start time which ensures that sufficiently many similarity exploration steps have been executed. After the initial cold start time, most of the nearest neighbors of u are of the same user type, and essentially no further cost is required to learn the neighborhoods. This is reflected in the lower bound (5), by the terms depending on T becoming negligible as T becomes large compared to T_{start}. Note the dependence of T_{start} on γ; the more similar the user types are (i.e., the closer γ is to 1), the longer it takes till User-CF is guaranteed to find good neighborhoods.

4.1. Dependence on p_{f} is nearly optimal

The cold start time of the User-CF algorithm guaranteed by Theorem 1 is proportional to $(1/p_{\mathrm{f}}^2)^{\frac{1}{1-\alpha}}$. For small learning rates α, this scaling can not be improved significantly, as the following results shows.

Proposition 1 *Suppose that there are more items than user types, i.e., $M \geq K$. Fix $\lambda \in (0,1)$. Then there is a set of users with at least $\frac{N}{2K}$ users of the same type, for each user type, with preference vectors such that for all $T \leq \frac{\lambda}{p_{\mathrm{f}}^2}$, the expected reward of any online algorithm is upper bounded by $\frac{\mathbb{E}[\mathrm{reward}(T)]}{TN} \leq \lambda + \frac{1}{K}$.*

Proposition 1 shows that if the cold start time is significantly smaller than $1/p_{\mathrm{f}}^2$, then there are problem instances for which any algorithm mostly recommends non-likable items. The proposition is a consequence of the fact that after making on the order of $1/p_{\mathrm{f}}^2$ recommendations, for many users we did not obtain any rating. A consequence of the proposition is that the cold start time of the User-CF algorithm is near optimal.

Recall that even after the initial cold start time, a constant fraction of $\frac{K}{N}\frac{c\log(M/\delta)}{p_{\mathrm{f}}\Delta^2}$ of the recommendations might be non-likable. This fraction is the cost for establishing b). Specifically, in order to ensure b), the User-CF algorithm needs to recommend sufficiently many items to the k neighbors of u so that $\hat{p}_{ui}$ indicates whether user u likes item i or not, or more precisely such that the maximum of $\hat{p}_{ui}$ over items not rated yet corresponds to a likable item. This is established by showing that $\hat{p}_{ui} > p_{\mathrm{f}}/2$ for all likable items $i \in \mathcal{Q}_q, q = 0, \ldots, \frac{t}{\eta Q} - 1$, and $\hat{p}_{ui} < p_{\mathrm{f}}/2$ for all other items. Since the expected number of positive ratings per recommendation is proportional to p_{f}, the number of ratings required to ensure that $p_{ui} > p_{\mathrm{f}}/2$ is proportional to $1/p_{\mathrm{f}}$.

4.2. One versus two class CF

Recall that $p_{\mathrm{f}} = 1$ implies that users provide positive and negative ratings, and that the User-CF algorithm is nearly optimal in p_{f}. Since the expected number of ratings obtained by a user in a given number of time steps is proportional to p_{f}, a consequence of our result is that receiving positive and negative ratings instead of only positive ones improves the number of ratings required for initial exploration by a factor of $1/p_{\mathrm{f}}$, which can be significant.

We finally note that Bresler et al. (2014) proved a performance guarantee for a closely related two-class collaborative CF algorithm termed Collaborative-Greedy. Bresler et al. (2014) consider the regime where the number of users is much larger than the number of items, i.e., $N = O(M^C), C > 1$ and additionally the number of user types obeys ($N = O(KM)$). For this regime, Theorem 1 particularized to $p_{\mathrm{f}} = 1$ essentially reduces to Theorem 1 in (Bresler et al., 2014) (there are some further minor differences). However, our result particularized to the two-class case also holds when the number of items is much larger than the number of users, and allows the number of user types to be near linear in the number of users. This improvement is due to differences in the preference explo-

$c_3 \frac{kp_{\mathrm{f}}\Delta^2}{\log(M/\delta)}$, for numerical constants c_1, c_2, and c_3.

ration strategies of the algorithms.

4.3. Alternative exploration strategies

While there are other sensible preference exploration strategies, the essential element of our approach is to split up the item space into subsets of items $\mathcal{Q}_0, \mathcal{Q}_1, \ldots$, start by exploring $\mathcal{Q}_0$, then allow for exploitation steps, continue with exploring $\mathcal{Q}_1$, again allow for exploitation steps and so forth. If one explores instead the whole item space $[M]$ at the beginning, the learning time required for $\hat{p}_{ui}$ to indicate whether an item is likable or not, is proportional to M, and can therefore be very large. To see this, consider a preference exploration step that recommends a single item to all users, chosen uniformly at random from the set of all items $[M]$. The expected number of ratings obtained by executing T_r such preference exploration steps relevant for estimating whether $p_{ui} > 1/2$, is the expected number of neighbors of u to which i has been recommend, and is therefore proportional to $T_r k/M$. To ensure that this expectation is larger than 0, T_r has to be on the order or M (provided that the other parameters are fixed).

5. Numerical results

In this section, we simulate an online recommender system based on real-world data in order to understand whether the User-CF algorithm behaves as predicted by Theorem 1, even when the data is not generated by the probabilistic model, but is based on real data. While an ideal dataset to validate our algorithm would consist of the ratings from all users for all items, the vast majority of ratings in standard CF datasets such as the Netflix or Movielens dataset (consisting of movie ratings) are unknown. To obtain a dataset with a higher proportion of ratings, following (Bresler et al., 2014), we consider the subset of the Movielens dataset corresponding to frequently rated items and to users that have rated many items. The Movielens dataset consists of 10 Million movie ratings in $\{1, 2, 3, 4, 5\}$; we took the ratings ≥ 4 as 1, ratings ≤ 3 as -1, and missing ratings as 0. Many of the items (movies) in the dataset have a significant bias towards a positive or negative rating. To make sure our results are not due to exploiting such biases, we only select frequently rated items out of the (approximately) unbiased items. Note that a nearest neighbor based algorithm, like the User-CF algorithm, performs well in case the item ratings are very biased even when the neighborhoods are randomly selected. Of the resulting dataset, denoted by $\mathbf{R}_{\text{ML}} \in \{-1, 0, 1\}^{1000\times 500}$, 18.1% of the ratings are 1, 17.1% are -1, and the remaining ones are unknown and therefore set to 0.

One class versus two class CF: We start with comparing the qualitative behavior of the User-CF algorithm to a two-class version of the User-CF algorithm. The two-class version of the User-CF algorithm differs from the one-class version in taking into account the negative ratings. Specifically, the ratings R_{vi} in (3) for the two-class version or the User-CF algorithm are set to $-1, 0$, and 1, if a negative, none, or a positive rating was obtained as a response to a recommendation. We performed the following experiment. If the User-CF algorithm recommends item i to user u, it obtains the rating $R_{ui} = 1$ in response provided that $[\mathbf{R}_{\text{ML}}]_{ui} = 1$ ($[\mathbf{R}_{\text{ML}}]_{ui}$ denotes the (u, i)-th entry of $\mathbf{R}_{\text{ML}}$), and $R_{ui} = 0$ otherwise, while the two-class User-CF algorithm obtains $R_{ui} = [\mathbf{R}_{\text{ML}}]_{ui}$ in response. We allow both algorithms to only recommend an item to a given user once, so after $M = 500$ time steps, all items have been recommend to all users. We measure performance in terms of the accumulated reward, defined as

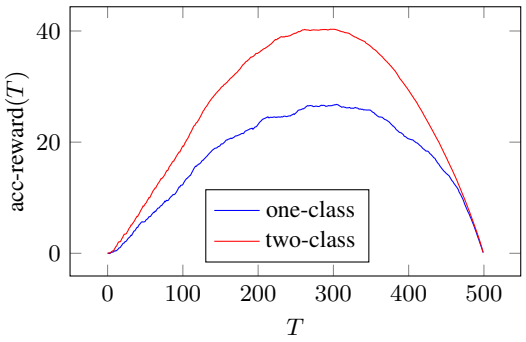

Figure 2. Comparison of one and two class recommenders.

$$\text{acc-reward}(T) := \sum_{t=0}^{T-1} \text{reward}(t),$$

$$\text{reward}(t) := \frac{1}{N} \sum_{u=0}^{N-1} [\mathbf{R}_{\text{ML}}]_{ui(u,t)}, \tag{6}$$

where $i(u, t)$ is the item recommended to user u by the corresponding variant of the User-CF algorithm. The results, depicted in Figure 3, show that the two-class recommender performs better, as expected, since it obtains significantly more ratings. Specifically, the expected number of ratings it obtains is almost twice the expected number of ratings the one-class User-CF algorithm obtains. After having recommend most of the likable items, mostly non-likable are left to recommend, which explains the inverse U-shape in Figure 3.

Dependence of User-CF on p_{f}: We next validate empirically that the cold start time and number of preference exploration steps (needed to learn the preferences of the users) scale as $1/p_{\text{f}}^2$ and $1/p_{\text{f}}$, respectively. We start with the former. To this end, we split the items into two random disjoint sets $\mathcal{I}_1 \subset [M]$ and $\mathcal{I}_2 \subset [M]$ of equal cardinality. We then perform the following experiment for

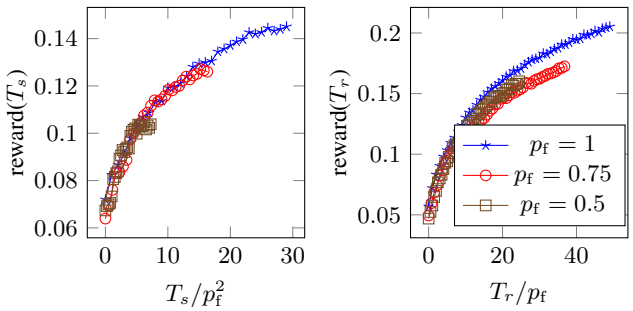

Figure 3. Left: Reward obtained from a single exploitation step, after performing T_s similarity exploration steps and a fixed number of preference exploration steps. Right: Reward obtained from a single exploration step, after performing T_r preference exploration steps and a fixed number of similarity exploration steps.

$p_f \in \{1, 0.75, 0.5\}$. We start by recommending $\frac{3M}{kp_f}$ items, chosen uniformly at random from $\mathcal{I}_2$ to each user, and, provided the corresponding rating is positive ($[\mathbf{R}_{\text{ML}}]_{ui} = 1$), we provide this rating to the User-CF algorithm with probability p_f. The expected number of *positive* ratings obtained is therefore independent of p_f. Those preference exploration steps make sure that the preferences of the items in $\mathcal{I}_2$ are explored well. We then perform T_s similarity exploration steps on the items in the sets $\mathcal{I}_1 = \{i_0, \dots, i_{M/2-1}\}$, by recommending item i_t to user u at $t = 0, \dots, T_s - 1$. If $[\mathbf{R}_{\text{ML}}]_{ui_t} = 1$ then we provide the rating $[\mathbf{R}_{\text{ML}}]_{ui_t}$ to the User-CF algorithm with probability p_f. After T_s such similarity exploration steps, we perform an exploitation step. In Figure 3 we plot the reward defined in (6) obtained by the exploitation step, over T_s/p_f^2. The results confirm that the cold start time required to find 'good' neighborhoods scales inversely proportional to p_f^2, since all three curves lie on top of each other.

Next, we demonstrate that the number of preference exploration steps required to learn the preferences of the users is proportional to $1/p_f$. To this end, we perform the same experiment as above, this time, however, we first perform $T_s = 25/p_f^2$ similarity exploration steps on the items in the set $\mathcal{I}_1$, and then perform T_r preference exploration steps by recommending T_r items, chosen uniformly at random from $\mathcal{I}_2$ to each user. As before, if $[\mathbf{R}_{\text{ML}}]_{ui} = 1$, the rating $[\mathbf{R}_{\text{ML}}]_{ui}$ is provided to the algorithm with probability p_f. In Figure 3 we plot the reward obtained from performing a single exploitation step after T_r such preference exploration steps over T_r/p_f. The results indicate that, as predicted by our theory, the number of preference exploration steps required to learn the preferences is proportional to p_f, as the curves for different p_f lie on top of each other. As mentioned previously, this is not surprising, as the number of positive ratings obtained is proportional to p_f.

References

Aggarwal, Charu C. *Recommender systems: The textbook*. Springer, 2016.

Bardenet, Rémi and Maillard, Odalric-Ambrym. Concentration inequalities for sampling without replacement. *Bernoulli*, 21(3):1361–1385, 2015.

Barman, Kishor and Dabeer, Onkar. Analysis of a collaborative filter based on popularity amongst neighbors. *IEEE Trans. Inf. Theory*, 58(12):7110–7134, 2012.

Bartók, Gábor, Foster, Dean P., Pál, Dávid, Rakhlin, Alexander, and Szepesvári, Csaba. Multi-armed bandit problems with dependent arms. *Math. Oper. Res.*, 39(4): 967–997, June 2014.

Biau, Gérard, Cadre, Benoît, and Rouvière, Laurent. Statistical analysis of k-nearest neighbor collaborative recommendation. *Ann. Stat.*, 38(3):1568–1592, 2010.

Bresler, Guy, Chen, George H, and Shah, Devavrat. A latent source model for online collaborative filtering. In *Advances in Neural Information Processing Systems*, pp. 3347–3355. 2014.

Bresler, Guy, Shah, Devavrat, and Voloch, Luis F. Regret guarantees for item-item collaborative filtering. *arXiv:1507.05371*, 2015.

Bubeck, Sébastien and Cesa-Bianchi, Nicolò. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5, 2012.

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

Deshpande, Yash and Montanari, Andrea. Linear bandits in high dimension and recommendation systems. In *Annual Allerton Conference on Communication, Control, and Computing*, pp. 1750–1754, October 2012.

Ekstrand, Michael D., Riedl, John T., and Konstan, Joseph A. Collaborative filtering recommender systems. *Found. Trends Hum.-Comput. Interact.*, 4(2):81–173, 2011.

Harper, F. Maxwell and Konstan, Joseph A. The MovieLens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, 2015.

Heckel, Reinhard, Vlachos, Michail, Parnell, Thomas, and Dünner, Celestine. Scalable and interpretable product recommendations via overlapping co-clustering. In *IEEE International Conference on Data Engineering*, 2017.

Hu, Yifan, Koren, Yehuda, and Volinsky, Chris. Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining*, pp. 263–272, 2008.

Kveton, Branislav, Szepesvari, Csaba, Wen, Zheng, and Ashkan, Azin. Cascading Bandits: Learning to Rank in the Cascade Model. In *International Conference on Machine Learning*, pp. 767–776, 2015.

Pan, Rong, Zhou, Yunhong, Cao, Bin, Liu, N.N., Lukose, R., Scholz, M., and Yang, Qiang. One-class collaborative filtering. In *IEEE International Conference on Data Mining*, pp. 502–511, 2008.

Pandey, Sandeep, Chakrabarti, Deepayan, and Agarwal, Deepak. Multi-armed Bandit Problems with Dependent Arms. In *International Conference on Machine Learning*, ICML '07, pp. 721–728, New York, NY, USA, 2007.

Sarwar, Badrul, Karypis, George, Konstan, Joseph, and Riedl, John. Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce*, pp. 158–167, 2000.