Combating the Cold-start User Problem in Collaborative Filtering Recommender Systems

by

Sampoorna Biswas

B.Tech. Computer Science, Indraprastha Institute of Information Technology, New Delhi, 2014

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

December 2016

Abstract

For tackling the well known cold-start user problem in collaborative filtering recommender systems, one approach is to recommend a few items to a cold-start user and use the feedback to learn her preferences. This can then be used to make good recommendations to the cold user. In the absence of a good initial estimate of the preferences, the recommendations are like random probes. If the items are not chosen judiciously, both bad recommendations and too many recommendations may turn off a user. We study the cold-start user problem for the two main types of collaborative filtering methods – neighbourhood-based and matrix factorization. We formalize the problem by asking what are the b (typically a small number) items we should recommend to a cold-start user, in order to learn her preferences best, and define what it means to do that under each framework. We cast the problem as a discrete optimization problem, called the *optimal interview design* (OID) problem, and study two variants – OID-NB and OID-MF – for the two frameworks. We present multiple non-trivial results, including NP-hardness as well as hardness of approximation for both. We further study supermodularity/submodularity and monotonicity properties for the objective functions of the two variants. Finally, we propose efficient algorithms and comprehensively evaluate them. For OID-NB, we propose a greedy algorithm, and experimentally evaluate it on 2 real datasets, where it outperforms all the baselines. For OID-MF, we discuss several scalable heuristic approaches for identifying the b best items to recommend to the user and experimentally evaluate their performance on 4 real datasets. Our experiments show that our proposed accelerated algorithms significantly outperform the prior art, while obtaining similar or lower error in the learned user profile as well as in the rating predictions made, on all the datasets tested.

Preface

This thesis is submitted in partial fulfilment of the requirements for a Master of Science degree in Computer Science. All the work presented in this dissertation are original and independent work of the author, performed under the supervision of Prof. Laks V. S. Lakshmanan

Chapter 4 is part of work conducted in collaboration with Prof. Laks V. S. Lakshmanan, and Prof. Senjuti Basu Roy. I was responsible for deriving all the technical results and the experimental evaluation.

Table of Contents

List of Tables

List of Figures

Acknowledgements

First of all, I would like to thank my supervisor Dr. Laks V.S. Lakshmanan for his invaluable guidance, advice and patience. This thesis would not be what it is without his help at every step of the way, from being a sounding board for my ideas to helping me write the thesis. He taught me how to critically analyze research, and write technical content, and I really value all the time he dedicated to our long research discussions.

I would also like to thank all the other professors and collaborators who helped me along the way, in particular, Dr. David Poole, for being my second reader and providing feedback on my work, and Dr. Senjuti Basu Ray and Dr. Nick Harvey, for insightful discussions and helping me polish my work.

I am also thankful for my labmates at the Data Mining and Management lab; I learned a lot from them. In particular I would like to thank Glenn Bevilacqua and Sharan Vaswani for all the tips, tricks and machine learning insider information without which my experiments may not have proceeded as smoothly. And finally I would like to thank my family, and friends all around the world, thanks for your love and support, and for giving me a break from research from time to time!

Chapter 1

Introduction

Recommender systems have emerged as a popular solution to the information overload problem and have been successfully deployed in many application domains such as recommendation of products (books, music, movies, etc.), services (e.g., restaurants), and content (e.g., search queries, hashtags, and other online content). Owing to their wide appeal, various approaches have been developed for generating good recommendations that are likely to appeal to their users.

Broadly, there are two main methods for producing recommendations

- Collaborative filtering (CF) systems [13], which exploit the intuition that users may like items preferred by users with tastes or interests that are similar to theirs, or that users may enjoy items similar to those that they have enjoyed in the past, or a combination of these ideas

- Content-based systems, which exploits the idea that users would like items that have similar properties (eg. movies of the same genre, books by the same author) to items that they have enjoyed previously or that users with similar properties (eg. same geographic location, similar demographics) would have similar tastes

Methods based on collaborative filtering can be further divided into [2]

- Memory-based methods [10], which recommend items based on heuristics aggregated over similar users or items

- Model-based methods, which build statistical models for the rating data and use those to make predictions

A common memory-based approach is to construct *neighbourhoods* of similar items or users, and recommend items that have been enjoyed by one user but not yet consumed by another user with similar tastes. Among model-based approaches, an approach that has been particularly successful is the so-called matrix factorization (MF) approach, which assumes a latent

factor model of low dimensionality for users and items, which are learned by factoring the matrix of observed ratings [26]. Compared to content-based filtering, collaborative filtering has been found to result in higher quality recommendations and is more scalable. Hence, CF is the focus of this thesis.

An important challenge faced by any recommender system is the so-called *cold-start user* and *cold-start item* problem [3, 42]. The former occurs when a new user joins the system and the latter when a new item becomes available or is added to the system's inventory. Without past user-item interactions, collaborative filtering cannot determine similarity of users/items, while content-based filtering requires additional metadata for every new user and item. In this thesis, we study the cold-start problem for users under both neighbourhood-based and matrix factorization settings.

A simple approach to mitigate this problem for CF techniques, is to present a certain minimum number of items to the cold-start user and obtain the user's feedback on them. A key question is *how to select items to recommend to a cold-start user*. Approaches that have been explored in the literature for tackling this problem have mainly tended to be ad hoc and heuristic in nature. E.g., [16, 23, 50] explore an offline strategy for selecting the items while [12, 30, 49] develop an online strategy for recommending items to cold-start users. In the online approach, the item recommended to a cold-start user takes into account her feedback on the previous recommendation, whereby the user profile is updated and used for making further recommendations. While these works report empirical results based experiments conducted on some datasets, unfortunately, these works do not formulate the item selection problem in a rigorous manner and do not analyze its computational properties. Furthermore, no comprehensive scalability experiments have been reported on their proposed strategies for item selection.

In this thesis, we focus on the cold-start user problem and formulate the item selection problem for cold-start users as a discrete optimization problem. We study this both for a latent factor model based on probabilistic matrix factorization [39], and for a neighbourhood-based system for our underlying recommender system. Since user attention and patience is limited, we assume that there is a budget b on the number of items which we can request feedback from a cold-start user. The main question we then study is, *how to select the b best items to recommend to such a user that will allow the system to learn the user's preferences as well as possible*. We formalize what it means to learn the user's preferences best under the two settings. For the neighbourhood-based system, we define that as being able to predict future ratings with low error, for the most number of items. The rationale

is that it is not very useful to know what the user likes or dislikes (given by rating prediction accuracy) if it is only on a handful of items; we want to maximize the number of items for which we can make accurate predictions, also known as prediction coverage [20]. For the matrix factorization system, we define that as learning the user's latent factor vector (the user's *profile*) with low error. The motivation is that if the user profile is learned well, it will pay off in allowing the system to make high quality recommendations to the user in the future. We formulate the item selection problem as a discrete optimization problem, called *optimal interview design* (OID), where the items selected can be regarded as questions selected for interviewing the cold-start user for her feedback on those items. For the two settings, we study two variants of the problem, which we call OID-NB and OID-MF for the neighbourhood-based system and the matrix factorization system respectively.

A challenge with OID-MF is defining the true user profile against which to measure the error of a learned profile. This is necessary for defining the objective function we need to optimize with our choice of b items. The difficulty is that there is no prior information on a cold-start user. We address this by showing that under reasonable assumptions, which will be made precise in Section 2.1, we can directly express the difference between the learned user profile and the true user profile in terms of the latent factors of the b items chosen. This allows us to reason about the quality of different choices of b items and paves the way for our optimization framework (Section 4.2).

We establish that both OID-NB and OID-MF problems are NP-hard. We show that the OID-NB problem is monotone and submodular, so the proposed greedy algorithm provides a $(1 - 1/e)$-approximation to the optimum. We subsequently also show that the OID-MF problem is NP-hard to approximate to within a factor $\frac{\alpha}{\theta}$, where α and θ depend on the problem instance (Section 4.3.2). Further, we show that its objective function, i.e., least squared error between the true and learned user profile, is neither submodular nor supermodular, suggesting efficient approximation algorithms may be unlikely to exist (Section 4.3.4).

Finally, we propose efficient algorithms and comprehensively evaluate them. For OID-NB, we propose a greedy algorithm (Section 3.3), and experimentally evaluate it on 2 real datasets (Section 3.4), where it outperforms all the baselines. For OID-MF, we discuss several scalable heuristic approaches for identifying the b best items to recommend to the user (Section 4.4) and experimentally evaluate their performance on 4 real datasets. Our experiments show that our proposed accelerated algorithms significantly out-

perform the prior art, while obtaining similar or lower error in the learned user profile as well as in the rating predictions made, on all the datasets tested (Section 4.5).

Chapter 2

Background and Related Work

2.1 Background

In this section, we summarize the relevant notions on collaborative filtering (CF) that are used in our problem formulation and further technical development.

2.1.1 Recommender Systems

Most recommender systems use a rating/interaction matrix, R, of size $n \times m$, of users, $\mathcal{U} = \{u_1, u_2, ..., u_m\}$, and items, $\mathcal{I} = \{v_1, v_2, ..., v_n\}$. Typically the matrix is sparse, and has only very few known ratings. Let each entry R_{ij} in the matrix be drawn from a rating scale and represent whether a user u_i likes a particular item v_j or not. This preference information can be gained through explicit or implicit feedback. For implicit feedback, it is typically binary, but for explicit feedback, the rating scale can be anything, for example, ranging from 1 to 5. We focus on learning the profile of a single cold-start user. This would be represented as an entire row of unknown ratings in R.

2.1.2 Neighbourhood-based Collaborative Filtering

Neighbourhood-based methods construct similarity *neighbourhoods* of users or items and aggregate statistics across them to predict future ratings. Let $\hat{R}_{ij}$ indicate the predicted rating for user u_i and item v_j. User based methods use u_i's similarity to other users who have rated v_j, to predict $\hat{R}_{ij}$, while item based methods use u_i's ratings on items similar to v_j to predict $\hat{R}_{ij}$.

For the former, we first construct u_i's neighbourhood by computing its similarity score, $w(u_i, u_k)$, with every other user $u_k \in \mathcal{U} \setminus u_i$. Let I_{u_i} be the set of items rated by u_i. For user-user similarity, Pearson correlation is used

[40] as follows,

$$w(u_i, u_k) = \frac{\sum_{v_j \in I_{u_i,u_k}} (R_{ij} - \bar{R}_i)(R_{kj} - \bar{R}_k)}{\sqrt{\sum_{v_j \in I_{u_i,u_k}} (R_{ij} - \bar{R}_i)^2} \cdot \sqrt{\sum_{v_j \in I_{u_i,u_k}} (R_{kj} - \bar{R}_k)^2}} \tag{2.1}$$

Here, I_{u_i,u_k} denotes the set of items rated by both u_i and u_k, i.e., $I_{u_i,u_k} = I_{u_i} \cap I_{u_k}$, and $\bar{R}_i$ denotes the average rating given by u_i. Note that $w(u_i, u_k) = w(u_k, u_i)$. Let $N_j(u_i)$ be the set of neighbours of u_i that have rated v_j. Neighbours can be selected in two ways: either we select users such that their similarity score is above some absolute threshold [43] which we call $\theta_{neighbourhood}$ in this thesis, or we select the top n most similar users [40]. With either technique, being able to select good quality neighbours is critical to achieving high prediction accuracy. Then we use weighted average to compute $\hat{R}_{ij}$ as follows,

$$\hat{R}_{ij} = \frac{\sum_{u_k \in N_j(u_i)} w(u_k, u_i) \cdot R_{kj}}{\sum_{u_k \in N_j(u_i)} abs(w(u_k, u_i))} \tag{2.2}$$

For item-item similarity on the other hand, adjusted cosine similarity score has been shown to perform better than other similarity measures such as cosine similarity and correlation-based scores [40]. Let U_{v_j} be the set of users who have rated v_j, and $U_{v_j,v_k} = U_{v_j} \cap U_{v_k}$. Then the similarity between two items v_j and v_k are given by,

$$w(v_j, v_k) = \frac{\sum_{u_i \in U_{v_j,v_k}} (R_{ij} - \bar{R}_i)(R_{ik} - \bar{R}_i)}{\sqrt{\sum_{u_j \in U_{v_j,v_k}} (R_{ij} - \bar{R}_i)^2} \cdot \sqrt{\sum_{u_j \in U_{v_j,v_k}} (R_{ik} - \bar{R}_i)^2}} \tag{2.3}$$

Note that $w(v_j, v_k) = w(v_k, v_j)$. Let $N_i(v_j)$ be the set of neighbours of v_j that have been rated by u_i. As with user-based collaborative filtering, neighbours can be selected using the following two methods: either select those with similarity scores above some absolute threshold $\theta_{neighbourhood}$, or by selecting the top n similar items. We can estimate $\hat{R}_{ij}$ by using a weighted average across this set.

$$\hat{R}_{ij} = \frac{\sum_{v_k \in N_i(v_j)} w(v_j, v_k) \cdot R_{ik}}{\sum_{v_k \in N_i(v_j)} abs(w(v_j, v_k))} \tag{2.4}$$

2.1.3 Matrix Factorization

To make predictions using neighbourhood-based collaborative filtering, the entire rating matrix R needs to be stored in memory, along with the similarity computations. In contrast, matrix factorization represents users and

items in terms of lower dimensional vectors of features inferred from the ratings, called latent features or latent factors [26]. Instead of storing the entire rating matrix in memory, it stores two lower dimensional factor matrices – the user latent factor matrix $U \in \mathbb{R}^{m \times d}$ and the item latent factor matrix $V \in \mathbb{R}^{n \times d}$, where d is the dimension of the latent feature-space. A certain user u_i's preference for an item v_j is represented by an inner product between their corresponding latent vectors, $V_j^T U_i$, which approximates R_{ij}. The goal in matrix factorization is to determine U, V that produce the lowest error between the observed ratings in R, and the estimated ratings $\hat{R}$. Typically gradient descent or alternating least squares is used to minimize regularized root mean square error (RMSE) over the set of known ratings in R [26].

2.1.4 Monotonicity and Submodularity/Supermodularity

Monotonicity and submodularity or supermodularity are important properties of constrained optimization objective functions. We study these properties for our problem in Sections 3.2.2, 3.2.3, 4.3.3, and 4.3.4. Here we review the definitions.

Definition 2.1.1. *For subsets $A \subset B \subset U$ of some ground set U, a function $f : 2^U \to \mathcal{R}$ is monotone decreasing if $f(A) \geq f(B)$ and monotone increasing if $f(A) \leq f(B)$.*

Definition 2.1.2. *For subsets $A \subset B \subset U$ of some ground set U, and $x \in U \setminus B$, a set function $f : 2^U \to \mathbb{R}_{\geq 0}$ is submodular if $f(B \cup \{x\}) - f(B) \leq f(A \cup \{x\}) - f(A)$. The function $f(.)$ is supermodular iff $-f(.)$ is submodular, or equivalently iff $f(B \cup \{x\}) - f(B) \geq f(A \cup \{x\}) - f(A)$.*

2.2 Related Work

We classify research related to the problem studied in this thesis under the following categories.

2.2.1 Cold Start Problem in Collaborative Filtering

The cold start problem in collaborative filtering recommender systems has been addressed using many different methods. A common approach combines CF with user content (metadata) and/or item content information to start off the recommendation process for cold users [25, 27, 28, 42, 47].

Other approaches leverage information from an underlying social network to recommend items to cold users [29, 31, 32]. Some researchers have proposed similarity measures tuned to cold-start users [3, 8], and use of feature-based regression [35]. In addition, online CF techniques, that incrementally update the latent vectors as new items or users arrive, have been proposed as a way to incorporate new data without retraining the entire model [1, 22, 41]. None of these works rigorously study the problem of selecting a limited number of items for a cold-start user as an optimization problem.

One exception is [4], which studies *the cold-start item problem* and formalizes it as an optimization problem of selecting users, to rate a given cold-start item. We borrow motivation from this paper and study the *cold-start user problem* by formalizing an optimization function in a probabilistic manner. Unlike them, our recommender model is based on probabilistic MF. Furthermore, they do not study the complexity or approximability of the user selection problem in their framework. They also do not run any scalability tests, and their experiments are quite limited. As part of our technical results, we show that our objective function is not supermodular. By duality between the technical problems of cold-start users and cold-start items, it follows that the objective used in their framework is not supermodular either, thus correcting a misclaim in their paper. A practical observation about the cold-start user problem is that it is easy and natural to motivate a cold-start user by asking her to rate several items in return for better quality recommendations using the learned profile. However, it is less natural and therefore harder to motivate users to help the system learn the profile of an item, so that it can be recommended to *other* users in the future.

2.2.2 Interactive Recommendation

Items may be recommended to a cold-start user in batch mode or interactive mode. In batch mode, the items are selected in one shot and then used for obtaining feedback from the cold-start user. E.g., this is the approach adopted in [4] (for user selection). The drawback of recommending a static set of items to every cold user, is that it assumes that the actual ratings given by the user do not affect the informativeness of the questions. That is, we cannot make use of the rating on one item, to choose the next few items. In interactive mode, feedback obtained on an item can be incorporated in selecting the next item. This can be further handled in two ways – online or offline. In the online setting, multi-armed bandit frameworks that interleave exploration with exploitation have been studied [11, 12, 30, 44, 49]. However, these approaches require re-training of the model after each item is

recommended. In contrast, offline approach considers all possible outcomes for feedback and prepares an "interview plan" in the form of a decision tree [16, 23, 50]. It is well known that constructing the optimal decision tree is NP-complete. Furthermore, the search space of possible decision trees for forming interview plans is exponential. While heuristic solutions are proposed in [16, 23, 50], large scale scalability experiments are not reported.

2.2.3 Complexity of Recommendation

The complexity of recommendation systems has been studied before under various assumptions. Non-negative matrix factorization (used in some CF approaches) has been shown to be NP-hard [45]. In [18], the authors find products that, between them, have been bought by as many people as possible, and recommend them as well as products similar to them to other users. They show that finding products that cover as many users as possible is NP-hard. Both papers above are not concerned with item selection for learning the profile of a cold-start user.

In [17], the authors study the problem of finding a limited number of users such that upon recommending a given item to them, if they rate it favorably, it will then be recommended to the maximum number of other users by the system. They show this problem is NP-hard and inapproximable. Finally, [5] shows that the problem of forming groups of users such that the group recommendations they receive maximize user satisfaction is NP-hard. These problems are orthogonal to the problem studied in this paper and do not directly address the item selection problem for cold-start users.

Chapter 3

Neighbourhood-based Methods

3.1 Preliminaries

3.1.1 Dominating Set

In graph theory, a dominating set for a graph $G = (V, E)$ is $D \subset V$ such that $\forall v \notin D$, there is an edge $e = (u, v)$ and $u \in D$. The domination number $\gamma(G)$ is the number of vertices in the smallest dominating set for G.

Given a graph G, and a number k, the Dominating Set problem asks to find D such that it dominates the maximum number of vertices and $|D| = k$.

3.1.2 Problem Statement

We focus on the item-item similarity model for the underlying recommender system. Let u_ℓ be a cold-start user whose preferences need to be learned by recommending a small number of items to u_ℓ. Each item recommended to u_ℓ can be viewed as a probe or "interview question" to gauge u_ℓ's interest. Since there is a natural limit on how many probe items we can push to a user before saturation or apathy sets in, we assume a budget b on the number of probe items. Our objective is to select b items that maximize our learning of the cold user's preferences, determined by achieving a high prediction accuracy on the maximum number of items.

We assume that ratings are described perfectly by the cold user's preference for similar items, modulo some noise. Then for some item $v_j, R_{\ell j} = \hat{R}_{\ell j} + \varepsilon_{\ell j}$, where $\varepsilon_{\ell j}$ is the noise term associated with the user-item pair (u_ℓ, v_j), and the predicted rating would be given by,

$$\hat{R}_{\ell j} = \frac{\sum_{v_k \in N_\ell(v_j)} w(v_k, v_j) \cdot R_{\ell k}}{\sum_{v_k \in N_\ell(v_j)} abs(w(v_k, v_j))} \tag{3.1}$$

Note that initially the number of v_j's neighbours rated by $u_\ell, |N_\ell(v_j)| = 0$, but after the interview, $|N_\ell(v_j)| \leq b$, as we obtain u_ℓ's feedback on b

items, and at most all b items can be neighbours of v_j. In this thesis we use absolute thresholding to select neighbours as a sufficiently large threshold ensures higher quality neighbours, and thus better prediction accuracy [20]. We denote the threshold used in absolute thresholding as $\theta_{neighbourhood}$. Two items with a similarity score $w(.,.) \geq \theta_{neighbourhood}$ are considered to be in each other's neighbourhoods. Note that due to the sparsity in ratings in recommender systems, using a lower value of $\theta_{neighbourhood}$ increases the number of items for which rating predictions can be made, as it increases the number of items with at least one neighbour, but it introduces more noise and can result in poorer prediction accuracy. Thus the value of $\theta_{neighbourhood}$ indirectly controls prediction accuracy. Given a judiciously chosen threshold $\theta_{neighbourhood}$, we would like to maximize the number of items for which we are able to predict the user's preference rating. This has clear motivation, as it provides a larger universe from which to recommend items to that user. This is known as prediction coverage [20]. For a set of items $\mathcal{A} \subseteq \mathcal{I}$, the prediction coverage of $\mathcal{A}, PC(\mathcal{A})$ is defined as follows,

$$PC(\mathcal{A}) = \left| \bigcup_{v_k \in \mathcal{A}} N_\ell(v_k) \right|.$$

Given a fixed threshold $\theta_{neighbourhood}$, our objective is to select b items that achieve maximum prediction coverage. We now formally state the problem we study.

Problem 1 (Optimal Interview Design - Neighbourhood Based (OID-NB))**.** *Given an item-item similarity matrix, $\theta_{neighbourhood}$ absolute threshold for neighbourhood selection, cold start user u_ℓ, and a budget b, find a set of b items B, to recommend to u_ℓ that maximizes prediction coverage $PC(B)$.*

3.2 Technical Results

3.2.1 Hardness

Theorem 3.2.1. *The optimal interview design problem for an item-based collaborative filtering recommender system (OID-NB, Problem 1) is NP-hard.*

Proof. We prove the hardness of OID-NB through a reduction from the dominating set problem (Section 3.1.1).

Reduction: From an instance $\mathcal{X}$ of dominating set, create an instance $\mathcal{Y}$ of OID-NB. For every node $w \in V$, create a corresponding item v_w in

$\mathcal{I}$, and for every $e_i = (u, w) \in E$, create a corresponding user u_i. Create a dummy item v_d and the cold-start user u_ℓ. Therefore, $|I| = |V| + 1$ and $|U| = |E| + 1$.

Now, construct the rating matrix R. We consider only binary rating values here, but the proof can be easily extended to any interval rating scale. For every user u_i corresponding to $e_i = (u, w) \in E$ in $\mathcal{X}$, assign a rating $R_{iu} = R_{iw} \in 0, 1$. Also assign rating $R_{id} \neq R_{iu}$. Therefore, every user rates exactly 3 items, out of which two receive the same rating, and the dummy item receives the opposite. Alternate this for each user. For example, if the first user rates the 2 items 1, and rates the dummy item 0, then the next user will rate the 2 items 0, and dummy item 1. For all other user-item pairs, the rating is considered null, or unknown. Let $\theta_{neighbourhood} = 1$. By construction, $w(v_d, v_k) < \theta_{neighbourhood} = 1$, for any $v_k \in I$. To see that this is true, w.l.o.g. consider a user u_i who has rated both v_d and some item v_k. $\bar{R}_i$ is either $\frac{1}{3}$ or $\frac{2}{3}$ (by construction). Hence $(R_{ik} - \bar{R}_i)(R_{di} - \bar{R}_i) = (r - \frac{1}{3})(1 - r - \frac{2}{3}) < 0$, where $r = \{0, 1\}$. Adding this up over all users who have rated both, we obtain $w(v_d, v_k) < 0 < 1$.

For $\theta_{neighbourhood} = 1$, we claim that we are able to produce predictions on an item if and only if the node in $\mathcal{X}$ corresponding to its neighbour, is in the dominating set D. Therefore, ratings received on the b items recommended to u_ℓ, allow us to make predictions on their neighbours. Then the b nodes that dominate the most number of nodes in $\mathcal{X}$, produces the highest prediction coverage in $\mathcal{Y}$.

To see that nodes are adjacent in $\mathcal{X}$, if and only if the corresponding items in $\mathcal{Y}$ are neighbours, we compute the adjusted cosine similarity of two items v_j, v_k in $\mathcal{X}$ such that there is an edge $e_i = (j, k)$ in $\mathcal{Y}$. By construction, $U_{v_j, v_k} = U_{v_j} \cap U_{v_k} = u_i$.

$$\begin{aligned} w(v_j, v_k) &= \frac{(R_{ij} - \bar{R}_i)(R_{ik} - \bar{R}_i)}{\sqrt{(R_{ij} - \bar{R}_i)^2}.\sqrt{(R_{ik} - \bar{R}_i)^2}} \\ &= \frac{(r - \bar{R}_i)(r - \bar{R}_i)}{\sqrt{(r - \bar{R}_i)^2}.\sqrt{(r - \bar{R}_i)^2}} \\ &= \frac{(r - \bar{R}_i)(r - \bar{R}_i)}{(r - \bar{R}_i).(r - \bar{R}_i)} = 1 \end{aligned} \tag{3.2}$$

If the edge did not exist in $\mathcal{X}$ but the nodes did, there would be no u_i in $\mathcal{Y}$, and $w(v_j, v_k)$ would be 0. On the other hand, $w(v_d, v_k) < 1$ as the node corresponding to v_d does not exist in $\mathcal{X}$. It also follows that, to

produce recommendations on an item v_j in $\mathcal{Y}$, the node corresponding to it must be dominated by at least one other node in $\mathcal{X}$. Otherwise, $N_\ell(v_j) = \phi$. Therefore, the b items that dominate the largest number of nodes in $\mathcal{X}$ produce the largest prediction coverage in $\mathcal{J}$. This is what we had to show. □

3.2.2 Monotonicity

We define monotonicity as given in Definition 2.1.1.

Consider the cold-start user u_ℓ. Let $f(A) = PC(A) = |\bigcup_{v_k \in A} N_\ell(v_k)|$ for some subset A of item set $\mathcal{I}$. First we show that $f(A)$ is normalized, that is, $f(\phi) = 0$. This is trivially true, as $|N_\ell(\phi)| = |\phi| = 0$. To show monotonicity, we consider $B = A \cup v_j$ where $v_j \in \mathcal{I} \setminus A$.

$$\begin{aligned} f(B) - f(A) &= \left| \bigcup_{v_k \in B} N_\ell(v_k) \right| - \left| \bigcup_{v_k \in A} N_\ell(v_k) \right| \\ &= \left| \bigcup_{v_k \in A \cup v_j} N_\ell(v_k) \right| - \left| \bigcup_{v_k \in A} N_\ell(v_k) \right| \\ &= \left| \bigcup_{v_k \in A} N_\ell(v_k) \cup N_\ell(v_j) \right| - \left| \bigcup_{v_k \in A} N_\ell(v_k) \right| \geq 0 \end{aligned}$$

The last equation is true as $|N_\ell(v_j)| \geq 0$. Hence $f(.)$ is a monotone increasing function.

3.2.3 Submodularity

We define submodularity as given in Definition 2.1.2.

Consider $A, B, f(.)$ as defined in Section 3.2.2. Under this definition, the objective function simplifies into the budgeted maximum covering problem, which has been shown to be submodular in previous work [21, 24].

3.2.4 Approximation

Theorem 3.2.2. *Consider an item-item similarity collaborative filtering system, with item set $\mathcal{I}$, cold user u_ℓ, and $\theta_{neighbourhood} = 1$. Let the set of b items producing optimal prediction coverage be $OPT_{OID-NB} = argmax_{A \subset \mathcal{I}, |A|=b} |\bigcup_{v_k \in A} N_\ell(v_k)|$, and let $\mathcal{A}$ be the solution produced by selecting the items one by one, each time selecting the one that covers the most*

number of as yet uncovered elements, and $|\mathcal{A}| = b$. *Then*

$$\left| \bigcup_{v_k \in \mathcal{A}} N_\ell(v_k) \right| \geq (1 - 1/e) \left| \bigcup_{v_k \in OPT_{OID-NB}} N_\ell(v_k) \right|.$$

Proof. In Sections 3.2.2 and 3.2.3, we show that the objective function is normalized, monotone increasing and submodular. Now we apply the general result on submodular function maximization with a cardinality constraint by Nemhauser et. al. which states that an algorithm that greedily selects sets (here items) can do no better than $(1 - 1/e)$ times the optimal solution [34]. In fact, it states that a better approximation factor does not exist. □

3.3 Algorithms

In this section, we present our algorithm for selecting items with which to interview a cold user so as to learn her preferences as well as possible. In view of the hardness and hardness of approximation results (Theorems 3.2.1 and 3.2.2), we present the following greedy selection algorithm: select items one at a time, each time selecting the item that increases the prediction coverage most. Let the solution produced by this algorithm be $\mathcal{A}$, and $\mathcal{A}_i$ be the set of items selected at iteration i. We can formally express this as, $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \{\arg\max_{v_j} \delta(v_j|\mathcal{A}_i)\}$, where $\delta(v_j|\mathcal{A}_i)$ denotes the increase in prediction coverage when v_j is added to $\mathcal{A}_i$.

The pseudocode is given in Algorithm 1. In Line 6, we compute the increase in prediction coverage, which utilizes the item-item similarity matrix and $\theta_{neighbourhood}$. In case no item increases the prediction coverage, the algorithm adds the last item by default (Line 4).

3.4 Experimental Evaluation

In this section, we describe the experimental evaluation for our algorithm, and compare them with prior art.

The development and experimentation environment uses a Linux Server with 2.93 GHz Intel Xeon X5570 machine with 98 GB of memory with OpenSUSE Leap OS.

Algorithm 1 Greedy Selection (GS)

Input: item set $\mathcal{I}$; budget b; rating matrix R; item-item similarity matrix; $\theta_{neighbourhood}$

Output: items subset $B, |B| = b$.

```
1: B ← φ
2: do
3:     max ← 0
4:     maxitem ← I[−1]
5:     for v_i ← 1 to |I| do
6:         δ(v_i|B) ← PC(B ∪ v_i) − PC(B)
7:         if δ(v_i|B) ≥ max then
8:             max ← δ(v_i|B)
9:             maxitem ← v_i
10:            I ← I \ v_i
11:        end if
12:        B ← B ∪ maxitem
13:    end for
14: while |B| ≤ b
```

3.4.1 Dataset and Model

For our experiments, we use two datasets – Movielens (ML) 100K and Movielens 1M [1], which are used for movie recommendations. We describe their characteristics in Table 3.1.

Table 3.1: Dataset Sizes

Dataset	# Ratings	# Users	# Items	Sparsity
ML 100K	100,000	943	1682	6.3%
ML 1M	1,000,209	6,040	3,900	4.25%

3.4.2 Model Parameters & Experimental Setup

For each dataset, we compute item-item similarity using adjusted cosine similarity for all items using only the ratings given by 70% of the users. We refer to them as the warm users, $\mathcal{U}$. We use an absolute threshold,

[1] Available at `http://grouplens.org/datasets/movielens/`. Source: [19]

$\theta_{neighbourhood}$, to compute neighbours, and study the effect of varying its value.

Experimental Setup: We simulate the cold user interview process as follows:

1. Set up the system
   (a) Randomly select 70% of the users in a given dataset to train the model ($\mathcal{U}$)
   (b) $R :=$ Matrix of ratings given by $\mathcal{U}$ only
   (c) For every pair of items (v_j, v_k), compute adjusted cosine similarity $w(v_j, v_k) = \frac{\sum_{u_i \in U_{v_j, v_k}} (R_{ij} - \bar{R}_i)(R_{ik} - \bar{R}_i)}{\sqrt{\sum_{u_j \in U_{v_j, v_k}} (R_{ij} - \bar{R}_i)^2} \cdot \sqrt{\sum_{u_j \in U_{v_j, v_k}} (R_{ik} - \bar{R}_i)^2}}$.
   (d) If $w(v_j, v_k) > \theta_{neighbourhood}$, add each item to the other's neighbourhood list
2. For each cold user $u_\ell \notin \mathcal{U}$,
   (a) Randomly split items they have rated, into candidate pool CP and test set $Test$ equally
   (b) Run item selection algorithm on CP with corresponding neighbours list and budget $= b$
   (c) $B :=$ items returned by algorithm to interview u_ℓ
   (d) Reveal $R_\ell := u_\ell$'s ratings on B
   (e) Evaluation: RMSE on $Test := \sqrt{\frac{1}{|Test|} \sum_{v_j \in Test} (R_{\ell j} - \hat{R_{\ell j}})^2}$, prediction coverage $PC(B) := |\bigcup_{v_k \in B} N_\ell(v_k)|$
3. Average prediction error and prediction coverage over all cold users

In Step 2e, we compute RMSE and prediction coverage on $Test$. Note that some items in $Test$ may not have any neighbours among B. In that case, we predict a default rating, such as the mean rating of the dataset.

3.4.3 Algorithms Compared

We study three versions of the Greedy Selection algorithm (`GS`) as described in Section 3.3, corresponding to three different $\theta_{neighbourhood} = \{0.4, 0.5, 0.6\}$. We call them (`GS4`), (`GS5`) and (`GS6`). We compare them against baselines Random Selection (`RS`), where the items are randomly sampled from the candidate pool, Popular (`Pop`), where the b most rated items are selected,

with the rationale being that obscure items may not be informative about users' likes and dislikes, while also being difficult for them to rate [16, 36], Entropy0 (`Ent0`) [36] (Equation 3.3) and EntPop (`EP`). The last two are modifications of entropy, which is a measure of informativeness. Items which are more controversial, and have a wider spread in their rating distribution, tend to be more informative about a user's preferences, while knowing that the user likes an item that everyone likes, does not tell us much. However, items with high entropy are often less popular. To balance informativeness with popularity, we use the measures Entropy0 (`Ent0`) as described in Equation 3.3, and EntPop (`EP`), where the b items with highest values of entropy $\times \log(\text{popularity})$ are selected. Both have been shown to perform better than pure entropy [36].

$$Entropy0(v_j) = -\frac{1}{\sum_i w_i} \sum_{i=0}^{5} p_i w_i log(p_i) \tag{3.3}$$

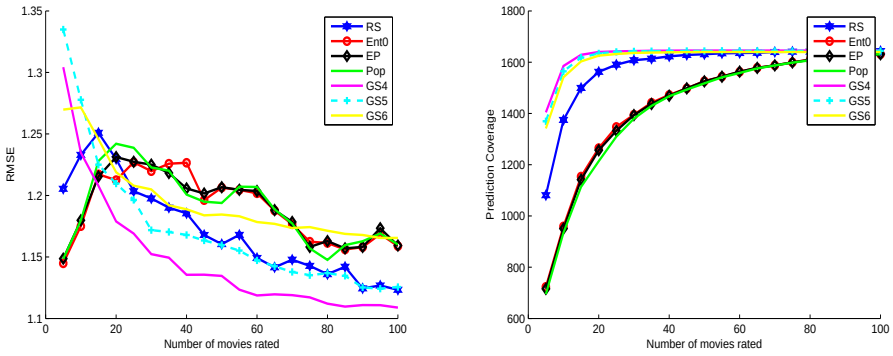
Here, p_i is the fraction of users who have rated v_j equal to i, starting from $i = 0$, referring to the class of users who have not rated v_j, to $i = 5$, the maximum possible rating in our datasets. w_i refers to the weights given to the 6 classes. We use $w = [0.5, 1, 1, 1, 1, 1]$, which was shown to work the best in [36].

This gives us a total of 7 algorithms to compare.

3.4.4 Results

The first thing we observe is that `GS4` performs better than all other algorithms after $b = 15$ (Fig. 3.1(a)). Up to that point, `Ent0`, `EP` and `Pop` perform the best. However, they consistently perform poorly on prediction coverage (Fig. 3.1(b)). We hypothesize that their "good performance" for lower values of b is not because they are able to accurately predict what the user would like or dislike, but due to the default predicted rating. The low prediction coverage implies that many items in $Test$ would not have any neighbours in the B selected by these algorithms, and so would be assigned the mean rating by default. Hence the "good performance" would only be due to the other algorithms making predictions that are worse than predicting the mean, as they have likely not yet learned the user's preferences. This hypothesis can also explain why `GS` performs worse as we increase $\theta_{neighbourhood}$ from 0.4 to 0.6. As the threshold is raised, fewer items are covered, and so more items are assigned a mean rating.

To test this hypothesis, we observe what happens when we change the default rating from the mean rating to an arbitrarily bad rating, such as 1.



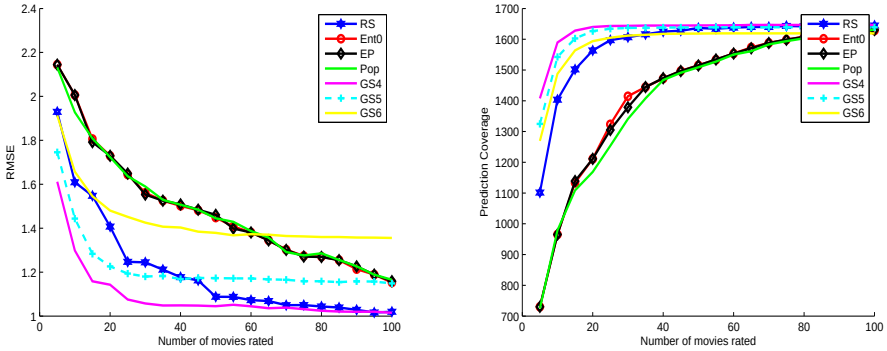
(a) Prediction Error (b) Prediction Coverage

Figure 3.1: ML 100K dataset with default mean rating

After doing that, our proposed algorithm `GS4` performs the best in terms of predicton accuracy as well as prediction coverage (Fig. 3.2). Interestingly, prediction error does not decrease monotonically as we increase the similarity threshold, as after $\theta_{neighbourhood} \geq 0.5$, the reduction in coverage reduces the number of neighbours, and increases the number of items for which we are unable to make any prediction. Thus we observe that raising $\theta_{neighbourhood}$ in fact leads to worse prediction error performance. We observe a similar trend with the ML 1M dataset as well (Fig 3.3) when we set the default rating to 1. The prediction accuracy for `GS5` and `GS6` worsens as the prediction coverage worsens.

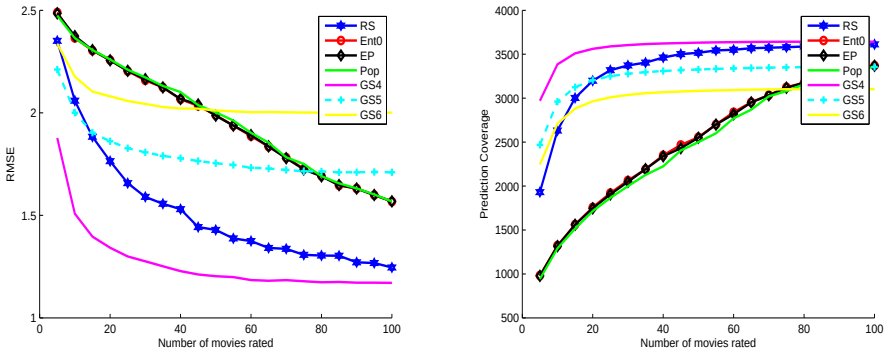
3.4.5 Discussion

Compared to real world recommender systems, the datasets we use in the experiments are quite small. Even then, and even with our efficient greedy selection algorithm and scalable baselines, the experiments took an extremely long time. The major roadblock was computing item-item similarity scores for all item pairs, which is a necessary pre-processing step. For large datasets, it becomes impractical to compute this information, and also to update it as new users and items enter the system. Some scalable methods have been explored using clustering [46], deriving similarity scores for multiple users/items at a time [6], and through parallel computing [15]. However, these approaches do not address the cold-start user problem, and further research needs to be done to adapt them for this purpose.



(a) Prediction Error (b) Prediction Coverage

Figure 3.2: ML 100K dataset with default rating = 1



(a) Prediction Error (b) Prediction Coverage

Figure 3.3: ML 1M dataset with default rating = 1

Chapter 4

Matrix Factorization Methods

4.1 Preliminaries

4.1.1 Probabilistic Matrix Factorization

MF treats the (latent) features of users and items as deterministic quantities. Probabilistic MF (PMF), on the other hand, assumes that these features are drawn from distributions. More precisely, PMF [39] expresses the rating matrix R as a product of two random low dimension latent factor matrices with the following zero-mean Gaussian priors:

$$\Pr[U|\Sigma_U] = \prod_{i=1}^{m} \mathcal{N}(U_i|\mathbf{0}, \sigma_{u_i}^2 I), \Pr[V|\Sigma_V] = \prod_{j=1}^{n} \mathcal{N}(V_j|\mathbf{0}, \sigma_{v_j}^2 I), \tag{4.1}$$

where $\mathcal{N}(x|\mu, \sigma^2)$ is the probability density function of a Gaussian distribution with mean μ and variance σ^2. It then estimates the observed ratings as $R = \hat{R} + \varepsilon = U^T V + \varepsilon$, where ε is a matrix of noise terms in the model. More precisely, $\varepsilon_{ij} = \sigma_{ij}^2$ represents zero-mean noise in the model.

The conditional distribution over the observed ratings is given by

$$\Pr[R|U, V, \Sigma] = \prod_{i=1}^{m} \prod_{j=1}^{n} [\mathcal{N}(R_{ij}|U_i^T V_j, \sigma_{ij}^2)]^{\delta_{ij}} \tag{4.2}$$

where Σ is a $d \times d$ covariance matrix, and δ_{ij} is an indicator function with value 1 if user u_i rated item v_j, and 0 otherwise.

Taking the log over the posterior gives us (see [39]):

$$\begin{aligned}\ln \Pr[U, V|R, \Sigma, \Sigma_V, \Sigma_U] = &-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\frac{\delta_{ij}}{\sigma_{ij}^2}(R_{ij} - U_i^T V_j)^2 \\ &-\frac{1}{2}\sum_{i=1}^{m}\frac{U_i^T U_i}{\sigma_{u_i}^2} - \frac{1}{2}\sum_{j=1}^{n}\frac{V_j^T V_j}{\sigma_{v_j}^2} \\ -\frac{1}{2}\left(\sum_{i=1}^{m}\sum_{j=1}^{n}\delta_{ij}\ln\sigma_{ij}^2 + d\sum_{i=1}^{m}\ln\sigma_{u_i}^2 + d\sum_{j=1}^{n}\ln\sigma_{v_j}^2\right) + C \qquad (4.3)\end{aligned}$$

Algorithms like gradient descent or alternating least squares can be used to optimize the resulting non-convex optimization problem.

4.1.2 Problem Statement

Consider a PMF model (U, V) trained on an observed ratings matrix R, by minimizing a loss function such as squared error between R and the predicted ratings $\hat{R} = U^T V$ (with some regularization). Let u_ℓ be a cold-start user whose profile needs to be learned by recommending a small number of items to u_ℓ. As in Section 3.1.2, to avoid saturation or apathy, we limit the number of probe items to a small number b. We denote the true profile of u_ℓ by U_ℓ and the learned profile (using her feedback on the b items) as $\hat{U}_\ell$. Our objective is to select b items that minimizes the error in the learned profile $\hat{U}_\ell$ compared to the true profile U_ℓ. We next formally state the problem studied in this paper.

Problem 2 (Optimal Interview Design - Matrix Factorization)**.** *Given user latent vectors U, item latent vectors V, cold start user u_ℓ, and a budget b, find the b best items to recommend to u_ℓ such that $E[||\hat{U}_\ell - U_\ell||_F^2]$ is minimized.*

4.2 Solution Framework

A first significant challenge in solving Problem 2 is that in order to measure how good our current estimate the user profile is, we need to know the actual profile of the cold user, on which we have no information! In this section, we devise an approach for measuring the error in the estimated user profile, which intelligently circumvents this problem (see Lemma 4.2.1).

Note that using the PMF framework in Section 4.1.1, we obtain low dimensional latent factor matrices U, V. In the absence of any further information, we assume that the latent vector of the cold user "truly" describes her profile.[2] Notice that the budget b on the number of allowed interview/probe items is typically a small number. Following prior work [4, 37, 41], we assume that the responses of the cold user u_ℓ to this small number of items does not significantly change the latent factor matrix V associated with items. Under this assumption, we can perform local updates to U_ℓ as the ratings from u_ℓ on the b probe items are available. A second challenge is that we consider a *batch setting* for our problem. This means that we should select the b items without obtaining explicit feedback from the cold user. We overcome this challenge by estimating the feedback rating the user u_ℓ would provide according to the current model. Specifically, we estimate cold user u_ℓ's rating on an item v_j as $R_{\ell j} = \hat{R}_{\ell j} + \varepsilon_{\ell j} = V_j^T U_\ell + \varepsilon_{\ell j}$, where $\varepsilon_{\ell j}$ is a noise term associated with the user-item pair (u_ℓ, v_j).

Let R_ℓ denote the vector containing the ratings of the cold user u_ℓ on the b items presented to her, and let V_B be the $d \times b$ latent factor matrix corresponding to these b items. We assume that the noise in estimating the ratings $\hat{R}$ depends on the item under consideration, i.e., $\mathbb{E}[\varepsilon_{ij}^2] = \sigma_{v_j}^2$, for all users u_i. This gives us the following posterior distribution,

$$\Pr[U_\ell | R_\ell, V_B, C_B] \propto \mathcal{N}(R_\ell | V_B^T U_\ell, C_B) \mathcal{N}(U_\ell | \mathbf{0}, \sigma_{u_\ell}^2 I)$$

where C_B is a $b \times b$ diagonal matrix with $\sigma_1^2, \sigma_2^2, ..., \sigma_b^2$ at positions corresponding to the items in B. Using Bayes rule for Gaussians, we obtain $\Pr[U_\ell | R_\ell, V_B, C_B] \propto \mathcal{N}(U_\ell | \hat{U}_\ell, \Sigma_B)$, where $\hat{U}_\ell = \Sigma_B V_B C_B^{-1} R_\ell$ and $\Sigma_B = (\sigma_{u_\ell}^{-2} I + V_B C_B^{-1} V_B^T)^{-1}$. Setting $\gamma = \sigma_{u_\ell}^{-2}$, the estimate $\hat{U}_\ell$ of the cold user's true latent factor vector U_ℓ can be obtained using a ridge estimate. More precisely,

$$\hat{U}_\ell = (\gamma I + V_B C_B^{-1} V_B^T)^{-1} V_B C_B^{-1} R_\ell \tag{4.4}$$

Here, γ is mainly used to ensure that the expression is invertible.

Under this assumption, we next show that solving Problem 2 reduces to minimizing $tr((V_B C_B^{-1} V_B^T)^{-1})$. More precisely, we have:

Lemma 4.2.1. *Given user latent vectors U, item latent vectors V, cold start user u_ℓ, and budget b, a set of b items B minimizes $E[||\hat{U}_\ell - U_\ell||_F^2]$ iff it minimizes $tr((V_B C_B^{-1} V_B^T)^{-1})$, where V_B is the submatrix of V corresponding to the b selected items.*

[2] There may be a high variance associated with U_ℓ.

Proof. Our goal is to select b items such that using her feedback on those items, we can find the estimate of the latent vector $\hat{U}_\ell$ of the cold user u_ℓ, that is as close as possible to the true latent vector vector U_ℓ.

Equation 4.4 gives us an estimate for $\hat{U}_\ell$. For simplicity, we will assume that $\gamma = 0$, and that $V_B C_B^{-1} V_B^T$ is invertible.

R_ℓ can be expressed as $V_B^T U_\ell + \varepsilon_B$, where ε_B is a vector of the b zero-mean noise terms corresponding to the b items. Replacing this in Equation 4.4, we get

$$\begin{aligned} \hat{U}_\ell &= U_\ell + (V_B C_B^{-1} V_B^T)^{-1} V_B C_B^{-1} \varepsilon_B \\ \Rightarrow \hat{U}_\ell - U_\ell &= (V_B C_B^{-1} V_B^T)^{-1} V_B C_B^{-1} \varepsilon_B \end{aligned} \tag{4.5}$$

From Equation 4.5, it is clear that the choice of the b interview items determines how well we are able to estimate $\hat{U}_\ell$. The expected error in the estimated user profile is

$$E[||\hat{U}_\ell - U_\ell||_F^2] = E[tr((\hat{U}_\ell - U_\ell)(\hat{U}_\ell - U_\ell)^T)] \tag{4.6}$$

Replacing Equation 4.5 in Equation 4.6 and simplifying, we get

$$\begin{aligned} E[tr((\hat{U}_\ell - U_\ell)(\hat{U}_\ell - U_\ell)^T)] = \\ E[tr((V_B C_B^{-1} V_B^T)^{-1} V_B C_B^{-1} \varepsilon_B \varepsilon_B^T (C_B^{-1})^T V_B^T (V_B C_B^{-1} V_B^T)^{-1})] \\ = tr((V_B C_B^{-1} V_B^T)^{-1}) \end{aligned} \tag{4.7}$$

The second equality above follows from from replacing $E[\varepsilon_B \varepsilon_B^T] = C_B$ and simplifying the algebra. The lemma follows. □

In view of the lemma above, we can instantiate Problem 2 and restate it as follows.

Problem 3 (Optimal Interview Design - Matrix Factorization (OID-MF)). *Given user latent vectors U, item latent vectors V, cold start user u_ℓ, and a budget b, find the b best items to recommend to u_ℓ such that $E[||\hat{U}_\ell - U_\ell||_F^2] = tr((V_B C_B^{-1} V_B^T)^{-1})$ is minimized.*

Since the lemma shows that Problem 2 is essentially equivalent to Problem 3, we focus on the latter problem in the rest of the thesis.

4.3 Technical Results

In this section, we study the hardness and approximation of the OID-MF problem we proposed.

4.3.1 Hardness

Our main result in this section is:

Theorem 4.3.1. *The optimal interview design problem under the matrix factorization setting (OID-MF, Problem 3) is NP-hard.*

The proof of this theorem is non-trivial. We establish this result by proving a number of results along the way. For our proof, we consider the special case where $\sigma_1^2 = \sigma_2^2 = ... = \sigma_b^2 = \sigma^2$ and $\lambda = \frac{\sigma^2}{\sigma_{u_\ell}^2}$. Then $C_B = \sigma^2 I$, and plugging it in to Equation 4.7 yields

$E[||\hat{U}_\ell - U_\ell||_F^2] = \sigma^2 \cdot tr((V_B V_B^T)^{-1})$. We prove hardness for this restricted case. The hardness of the general case follows.

The proof of hardness is performed by reduction from the well-known NP-complete problem Exact Cover by 3-Sets (X3C) [14].

Reduction: Given a collection $\mathcal{S}$ of 3-element subsets of a set X, where $|X| = 3q$, X3C asks to find a subset $\mathcal{S}^*$ of $\mathcal{S}$ such that each element of X is in exactly one set of $\mathcal{S}^*$. Let $(X, \mathcal{S})$ be an instance of X3C, with $X = \{x_1, ..., x_{3q}\}$ and $\mathcal{S} = \{S_1, ..., S_n\}$. Create an instance of OID-MF as follows. Let the set of items be $\mathcal{I} = \{a_1, ..., a_n, d_1, ..., d_k\}$, where $k = 3q$, item a_j corresponds to set S_j, $j \in [n]$, and d_j are dummy items, $j \in [k]$. Convert each set S_j in $\mathcal{S}$ into a binary vector $\mathbf{u_j}$ of length k, such that $\mathbf{a_j}[i] = 1$ whenever $x_i \in S_j$ and $\mathbf{a_j}[i] = 0$ otherwise. Since the size of each subset is exactly 3, we will have exactly three 1's in each vector. These vectors correspond to the item latent vectors of the n items $a_1, a_2, ..., a_n$. We call them *set vectors* to distinguish them from the vectors corresponding to the dummy items, defined next: for a dummy item d_j, the corresponding vector $\mathbf{d_j}$ is such that $\mathbf{d_j}[j] = \eta$ and $\mathbf{d_j}[i] = 0$, $i \neq j$. Let $\mathcal{W}$ be the set of all vectors constructed. We will set the value of η later. Thus, $\mathcal{W}$ is the transformed instance obtained from $(X, \mathcal{S})$. Assuming an arbitrary but fixed ordering on the items in $\mathcal{I}$, we can treat $\mathcal{W}$ as a $k \times (n+k)$ matrix, without ambiguity. Let $\mathcal{A} = \{\mathbf{a_1}, ..., \mathbf{a_n}\}$ and $\mathcal{D} = \{\mathbf{d_1}, ..., \mathbf{d_k}\}$ resp., denote the sets of set vectors and dummy vectors constructed above. We set the budget to $b := q + k$. For a set of items $B \subset \mathcal{I}$, with $|B| = b$, we let $\mathcal{B}$ denote the $k \times (q + k)$ submatrix of $\mathcal{W}$ associated with the items in B. Formally, our problem is to find b items $B \subset \mathcal{I}$ that minimize $tr((\mathcal{B}\mathcal{B}^T)^{-1})$.

For a matrix $\mathcal{M}$, define $f(\mathcal{M}) = tr((\mathcal{M}\mathcal{M}^T)^{-1})$. Define

$$\theta := \frac{q}{3 + \eta^2} + \frac{k - q}{\eta^2}. \tag{4.8}$$

We will show the following claim.

Claim 1. *Let $B \subset \mathcal{I}$, such that $|B| = k + q$. Then $f(\mathcal{B}) = \theta$ if $(\mathcal{B} \setminus \mathcal{D})$ encodes an exact 3-cover of X and $f(\mathcal{B}) > \theta$, otherwise.*

We first establish a number of results which will help us prove this claim.

Notice that Theorem 4.3.1 follows from Claim 1: if there is a polynomial time algorithm for solving OID-MF, then we can run it on the reduced instance of OID-MF above and find the b items B that minimize $f(\mathcal{B})$. Then by checking if $f(\mathcal{B}) = \theta$, we can verify if the given instance of X3C is a YES or a NO instance.

In what follows, for simplicity, we will abuse notation and use $\mathcal{A}, \mathcal{B}, \mathcal{W}$ both to denote sets of vectors and the matrices formed by them, relative to the fixed ordering of items in $\mathcal{I}$ assumed above. We will freely switch between set and matrix notations.

Recall the transformed instance $\mathcal{W}$ of OID-MF obtained from the given X3C instance. The next claim characterizes the trace of $\mathcal{B}\mathcal{B}^T$ for matrices $\mathcal{B} \subset \mathcal{W}$ that include all k dummy vectors of $\mathcal{W}$.

Claim 2. *Consider any $\mathcal{B} \subset \mathcal{W}$ such that $|\mathcal{B}| = k + q$ and $\mathcal{B}$ includes all the k dummy vectors. Then $tr(\mathcal{B}\mathcal{B}^T) = k + k \cdot \eta^2$.*

Proof. Let $\mathcal{B}' = \mathcal{B} - \mathcal{D}$. We have $tr(\mathcal{B}\mathcal{B}^T) = tr(\mathcal{B}'\mathcal{B}'^T + \mathcal{D}\mathcal{D}^T) = tr(\mathcal{B}'\mathcal{B}'^T) + tr(\eta^2 I) = \sum_{i=1}^{k}\sum_{j=1}^{q} b_{ij}^2 + k\eta^2$. As $\mathcal{B}'$ is a binary matrix, $\sum_{i=1}^{k}\sum_{j=1}^{q} b_{ij}^2 = \sum_{i=1}^{k} ||b_{*i}||_0$, where b_{*i} is the ith row, and $||\cdot||_0$ is the l_0-norm. This is nothing but the total number of 1's in $\mathcal{B}'$, which is $3q = k$. Thus, $tr(\mathcal{B}\mathcal{B}^T) = k + k\eta^2$. □

The next claim shows that among such subsets $\mathcal{B} \subset \mathcal{W}$, the ones that include all dummy vectors have the least $f(.)$-value, i.e., have the minimum value of $tr((\mathcal{B}\mathcal{B}^T)^{-1})$. Recall that $\mathcal{D} = \{\mathbf{d_1}, ..., \mathbf{d_k}\}$ is the set of dummy vectors constructed from the given instance of X3C.

Claim 3. *For any subset $\mathcal{A} \subset \mathcal{W}$, with $|\mathcal{A}| = k + q$, such that $\mathcal{D} \not\subset \mathcal{A}$, there exists $\mathcal{A}'$, with $|\mathcal{A}'| = k + q$ and $\mathcal{D} \subset \mathcal{A}'$, such that $f(\mathcal{A}') < f(\mathcal{A})$.*

Proof. By Claim 2, $tr(\mathcal{A}'\mathcal{A}'^T) = k + k\eta^2$. By assumption, $\mathcal{A}$ has at least 1 fewer dummy vectors than $\mathcal{A}'$ and correspondingly more set vectors than $\mathcal{A}'$. Since each set vector has exactly 3 ones, we have $tr(\mathcal{A}\mathcal{A}^T) \leq k + k\eta^2 + 3 - \eta^2$ for $\eta^2 > 3$. Let us consider the way the trace is distributed among the eigenvalues. The distribution giving the least $f(.)$ is the uniform distribution. For $\mathcal{A}\mathcal{A}^T$, this is $\lambda_1 = \lambda_2 = ... = \lambda_k = tr(\mathcal{A}\mathcal{A}^T)/k$. The distribution yielding the maximum $f(.)$ is the one that is most skewed. For $\mathcal{A}'\mathcal{A}'^T$, this happens when there are two distinct eigenvalues, namely η^2

with multiplicity $(k-1)$ and $k+\eta^2$ with multiplicity 1. This is because, the smallest possible eigenvalue is η^2 and the trace must be accounted for.[3] We next show that the largest possible value of $f(\mathcal{A}')$ is strictly smaller than the smallest possible value of $f(\mathcal{A})$, from which the claim will follow.

Under the skewed distribution of eigenvalues of $\mathcal{A}'\mathcal{A}'^T$ assumed above, $f(\mathcal{A}') \leq \frac{k-1}{\eta^2} + \frac{1}{k+\eta^2}$. Similarly, for the uniform distribution for the eigenvalues of $\mathcal{A}\mathcal{A}^T$ assumed above, $f(\mathcal{A}) \geq k \times k/tr(\mathcal{A}\mathcal{A}^T) \geq \frac{k^2}{k+k\eta^2+3-\eta^2}$. Set η to be any value $\geq \sqrt{(k+3)}$. Then we have

$$\begin{aligned} f(\mathcal{A}') &\leq \frac{(k-1)}{(k+3)} + \frac{1}{(2k+3)} \\ &= \frac{2k(k+1)}{(k+3)(2k+3)} \\ f(\mathcal{A}) &\geq \frac{k^2}{(k+k(k+3)+3-k-3)} \\ &= \frac{k}{(k+3)}. \end{aligned} \tag{4.9}$$

Now, $2(k+1) < (2k+3)$. Multiplying both sides by $k(k+3)$ and rearranging, we get the desired inequality $f(\mathcal{A}') \leq \frac{2k(k+1)}{(k+3)(2k+3)} < \frac{k}{(k+3)} \leq f(\mathcal{A})$, showing the claim. We can obtain a tighter bound on η by solving $\frac{k-1}{\eta^2} + \frac{1}{k+\eta^2} \leq \frac{k^2}{k+k\eta^2+3-\eta^2}$, which gives us $\eta^2 \geq \frac{1}{2}[\sqrt{5k^2+4} - k + 4]$. □

In view of this, in order to find $\mathcal{B} \subset \mathcal{W}$ with $|\mathcal{B}| = k+q$ that minimizes $f(\mathcal{B})$, we can restrict attention to those sets of vectors $\mathcal{B}$ which include all the k dummy vectors.

Consider $\mathcal{B} \subset \mathcal{W}$, with $|\mathcal{B}| = k+q$ that includes all k dummy vectors. We will show in the next two claims that the trace $tr(\mathcal{B}\mathcal{B}^T) = k+k\eta^2$ will be evenly split among its eigenvalues iff $\mathcal{B} - \mathcal{D}$ encodes an exact 3-cover of X. We will finally show that it is the even split that leads to minimum $f(\mathcal{B})$.

Claim 4. *Consider a set $\mathcal{B}$, with $|\mathcal{B}| = k+q$, such that $\mathcal{B}$ includes all the k dummy vectors. Suppose the rank q matrix $\mathcal{B}' = \mathcal{B} - \mathcal{D}$ does not correspond to an exact 3-cover of X. Then $\mathcal{B}'\mathcal{B}'^T$ has q non-zero eigenvalues, at least two of which are distinct.*

[3] Such extreme skew will not arise in reality since this corresponds to all q set vectors of $\mathcal{A}$ being identical (!), but this serves to prove our result.

Proof. The q column vectors in $\mathcal{B}'$ are linearly independent, so $rank(\mathcal{B}') = rank(\mathcal{B}'\mathcal{B}'^T) = q$. Since $\mathcal{B}'\mathcal{B}'^T$ is square, it has q non-zero eigenvalues. It is sufficient to show that at least two of those eigenvalues, say λ_1 and λ_2, are unequal. As $\mathcal{B}'$ does not correspond to an exact 3-cover, at least one row has more than one 1, and so at least one row is all 0's. The corresponding row and column in $\mathcal{B}'\mathcal{B}'^T$ will also be all 0's.

Define the weighted graph induced by $\mathcal{B}'\mathcal{B}'^T$ as $G = (V, E, w)$ such that $|V| = k$, $w(i,j) = (\mathcal{B}'\mathcal{B}'^T)_{ij}, \forall i,j \in [k]$. The all-zero rows correspond to isolated nodes. We know that the eigenvalues of the the matrix $\mathcal{B}'\mathcal{B}'^T$ are identical to those of the induced graph G, which in turn are the same as those of the connected components of G. Consider a non-isolated node i. Since each row of $\mathcal{B}'$ is non-orthogonal to at least two other rows, it follows that $(\mathcal{B}'\mathcal{B}'^T)_{ij} \geq 1$ for at least 2 values of $j \neq i$. Thus, each non-isolated node is part of a connected component of size ≥ 3 and since there are isolated nodes, the number of (non-isolated) components is $< q$. Thus, the q non-zero eigenvalues of G are divided among the $< q$ components of G. By the pigeonhole principle, there is at least one connected component with ≥ 2 eigenvalues, call them λ_1, λ_2, say $\lambda_1 \geq \lambda_2$. We know that a component's largest eigenvalue has multiplicity 1, from which it follows that $\lambda_1 \neq \lambda_2$, as was to be shown. □

We next establish two helper lemmas, where $\mathcal{M}$ denotes a $k \times k$ symmetric matrix.

Lemma 4.3.2. *Let $\mathcal{M}$ be a positive semidefinite matrix of rank q. Suppose that it can be expressed as a sum of rank one matrices, i.e., $\mathcal{M} = \sum_{i=1}^{q} \mathbf{a_i} \cdot \mathbf{a_i}^T$, where $\mathbf{a_i}$ is a column vector, and $\forall i,j \in [k], i \neq j, \mathbf{a_i} \cdot \mathbf{a_j}^T = 0$, and $\mathbf{a_i} \cdot \mathbf{a_i}^T = s$. Then the q eigenvalues of $\mathcal{M}$ are identical and equal to s.*

Proof. The spectral decomposition of a rank q matrix $\mathcal{M}$ is given as

$$\mathcal{M} = \sum_{i=1}^{q} \lambda_i \mathbf{u_i} \mathbf{u_i}^{\mathbf{T}} \tag{4.10}$$

where λ_i are eigenvalues and $\mathbf{u_i}$ are orthonormal vectors. From the hypothesis of the lemma, we have $\frac{1}{s} \cdot \mathbf{a_i} \cdot \mathbf{a_i}^{\mathbf{T}} = \mathbf{1}$.

$$\mathcal{M} = \sum_{i=1}^{q} \mathbf{a_i} \mathbf{a_i}^T = \sum_{i=1}^{q} s \times \frac{\mathbf{a_i}}{\sqrt{s}} \frac{\mathbf{a_i}}{\sqrt{s}}^T \tag{4.11}$$

where $\frac{\mathbf{a_i}}{\sqrt{s}}$ are orthonormal. Comparing this with Eq. 4.10, the eigenvalues of $\mathcal{M}$ are $\lambda_1 = \lambda_2 = ... = \lambda_q = s$. □

Lemma 4.3.3. *Let $\mathcal{M}$ be a symmetric rank k matrix and suppose that it can be decomposed into $\sum_{i=1}^{q} \mathbf{a_i}\mathbf{a_i}^T + \kappa \cdot I$, for some constant κ. Then it has $(k-q)$ eigenvalues equal to κ.*

Proof. Let the eigenvalues of $\mathcal{M}$ be $\lambda_1, \lambda_2, ..., \lambda_k$. Let λ any eigenvalue of $\mathcal{M}$, and v the corresponding eigenvector. Then we have $(\mathcal{M} + \kappa I)v = (\sum_{i=1}^{q} \mathbf{a_i}\mathbf{a_i^T} + \kappa I)v = (\lambda + \kappa)v$. Since $\sum_{i=1}^{q} \mathbf{a_i}\mathbf{a_i}^T$ results in a rank q symmetric matrix, it has q non-zero eigenvalues. Adding κ to all of them, we get, $\lambda_{q+1} = ... = \lambda_k = \kappa$. □

Proof of Claim 1: Consider any set of vectors $\mathcal{B} \subset \mathcal{W}$: $|\mathcal{B}| = k + q$. By Claim 3, we may assume w.l.o.g. that $\mathcal{B}$ includes all k dummy vectors. Suppose $\mathcal{B}' := \mathcal{B} - \mathcal{D}$ encodes an exact 3-cover of X. Then $\mathcal{B}\mathcal{B}^T$ can be decomposed into the sum of q rank one matrices and a diagonal matrix: $\mathcal{B}\mathcal{B}^T = \sum_{j=1}^{q} \mathbf{b_j} \cdot \mathbf{b_j^T} + \eta^2 I$. Here $\mathbf{b_i}$ refers to the ith column of $\mathcal{B}$, which is a set vector. Since $\mathcal{B}'$ is an exact 3-cover, we further have that $\mathbf{b_i} \cdot \mathbf{b_i^T} = 3$, $i \in [q]$, and $\mathbf{b_i} \cdot \mathbf{b_j^T} = 0$, $i \neq j$. By Lemma 4.3.2, since $\mathcal{B}'\mathcal{B}'^T$ is also a positive semidefinite matrix of rank q, we have $\lambda_1^{\mathcal{B}'} = \cdots \lambda_q^{\mathcal{B}'} = 3$, where $\lambda_i^{\mathcal{B}'}$ are the eigevalues of $\mathcal{B}'\mathcal{B}'^T$. The corresponding q eigenvalues of $\mathcal{B}\mathcal{B}^T$ are all $\eta^2 + 3$. Furthermore, by Lemma 4.3.3, the remaining $k-q$ eigenvalues of $\mathcal{B}\mathcal{B}^T$ are all equal to η^2. That is, the eigenvalues of $\mathcal{B}\mathcal{B}^T$ are $\lambda_1^{\mathcal{B}} = \cdots = \lambda_q^{\mathcal{B}} = \eta^2 + 3$ and $\lambda_{q+1}^{\mathcal{B}} = \cdots = \lambda_k^{\mathcal{B}} = \eta^2$. For this $\mathcal{B}$, $f(\mathcal{B}) = tr((\mathcal{B}\mathcal{B}^T)^{-1}) = \frac{q}{\eta^2+3} + \frac{k-q}{\eta^2} = \theta$ (see Eq. 4.8).

Now, consider a set of vectors $\mathcal{A} \subset \mathcal{W}$, with $|\mathcal{A}| = k + q$, such that that $\mathcal{A}$ includes all k dummy vectors. Suppose $\mathcal{A}' := \mathcal{A} - \mathcal{D}$ does not correspond to an exact 3-cover of X. Notice that $\mathcal{A}$ is a symmetric rank k matrix which can be decomposed into $\mathcal{A} = \sum_{j=1}^{q} \mathbf{a_i} \cdot \mathbf{a_i^T} + \eta^2 I$, so $\lambda_{q+1}^{\mathcal{A}} = \cdots = \lambda_k^{\mathcal{A}} = \eta^2$, where $\lambda_i^{\mathcal{A}}$, $i \in [q+1, k]$, are $k - q$ of the eigenvalues of $\mathcal{A}\mathcal{A}^T$. Since both $\mathcal{B}$ and $\mathcal{A}$ include all k dummy vectors and q of the set vectors, by Claim 2, $tr(\mathcal{B}\mathcal{B}^T) = tr(\mathcal{A}\mathcal{A}^T) = k + k\eta^2$. We have $\sum_{j=q+1}^{q} \lambda_j^{\mathcal{B}} = (k-q)\eta^2 = \sum_{j=q+1}^{k} \lambda_j^{\mathcal{A}}$ and so $\sum_{j=1}^{q} \lambda_j^{\mathcal{A}} = \sum_{j=1}^{q} \lambda_j^{\mathcal{B}} = q(\eta^2 + 3)$. Now, $f(\mathcal{B}) = \sum_{j=1}^{k} \frac{1}{\lambda_j^{\mathcal{B}}} = \frac{q}{\eta^2+3} + \frac{k-q}{\eta^2}$, whereas $f(\mathcal{A}) = \sum_{j=1}^{k} \frac{1}{\lambda_j^{\mathcal{B}}} = \sum_{j=1}^{q} \frac{1}{\lambda_j^{\mathcal{A}}} + \frac{k-q}{\eta^2}$. Thus, to show that $f(\mathcal{B}) < f(\mathcal{A})$, it suffices to show that $\frac{q}{\eta^2+3} < \sum_{j=1}^{q} \frac{1}{\lambda_j^{\mathcal{A}}}$. LHS $= q \times \frac{1}{AM(\lambda_1^{\mathcal{B}}, ..., \lambda_q^{\mathcal{B}})} = q \times \frac{1}{AM(\lambda_1^{\mathcal{A}}, ..., \lambda_q^{\mathcal{A}})}$, where $AM(.)$ denotes the arithmetic mean. RHS $= q \times \frac{1}{HM(\lambda_1^{\mathcal{A}}, ..., \lambda_q^{\mathcal{A}})}$, where $HM(.)$ denotes the harmonic mean. It is well known that $AM(.) \geq HM(.)$ for a given collection of positive real numbers and the equality holds iff all numbers in the collection are identical. On the other hand, we know that since $\mathcal{A}'$ does not correspond to

an exact 3-cover of X, by Claim 4, not all eigenvalues of $\mathcal{A}'$ are equal, from which it follows that $LHS < RHS$, completing the proof of Claim 1 as also Theorem 4.3.1. □

After this, we show our inapproximability result for OID-MF.

4.3.2 Hardness of Approximation

Theorem 4.3.4. *It is NP-hard to approximate the optimal interview design problem in polynomial time within a factor less than* $\frac{\alpha}{\theta}$*, where* $\alpha = \theta + \frac{2}{(2+\eta^2)(4+\eta^2)(3+\eta^2)}, \eta^2 \geq \frac{1}{2}[\sqrt{5k^2+4} - k + 4]$*,* $\theta = \frac{q}{3+\eta^2} + \frac{k-q}{\eta^2}, k = 3q$ *is the dimension of the latent vector-space.*

First we define a variant of the X3C problem which we refer to as Max q-Cover by 3-Sets (M3C), which will be convenient in our proof.

Definition 4.3.1. *Given a number* q *and a collection of sets* $\mathcal{S} = S_1, S_2, ..., S_n$*, each of size 3, is there a subset* $\mathcal{S}^*$ *of* $\mathcal{S}$ *such that the cover* $C = |\bigcup_{s \in \mathcal{S}^*} s| = 3q$ *and* $|\mathcal{S}^*| \leq q$*?*

Since each set has 3 elements, with $|\mathcal{S}^*| \leq q$, we get $C = 3q$ if and only if $|\mathcal{S}^*|$ is an exact cover. Thus X3C can be reduced to M3C, making M3C NP-hard.

We convert an instance x of M3C to an instance of OID-MF, $h(x)$, in the same way as described in the NP-Hardness proof: let the set of items be $\mathcal{I} = \{a_1, ..., a_n, d_1, ..., d_k\}$, where $k = 3q$, item a_j corresponds to set S_j, $j \in [n]$, and d_j are dummy items, $j \in [k]$. Let the dummy vectors be defined as above, and $b := q + k$. As shown previously in Claim 3, we need to only consider those sets of vectors $\mathcal{B}$ that have all k dummy vectors. Similarly, we can transform a solution y of OID-MF, back to a solution of M3C, $g(y)$, in the following manner: discard the chosen dummy vectors, and take the sets corresponding to the q set vectors.

As an YES instances of M3C correspond to YES instances of X3C, an instance x with $C = 3q$ corresponds to $f(\mathcal{B}) = \theta$.

For the NO instances of M3C, $C \leq 3q - 1$ (by the definition). Unfortunately, a similar one-to-one mapping does not exist in such cases: with the same C, there could be multiple instances of M3C that corresponds to different instances of OID-MF and correspondingly $f(\mathcal{B})$. From Theorem 4.3.1, we know that it is NP-hard to determine whether $f(\mathcal{B}) \leq \theta$ for a given instance of OID-MF $h(x)$.

To find the lowest $f(\mathcal{B})$ of a NO instance of OID-MF, we first prove an intermediate result that shows that among the set of different $f(\mathcal{B})$ giving the same cover value C, the lowest possible $f(.)$ value increases as C decreases.

Claim 5. *As the cover value increases, the best (i.e., lowest) f(.) value among all the solutions with the same cover value decreases.*

Proof. Let $\mathcal{B}' = \mathcal{B} \setminus \mathcal{D}$.

By interpreting $\mathcal{B}'\mathcal{B}'^T$ as a $(k \times k)$ adjacency matrix, the dimensions correspond to the k nodes in the graph. Dimensions that are uncovered are isolated nodes, and dimensions that are covered are part of a connected component. Sum of degrees of the entire graph $= 3k$ (sum of all entries in the adjacency matrix $\mathcal{B}'\mathcal{B}'^T$) which is a constant given k.

From this, given that the sum of the degrees over the graph is $3k$ (which is a constant), we argue that with more uncovered dimensions/nodes, average degree (d_{avg}) (ignoring the isolated nodes) and maximum degree (d_{max}) increase. From this, it follows that each non-isolated node has degree at least 3, hence the average degree for such nodes is greater than 3 for any $\mathcal{B}'\mathcal{B}'^T$. If there are multiple components in a given graph, considering the one with the highest average degree, $\lambda_1 \geq max(d_{avg}^{max}, \sqrt{d_{max}})$, where d_{avg}^{max} is the highest average degree among all components.

For a NO instance, the highest average degree among all connected components is greater than 3, since the vectors must overlap at least over 1 dimension. For a given cover C, the lowest value of λ_1 is thus lower bounded by $d_{avg} > 3$, which increases as the overlap increases. In turn, a higher value of λ_1 makes the distribution of eigenvalues more skewed, leading to a higher $f(.)$. To have a lower $f(.)$, we must have λ_1 as close to 3 as possible, by decreasing d_{avg} and d_{max}, thereby, increasing coverage. □

Following this claim, among the NO instances of OID-MF, it is sufficient to show that the lowest $f(.)$ corresponds to the highest C, where $C = 3q-1$. Next we calculate its corresponding $f(.)$.

Given a NO instance with $C = 3q-1$, we next show that such an instance gives rise to a unique OID-MF solution. For this scenario, it could be shown that there are exactly $q-2$ disjoint sets, and 2 sets cover exactly one element twice. This can only be obtained from a solution y of OID-MF, if in the given solution, $q-2$ set vectors are disjoint, and 2 have exactly one 1 in the same position. The following example illustrates this.

Example 1. *For an instance with $q = 3$, a solution with exactly two vectors overlapping on one dimension could look like*

$$\mathcal{B}'^T = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Then

$$\mathcal{B}'\mathcal{B}'^T = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Next, we show what $f(\mathcal{B})$ of such a solution would be. As before, let $\mathcal{B}' := \mathcal{B} \setminus \mathcal{D}$. Interpreting $\mathcal{B}'\mathcal{B}'^T$ as the adjacency matrix of a graph G, we know that the eigenvalues of $\mathcal{B}'\mathcal{B}'^T$ are the same as those of G, given by the multi-set union of its components, which are: $q-2$ corresponding to the disjoint set vectors, and 1 corresponding to the two over-lapping vectors. The first $q-2$ components each form a 3-regular graph which contributes an eigenvalue of 3 each. It could be shown that the last one, which corresponds to the overlap, contributes to $(0, 0, 0, 0, 2, 4)$. Therefore, $f(\mathcal{B}) = \alpha = \frac{2q}{\eta^2} + \frac{q-2}{3+\eta^2} + \frac{1}{2+\eta^2} + \frac{1}{4+\eta^2} = \theta + \frac{2}{(2+\eta^2)(4+\eta^2)(3+\eta^2)}$. □

It follows from our arguments, that $f(\mathcal{B}) \geq \alpha$ if and only if $C \leq 3q - 1$.

Let $\mathcal{A}$ be an approximation algorithm that approximates OID-MF to within $c < \frac{\alpha}{\theta}$, returns a value v such that $OPT_{OID-MF}(h(x)) \leq v \leq c \times OPT_{OID-MF}(h(x))$.

Claim 6. *x is a YES instance of M3C if and only if $\theta \leq v < \alpha$.*

Proof. If $h(x)$ is a YES instance of OID-MF, $OPT_{OID-MF}(h(x)) = \theta$, so $\theta \leq v \leq c \times \theta$. Since $c < \frac{\alpha}{\theta}$, $\theta \leq v < \alpha$. If $h(x)$ is a NO instance of OID-MF, $\alpha \leq OPT_{OID-MF}(h(x))$, so $\alpha \leq v$. Since the intervals are disjoint, the claim follows. □

Thus if such an approximation algorithm $\mathcal{A}$ existed, we would be able to distinguish between the YES and NO instances of M3C in polynomial time. However as that is NP-hard, unless P = NP, $\mathcal{A}$ cannot exist.

4.3.3 Monotonicity

We define monotonicity as in Definition 2.1.1.

In [4], for a similar objective function for the user selection problem for a cold-start item, the authors prove it is monotone decreasing.

4.3.4 Supermodularity and Submodularity

We define submodularity/supermodularity as in Definition 2.1.2

In [4], for a similar objective function for the user selection problem for a cold-start item, the authors claimed that their objective function is supermodular. The following lemma shows that the objective function $f(.)$ for our OID-MF problem is not supermodular.

Lemma 4.3.5. *The objective function* $f(B) = tr((V_B V_B^T)^{-1})$ *of the OID-MF problem is* not supermodular.

Proof. We prove the result by showing that the function $tr(\mathcal{M}\mathcal{M}^T)^{-1})$ is in general not supermodular. Consider the following matrices:

$$\mathcal{M}_1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\mathcal{M}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix},$$

and vector

$$x^T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Notice that $\mathcal{M}_1$, viewed as a set of column vectors, is a subset of $\mathcal{M}_2$, viewed as a subset of column vectors. Now, $f(\mathcal{M}_1) = tr(\mathcal{M}_1\mathcal{M}_1^T) = 12, f(\mathcal{M}_1 \cup \{x\}) = 10.333, f(\mathcal{M}_2) = 6.6250, f(\mathcal{M}_2 \cup \{x\}) = 4.4783$.

Clearly, $f(\mathcal{M}_1 \cup \{x\}) - f(\mathcal{M}_1) = 10.333 - 12 = -1.6667$ and $f(\mathcal{M}_2 \cup \{x\}) - f(\mathcal{M}_2) = 4.4783 - 6.6250 = -2.1467$, which violates $f(\mathcal{M}_1 \cup \{x\}) - f(\mathcal{M}_1) \leq f(\mathcal{M}_2 \cup \{x\}) - f(\mathcal{M}_2)$, showing $f(.)$ is not supermodular. □

We remark that the lack of supermodularity of $f(.)$ is not exclusive to binary matrices; supermodularity does not hold for real-valued matrices $\mathcal{M}$ as well. As a consequence, by the duality between the technical problems of cold-start users and cold-start items, the lemma above disproves the claim in [4] about the supermodularity of their objective function.

Lemma 4.3.6. *The objective function of* $f(B) = tr((V_B V_B^T)^{-1})$ *the OID-MF problem is* not sub-modular.

Proof. Consider the same matrix $\mathcal{M}_2$ and x as above and the following matrix $\mathcal{M}_1$.

$$\mathcal{M}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

As before, regarded as sets of column vectors, $\mathcal{M}_1 \subset \mathcal{M}_2$. However, $f(\mathcal{M}_1) = 20, f(\mathcal{M}_1 \cup \{x\}) = 16.333, f(\mathcal{M}_2) = 6.6250, f(\mathcal{M}_2 \cup \{x\}) = 4.4783$. Hence $f(\mathcal{M}_1 \cup \{x\}) - f(\mathcal{M}_1) = 16.333 - 20 = -3.6667$ and $f(\mathcal{M}_2 \cup \{x\}) - f(\mathcal{M}_2) = 4.4783 - 6.6250 = -2.1467$. Clearly, submodularity fails to hold, since $f(\mathcal{M}_1 \cup \{x\}) - f(\mathcal{M}_1) \geq f(\mathcal{M}_2 \cup \{x\}) - f(\mathcal{M}_2)$ is violated. □

4.4 Algorithms

In this section, we present algorithms for selecting items with which to interview a cold user so as to learn her preferences as well as possible. In view of the hardness and hardness of approximation results (Theorems 4.3.1 and 4.3.4), and the fact that the objective function is neither submodular nor supermodular (see Section 4.3.4), efficient approximation algorithms are unlikely to exist. We present scalable heuristic algorithms for item selection.

4.4.1 Accelerated Backward Greedy

As discussed earlier, in [4], the authors study the problem of user selection for the cold-start item problem. They propose two backward greedy algorithms, called Backward Greeedy (BG) and Backward Greedy2 (BG2).

Recall that for a set of items $S \subseteq \{v_1, ..., v_n\}$, we denote by V_S the submatrix of the item latent factor matrix corresponding to the items in S and $f(V_S) := tr((V_S V_S^T)^{-1})$ is the profile learning error that we seek to

minimize by selecting the best items. It can be shown, following a similar result in [4] (Proposition 3) that $f(.)$ is monotone decreasing, i.e., for item sets $S \subseteq T$, $f(V_T) \leq f(V_S)$.

Overview. Backward greedy algorithms essentially remove the worst items from the set of all items and use the remaining ones as interview items. The core idea is to start with the set of all items and successively remove an item with the smallest increase in the error, until no more than b items are left, where b is the budget. It was claimed in [4] that the backward greedy algorithms are approximation algorithms. This is incorrect since their claim relies on the error function being supermodular. Unfortunately, their proof of supermodularity is incorrect as shown by our counterexample in Section 4.3.4. Thus, backward greedy is a heuristic for their problem as well as our OID problem.

Acceleration. In this section, we propose accelerated versions of BG and BG2, by incorporating two optimizations: (i) we speed up the matrix inversion step, necessary for evaluating the error function $f(.)$, by using the Sherman-Morrison formula with a rank-one update; (ii) we borrow ideas from the classic lazy evaluation approach, originally proposed in [33] to save on function evaluations. We briefly describe these optimizations next.

Sherman-Morrison. Suppose the current set of items is S. Let $v \in S$ be an item that is removed from S to give $S' = S \setminus \{v\}$. Suppose $(V_S V_S^T)^{-1}$ is already computed. Then $(V_{S'} V_{S'}^T)^{-1}$ can be computed as

$$\begin{aligned}(V_{S'} V_{S'}^T)^{-1} &= (V_S V_S^T - vv^T)^{-1} \\ &= (V_S V_S^T)^{-1} + \frac{(V_S V_S^T)^{-1} vv^T (V_S V_S^T)^{-1}}{1 - v^T (V_S V_S^T)^{-1} v}.\end{aligned}$$

Lazy evaluation. Lazy evaluation, originally proposed as a way to speed up the greedy algorithm for submodular function maximization, can be easily adapted to a supermodular function minimization as follows. Suppose $g(S)$ is a supermodular set function that is monotone decreasing and we want to find a set S of size $\leq b$ that minimizes $g(S)$, using the backward greedy framework. Let S_i be the set of items in iteration i of a backward greedy algorithm. Then for $j > i$, clearly $S_j \subset S_i$. Define $\tilde{g}(u|S) := g(S \setminus \{u\}) - g(S)$, i.e., increase in error from dropping item u from set S. If g is supermodular, it follows that $\tilde{g}(u|S_i) \leq \tilde{g}(u|S_j)$, whenever $u \in S_j$, $i < j$. This follows upon noting that the function $\tilde{g}(u|S)$, denoting increase in error on dropping u from S is actually the negative of the marginal gain of adding u to $S \setminus \{u\}$, i.e., $\tilde{g}(u|S) = -g(u|S)$. Suppose there are items $u \in S_i$ and $v \in S_j$, with $i < j$, such that $\tilde{g}(u|S_i) \geq \tilde{g}(v|S_j)$. Then there is no need to evaluate

$\tilde{g}(u|S_j)$, since $\tilde{g}(u|S_j) \geq \tilde{g}(u|S_i) \geq \tilde{g}(v|S_j)$. This saving on evaluations can be implemented by keeping track of when each error increment (or marginal gain) was last updated and making use of a priority queue for picking the element with the least error increment.

Recall that our error function $f(.)$ is actually *not* supermodular. Our main goal in applying lazy evaluation to it is not only to accelerate item selection, but also explore the impact of lazy evaluation on the error performance.

We give the pseudocode for the accelerated version of BG2 in Algorithm 2. The algorithm corresponding to accelerated BG can be easily obtained from this, as we explain below.

We start with B initialized to all items. We use a priority queue for efficient implementation of lazy evaluation. The "freshness" check in Line 10 in Algorithm 2 checks that the error increment in α_j was computed in the latest iteration to make sure v_j is indeed the item with the least error increment. Notice that we use the notation $\tilde{f}(v_j|V_B, C_B)$ where V_B is the latent factor matrix corresponding to the items in B and C_B is the covariance matrix. C_B is initialized to the diagonal matrix of noise terms C, where $C_{jj} = \sigma_j^2$, $v_j \in B$.

The use of Sherman-Morrison optimization allows us to save on repeated invocation of matrix inverse, and instead allows it to be computed incrementally and hence efficiently using rank one update. The use of lazy evaluation saves on evaluations of error increments that are deemed redundant, assuming (pretending, to be more precise) that $f(.)$ is supermodular. We will evaluate both the prediction and profile error performance as well as the running time performance of these optimizations in Section 4.5.

The algorithm for the accelerated version of basic Background Greedy (BG) differs from Algorithm 2 by simply assuming that the noise terms are identical, i.e., $\sigma_1^2 = \cdots = \sigma_n^2 = \sigma^2$, i.e., we set C to $\sigma^2 \cdot I$, where I is the identity matrix. This saves some work compared to Algorithm 2. We refer to this modified algorithm as ABG1 and omit its pseudocode for brevity.

4.4.2 Accelerated Forward Greedy

In a real recommender system, the number n of items may be in the millions and b may be $<< n$. One key shortcoming of the backward greedy family of algorithms (BG and BG2 and their accelerated versions) is that they need to sift through a large number of items and eliminate them one by one till the budget b is reached. One approach for remedying this is to consider a forward greedy approach.

Algorithm 2 Accelerated Backward Greedy 2 (ABGD2)

Input: item set $\mathcal{I}$ and corresponding matrix V; budget b; diagonal matrix of noise terms C.

Output: items subset $B, |B| = b$.

$V_B \leftarrow V$
$B \leftarrow \mathcal{I}$
$C_B \leftarrow C$
for $j \leftarrow 1$ to $|\mathcal{I}|$ **do**
    insert $(v_j, \tilde{f}(v_j|V_B, C_B))$ into priority queue Q
end for
do
    pop (v_j, α_j) from Q
    if α_j not "fresh" **then**
        recompute $\alpha_j = \tilde{f}(v_j|V_B, C_B)$
    end if
    if $\alpha_j <$ MIN(Q) **then**
        $V_B \leftarrow V_B \setminus v_j$
        $B \leftarrow \mathcal{I} \setminus \{v_j\}$
        $C_B \leftarrow C_B \setminus \sigma_j^2$
    else
        insert (v_j, α_j) into Q
    end if
while $|V_B| > b$

Overview. Recall that the function $f(.)$ is monotone decreasing, so $-f(.)$ is monotone increasing. We start with B initialized to the empty set of items, which has the highest error and hence $-f(\emptyset)$ has the smallest value. At each iteration, we add to B an item that has the maximum marginal gain w.r.t. $-f(.)$. That is, we successively add

$$\begin{aligned} v^* &= \arg\max_{v \in \mathcal{I} \setminus B} [-f(V_{B \cup \{v\}} | C_{B \cup \{v\}}) - (-f(V_B) | C_B)] \\ &= \arg\max_{v \in \mathcal{I} \setminus B} [f(V_B | C_B) - f(V_{B \cup \{v\}} | C_{B \cup \{v\}})] \end{aligned}$$

to B until the budget b is reached. In the algorithm, we use $-\tilde{f}(v_j | V_B, C_B)$ to denote $[f(V_B|C_B) - f(V_{B\cup\{v\}}|C_B \cup \{v\})]$.

As with accelerated backward greedy, Sherman-Morrison formula and lazy evaluation are used to optimize forward greedy. The resulting algorithm, referred to as Accelerated Forward Greedy 2, is depicted in Algorithm 3. It can be also be adapted to get Accelerated Forward Greedy 1 (AFG1), by setting $C = \sigma^2 \cdot I$, as we did for ABG1.

In the next section, we conduct an empirical evaluation of the forward and backward greedy algorithms as well as their accelerated versions proposed here and compare them against baselines.

4.5 Experimental Evaluation

In this section, we describe the experimental evaluation for our algorithms, and compare them with prior art. We evaluate our solutions both qualitatively and scalability-wise: quality evaluation is done by measuring *prediction error* and *user profile estimation error* (see Section 4.5.2), whereas, the scalability study is conducted by measuring the running time.

The development and experimentation environment uses a Linux Server with 2.93 GHz Intel Xeon X5570 machine with 98 GB of memory with OpenSUSE Leap OS.

4.5.1 Dataset and Model

For our experiments, we use datasets from the movie recommendation domain – Netflix [4] and Movielens (ML) [5].Moreover, we use three different available ML datasets that gives us a total of four different datasets. We describe their characteristics in Table 4.1.

[4]The original dataset was released as part of The Netflix Prize [7] but has since been removed from the public domain due to privacy concerns.

[5]Available at `http://grouplens.org/datasets/movielens/`. Source: [19]

Algorithm 3 Accelerated Forward Greedy 2 (AFG2)

Input: item set $\mathcal{I}$ and corresponding matrix V; budget b; diagonal matrix of noise terms C.

Output: items subset $B, |B| = b$

$V_B \leftarrow \phi$
$B \leftarrow \phi$
for $j \leftarrow 1$ to $|\mathcal{I}|$ **do**
    insert $(v_j, -\tilde{f}(v_j|V_B, C_B))$ into priority queue Q
end for
do
    pop (j, α_j) from Q
    if α_j not "fresh" **then**
        recompute $\alpha_j = -\tilde{f}(v_j|V_B, C_B)$
    end if
    if $\alpha_j >$ MAX(Q) **then**
        $V_B \leftarrow V_B \cup \{v_j\}$
        $B \leftarrow \mathcal{I} \cup \{v_j\}$
    else
        insert (v_j, α_j) into Q
    end if
while $|V_B| < b$

Table 4.1: Dataset Sizes

Dataset	# Ratings	# Users	# Items	Sparsity
ML 100K	100,000	943	1682	6.3%
ML 1M	1,000,209	6,040	3,900	4.25%
ML 20M	20,000,263	138,493	27,278	0.53%
Netflix	100,480,507	480,189	17,770	1.18%

4.5.2 Model Parameters & Experimental Setup

For each dataset, we train a probabilistic matrix factorization model [39] on only the ratings given by 70% of the users. We refer to them as the warm users, $\mathcal{U}$. We use gradient descent algorithm [48] to train the model, with latent dimension = 20, momentum = 0, regularization = 0.1 and linearly decreasing step size for faster convergence. This allows us to use large steps while initially moving towards the minima, decreasing as we approach it to avoid overshooting. We report the number of warm users, number of items rated by them, and RMSE obtained, for the different datasets in Table 4.2. Note that # Items in Table 4.1 is different from $|\mathcal{I}|$ in Table 4.2. This is because some items were not rated by the warm users. We had to discard such items from consideration, as we could not build item profiles for them.

Table 4.2: Experiment Sizes

Dataset	# Warm Users $\lvert\mathcal{U}\rvert$	$\lvert\mathcal{I}\rvert$	RMSE
ML 100K	702	1647	0.9721
ML 1M	4,473	3,666	0.8718
ML 20M	102,628	25,529	0.7888
Netflix	355,757	17,770	0.8531

Experimental Setup: We simulate the cold user interview process as follows:

1. Set up the system
   (a) Randomly select 70% of the users in a given dataset to train the model ($\mathcal{U}$)
   (b) $R :=$ Matrix of ratings given by $\mathcal{U}$ only

(c) Train a PMF model on R, to obtain U, V

2. Construct item covariance matrix C given by,
$\sigma_j^2 := \frac{1}{|R_{*j}|} \sum_{i \in R_{*j}} (R_{ij} - \hat{R}_{ij})^2$, where R_{*j} refers to the column(set) of ratings received by item j

3. For each cold user $u_\ell \notin \mathcal{U}$,

   (a) Construct U_ℓ using gradient descent method [37], and using the item latent factor matrix V

   (b) Randomly split items they have rated, into candidate pool CP and test set $Test$

   (c) Run item selection algorithm on CP with corresponding V and budget $= b$

   (d) $B :=$ items returned by algorithm to interview u_ℓ

   (e) Reveal $R_\ell := u_\ell$'s ratings on B

   (f) Construct $\hat{U}_\ell := (\gamma I + V_B C_B^{-1} V_B^T)^{-1} V_B C_B^{-1} R_\ell$

   (g) Evaluation: RMSE on $Test := \sqrt{\frac{1}{|Test|} \sum_{v_j \in Test} (R_{\ell j} - V_j^T \hat{U}_\ell)^2}$, profile error $:= ||\hat{U}_\ell - U_\ell||_F^2$

4. Average prediction and profile error over all cold users

In Step 2, we estimate the noise terms using the method outlined in [4]. For the case where the covariance matrix $C = \sigma^2 I$, we estimate σ^2 that best fit a validation set (a randomly chosen subset of the cold user ratings).

In Step 3a we compute *true* latent vectors of the cold users. We cannot compute that using the PMF model in Step 1c, as these ratings are hidden at that stage. Moreover, since each cold user is independent, we cannot train a model on their combined pool of ratings. Instead, we adopt the gradient descent based method given in [37], to generate the latent vector for a new user keeping everything else constant. This method produces results comparable to retraining the entire model globally (with 1% error in the worst case), in a few milliseconds [37]. The user profiles thus generated are used as *true profiles*.

In Step 3g, the estimated rating is computed by taking an inner product between the item and the cold user vector $\hat{U}_\ell$. For the real rating $R_{\ell j}$, we study two different settings: in one we use ratings that were provided by u_ℓ on $Test$, but were kept hidden from the system until the evalution stage, and in the other $R_{\ell j} = V_j^T U_\ell$. We call the first one the *real* setting, as

we use the real ratings provided by the cold user, and we call the second one the *ideal* setting, as it corresponds to an ideal, zero error MF model. Studying our algorithms under the *ideal* setting has two advantages: first, it allows us to decouple our problem from the problem of tuning a matrix factorization model. This way, the model error from using a possibly less than perfect PMF model does not percolate to our problem. Second, we do not have to be limited to only selecting the items for which we have ratings in our dataset, as we can generate ratings for all items. This is especially crucial for sparse datasets, like ours.

4.5.3 Algorithms Compared

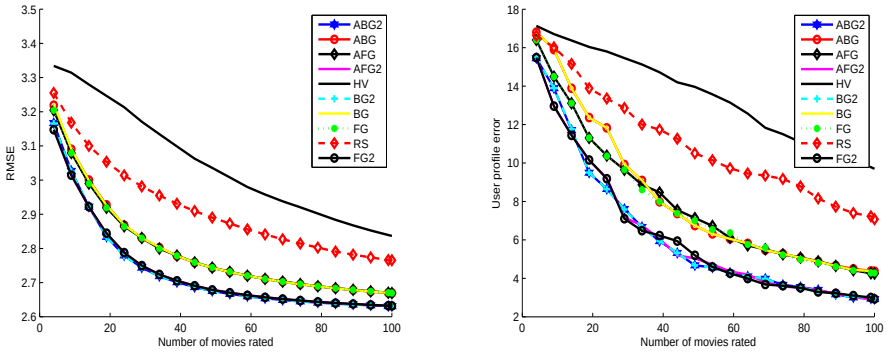
We compare the following algorithms against their accelerated versions, as described in Section 4.4: Backward Greedy Selection 1 (`BG`), Backward Greedy Selection 2 (`BG2`), Forward Greedy Selection (`FG`) and Forward Greedy Selection 2 (`FG2`). Further, we use the following two heuristics as baselines: Random Selection (`RS`), where the items are randomly sampled from the candidate pool, and High Variance (`HV`), where b items with the highest variance in rating prediction among warm users are selected. This gives us a total of 10 algorithms to compare.

4.5.4 Quality Experiments

Results: We run quality experiments to measure prediction error and profile error for all five datasets. For the datasets ML 100K and ML 1M, we compare all 10 algorithms under the *ideal* setting, where we note that `BG` performs almost exactly the same as `FG` (see Fig. 4.1), while `BG2` and `FG2` perform better for both profile and prediction error. For the larger datasets Netflix and ML 20M, we compare algorithms under the *real* setting. For both, we observe that `FG2` outperforms `BG2` for both prediction and profile error, and `FG` outperforms `BG` for smaller values of b.

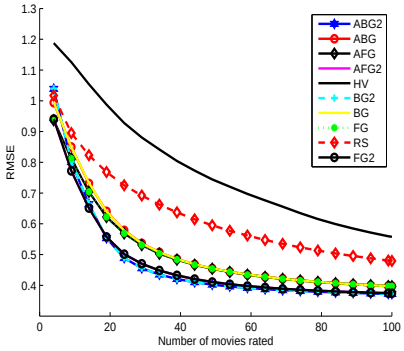
We also observe that algorithms that produce low profile error also produce low prediction error. The only exception is `RS` and `HV` in Figures 4.2, 4.3.

Despite the lack of supermodularity or submodularity, the accelerated variants of all the algorithms perform akin to their non-accelerated variants on both prediction and profile error, for all four datasets (Fig. 4.1, 4.2, 4.3). This suggests that the objective function may be close to satisfying supermodularity. This requires further investigation.

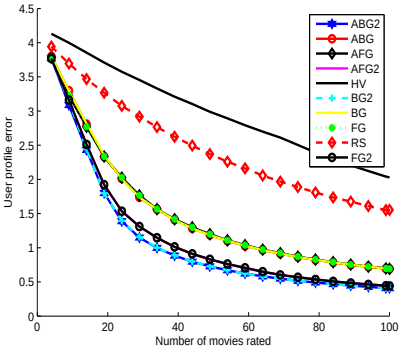


(a) Prediction Error – ML 100K

(b) Profile Error – ML 100K

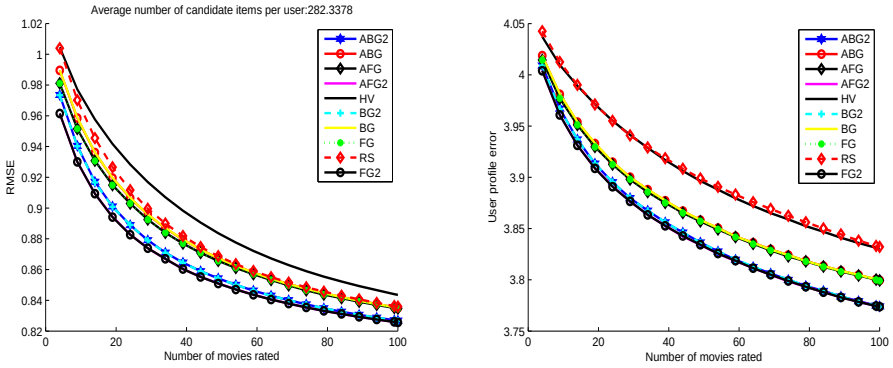


(c) Prediction Error – ML 1M



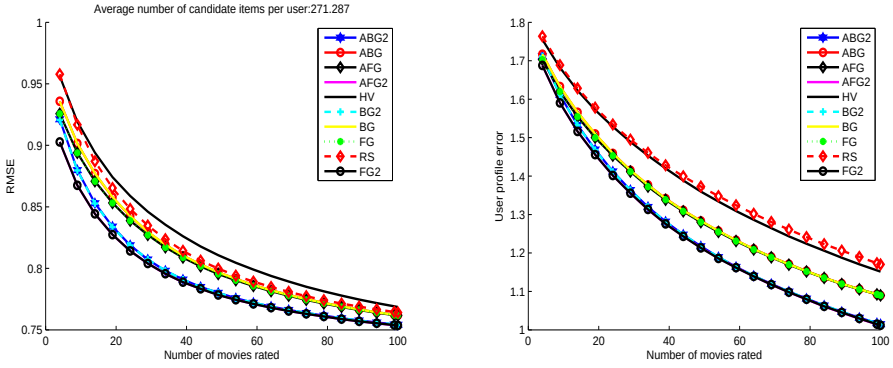
(d) Profile Error – ML 1M

Figure 4.1: Movielens 100K and 1M Datasets



(a) Prediction Error (b) Profile Error

Figure 4.2: Netflix Dataset



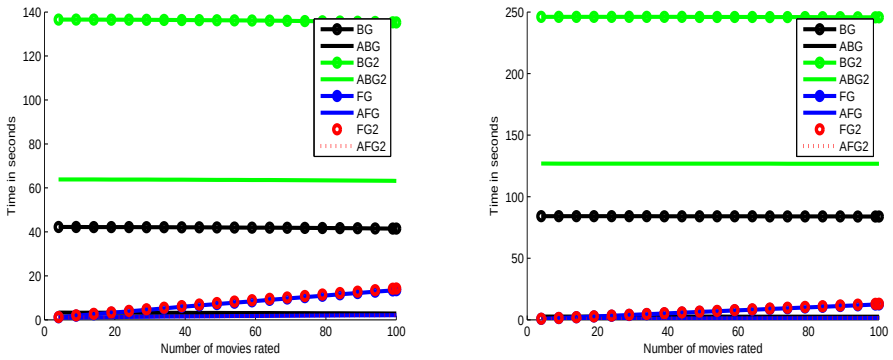
(a) Prediction Error (b) Profile Error

Figure 4.3: Movielens 20M Dataset

4.5.5 Scalability Experiments

To test scalability of our proposed solutions we run all 10 algorithms on 2 of the datasets, ML 100K and ML 1M under *ideal* setting and on Netflix and ML 20M under *real* setting, and measure running times with varying budget. Note that due to the datasets' sparsity, the average number of items per cold user that the algorithms sift through in the *real* setting ranges from 271 to 282, while in the *ideal* setting, it is significantly more (1647 and 3666 for ML 100K and ML 1M respectively).

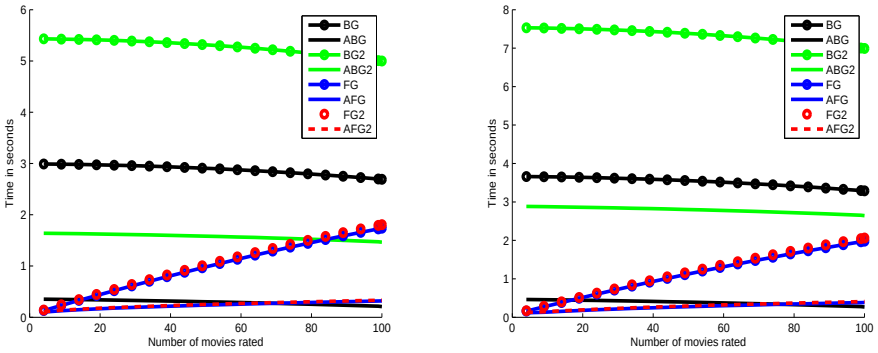
Results: In all cases, the accelerated algorithms produce error similar to their un-accelerated counterparts (Fig. 4.1, 4.2, 4.3), but running time performance is far superior (Fig. 4.4, 4.5). Among all algorithms, `FG2` (both accelerated and unaccelerated) has the best qualitative performance, with prediction and profile error comparable to `BG2` (Fig. 4.1) or better (Fig. 4.2, 4.3), and is significantly faster than `BG2` in terms of running time. In fact, even for ML 100K, our smallest dataset, under the *ideal* setting, the time taken by unaccelerated `FG2` for $b = 100$ is approximately a sixth of the time taken by `ABG2` for $b = 4$. Moreover, running times of all backward greedy algorithms increase significantly as we decrease b (see Fig. 4.4, 4.5), which makes them unsuitable for use in a real world system, where b would typically be very small.



(a) Running Time – ML 100K

(b) Running Time – ML 1M

Figure 4.4: We measure the running time by varying b from 4 to 100 for ML 100K and ML 1M, averaged across 200 and 1000 cold users respectively. The plots show the performance of all 10 greedy algorithms on a candidate pool of 1682 and 3952 items respectively.



(a) Running Time – Netflix

(b) Running Time – ML 20M

Figure 4.5: We measure the running time by varying b from 4 to 100 for Netflix and ML 20M, averaged across 5000 cold users.

Chapter 5

Conclusion

In this thesis, we studied theoretical properties of the cold-start user problem in collaborative filtering recommender systems. We studied this problem under the two most prevalent frameworks for collaborative filtering – neighbourhood-based methods and matrix factorization. More specifically, we studied which items to interview a cold-start user with, so as to best learn their preferences. We call this the optimal interview design (OID) problem, and we formalized what it means to learn the user's preferences best under the two different frameworks – OID-NB for neighbourhood-based methods and OID-MF for matrix factorization.

We proved that both OID-NB and OID-MF are NP-hard, and proved bounds on their approximability. We also studied their monotonicity and submodularity/supermodularity properties, proposed various efficient algorithms and evaluated their performance comprehensively on two and four real world datasets, for the neighbourhood based framework and the matrix factorization framework respectively. For the first, we considered only item-item similarity collaborative filtering and absolute thresholding for neighbour selection, but it would be interesting to study the same problem with user-user similarity and top-n neighbour selection methods. In our experiments, we found that the item-item similarity matrix computation was a serious bottleneck. Although some scalable alternatives have been proposed [6, 15, 46], further research is needed to adapt them for the cold-start problem.

For the second part, we demonstrated the importance of learning the cold-start user's profile, and how that translates to improved rating prediction performance. Our proposed algorithm not only outperformed the state of the art in terms of learning the user's preferences, but was also many times faster. We also observed that our proposed accelerated algorithms performed akin to the unaccelerated variants, suggesting that the objective function is close to supermodular. One future research direction would be to see whether it satisfies weak supermodularity [9].

This work can be also extended to the cold-start item problem, which is analogous to what we studied and the cold-start system problem as well.

Another direction to explore would be to study multiple cold-start users at the same time. In this work, we assumed that each cold-start user was independent. Different noise, ratings and covariance models can also be explored.

Lastly, the OID problem can be studied in an interactive setting, where the response from the user on one item is used to determine the next. It could be combined with an implicit rating model, where the user does not need to explicitly rate the items served to her, but indicates her preference through actions such as clicking on the item, making it a favorite, or buying it.

Bibliography

[1] Jacob Abernethy, Kevin Canini, John Langford, and Alex Simma. Online collaborative filtering. *University of California at Berkeley, Tech. Rep*, 2007.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[3] Hyung Jun Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, 178(1):37–51, 2008.

[4] Oren Anava, Shahar Golan, Nadav Golbandi, Zohar Karnin, Ronny Lempel, Oleg Rokhlenko, and Oren Somekh. Budget-constrained item cold-start handling in collaborative filtering recommenders via optimal design. In *Proceedings of the 24th International Conference on World Wide Web*, pages 45–54. International World Wide Web Conferences Steering Committee, 2015.

[5] Senjuti Basu Roy, Laks VS Lakshmanan, and Rui Liu. From group recommendations to group formation. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1603–1616. ACM, 2015.

[6] Robert M Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 43–52. IEEE, 2007.

[7] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[8] JesúS Bobadilla, Fernando Ortega, Antonio Hernando, and JesúS Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26:225–238, 2012.

[9] Christos Boutsidis, Edo Liberty, and Maxim Sviridenko. Greedy minimization of weakly supermodular set functions. *arXiv preprint arXiv:1502.06528*, 2015.

[10] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

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

[12] Stéphane Caron and Smriti Bhagat. Mixing bandits: A recipe for improved cold-start recommendations in a social network. In *Proceedings of the 7th Workshop on Social Network Mining and Analysis*, page 11. ACM, 2013.

[13] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. *Collaborative Filtering Recommender Systems*. Now Publishers Inc, 2011.

[14] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[15] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4–pp. IEEE, 2005.

[16] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 595–604, New York, NY, USA, 2011. ACM.

[17] Amit Goyal and Laks VS Lakshmanan. Recmax: Exploiting recommender systems for fun and profit. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1294–1302. ACM, 2012.

[18] Mikael Hammar, Robin Karlsson, and Bengt J Nilsson. Using maximum coverage to optimize recommendation systems in e-commerce. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 265–272. ACM, 2013.

[19] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2016.

[20] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[21] Dorit S Hochba. Approximation algorithms for np-hard problems. *ACM SIGACT News*, 28(2):40–52, 1997.

[22] Yanxiang Huang, Bin Cui, Jie Jiang, Kunqian Hong, Wenyu Zhang, and Yiran Xie. Real-time video recommendation exploration. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 35–46. ACM, 2016.

[23] Rasoul Karimi, Alexandros Nanopoulos, and Lars Schmidt-Thieme. A supervised active learning framework for recommender systems based on decision trees. *User Modeling and User-Adapted Interaction*, 25(1):39–64, 2015.

[24] Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[25] Heung-Nam Kim, Ae-Ttie Ji, Inay Ha, and Geun-Sik Jo. Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation. *Electronic Commerce Research and Applications*, 9(1):73–83, 2010.

[26] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[27] Xuan Nhat Lam, Thuc Vu, Trong Duc Le, and Anh Duc Duong. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 208–211. ACM, 2008.

[28] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, 2014.

[29] Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 283–292. ACM, 2013.

[30] Jérémie Mary, Romaric Gaudel, and Philippe Preux. Bandits warm-up cold recommender systems. *CoRR*, abs/1407.2806, 2014.

[31] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24. ACM, 2007.

[32] Paolo Massa and Bobby Bhattacharjee. Using trust in recommender systems: an experimental analysis. In *Trust Management*, pages 221–235. Springer, 2004.

[33] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, pages 234–243. Springer, 1978.

[34] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[35] Seung-Taek Park and Wei Chu. Pairwise preference regression for cold-start recommendation. In *Proceedings of the third ACM conference on Recommender systems*, pages 21–28. ACM, 2009.

[36] Al Mamunur Rashid, George Karypis, and John Riedl. Learning preferences of new users in recommender systems: an information theoretic approach. *ACM SIGKDD Explorations Newsletter*, 10(2):90–100, 2008.

[37] Steffen Rendle and Lars Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 251–258. ACM, 2008.

[38] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. Trust management for the semantic web. In *International semantic Web conference*, pages 351–368. Springer, 2003.

[39] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. Citeseer, 2011.

[40] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[41] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28. Citeseer, 2002.

[42] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[43] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating word of mouth. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

[44] Linqi Song, Cem Tekin, and Mihaela van der Schaar. Clustering based online learning in recommender systems: a bandit approach. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 4528–4532. IEEE, 2014.

[45] Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.

[46] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121. ACM, 2005.

[47] Mi Zhang, Jie Tang, Xuchen Zhang, and Xiangyang Xue. Addressing cold start in recommender systems: A semi-supervised co-training algorithm. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 73–82. ACM, 2014.

[48] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116. ACM, 2004.

[49] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. Interactive collaborative filtering. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1411–1420. ACM, 2013.

[50] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 315–324, New York, NY, USA, 2011. ACM.