Cold-Start Collaborative Filtering

Xiaoxue Zhao

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of
University College London.

Department of Computer Science
University College London

18 January 2016

I, Xiaoxue Zhao, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Collaborative Filtering (CF) is a technique to generate personalised recommendations for a user from a collection of correlated preferences in the past. In general, the effectiveness of CF greatly depends on the amount of available information about the target user and the target item. The cold-start problem, which describes the difficulty of making recommendations when the users or the items are new, remains a great challenge for CF. Traditionally, this problem is tackled by resorting to an additional interview process to establish the user (item) profile before making any recommendations. During this process the user's information need is not addressed. In this thesis, however, we argue that recommendations would be preferably provided right from the beginning. And the goal of solving the cold-start problem should be maximising the overall recommendation utility during all interactions with the recommender system. In other words, we should not distinguish between the information-gathering and recommendation-making phases, but seamlessly integrate them together. This mechanism naturally addresses the cold-start problem as any user (item) can immediately receive sequential recommendations without providing extra information beforehand.

This thesis solves the cold-start problem in an interactive setting by focusing on four interconnected aspects. First, we consider a continuous sequential recommendation process with CF and relate it to the exploitation-exploration (EE) trade-off. By employing probabilistic matrix factorization, we obtain a structured decision space and are thus able to leverage several EE algorithms, such as Thompson sampling and upper confidence bounds, to select items. Second, we extend the sequential recommendation process to a batch mode where multiple recommendations are made at each interaction stage. We specifically discuss the case of two consecutive interaction stages, and model it with the partially observable Markov decision process (POMDP) to obtain its exact theoretical solution. Through an in-depth analysis of the POMDP value iteration solution, we identify that an exact solution can be abstracted as selecting users (items) that are not only highly relevant to the target according to the initial-stage information, but also highly correlated with other *potential* users (items) for the next stage. Third, we consider the intra-stage recommendation optimisation and focus on the problem of personalised item diversification. We reformulate the latent factor models using the mean-variance analysis from the portfolio theory in economics. The resulting portfolio ranking algorithm naturally captures the user's interest range and the uncertainty of the user preference by employing the variance of the learned user latent factors, leading to a diversified item list adapted to the individual user. And, finally, we relate the diversification algorithm back to the interactive process by considering inter-stage joint portfolio diversification, where the recommendations are optimised jointly with the user's past preference records.

Acknowledgements

Though only my name appears on the cover of this thesis, a great many people have contributed to its production. I would like to acknowledge all those who have encouraged, inspired, supported, assisted, and brought sacrifices to help my pursuit of the Ph.D. degree.

My deepest gratitude goes to my first supervisor, Dr. Jun Wang, for his guidance, expertise, and patience. I have been very fortunate to have a supervisor who allowed me a great amount of leeway to explore on my own, and at the same time guided me onto the right track throughout the course of my studies. Jun introduced me to the field of recommender systems and information retrieval. He also taught me how to ask questions, formulate problems and present ideas. His patience and support helped me overcome many research bottlenecks and finally complete my Ph.D. journey.

I am also very grateful to my second supervisor, Dr. Ingemar Cox, for the insightful discussions that helped me gain a bigger picture of my work. I am also thankful to him for his comments on the structure, grammars and notations in my writings.

Beside my supervisors, I would like to thank Dr. Emine Yilmaz for reviewing my first-year and transfer vivas and for her critical advice. I would like to express my sincere gratitude to Dr. Licia Capra and Dr. Mark Levene for sparing their time to review my thesis as my Ph.D. committee members. I would also like to thank Dr. Zhengxiao Guo for his advice on my academic career.

In addition, I have also crossed paths with many other doctorate students and post-doctoral researchers who have influenced and enhanced my research. My Ph.D. studies started with intensive collaborations with Dr. Yue Shi from Delft University of Technology, which were vital for me to adapt to the new research area. Throughout my Ph.D., I have had a great many of enjoyable discussions with Weinan Zhang, who has provided me the benefits of his technical expertise and his academic acumen. I have expanded my horizon to venture finance recommendation thanks to Dr. Thomas Stone, and to online advertising research thanks to Dr. Shuai Yuan and Dr. Bowei Chen. I have learnt a lot about theoretical optimal control through discussions with Dr. Thomas Furmston. I would also like to thank two former members of my research team, Dr. Tamas Jambor, and Dr. Jagadeesh Gorla for always being there to help me in both my research and my career.

The influence of those with whom I worked as a master student before coming to UCL also continues to be important. My former supervisor Dr. Jiping Huang of Fudan University invited me to give a presentation of my current research to his group and allowed me to exchange ideas with his current students. It was also him who first guided me into academic research. I have also been lucky enough to be an exchange student to Kyoto University, Hokkaido University, and The Chinese University of Hong

Kong during my master degree, and have benefited from supervision under Dr. Shigeru Shinomoto, Dr. Kousuke Yakubo and Dr. Kin Wah Yu respectively.

I would also like to thank my friends Dr. Xiaoyu Han, Yiqi Deng, Runing Ye, Tingting Lu, Haitang Luo, Dr. Lingjian Yang, Lijuan Guan, Dr. Caifa Guo, Dr. Xiaobei Zheng, and all who have shared my happiness and supported me through difficult times.

My family has been always encouraging, supportive and shown belief in me and my work. My parents are themselves academic scholars, and we shared almost everything during my research. Especially because my mother herself is a dedicated researcher in computer science, I could therefore have long discussions on many topics with her. But her most important influence is a curious mind, a critical attitude and a genuine passion for research. I feel deeply indebted to them for allowing their only child to travel to a country so far for the pursuit of her career. At the same time, I also feel that my coming to UCL was fateful, since here I met my better half, Tom. Tom has always made me feel beloved and supported, cared for and encouraged. He eased my anxiety in difficult times. He was also the first reader of my thesis, pointing out and correcting grammar and spelling mistakes. My cousin, Annie, who accommodated me when I first arrived in London, was also very encouraging and supportive throughout my studies.

Without the help of the many people mentioned above, this thesis would not have been possible.

Contents

List of Figures

List of Tables

Notation

The following notations are used throughout this thesis. In addition to their definitions here, they are also described in their first occurrences in each chapter. For the reader's convenience, we describe chapter-specific notations separately in Table 3.1, Table 4.1, Table 5.1 and Table 6.1 for Chapters 3-6.

Notation	Description
$\mathbf{X}$	A matrix (a bold upper-case letter)
$\boldsymbol{x}$	A vector (a bold italic lower-case letter)
$\boldsymbol{X}$	A random vector (a bold italic upper-case letter)
$\|\mathbf{X}\|$	Determinant of $\mathbf{X}$
$\mathbf{X}^T$	Transpose of $\mathbf{X}$
diag$[\mathbf{X}]$	A vector composed of the diagonal elements of $\mathbf{X}$
diag$[\boldsymbol{x}]$	A diagonal matrix with its diagonal elements as $\boldsymbol{x}$
Dg$[\mathbf{X}]$	A diagonal matrix with the diagonal elements of $\mathbf{X}$
$p(\cdot)$	Density function
$\mathbb{E}[\cdot]$	Expectation
$\mathrm{Var}[\cdot], \mathrm{Cov}[\cdot]$	Variance, covariance matrix
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$\sum, \prod$	Sum, product
C_k^n	The number of k-combinations from a given set of n elements
$\mathbb{I}[\cdot]$	Indicator function
$\mathbf{I}$	Identity matrix
$\emptyset$	Empty set
ξ	A random noise with zero mean

Chapter 1

Introduction

For approximately the last two decades, information retrieval has fundamentally transformed the way in which people seek and work with information. Roughly speaking, there are two types of information retrieval (IR) systems [3]. On one hand, we have *ad hoc* information retrieval, e.g., web search [4], which deals with a relatively fixed collection of information items (webpages, documents, images, product descriptions etc.) and explicit user information requests. On the other hand, there are information filtering systems, such as recommender systems [5, 6], to address the situation where the information is actively filtered for users based on their preferences and implicit behavioural data without explicit personal need. Nevertheless, in either case, the fundamental problem remains the same, which is how to compute and find the *match* between the information items and information requests [7].

The task of recommender systems in general can be illustrated as in Figure 1.1. In this graph, red and blue blocks represent the previously expressed preferences by the users. The blank blocks are unknown preferences that need to be predicted by the system. There are mainly two ways to make the preference prediction. One way is to make use of the demographic information of users and the content information of items, such as the gender, age and location of users, and the genre, release date and tags of items; this is referred to as content-based recommendation [8]. The other way purely relies on the collective rating information between users and items without any forms of content information; this is referred to as *Collaborative Filtering* (CF) [9]. The intuition behind CF is the real-world "word of mouth" phenomenon: users who had similar taste in the past are likely to have similar preferences in the future, and likewise for the items. The name "collaborative filtering" was coined in 1992 in a pioneer work on an automatic electronic mail filtering system called *Tapestry* [10], one of the first recommender systems. The term "collaborative" suggests the collective usage of the information involving multiple users and items in such systems, instead of explicit collaboration between the system and the users [5]. CF was popularised through the Netflix competitions which started in 2006*, in which it has played a central role to provide efficient and accurate recommendation models [11]. Compared to a content-based recommender system, CF is not limited to the availability of the content information, and it can also overcome the over-specification problem that is usually suffered by a content-based recommender system [8].

*http://www.netflixprize.com/

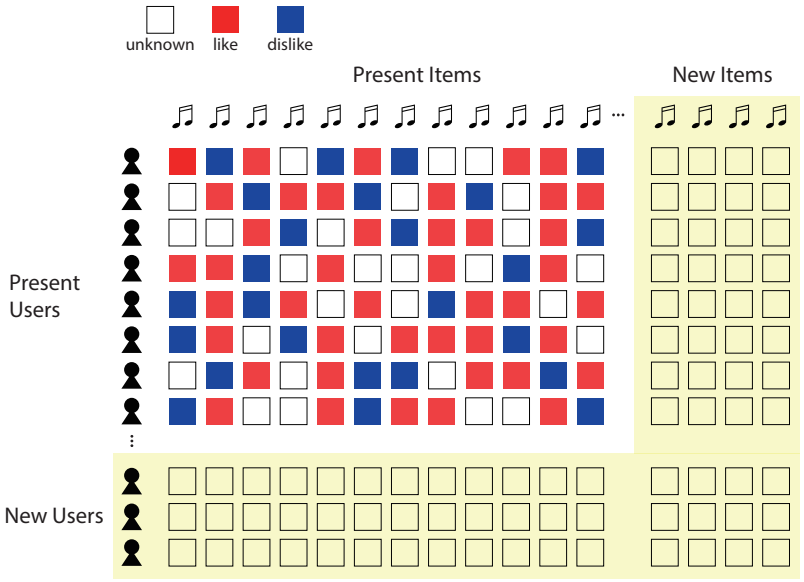


Figure 1.1: A schematic illustration of the task of a recommender system. Positive and negative preferences are shown in red and blue respectively. Blank blocks are unknown preferences which are to be predicted by the system. The shaded areas represent the situations when new users and new items join the system with no preference information available yet. How to predict the related preference values is referred to as the cold-start problem.

Despite its many advantages, a major problem for CF is that the quality of recommendations largely depends on whether there is sufficient rating information on the users and items [12]. Especially, when new users or new items just join the system and there is no rating information about them yet, it is difficult to initiate recommendations by CF. This is referred to as the cold-start problem [13]. In Figure 1.1, we have used shaded areas to highlight these users and items as well as the associated preferences to predict. To make recommendations for these users/items, the preferences within the entire shaded area need to be predicted, whereas no existing preferences about them can be used to support this process.

Many efforts have been made to solve the cold-start problem. Especially, much work has been done on utilising additional information about the users and items. In [14, 15], sources such as age and gender of a user and genres and tags of an item are incorporated into their rating prediction model. In [16], social information, such as twitter following relationship, is used to make recommendations. However, such content-based information is not always available, requiring pure CF-based algorithms for solving cold-start problems.

Within the scope of CF, an additional process is usually introduced to solve the cold-start problem, e.g., an interview process [17, 18] to first learn the user profile, and then to make recommendations based on the established profile. Usually the objective is to learn the user profile as much as possible through the initial stage, rather than to satisfy the user's information need at the same time [19]. In this thesis, we argue that, a more integrated view should not ignore the user's information need even at the earliest stage; we should take into account the user's information need throughout the whole process. Also, recommendations would preferably be provided right from the outset and user interests acquired by employing a less intrusive method that gradually learns the user profile. Therefore, we focus on a more

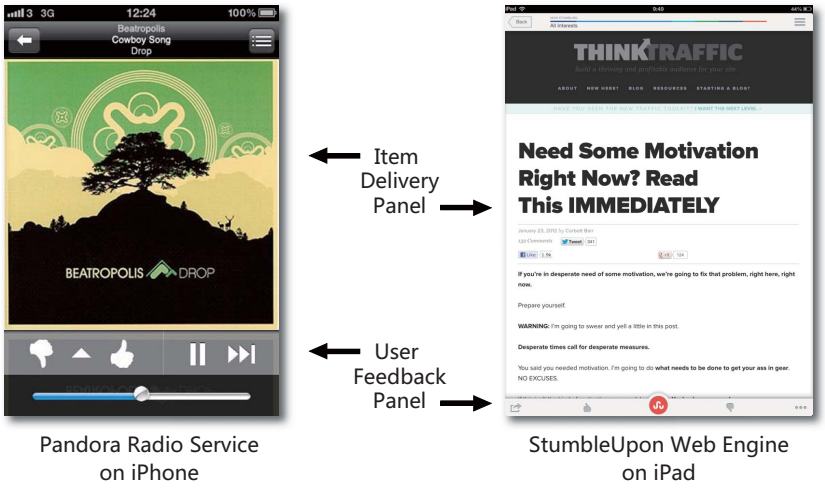


Figure 1.2: Examples of the interactive recommendation interface. Left: Pandora. Right: StumbleUpon. This interface promotes recommendation-oriented interactions. The service itself is to provide recommendations to the user, and the user can only interact with the items (music in Pandora and webpages in StumbleUpon) provided by the service.

integrated view of the cold-start problem which does not distinguish between the information-gathering and the recommendation-making phases, but continuously learn/detect the user profile while trying to satisfy the user at the same time.

Recently a type of interaction-based Web service interface has emerged. This interface enables pure recommendation-oriented interactions, which means that, all the services are comprised of direct user interactions with the recommender system over time. In such systems, there is no need (and in many systems, no way) for users to actively search for content. Instead, information items, such as webpages (StumbleUpon.com), songs (Pandora.com), ask and answer (Jelly.co) or dating matches (goTinder.com), are sequentially recommended to individual users, while feedback on recommended items is continuously observed (see Figure 1.2). Then the feedback collected during interactions can be utilised to improve the recommendation quality for the users.

We refer to this interface as the Interactive Recommender Interface. We argue that the interactive recommendation interface itself has suggested a more natural way to solve cold-start problems, i.e.,, to use an interactive recommendation process and learn the user's profile through her feedback on the recommended items. Meanwhile, it also makes the recommendation problem more challenging, because the recommendations are expected to be made right from the beginning rather than after a warming-up phase. Therefore, the recommended item(s) provided during each interaction stage need(s) to be informative for both the new user and the system. This way, the system can update gradually to establish the user's profile while satisfying the user's information need throughout the entire interactive process.

1.1 Research Problems

This thesis focuses on the cold-start problem and proposes to solve it through an interactive recommendation process (see Figure 1.3) with the integrated objective of maximising the user satisfaction over a

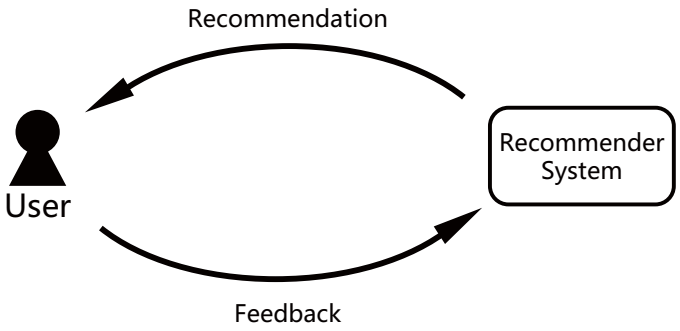


Figure 1.3: The interactive recommendation process. The system recommends item(s) to the user, then the user gives feedback on the recommended item to the system, based on which the system refines its model of the user and improves its recommendation quality.

period of time. To achieve this objective, we identify the following associated research problems:

1. **Sequential Recommendation**. The sequential recommendation process is the simplest form of the interactive recommendation process, where the system sequentially provides one item at each interactive round. The research problem of this process is how to decide the item to show to the user at each interaction in order to achieve maximal total ratings collected over T timesteps. For this process, a successful solution should balance between the two interconnected aspects, recommending and learning, in delivering each item during the process. It requires us model not only the recommended item's potential utility for the user, but also the potential information to be gained from the user's feedback on it. For the former, we need to establish a preference prediction model, and for the latter, we need to model the uncertainty in the prediction model.

2. **Sequential Batch Recommendation**. The above interactive recommendation process requires the users to actively provide feedback on the recommended items, and the system needs to update whenever a new rating is received, which can be computationally expensive for practical applications. Meanwhile, this process is also impractical for solving the item cold-start problem. Unlike users, cold-start items cannot actively obtain feedback from users; on the contrary, we need to wait for the users to give feedback on them. As users differ in their response times, waiting for one user's response before targeting to another is impractical. These concerns suggest us to integrate a batch solution into the sequential recommendation, i.e., to adopt a sequential batch process. The research problem is how to select the batch of recommendations (a batch of items for a cold-start user, or a batch of users for a cold-start item) in each interactive round.

3. **Item Combination and Diversification**. With a sequential batch recommendation process where multiple items are recommended at each interaction, the top-N recommendation diversification problem naturally emerges. The probability ranking principle (PRP) asserts that the optimal ranking of a list of recommended items should be in order of decreasing probability of relevance to the user [20], which is, however, based on the assumption that the recommended items are independent of each other [20]. Diversification, on the other hand, addresses the correlations between items

and suggests that a more diverse item combination may promote novel content exploration and discovery [21] and finally lead to better user experience. Achieving the optimally ranked list thus requires us to balance between accuracy-based criteria such as relevance and the diversification need of the user.

4. **Diversification over Multiple Interactions**. When we consider the recommendation process as a temporal process, the problem emerges of whether the diversification of items should be considered alone or jointly with respect to the feedback received previously. It is therefore interesting to investigate the effect of diversifying the combined item list summarising the user's past interests and the potential interests in the future.

1.2 Approaches

This thesis aims to address the cold-start CF problem in an interactive setting. We propose to approach the problem from the above-mentioned angles and provide theoretical understanding on each of them. We start with the sequential recommendation process by proposing an interactive CF framework and relate it to the multi-armed bandit problem [22, 23]. Then, in order to address multiple items at each interaction, we extend the framework by formulating it into a partially observable Markov decision process (POMDP) [24] to search for the exact solution and its implications. After that, we focus on the item diversification problem and relate it to the portfolio retrieval [25]. Finally, we consider joint item diversification with a case study in venture finance.

1.2.1 Interactive CF

We first propose an interactive CF (ICF) framework that aims to study CF in an interactive setting. According to the framework, the recommender system sequentially recommends items to the target user and iteratively updates the user model with the received feedback. The goal is to maximise the overall recommendation accuracy over a period of time. This mechanism naturally addresses the cold-start problem as any user can immediately receive sequential recommendations without providing ratings beforehand.

The integrated goal of maximising the overall recommendation performance over a period of time covers both the learning and recommending aspects, and is closely related to the Exploitation-Exploration (EE) problems [22, 23, 26]. EE problems describe the dilemma of whether, for each interaction, we should try to satisfy the user's interest with the best-guessed item according to current knowledge or whether we should try some sub-optimal yet discriminative items to gain more knowledge about the user. EE problems have been intensively studied in the machine learning and statistics communities, with multi-armed bandits as the generic setting. There are mainly two types of approaches. One type assumes no correlations between individual arms, such as probability-based methods including ϵ-greedy [22], epoch-greedy [27], Exp3 and Exp4 [28], and index-based methods including Gittins Index [23] and upper confidence bounds [29, 22]. The other type, the contextual bandit, assumes that the reward of "pulling an arm" is based on both the arm and the context, which share a common feature space. The concept of the contextual bandit has been applied to personalised news article recommendation [26],

where the context is defined as content features (texts) of the news articles and the browsing contexts of the users.

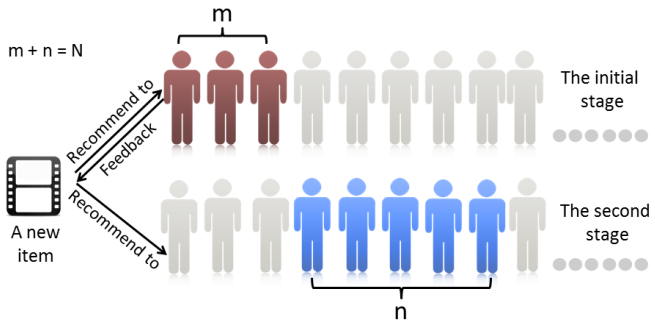
However, it is unclear as to how to model the interaction in pure CF settings where there is no content data to represent users and items and the only observations are ratings. In order to naturally integrate with existing CF approaches, we address the ICF problem under the popular matrix factorization framework, which has been proven to be effective in various recommendation competitions [30]. Specifically, we utilise the probabilistic matrix factorization (PMF) [31], investigating into the probabilistic model of the user-item ratings. The reason of using PMF is due to its capability of modelling both the expectation and the uncertainty of the user/item feature vectors, which further indicate the extents of exploitation and exploration respectively. According to PMF, the uncertainty of a predicted rating comes from both the related user feature vector and the related item feature vector. To consider both uncertainties, we adopt empirical algorithm Thompson sampling [32] which samples user/item feature vectors from their distributions. With sampled feature vectors, both the expectation and uncertainty of each recommendation choice are addressed during the decision process. Then, we assume the item feature vectors are well-learnt and thus the item-side uncertainties could be disregarded. The problem then falls into a linear form in the item feature vectors, and thus can be solved by various linear bandit algorithms [29], including a variation of ϵ-greedy algorithm, linear confidence bound and generalized linear upper confidence bound.

1.2.2 Two-Stage CF

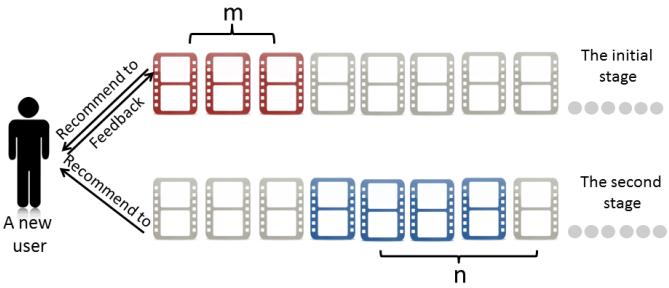
The above interactive recommendation process can be impractical for solving an item cold-start problem because ratings on the new item cannot be actively obtained. Rather, we need to wait for the users to respond, and the response times may differ. Also, updating the system whenever new feedback is registered can also be expensive. These concerns motivate us to combine a sequential interactive recommendation process with a batch approach – to recommend a batch of items (for a cold-start user) or a batch of users (for a cold-start item) at each interaction stage.

On the other hand, we can also view the cold-start problem as a resource allocation problem. In a short period of time, the number of recommendations (either for a new item or to a new user) is usually much smaller than the size of the available pool. As such, only a small portion can be selected due to the limited resources. For example, advertisements of a new fresh item can only be sent to a limited number of users, and a new user can only rate a limited number of items when joining a new web service. It is important to utilise the limited recommendation resources wisely.

We thus propose a simple yet practical two-stage interaction process for solving the cold-start problem (See Figure 1.4). During the initial stage, we use a portion of recommendation allocations to estimate the new item's (user's) model (with also considerations on its utility). After that, in the second stage, we use the remaining recommendation allocations. The goal, again, is to maximise the total feedback from the two stages (the overall recommendation quality). Note that though this process is also two-stage, it is fundamentally different from the traditional two-stage approaches for cold-start problems where the recommendation objective is placed onto only the second stage [33, 34, 35, 17, 36]. We impose the



(a) A cold-start item



(b) A cold-start user

Figure 1.4: Schematic figures of the two-stage recommendation process for (a) a cold-start item; and (b) a cold-start user. The total N resources are allocated in two stages. At the initial stage, m users (items) are selected, with their feedback used to update the profile of the new item (user). Then another n users (items) are selected in the second stage to exploit the updated profile. The target is to maximise the overall feedback over two stages.

objective onto the whole process, and thus the initial stage involves both learning and recommending.

We formulate the problem with POMDP and provide its exact solution. A POMDP models a Markov decision process where the true state of the system is partially unobservable [24]. In the scenario of item cold-start recommendation, we define the true state of the system as each user's genuine (potential) preference regarding the new item, which is unknown for unobserved users. As such, we can use POMDP to describe the decision process by defining the following (known as the POMDP tuple) [24]: (i) the states (the continuous space of all possible preferences from the users), (ii) the actions (recommendations), (iii) the observations (the ratings received from targeted users), (iv) the reward function (the expected total rating), (v) the state-transition function (how the system updates its models of unobserved users), and (vi) the observation function (the probabilistic model that generates the observed ratings).

We base the POMDP on both a correlated-user (CU) model and a PMF model; the former can be seen as a probabilistic representation of the memory-based CF, and the latter is a typical latent factor model for CF. In both cases, we argue that the user-user correlation plays an central role in the decision process as it determines how the feedback from selected users can update the expected feedback from others; it is directly modelled through the multivariate Gaussian distribution in the CU model, and can be easily inferred from the latent feature vectors in PMF. The update of system's belief from one set of users to the other can be well-captured by the POMDP formulation which enables us to find an exact solution to the two-stage recommendation process.

However, an exact value iteration solution of the POMDP is intractable and is PSPACE-complete

[37], as it requires us to iterate through all possible selections and observations. However, as we closely analyse each term of the exact solution from the perspective of correlation between users, we reveal an important insight: the users chosen in the initial stage should be those not only highly relevant according to the initial-stage information, but also able to potentially guide us to find users with high expected values in the next stage. This ability of guidance can be further abstracted as a strong correlation between the initial-stage users and potential second-stage users, no matter positive or negative. With this insight, we propose the approximation method *guided* EE to ease the computational complexity brought by the exact solution.

The proposed process and its solution are also applicable to other scenarios. For example, in IR, when a query is registered, the system shows two subsequent pages to the user such that the second-stage results can be refined [38, 39]. And, in online display advertising, for a new campaign, in order to understand which users should be targeted, the advertiser can spend some budget to show the ads to different users and collect their feedback (i.e., ad click or conversion). Then after the warming-up stage, we can leverage the users' feedback and refine the target user groups for higher advertising performance [40].

1.2.3 Diversification

When multiple items are concerned, the correlation between them should also be taken into account. This leads us to consider the item diversification problem. The rationale behind promoting result diversification has been explored by the researchers of both IR and recommender systems. For example, in text retrieval, some regard diversifying the search results as a way of reducing redundancy and improving information novelty in the results [41], as in the work on sub-topic retrieval [42]. Others consider it as the means of managing uncertainty and risk in the ranked list [43, 25]. In parallel to text retrieval and Web search, diversification of the recommendation results has further recently been identified as a critical factor that significantly influences end-user satisfaction [44, 45, 21, 46].

In the past, diversification of recommended items was usually achieved in an explicit manner. For example, a similarity measure is usually introduced first, then the diversification is increased by reducing the in-list similarity [46, 47]. The balance between diversity and other criteria such as relevance, however, has not been systematically discussed [41, 48, 25]. In other words, the previously proposed methods have focused on the question "how to diversify", but failed to answer "when to diversify". More specifically, the diversification need of different users may be different and thus the level of diversification should be adaptive. A recent study in web search has found that different queries could benefit from different diversification strategies [49, 50]. In recommendation, it is even more useful to make the diversification adaptive to individual users' tastes.

The usefulness of adaptive diversification has two aspects. First, user tastes have different scope and coverage of the underlying topics/factors, indicated by the rated items. Some users' tastes are more specific to a few topical areas, while others are more diversified across various topics. To see this, consider the following examples where users are required to provide two rated movies to describe their movie tastes and use them as the ground for recommendation. Suppose a user was in favor of the two

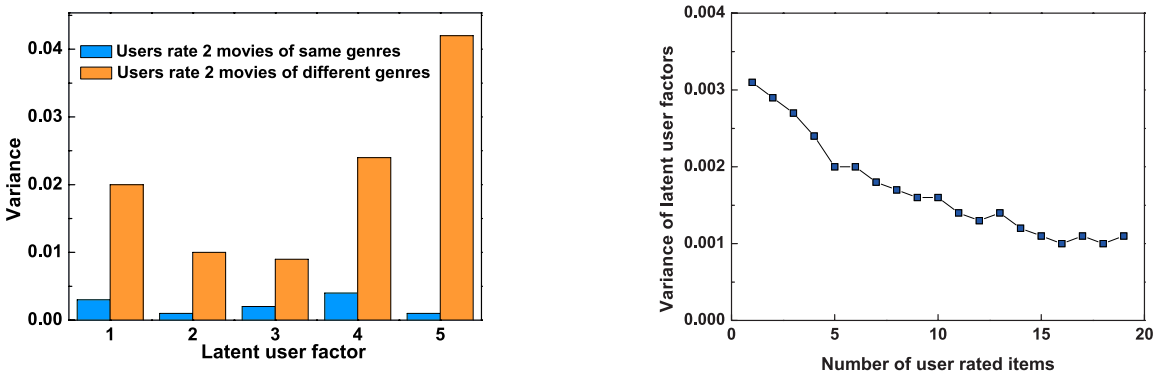


Figure 1.5: Comparisons of the variances in latent factors for different users. The latent factors are obtained by PureSVD with latent dimensionality 5 [1] on the MovieLens 1M dataset [2]. Left: The variance of each latent user factor for users who rated 2 movies with the same or different genres. Users who rate 2 movies of the different genres tend to have higher variances in their latent factors. Right: The relationship between the average variance of latent user factors against the number of movies that the user rated. Users who have fewer numbers of ratings tend to have higher average variances in their latent factors.

movies "Underworld" and "Eclipse". For this user, we may provide a recommendation list containing less diversified items as it is likely that the user's taste is more concentrated on a few specific topical areas (likely to prefer Fantasy/Thriller kind movies). By contrast, if the user liked "The Social Network" and "Taken", then a more diversified recommendation would fit the user's taste better, implied by the fact that the preferred two movies are in quite different genres. These examples suggest that the diversification level should rely on the underlying topic distribution and the individual user's taste. This is similar to the notions of *exhaustivity* and *specificity* discussed in [51].

Second, a target user's "true" taste is hidden and can be inferred only from the rated items (the user profile). Thus, our understanding of the target user's taste varies and depends on the ambiguity of the provided user profile. For a cold-start user with no or only a small number of items, the information is not enough to infer the user's exact taste, so a more diversified recommendation list would be a safer bet. Also, a diversified recommendation list can help to clear the ambiguity of the user profile, which will further assist the system's knowledge about the user.

The above two considerations can be further illustrated in Figure 1.5 which shows our intuition by employing a latent factor model on a movie rating data set. The uncertainty of learned user tastes is measured by their variances (the exact definition can be found later in Chapter 5). First, we can see that variances of latent user factors are obviously higher for the users who rated two movies with different genres than for those who rated two movies within the same genre. Second, we can also see that the average variance of latent user factors decreases as the user rates more items, indicating a reduction of uncertainty in the user profile with more collected information. It should however be noted that more ratings in a profile may not necessarily give us more information about the user preference. It also depends on what items the user has rated – some ratings are more informative than others [35].

Taking the above aspects into account, we propose the latent factor portfolio (LFP) framework to address adaptive diversification which connects the latent factor models [52, 9, 53, 31, 54] with portfolio

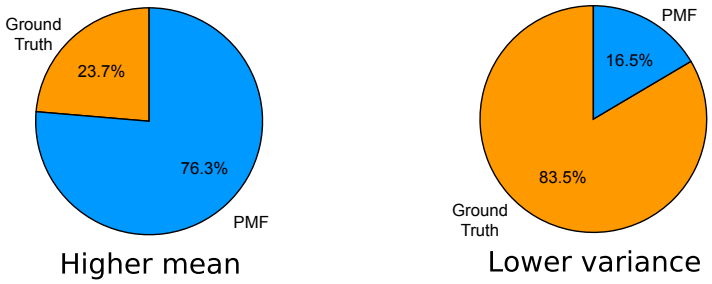


Figure 1.6: Mean and variance comparison between the PMF recommendations and groundtruths of VCs' investments. Left: percentage of PMF/groundtruths to have higher mean for individual users. Right: percentage of PMF/groundtruths to have lower variance for individual users.

retrieval [25]. We use latent factor models because they are a more general form of MF models and have been widely used for CF due to their accuracy and scalability. The goal of LFP is to optimise the trade-off of the expected return and uncertainty, modelled as the variance. In the proposed framework, the coverage of a user's preferences is modelled by the distribution of latent factors and the uncertainty is represented by using the variances of latent factors. Our derivation then shows that the distribution and the uncertainty of latent factors in a user profile determine the final uncertainty of the ranked list. The diversification level should finally reflect the distribution and the variance (uncertainty) of latent factors in a user profile that has been given. The proposed solution incorporates the concept of portfolio optimisation from finance, which integrates the goals of maximising expected return and reducing uncertainty into a unified optimisation problem. It provides a systematic way of achieving diversification in a personalised way. Combining the latent factor models and portfolio retrieval enables us to gain a clear understanding of the adaptive diversification, which otherwise would not be derived from either approach independently.

1.2.4 Risk-Hedging

To relate item diversification back to the temporal recommendation process, we investigate the effect of diversification over time. Inspired by the investment practices in finance, we argue that it could be beneficial if the recommended items are chosen jointly with the users' past preference records, instead of being optimised alone. Our motivation comes from the concept of "hedging" commonly practised by investors in their investment activities. Investors may choose an investment that can offset the potential risk of any adverse price movements in the investments that they are already holding. As a case study, we consider venture capital investment recommendation using the dataset obtained from CrunchBase, a repository of startup companies, individuals and investors focusing on US high-tech sectors[†]. In this case, we aim to enhance the recommendation quality by recommending startups that, to some extent, incorporate compensating features to those in the user's investment history.

Figure 1.6 further illustrates our motivation. This figure shows the statistics of the mean/variance comparison between the following two portfolios for each user: (i) the joint portfolio that contains

[†]https://www.crunchbase.com/

previously-selected items by each user and the top-N recommendations suggested by PMF (without diversification) to her; and (ii) the joint portfolio that contains previously-selected items by each user and the groundtruths in the test set (i.e., all the groundtruths for each user). The mean and variance are calculated based on the PMF model, and the same number of PMF recommendations are made to match the number of groundtruths in order to make the comparison. We can see that for 76.3% users the recommendations made by PMF have higher mean values, whereas in 83.5% cases the users' groundtruths have lower variances. This figure suggests that, first, promoting items with the highest expected returns may fail to produce the most desirable result; and, second, it may be possible to estimate the investor's actual risk appetite according to the probabilistic model PMF, which connects the *abstract* risk obtained by CF with the *actual* risk from a financial perspective.

We propose a joint portfolio optimisation process based on the probabilistic model PMF, which optimises the overall diversification of the items comprising both the user's previous rating records and the proposed recommendations. Similar to LFP, this optimisation process involves both the goals of maximising the expected return and reducing risk. In other words, the utility of a recommendation list is a combination of the expected return and the risk factor. In order to determine the optimal recommendations, we need to first find the maximal utility to be brought by a candidate recommendation list. This requires us to conduct weight optimisation for the joint portfolio. Next, by iterating through all candidate recommendation lists, we can further determine the optimal list. As iterating through all possible combinations of items is infeasible in practice, we further propose several approximate solutions. These include an index-based ranking solution, sampling, sequential selection, weight-based ranking, and filtering.

The case study is conducted on a financial dataset, but similar arguments can be made in a traditional recommendation scenario. When we optimise the joint portfolio over time, we may catch the temporal dynamic change in the user's taste [45]. Thus, we could consistently avoid the over-specification of recommended items over time.

The relationship between the four aspects is shown in Figure 1.7. These aspects can be consolidated into two factors: the number of stages involved in the process (the x-axis in the figure) and the number of items to recommend at each stage (the y-axis in the figure). First, we propose ICF as a sequential recommendation process in which one item is recommended during each stage. Then, in the two-stage CF, we extend this process to include two stages, each involving a batch of recommendations. After that, we focus on the intra-stage recommendation by considering diversification. Finally, we consider risk-hedging diversification which relates the diversification back to the interactive process.

1.3 Contributions

The contribution of this thesis is to build a number of algorithms following the sub-objectives introduced above. These algorithms are presented throughout the following publications which form the main part of this thesis:

- Yue Shi, Xiaoxue Zhao, Jun Wang, Martha Larson, and Alan Hanjalic. "Adaptive diversification

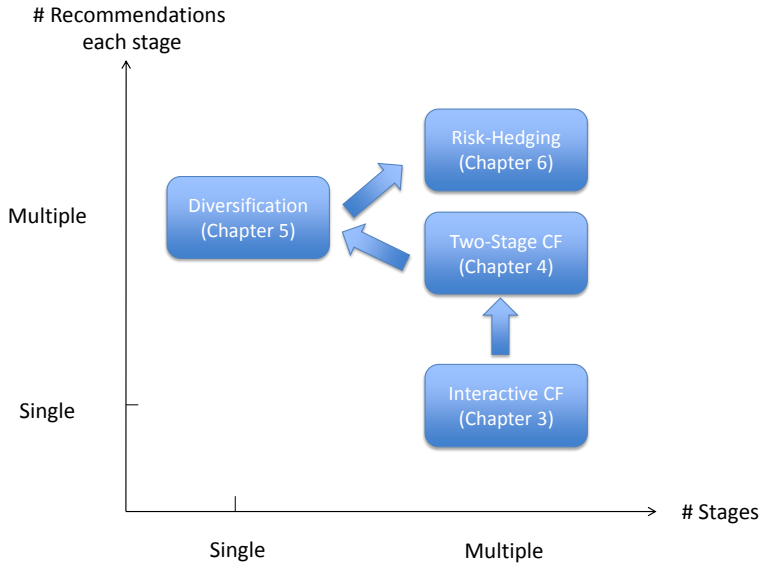


Figure 1.7: The histogram of the four interconnected aspects discussed in this thesis.

of recommendation results via latent factor portfolio." *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2012.

- Xiaoxue Zhao, Weinan Zhang, and Jun Wang. "Interactive collaborative filtering." *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM)*, 2013. (Nominated as the best student paper candidate.)

- Xiaoxue Zhao, Weinan Zhang, and Jun Wang. "Risk-hedged venture capital investment recommendation." *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys)*, 2015.

- Xiaoxue Zhao, and Jun Wang. "A theoretical analysis of two-stage recommendation for cold-start collaborative filtering." *Proceedings of the 2015 Conference on the Theory of Information Retrieval (ICTIR)*, 2015.

I was also involved in other publications during my PhD which are relevant to CF, but not directly related to this thesis:

- Weinan Zhang, Jun Wang, Bowei Chen, and Xiaoxue Zhao. "To personalize or not: a risk management perspective". *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys)*, 2013.

- Shuai Yuan, Jun Wang, and Xiaoxue Zhao. "Real-time bidding for online advertising: measurement and analysis". *Proceedings of the 7th International Workshop on Data Mining for Online Advertising (ADKDD)*, 2013.

- Thomas Stone, Weinan Zhang, and Xiaoxue Zhao. "An empirical study of top-N recommendation for venture finance." *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM)*, 2013.

1.4 Structure

The rest of the thesis is organised as follows:

In Chapter 2, we give the background of the research domain. We specifically focus on the literature on CF, cold-start problems, the PRP principle in IR, relevance feedback, EE problems, POMDP, diversification and relevant economic concepts that are addressed in this thesis.

In Chapter 3, we focus on the first research problem, sequential interactive recommendation, and propose the **Interactive Collaborative Filtering** framework. We first formulate the objective function that is to achieve the maximal overall performance over a period of time. Next, we provide the PMF model to obtain the distributions of the user and item feature vectors. Then we employ Thompson sampling to address the uncertainties of both the user and the item models. After that, we assume that the uncertainty in ratings comes only from the cold-start user model, and connect our model to a series of UCB algorithms. In the experiment part, we compare our proposed EE algorithms to several baseline methods including SVD, ϵ-greedy, active learning and interview methods. We conduct our experiments on cold-start users as well as warm-start users with drifting tastes.

In Chapter 4, we address the second research problem, sequential batch recommendation, and propose the **Two-Stage Collaborative Filtering** process. We first formulate the two-stage recommendation with a POMDP framework. We then derive the exact solutions by value iteration for both the CU model and the PMF model, along with discussing on the link between them. After that, we present our theoretical conclusion on how to choose the users in the initial stage: they should be not only highly relevant according to the initial-stage information, but also highly correlated to potential second-stage users. With this finding, we propose the approximation method *guided* EE (GEE) for both the CU and the PMF models. In the experiment part, we compare the proposed algorithm with several baselines including greedy, active learning and UCB algorithms, on both a synthetic dataset and a real dataset.

In Chapter 5, we investigate the third research problem, recommendation result diversification, and propose the **Item Portfolio Diversification** which introduces portfolio theory into the diversification of recommendation lists. We start with deriving the variance (covariance) of (between) the rating estimations, as a function of the user and item latent factors. Then, we formulate the optimal ranking function by introducing the trade-off factor to balance the return and uncertainty. A sequential selection algorithm is then proposed which determines the selection and ranking of the items in the recommendation list. In the experiment part that follows, we focus on discovering the relations between the accuracy and diversity and on how the diversity level is adaptively achieved according to each user's personal need.

In Chapter 6, we discuss the fourth research problem concerning diversification over time. We propose a **Risk-Hedging Diversification** framework which jointly optimises a portfolio comprised of the proposed recommendations and the previous user rating records. We first give a brief introduction of the background of the investment screening process and the CrunchBase dataset that we use in the experiments. Then, we formulate the problem by decomposing it into two steps, and propose solutions for each of them. Then we propose several approximate solutions. In the experiment part, we conduct a thorough empirical analysis of the effect of parameters and a series of performance comparisons between

the proposed method and several baselines.

In Chapter 7, we sum up the conclusions and present several directions for future research.

Chapter 2

Related Work

Because this thesis seeks solutions within collaborative filtering (CF), it falls into the scope of the CF research. Due to the unique formulation of the cold-start problem in this thesis, we also touch on the topics of multi-armed bandit, decision theory, exploitation-exploration and diversification. Consequently, we first focus on CF and cold-start problems; then we provide a brief overview of several issues related to the methods proposed in this thesis.

2.1 Collaborative Filtering

CF aims to utilise preferences previously expressed by users in regard to items to infer possible future preferences [55]. The preferences can be explicitly-expressed ratings such as scores or "like"s and "dislike"s, or implicit user behavioural information such as clicking, viewing and purchasing, etc. CF exclusively relies on the preference matrix (see Figure 1.1) instead of any content-based information (e.g., the topic, tags, release date, title of an item, or the location, gender, age, nationality of a user).

The idea behind CF is that users who had similar preferences in the past are likely to have similar preferences in the future; and that the more similar the users were in the past, the more likely they would agree with each other in the future. It heuristically implements the real-world "word of mouth" phenomenon. Similarly, the items who shared similar users in the past would also attract the similar group of users in the future. It is argued that, compared to content-based methods, CF can catch more subtle relations and has a higher potential for serendipity [55].

CF can be achieved by mainly three approaches [11]: memory-based approaches such as neighbourhood-based CF (user-based and item-based) [2, 56], model-based CF [30, 57, 57, 58], and hybrid methods [59] that combine the memory-based and model-based approaches together.

2.1.1 Memory-Based CF

Memory-based CF examplifies the "word-of-mouth" heuristic directly. As the term itself suggests, it keeps a complete record of the user-item preferences. From the rating data the similarity between users or items is calculated for making recommendations. These methods are widely applied to many recommender systems such as movies [55, 60], news [61], online shopping [62], etc.

Typical examples of memory-based CF are neighbourhood-based CF, which include mainly two types: user-based CF and item-based CF. For user-based CF, the prediction of a potential user-item

preference score is calculated as the weighted summation of the preferences expressed by the user's neighbours, the users similar to the target user, weighted by the similarity [2, 63, 55, 64]. Let us denote the user as u, item as i, and the rating to predict as $r_{u,i}$, then the basic form of a user-based CF can be written as

$$\hat{r}_{u,i} = \frac{\sum_{v \in Nei(u)} sim(u,v) \times r_{v,i}}{\sum_{v \in Nei(u)} sim(u,v)}, \tag{2.1}$$

where $\hat{r}_{u,i}$ is the predicted rating, $Nei(u)$ represents u's neighbours who have rated i, $sim(u,v)$ is the similarity between user u and v, and $r_{v,i}$ is the known preference expressed by u's neighbour v. The denominator is used to limit the prediction to the desired range. The similarity measure can be cosine similarity or Pearson correlation. For the latter, the estimation function is usually altered as

$$\hat{r}_{u,i} = \overline{r}_u + \frac{\sum_{v \in Nei(u)} sim(u,v) \times (r_{v,i} - \overline{r}_v)}{\sum_{v \in Nei(u)} sim(u,v)}, \tag{2.2}$$

where the ratings by neighbours are centred by their averages. In some practices, the rating scores are also normalised by their standard deviations, which is referred to as the Z-score normalisation [65].

Item-based CF is largely symmetric to user-based CF, aside from the fact that sometimes the weighted average is taken for all available neighbours (i.e., all the items that have been rated by the same user) instead of for the most similar ones [56, 56, 64].

Elements in memory-based CF include the similarity measure, the neighbourhood selection and the normalisation of ratings. Discussions have concerned the choices of similarity measures, including the use of cosine similarity and Pearson correlation as mentioned above [64, 55], and adjusted cosine similarity for item-based CF [66]. These measures are mainly correlation-based, and the difference mainly lies in whether the rating bias is considered in the user/item ratings. Other similarity measures include mean-square difference [67], Spearman rank correlation [68], frequency-based Pearson correlation [63] spectral clustering techniques [69] and entropy [70]. An empirical comparison of different similarity measures can be found in [65]. Concerning neighbourhood selection, there are mainly two popular neighbour selection methods for the user-based CF: the k-nearest neighbours strategy [71] and threshold-based neighbour selection [72]. In regard to rating normalisation, the deviation from the mean rating and the Z-score normalisation are usually adopted for the mean and the spread of ratings [65, 73]. Empirical comparisons can be found in [2, 74]. Though mainly a heuristic method, in [75] the authors provided a probabilistic framework to explain memory-based CF, and we will further discuss the theoretical basis of memory-based CF in Chapter 4.

It is argued that the choice between a user-based and an item-based CF largely depends on the ratio of user-item numbers in the system. If the system has much more users than items, the item-item similarities can be more reliable than the user-user similarities, and thus an item-based recommender system is more suitable, and vice versa [62, 55]. Even though customarily either a user-based or an item-based CF system is used, researchers have also unified the two perspectives by integrating the predictions from similar users and similar items together [76].

Memory-based CF has many advantages. Foremost, it is intuitive and thus explainable [64]. Second, fewer parameters are needed to tune it [77]. However, memory-based CF usually suffers from aspects of scalability [11, 62] and sparseness of the rating data [78, 79]. These can be eased by adopting a model-based method.

2.1.2 Model-Based CF

While memory-based approaches make rating predictions based on the entire collection of previous ratings, model-based methods first abstract the rating information into a model which is then used to predict future preferences. A number of approaches can be adopted to build the model, such as machine learning [80], clustering [81, 82, 83], data mining [84, 85], Bayesian network [86, 87], and dimension reduction methods [30, 88]. For example, in a Bayesian Network approach, each node in the network is represented by an item and the state is the preference value for each item. The algorithm then searches over different network structures and dependencies which are used for further predictions [63]. In clustering-based models [81, 82, 83], users are first grouped into clusters, and the user's preference to different items is conditionally independent given the class of the user. In [89], the authors proposed a restricted Boltzmann machine to model the user's ratings of movies, with binary hidden units and softmax visible units.

A number of methods can be further categorised as latent factor models, in which the users and items are represented by a small number of "latent factors" and the preference prediction is then calculated based on the latent factors. For example, latent semantic models (aspect models) proposed by Hofmann [90] introduce a latent class variable associated to each observation, and the preference between the user-item pair is conditionally independent given the aspect. Later, Hofmann proposed a probabilistic latent semantic model for CF [91] which models the observed user ratings as a mixture of user communities where users participate probabilistically in one or more groups. Based on aspect models, in [92], the authors proposed a three-way aspect model to address the effect of content information. In [93], a generative latent variable model is proposed which models each user as a mixture of user attitudes distributed by Dirichlet allocation.

2.1.3 Matrix Factorization

This thesis mainly adopts matrix factorization (MF) [30] which is probably the most widely adopted latent factor model. MF first decomposes the user-item preference matrix, projects the users and items onto a lower dimension space, and then calculates the user-item preferences as the inner products between the user and item factors in the latent space [30]. Supposing we have the user and item vectors as $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$ respectively, we can estimate the rating score as

$$\hat{r}_{u,i} = \boldsymbol{p}_u^T \boldsymbol{q}_i. \tag{2.3}$$

The MF method for CF is sometimes alternatively referred to as SVD (singular vector decomposition) because SVD serves as one of the basic processes to obtain the user/item factors [30], as shown in Eq. (2.4). Supposing that the user-item rating matrix $\mathbf{R}$ consists of M users and N items, the rating

matrix $\mathbf{R}$ can be decomposed into three low-rank matrices, $\mathbf{U}^{M\times K}$, $\mathbf{S}^{K\times K}$ and $\mathbf{V}^{K\times K}$:

$$\mathbf{R} \approx \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{2.4}$$

$$\mathbf{P} = (\mathbf{U}\mathbf{S}^{1/2})^T, \mathbf{Q} = \mathbf{S}^{1/2}\mathbf{V}^T. \tag{2.5}$$

Then, the latent factors of users can be denoted as $\mathbf{P}$, and the latent factors of items can be denoted as $\mathbf{Q}$, as shown in Eq. (2.5). Each column vector in $\mathbf{P}$ (e.g., $\boldsymbol{p}_u$ for user u) or $\mathbf{Q}$ (e.g., $\boldsymbol{q}_i$ for item i) represents the corresponding user or item. We refer to the direct matrix decomposition method as pureSVD [52, 94] to differentiate it from a stochastic gradient descent version described below (which is, however, also widely referred to as SVD within the CF research community). Although PureSVD is the most basic latent factor model, its recommendation performance for top-N tasks is competitive according to a recent empirical study [52].

Due to the sparseness of the data, some early work suggested to first fill in the sparse preference matrix with imputations and then apply matrix factorization methods [88], but recent work has proved that focusing only on the observed elements in the matrix directly can produce better results [95, 57, 96, 97]. Stochastic gradient descent and alternating least squares (ALS) [57] are two popular approaches to obtain the desired user and item vectors. Both approaches incorporate a regularised term to avoid overfitting

$$\min_{\boldsymbol{p}_{(\cdot)},\boldsymbol{q}_{(\cdot)}} \sum_{\text{observed } r_{u,i}} (r_{u,i} - \boldsymbol{p}_u^T\boldsymbol{q}_i)^2 + \lambda(||\boldsymbol{p}_u||^2 + ||\boldsymbol{q}_i||^2), \tag{2.6}$$

where $||\boldsymbol{p}_u||^2$ and $||\boldsymbol{q}_i||^2$ denote the Euclidean length of the vectors $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$ respectively and λ is a regularisation parameter. In stochastic gradient descent [98], the user and item latent features are first initialised with random vectors. Then, in each training round, the algorithm loops through all available ratings in the training set, and, for each rating, calculates the prediction error

$$e_{u,i} = r_{u,i} - \boldsymbol{p}_u^T\boldsymbol{q}_i, \tag{2.7}$$

and updates the concerned feature vectors $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$ with the following modifications

$$\boldsymbol{p}_u \leftarrow \boldsymbol{p}_u + \gamma(e_{u,i}\boldsymbol{q}_i - \lambda\boldsymbol{p}_u), \tag{2.8}$$

$$\boldsymbol{q}_i \leftarrow \boldsymbol{q}_i + \gamma(e_{u,i}\boldsymbol{p}_u - \lambda\boldsymbol{q}_i), \tag{2.9}$$

where γ controls the magnitude of the modification rate to the opposite direction of the gradient.

The above update rule is obtained because the partial derivatives of the objective function Eq. (2.6) can be written as

$$\boldsymbol{p}'_u = -2\left(e_{u,i}\boldsymbol{q}_i - \lambda\boldsymbol{p}_u\right), \tag{2.10}$$

$$\boldsymbol{q}'_i = -2\left(e_{u,i}\boldsymbol{p}_u - \lambda\boldsymbol{q}_i\right), \tag{2.11}$$

and thus the approximate solution of $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$ are modified by moving towards the opposite direction of the gradient. The above process is thus repeated several rounds on the training set and can be terminated by cross-validation on the test set.

The ALS method alternates between fixing the user feature vectors to obtain the least-squares solution of the item feature vectors, and fixing the item feature vectors to obtain the least-squares solution of the user feature vectors according to Eq. (2.6) until convergence. We will further discuss ALS in regard to probabilistic matrix factorization (PMF) in Chapter 3.

It is worth mentioning that the quadratic form of the objective function with regularisation as shown in Eq. (2.6) is widely used in MF and has many variations, adding flexibility and elaboration to the model. A popular variation is to add bias terms for the user (b_u) and the item (b_i), plus a global bias term (b_g), to catch the deviations of user u and item i from the average, and the average of all ratings respectively:

$$\hat{r}_{u,i} = b_g + b_u + b_i + \boldsymbol{p}_u^T \boldsymbol{q}_i. \tag{2.12}$$

The corresponding optimisation objective thus becomes

$$\min_{\boldsymbol{p}_{(\cdot)},\boldsymbol{q}_{(\cdot)},b_{(\cdot)}} \sum_{\text{observed } r_{u,i}} (r_{u,i} - b_u - b_i - b_g - \boldsymbol{p}_u^T \boldsymbol{q}_i)^2 + \lambda(||\boldsymbol{p}_u||^2 + ||\boldsymbol{q}_i||^2 + b_u^2 + b_i^2), \tag{2.13}$$

which is optimised through learning additional bias features, and hence adds to additional flexibility in the model [30].

Another variation is to modify the objective function to adapt to the task of implicit rating matrix decomposition. In [99], the authors proposed the following optimisation objective

$$\min_{\boldsymbol{p}_{(\cdot)},\boldsymbol{q}_{(\cdot)}} \sum_{\text{observed } r_{u,i}} c_{u,i}(r_{u,i} - \boldsymbol{p}_u^T \boldsymbol{q}_i)^2 + \lambda(||\boldsymbol{p}_u||^2 + ||\boldsymbol{q}_i||^2), \tag{2.14}$$

in which case $r_{u,i}$ is a binary preference showing whether ($r_{u,i} = 1$) or not ($r_{u,i} = 0$) the user has indicated any interest in the item (through behavioural indicators such as purchasing, listening and clicks, etc.). $c_{u,i}$ is the confidence level to modify the weight of this implicit feedback in the model optimisation.

In addition, probabilistic latent semantic analysis [100] and latent Dirichlet allocation [58] are also among famous approaches for conducting MF, but they are less related to this thesis.

2.2 Cold-Start Problems for CF

When no or very few ratings are available to infer the interest/property of a new user/item, it is difficult to initiate accurate recommendations. It is referred to as the cold-start problem [13], a major challenge for CF. It is the extreme form of data sparseness which is a main factor limiting the effectiveness of CF models. Former approaches usually handled cold-start problems by employing additional information, such as demographic information of a new user (for the user cold-start problems) [101, 102] and content information for a new item (for the item cold-start problems) [103]. However, fewer work has tackled the cold-start problems fully within the scope of CF which relies exclusively on the rating information.

In CF, the prediction of ratings depends entirely on the collected ratings about the related user and item, which at their introduction stage are totally unknown. Thus, solving cold-start problems within CF is even more challenging (see the complete blank parts to the right and bottom in Figure 1.1.). Usually efforts are made either on solving a new user problem or a new item problem, referred to as as the user cold-start [104, 13] or the item cold-start [8] problem respectively.

2.2.1 User Cold-Start Problems

There is comparatively more literature on user cold-start problems than on item cold-start problems within the scope of CF, due to the fact that a new user can actively participate in the process to assist the system to learn. A common method used is to adopt an additional "interview" stage prior to the recommendation stage in order to collect sufficient information to initiate the first recommendations [105] . The interview questions may ask the user to provide additional information (e.g. favorite genre) or to rate a set of informative items. For the latter, the items used in the interview can be selected on a static-basis, such as by popularity, entropy or coverage [105, 106]. The interview phase can also be more intelligent, such as decision-tree based methods [105, 17, 36]. In decision-tree based methods, the new user encounters the first question at the top node; and, depending on the answer to the question (such as "like", "dislike" and "unknown"), the system provides further questions. Some work has discussed the case of showing multiple interview problems at once, to avoid the extremely overloaded "unknown" branch for the user [36].

Active Learning

Active learning (AL) methods form an important branch for designing the interview questions [107, 35, 33, 108]. They are also referred to as optimal design by statisticians [109]. AL presents a limited number of items (usually much smaller than the total number of available items) to the target user for review, and then learns the user's preferences on the remaining items based on her feedback on them. Because the number of items to review is limited, the user model's accuracy largely depends on the training points selected [107, 109]. The objective of active learning is usually represented by a statistical measure on the prediction, such as achieving minimal mean squared error in the model estimation (A-optimality criterion) [34], minimal 2-norm of the inverse of the information matrix (E-optimality criterion) [33] or minimal determinant of resulting covariance matrix of the system (D-optimality criterion) [33]. Achieving the global statistical measure is usually equivalent to maximising the information gain in the learning stage [35, 110].

Usually two targets for the interview process are considered. First, the interview process should be adequate, so that after receiving the feedback, the user profile should be sufficiently learnt for providing sound recommendations. Second, the interview process should also be kept minimal, so that the user will not be bored during the process and quit in the middle. Therefore, it usually emphasises on maximising the information gain during the learning stage [105]. With this target, the learning efficiency, rather than the user's information need is emphasised. In this thesis, instead, we use a unified goal that takes into account the user's information need from the beginning, leading to algorithms that can automatically balance between the learning and the recommending goals. This objective will be elaborated throughout

Table 2.1: A summary of previous studies on cold-start problems within the scope of CF.

	User Cold-start	Item cold-start
Active Learning	D-optimality [33] E-optimality [33] Model entropy minimisation [35] Rating prediction divergence minimisation [33]	A-optimality [34]
Decision Trees	Functional matrix factorization [17] Adaptive bootstrapping [36]	
Others	Popularity-based [105] Coverage [105]	Popularity-based [105] Coverage [105]

this thesis.

2.2.2 Item Cold-Start Problems

Content information is usually employed for tackling the item cold-start problem so that items with similar content information to the user's previously liked items are recommended [8]. [103] proposed a content-based hybrid approach to integrate content information about item domains into CF. [111] modelled the user's votes on different items with Boltzmann machines that use content-based parameter tying. In [112], the author suggested that users who share item preferences also share similar taxonomic preferences and thus used the item taxonomy information to assist the inference of the users preferences. In [113], the authors proposed a feature-based regression method that leverages all available information on users and items to address both the item and user cold-start problems. This can also extend to the problem of predicting preferences between new items and new users.

Active learning has also been employed to find informative users for the recommender system to learn about a new item [34]. This work falls in the "A-optimality" criterion for optimal design as mentioned before. Similarly, we argue that in order to maximise the overall performance right from the introduction of a new item, the learning stage and the recommendation stage should not be separated. This argument will be elaborated in Chapter 4.

A brief summary of previous studies on cold-start problems is shown in Table 2.1.

2.3 Other Related Issues

2.3.1 Probability Ranking Principle in CF

Originating from information retrieval [114], the probabilistic ranking principle (PRP) has been also related to CF [20]. PRP implies documents to be ranked in descending order by their probabilities of relevance can produce optimal performance under the "independent document" assumption [115].

This thesis will show that PRP is not optimal as the correlations between users play an important role in making recommendation decisions, updating the system, and optimisation in terms of diversification. First, according to PRP, in the user (item) cold-start problem scenario, supposing the rating is proportional to the relevance probability, the list of items (users) to recommend should be ranked accord-

ing to the prior information on them, e.g., the average ratings they have received (rated). However, under the interactive collaborative filtering (ICF) framework proposed in this thesis, such ranking strategy is not optimal as it does not consider the learning capacity of the system inherit in recommending these items (users). We argue in this thesis that the correlations between items (users) play an important role for the system to update, and thus a more comprehensive representation of item-item (user-user) relations should be adopted. Second, concerning the diversification problem, the item-item correlations play a central role in optimising a list of items according to the portfolio theory from economics.

Therefore, this thesis especially considers the cases where the correlations should be considered and thus the PRP principle does not apply.

2.3.2 Relevance Feedback in IR

The process of refined item selection and user group targeting is related to the concept of relevance feedback in IR [116, 117, 118]. Relevance feedback is used to involve the user into the retrieval process. It takes the feedback from a given query and uses it as the information to improve retrieval performance in the future. The procedures starts when the user issues a query, after which the system first shows some retrieval results; then the user marks some retrieved items to be relevant, with which the system refines the retrieval results. This process usually repeats for one or two iterations.

Similar to relevance feedback, the interactive recommendation process (Figure 1.3) proposed in this thesis also designs a feedback loop to refine the recommendation result over interactions. However, the interactive recommendation process differs from that of relevance feedback in the following aspects.

First, the procedure of relevance feedback mentioned above is usually limited to only one or two iterations [39]. It is because in IR, a query is input by the user only when the user is searching the information explicitly. As such, the user usually only interacts with the system a few times until locating the desired result. Conversely, recommender systems continuously provide information filtering services without users expressing their information need explicitly, and they collect feedback over time to keep improving recommendation quality.

Second, for relevance feedback, the objective is to locate the relevant search result as soon as possible, whereas for interactive recommendation process the target is to satisfy the user's information need during the whole interactive process. Therefore, the evaluations are essentially different. For relevance feedback, it is straightforward to use a time-based evaluation to evaluate the effectiveness of the relevance feedback, such as how soon the system can provide a relevant document for the user. However, in the interactive recommendation process and the cold-start scenarios, the overall recommendation utility over all interactions should be used for evaluating the system because the task is to continuously provide useful information items over time.

The essential difference lies in the functions of an IR system and a recommender system. While an IR system's main goal is to find the result as soon as possible, while a recommender system is required to actively filter the information for the user without explicit queries.

2.3.3 Exploitation-Exploration Problems

The trade-off between exploitation and exploration is referred to as the exploitation-exploration (EE) problem. EE problems are related to this thesis because in order to maximise the overall utility over a series of recommendation-feedback iterations, the recommended items should, on one hand, match the user's information need with the current knowledge (exploitation), and, on the other hand, be useful for the system to improve its knowledge about the user (exploration). Similarly, in the item cold-start problem scenario, the users to target to in the first stage should be both relevant to the target (exploitation) as well as useful for learning the target's properties (exploration).

The EE problem has been intensively studied in the literature of multi-armed bandit (MAB) problems [119, 27, 29]. In MAB, the agent needs to decide which slot machine among a series to play at each time step, in order to maximise the sum of rewards earned through a sequence of pulls [120, 29]. Gittins has provided an optimal index-based solution, referred to as Gittins index [23]. However, calculating the index is intractable in practice and researchers have endeavoured to find approximate solutions instead.

ϵ-greedy [121] is one of the simplest algorithms. It chooses the arm according to the greedy strategy* with probability $1-\epsilon$ and chooses a random arm otherwise. ϵ-greedy is proven to be able to achieve linear regret bound ($\widetilde{O}(T)$) if ϵ is a constant [121]. In Softmax methods, each arm is picked with a probability that is proportional to its expectation, and thus arms with higher expectations are picked with higher probabilities [122]. A variation of Softmax method, referred to as the Boltzmann exploration, chooses the arm with the following probability [123]

$$p_i(t+1) = \frac{\mathbf{e}^{\hat{\mu}_i(t)/\tau}}{\sum_j \mathbf{e}^{\hat{\mu}_j(t)/\tau}}, \tag{2.15}$$

where $p_i(t+1)$ denotes the probability of choosing i at the next time step $t+1$, $\hat{\mu}_i(t)$ is the expected return of i calculated at time t, and τ is a temperature parameter which controls the randomness of the choice [121].

Upper confidence bounds (UCB) form an important and popular category of approximate solutions for EE. In UCB, usually an "arm index" is defined as an combination of an expectation term (the exploitation component) and an uncertainty term (the exploration component), and then the arm with the largest index is selected in each round. For example, UCB1 [28] suggests to first pull each arm once, and then choose the arm according to

$$i_{UCB1}(t+1) = \arg\max_i \hat{\mu}_i(t) + \sqrt{\frac{2\ln(t)}{n_i(t)}}, \tag{2.16}$$

where $n_i(t)$ denotes the number of times that i has been played until the current step. The term $\sqrt{\frac{2\ln(t)}{n_i(t)}}$ here acts as the estimation of uncertainty in the expectation estimation $\hat{\mu}_i(t)$, which indicates of how well

*The greedy strategy is to choose the arm at time $t+1$ with the highest value of expectation summarised until the current time step:

$$i_{greedy}(t+1) = \arg\max_i \hat{\mu}_i(t),$$

where $\hat{\mu}_i(t)$ denotes the expectation of the reward by pulling arm i, calculated with information until time step t.

the expectation is estimated. Auer et al. have proven that UCB1 can achieve a logarithmic regret bound ($O(\ln T)$). UCB1 assumes no preliminary knowledge about the reward distributions and the results hold even when the arms are dependent to each other. Auer et al. also proposed UCB-normal, which makes use of the Chernoff-Hoeffding bounds to compute the index [22], for the special case that the bandit rewards are normally distributed.

In addition, epoch-greedy method was proposed for the case that the total time T is unknown [27]. In epoch-greedy, exploitation and exploration take place alternatively in each epoch in order to minimise the regret and a regret of $\widetilde{O}(T^{2/3})$ is achieved.

Contextual Bandit

For the above-mentioned approaches, when the number of arms is getting large, exploration becomes more difficult. On the other hand, in real-world problems, there is usually a structure underlying the arms that can be made use of. In the contextual bandit model, the structure is modelled by a common feature space for the arms and the context information [124]. A more general linear setting is discussed in [125]. Assuming the expected reward as a linear function, [29] proposed the LinRel algorithm with a regret of $\widetilde{O}(\sqrt{T})$.

A special case is where rewards can be modelled as a Gaussian process, based on which GP-UCB [126] emerged. GP-UCB models the arms to follow a multivariate Gaussian distribution. Observations about some of them will update the distribution which remains to be multivariate Gaussian. The decision is made based on the linear combination of the expectation and the standard deviation of the reward of each arm. The regret bound is $O(\sqrt{T})$.

Contextual bandit algorithms are most related to this thesis because, in our proposed scenarios, each available item to recommend corresponds to one arm. Therefore the total number of arms can be very large in number, making it different to use any of the MAB algorithms without assuming structure among them. Moreover, by using CF techniques, the structure (dependency) of (between) rewards (ratings) of different recommended items can be modelled. Actually, contextual bandit has already been applied for news article recommendation [26] and online advertising [127]. In both cases, a context (feature) is revealed at each timestep, and an arm (either a piece of news or advertisement) is selected based on the context. The context can be, for example, the user's demography or location information and the item's textual description.

In this thesis, however, we consider a domain-free scenario for the cold-start problem. Therefore, we need to derive a sensible representation for the correlated arms (items) with only the collaborative rating information without the help of content-based information. More discussions can be found in Chapter 3.

2.3.4 (Partially Observable) Markov Decision Processes

A Markov Decision Process (MDP) models discrete time stochastic control, where the agent makes decisions based on the state of the system, which then partly determine the resulting state of the system whereas the resulting states is also partly random [24]. A MDP can be described as a tuple $< \mathcal{S}, \mathcal{A}, Tran, Reward >$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $Tran$ is the state-

transition function describing the probability of the ending state s' given the starting state s and action a ($T(s, a, s') = p(s'|a, s)$), and $Reward$ is the reward function that gives the expected immediate reward when the agent takes action a at state s (R(s,a)). The target of an MDP is to maximise the total expected reward gained over a period of time. Exact solutions for an MDP include value iteration [128], which works from the terminal states and propagates the value estimation backwards, and policy iteration [129], which continuously improves the policy until convergence. Approximate solutions include determination-based approaches [130], sampling-based methods [131], heuristic search [132] and dimensionality reduction approaches [133].

MDPs are useful for studying a wide range of optimisation problems solved via dynamic programming and reinforcement learning, such as Robot navigation [134, 135], language dialog strategy design [136, 137], dynamic power management [138], and so on. In the scope of recommender systems, there is fewer work related to MDPs, which, to the best of our knowledge, includes only three papers [139, 140, 141]. [139] seeks to predict the user's next action (accepting a recommendation or selecting a non-recommended item) based on the state defined as the sequence of the user selections in the past, and the system action is to decide which item to recommend next. [140] and [141], on the other hand, consider a case study of a query tightening process which assists the user in building a personalised travel plan through her conversation with the recommender. In [140] and [141], the possible user actions, such as add a product to the cart, modify the current query, etc., are defined as the state space, and the system action includes showing the query, executing the current query and adding a product to cart.

In many real-world cases, the state of the system is unobservable and has to be inferred from the observations upon actions, which leads to the partially observable Markov decision process (POMDP) [24]. A POMDP can be easily transformed into a MDP by defining the belief state, which transforms the POMDP's true states into the probabilistic distributions of them [24]. POMDPs have been applied to decision support systems for preference elicitation by sequential query selection [142, 143]. We argue that for a recommender system, as the true preferences from users are only partially known for the observed users (users who have rated the item), it naturally forms a POMDP problem, and we can formulate the recommendation problem using POMDP to seek its exact solution. We will discuss this further in Chapter 4.

2.3.5 Ranking and Diversification

Both the conventional recommendation scenario and the interactive recommendation scenario encounter the ranking problem of items if the system provides several items in a list. For example, the PRP principle suggests that top-N relevant items should be shown to the user to achieve optimised ranking [62, 20, 144], but this is based on the assumption of independent items in the list. Traditionally, research for recommender systems has focused on improving the accuracy of the rating estimation for all the unknown ratings. However, in a ranking scenario, the top-ranked items and their order are the most important factors and thus accuracy-based algorithms can be insufficient [145, 146], leading to the *learning to rank* techniques [147] tailored for recommender system scenarios. The related recommendation problem usually referred to as *top-N recommendation* [56, 148, 46, 149].

One important factor of top-N recommendation is to provide a diversified item list to satisfy the user's information need. Diversity is realised as one of the most important aspects for the recommendation quality [45, 21, 46, 47]. It is the key factor to help users explore new interests that they might not discover by themselves and thus enhance user experience. Ziegler et al. proposed a re-ranking algorithm for topic diversification [46] which explicitly introduces an intra-list similarity metric based on content-based features of the items. Then the topic diversification is improved by reducing the in-list similarity [46]. According to their studies, diversity of a recommendation list may hamper precision to some degree, but will improve user satisfaction as a whole. Zhang and Hunley formalised the intra-list topic diversification problem by addressing a multi-objective optimisation problem on diversity and preference similarity, in which a relevant list is first provided and the items are re-arranged according to the distance with the user's profile [47]. On the other hand, in [45], Lathia et al. discussed the diversification of recommended items as a temporal process so that the system delivers novel items with respect to the recommendations made in the past. They provided a hybrid algorithm that can offer dynamic recommendations, and they also discovered the negative correlation between user profile length and the degree of recommendation diversity. The issue of evaluating the novelty and the diversity of recommendations has also been raised [21]. Meanwhile, diversification of search results has been extensively studied in the information retrieval community [41, 48, 150, 50], resulting in a fruitful set of diversification methods and evaluation measures. We encounter the diversification problem when multiple items are shown at one interaction interface; and, as such, the ranking of items is highly related to our proposed scenario [144].

2.3.6 Relevant Economic Concepts

Economic theory has been proven to be useful for information retrieval [151]. For example, portfolio theory [152] has been applied to optimise a list of retrieved results [25]. The basic idea behind the portfolio theory is to control the overall uncertainly inevitable in the estimations. By adopting portfolio theory, the list of retrieved documents can be diversified such that the overall risk is reduced. The efficient frontier concept was also introduced in [25], and the trade-off between accuracy and diversity was discussed. In addition, Wang et al. adopted portfolio theory for multimedia fusion [153] to address the uncertainty and correlation among different modalities in existing fusion methods. Marc et al. proposed a dynamical information retrieval model with a portfolio-armed bandit machine approach [154]. Recent increasing attention on exploiting economic principles in for IR [155] may also fall under the same direction.

2.4 Summary

This section aimed to provide an overview of the related areas in recommender system and their connections to this thesis. We first reviewed CF techniques, especially memory-based CF and MF methods, which will form the building blocks of our proposed algorithms. Then we discussed previous work on cold-start problems and concluded that mainly it either used supplementary content-based information or adopted interview processes before starting recommendation. After that, we discussed several interconnected issues that emerge in this thesis, including the PRP principle in recommender systems,

relevance feedback in IR, the EE trade-off, (PO)MDP processes, diversification of a recommendation list and related economic concepts.

From the literature review, we concluded that until now most work has addressed the cold-start problem by adopting a pre-recommendation stage (these methods have been summarised in Table 2.1). We argue that this strategy has neglected the user's information need at the initial stage. On the contrary, we propose an interactive recommendation mechanism and define the target to be maximising the overall satisfaction during a period of time. This goal combines both the exploitation aspect in which recommended items should be of the user's interest right away and the exploration aspect where the recommended items should also be useful for learning the user's profile. As such, we reviewed existing techniques for EE problems including approximate solutions such as upper confidence bound methods for MAB and solutions to (PO)MDP. When multiple items are recommended in each round, item diversification is naturally concerned. We then reviewed several related economic concepts including the portfolio theory for conducting item diversification. In addition, we also compared our proposed framework with relevance feedback in IR. And, last but not least, all of the discussions in this thesis are based on the fact that ratings are correlated, which is the basis for the system to update through interactions. This is a chief difference from the PRP principle for recommender systems.

Chapter 3

Interactive Collaborative Filtering

In this chapter, we introduce the interactive collaborative filtering (ICF) framework which we use to tackle the user cold-start problem. ICF aims to incorporate an interactive mechanism into the collaborative filtering (CF) process. While users receive sequential recommendations, the recommendation predictions are constantly refined using up-to-date feedback on the recommended items. We formulate the objective of this new type of recommendation mechanism as maximising the overall feedback over a period of time. This goal naturally leads to the trade-off between the two interconnected aspects: (i) learning about the user, and (ii) recommending informative items to her. As such, there is no need to use an additional per-recommendation "interview" procedure to learn about the cold-start user, as commonly adopted in previous work [110, 17]. In addition, with the interactive mechanism, the system can also discover interesting items for individual users when and if the user's personal preferences and contexts evolve over time.

We start from the objective function of ICF. Then we derive the probabilistic distributions of the user and item latent factors with the probabilistic matrix factorization (PMF) model. Based on the probabilistic model, we leverage several exploitation-exploration algorithms to obtain several sub-optimal decision policies, including the empirical Thompson sampling and upper confidence bound algorithms (UCB). We conduct experiments on both cold-start users and warm-start users with drifting tastes. Results show significant improvements of our methods over several strong baselines for the MovieLens, EachMovie and Netflix datasets.

3.1 Objective Function

Suppose the system has N items and M users in record (for the reader's convenience, we also provide a detailed notation list in Table 3.1). The ratings between them are recorded in the preference matrix $\mathbf{R}$ in which each element $r_{u,i}$ is the observed rating from the user u to the item i. Without loss of generality, we consider the following process in discrete timesteps. Suppose the target user is now denoted simply by u. At each timestep $t \in [1, 2, \ldots, T]$, the system delivers (recommends) an item to the target user. The user will then give feedback in the form of ratings, or "like"s and "dislike"s, or ignore the recommendation ("unknown"s). In either way, we denote the feedback as $r_{u,i(t)}$, the rating collected by the system from user u in regard to the recommended item $i(t)$ at timestep t. In other words, $r_{u,i(t)}$ is the "reward"

Table 3.1: Summary of key notations in Chapter 3.

Notation	Description
N, M	The number of items, the number of users
K	The dimension of the latent space for user/item feature vectors
$\mathbf{R}$	The preference matrix
$t \in [1, T]$	Discrete timestep. One item is recommended at each t
$i(t)$	The index of the item delivered to the user at time t
σ_0^2	The variance of the observation noise in the PMF
σ_p^2, σ_q^2	The prior variance of the user, item feature vectors for the PMF
λ_p	$\lambda_p = \sigma_0^2/\sigma_p^2$
$\boldsymbol{p}_u, \boldsymbol{q}_i$	User u's feature vector, item i's feature vector
$\boldsymbol{\nu}_i, \boldsymbol{\Psi}_i$	The mean and covariance matrix of item i's feature vector
$\boldsymbol{\mu}_{u,t}, \boldsymbol{\Sigma}_{u,t}$	The mean and covariance matrix of user u's feature vector calculated at time t
$\mathbf{D}_{u,t}$	The observation matrix for user u at time t, with each row being a recommended item's feature vector
$\boldsymbol{r}_{u,t}$	Observed ratings from the recommended items until t
α, c, ϵ	Exploration rate for LinUCB, GLM-UCB and ϵ-greedy respectively

collected by the system from this target user. After receiving the feedback, the system updates its model and decides on which item to recommend next.

To formulate the decision process, let us denote $\mathcal{H}(t)$ as the available information at t the system has for the target user

$$\mathcal{H}(t) = \{i(1), r_{u,i(1)}, \ldots, i(t-1), r_{u,i(t-1)}\} .$$

The item is selected according to a strategy π, which is defined as a function from the current information to the selected item

$$i(t) \equiv \pi(\mathcal{H}(t)) .$$

The optimal strategy should maximise the cumulated expected reward during T timesteps,

$$\boldsymbol{i}^*(\cdot) = \arg\max_{\boldsymbol{i}(\cdot)} \sum_{t=1}^{T} \mathbb{E}[r_{u,i(t)}], \tag{3.1}$$

where $\boldsymbol{i}(\cdot) = \{i(1), i(2), \ldots, i(T)\}$ and $i(t)$ is the item selected at timestep t. Because of the nature of recommender systems, here we use reward rather then regret to express the objective function, and maximising cumulative reward is equivalent to minimising cumulative regret. Here we consider the quality of recommendations at different timesteps as equally important, and summarise the user's overall satisfaction over a given period T. In our experiments, we show that a higher level of exploration is required to achieve a longer-term cumulative reward.

This objective falls into the ambit of the multi-armed bandit problem, where we regard each item as an arm of the bandit. The next questions are how to estimate the reward and how to optimise the objective function. Using the latent factor model [156], the rating is estimated as the product of user

and item feature vectors of dimension K: $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$ where $\boldsymbol{p}_u = (p_{u,1}, p_{u,2}, \ldots, p_{u,K})^T$ and $\boldsymbol{q}_i = (q_{i,1}, q_{i,2}, \ldots, q_{i,K})^T$. This is widely used in many CF algorithms:

$$r_{u,i} = \boldsymbol{p}_u^T \boldsymbol{q}_i + \xi, \tag{3.2}$$

where $\xi \sim \mathcal{N}(0, \sigma_0^2)$ is the observation noise. The objective function is then re-formulated as follows:

$$\boldsymbol{i}^*(\cdot) = \arg\max_{\boldsymbol{i}(\cdot)} \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{p}_u, \boldsymbol{q}_{i(t)}} [\boldsymbol{p}_u^T \boldsymbol{q}_{i(t)} | t] \, . \tag{3.3}$$

The question now is how to optimise the objective function.

3.2 Item Selection via Sampling

Both $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$ are random variables following certain distributions $p(\boldsymbol{p}_u, \boldsymbol{q}_i | t)$. A heuristic solution of Eq. (3.3) is to *sample* an item based on its probability of being optimal [32],

$$p\left(i(t) = i\right) = \int \mathbb{I}\left[\mathbb{E}(r_{u,i} | \boldsymbol{p}_u, \boldsymbol{q}_i) = \max_j \mathbb{E}(r_{u,j} | \boldsymbol{p}_u, \boldsymbol{q}_j)\right] \cdot p(\boldsymbol{p}_u, \boldsymbol{q}_i | t) d\boldsymbol{q}_i d\boldsymbol{p}_u, \tag{3.4}$$

where $\mathbb{I}$ is the indicator function [32]; it is 1 when the equality holds (i.e., when item i has the highest expected rating given $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$); otherwise 0. Thus, integrating $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$ out gives the probability of being optimal at t for item i. The integration is usually computationally expensive [32], but in practice, no need to compute explicitly. Here, we leverage a sampling method, Thompson sampling, to approximate the integration in Eq. (3.4). A nice property of Thompson sampling is that the integration is circumvented by sampling both the user and item feature vectors together from their distributions (considering the uncertainty from both aspects) and picking the item that leads to the largest expectation of the reward:

$$i^*(t)_{ts} = \arg\max_i \mathbb{E}(r_{u,i} | \tilde{\boldsymbol{p}}_u, \tilde{\boldsymbol{q}}_i), \tag{3.5}$$

where $\tilde{\boldsymbol{p}}_u$ and $\tilde{\boldsymbol{q}}_i$ are the sampled user and item feature vectors, which will be described in the next section.

3.2.1 Distributions of User and Item Feature Vectors

In this section, we adopt the PMF model [31] to build the distributions for the user and the item feature vectors, which are then used to generate the samples. According to PMF, the conditional probability distribution of the rating given the user and item feature vectors follows a Gaussian distribution

$$p(r_{u,i} | \boldsymbol{p}_u^T \boldsymbol{q}_i, \sigma_0^2) = \mathcal{N}(r_{u,i} | \boldsymbol{p}_u^T \boldsymbol{q}_i, \sigma_0^2) \, . \tag{3.6}$$

We denote $\mathbf{P}$ ($\mathbf{Q}$) as the user (item) feature vector matrix, where each row vector represents a user (item) feature vector ($\mathbf{P} = [\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_M]^T$, $\mathbf{Q} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_N]^T$). The distribution of the preference matrix $\mathbf{R}$ given $\mathbf{P}$ and $\mathbf{Q}$ is then the joint probability, i.e.,

$$p(\mathbf{R} | \mathbf{P}, \mathbf{Q}, \sigma_0^2) = \prod_{u=1}^{M} \prod_{i=1}^{N} [\mathcal{N}(r_{u,i} | \boldsymbol{p}_u^T \boldsymbol{q}_i, \sigma_0^2)]^{\delta_{u,i}},$$

where $\delta_{u,i} = 1$ if user u rated item i and $\delta_{u,i} = 0$ otherwise.

Similar to the PMF model [31], we define the prior distributions of the user and the item feature vectors as Gaussian with prior variances σ_p^2 and σ_q^2

$$p(\boldsymbol{p}_u|\sigma_p^2) = \mathcal{N}(\boldsymbol{p}_u|\mathbf{0}, \sigma_p^2\mathbf{I}), \tag{3.7}$$

$$p(\boldsymbol{q}_i|\sigma_q^2) = \mathcal{N}(\boldsymbol{q}_i|\mathbf{0}, \sigma_q^2\mathbf{I}) \ . \tag{3.8}$$

By observing the rating matrix $\mathbf{R}$, we can obtain the posterior distributions for the user and the item feature vectors [31]. Here we focus on the conditional distribution of the user (item) feature vectors, given the current item (user) feature vectors to implement Markov chain Monte Carlo and Gibbs Sampling (MCMC-Gibbs):

$$\begin{aligned}
&p(\mathbf{P}|\mathbf{R}, \mathbf{Q}, \sigma_0^2, \sigma_p^2) \propto p(\mathbf{R}|\mathbf{P}, \mathbf{Q}, \sigma_0^2, \sigma_p^2) \cdot p(\mathbf{P}|\mathbf{Q}, \sigma_0^2, \sigma_p^2) \\
&\propto \prod_{u=1}^{M} \mathcal{N}(\boldsymbol{p}_u|\mathbf{0}, \sigma_p^2\mathbf{I}) \prod_{i=1}^{N} [\mathcal{N}(r_{u,i}|\boldsymbol{p}_u^T\boldsymbol{q}_i, \sigma_0^2)]^{\delta_{u,i}} \\
&\propto \prod_{u=1}^{M} \exp\left[-\frac{1}{2\sigma_0^2}\Big(\frac{\sigma_0^2}{\sigma_p^2}\boldsymbol{p}_u^T\boldsymbol{p}_u + \sum_{\delta_{u,i}=1}(r_{u,i} - \boldsymbol{p}_u^T\boldsymbol{q}_i)^2\Big)\right] \\
&\propto \prod_{u=1}^{M} \exp\left[-\frac{1}{2\sigma_0^2}\Big(\boldsymbol{p}_u^T(\sum_{\delta_{u,i}=1}\boldsymbol{q}_i\boldsymbol{q}_i^T + \frac{\sigma_0^2}{\sigma_p^2}\mathbf{I})\boldsymbol{p}_u - 2\sum_{\delta_{u,i}=1} r_{u,i}\boldsymbol{q}_i^T\boldsymbol{p}_u\Big)\right].
\end{aligned}$$

This means that, for each user, its feature vector follows a multivariate Gaussian distribution given the feature vectors of the items rated by the user:

$$p(\boldsymbol{p}_u|\mathbf{R},\mathbf{Q}, \sigma_0^2, \sigma_p^2) = \mathcal{N}(\boldsymbol{p}_u|\boldsymbol{\mu}_u, \boldsymbol{\Sigma}_u), \tag{3.9}$$

$$\boldsymbol{\mu}_u = (\mathbf{D}_u^T\mathbf{D}_u + \lambda_p\mathbf{I})^{-1}\mathbf{D}_u^T\boldsymbol{r}_u, \tag{3.10}$$

$$\boldsymbol{\Sigma}_u = (\mathbf{D}_u^T\mathbf{D}_u + \lambda_p\mathbf{I})^{-1}\sigma_0^2 \ . \tag{3.11}$$

Here, $\mathbf{D}_u$ is the observation matrix for the user, with each row being the feature vector of an item rated by the user, sampled from its posterior; $\boldsymbol{r}_u$ denotes the vector of corresponding ratings to these items from the user; and $\lambda_p = \sigma_0^2/\sigma_p^2$.

Similarly, the posterior distribution for the item feature vector, $\boldsymbol{q}_i$, conditioned on the sampled user feature vectors, can be obtained as

$$p(\boldsymbol{q}_i|\mathbf{R},\mathbf{P}, \sigma_0^2, \sigma_q^2) = \mathcal{N}(\boldsymbol{q}_i|\boldsymbol{\nu}_i, \boldsymbol{\Psi}_i), \tag{3.12}$$

$$\boldsymbol{\nu}_i = (\mathbf{B}_i^T\mathbf{B}_i + \lambda_q\mathbf{I})^{-1}\mathbf{B}_i^T\boldsymbol{r}_i, \tag{3.13}$$

$$\Psi_i = (\mathbf{B}_i^T\mathbf{B}_i + \lambda_q\mathbf{I})^{-1}\sigma_0^2, \tag{3.14}$$

where $\mathbf{B}_i$ is the observation matrix with each row being a sampled user feature vector.

The distributions converge by alternatively sampling the item and the user feature vectors according

Algorithm 3.1: Thompson sampling

Require: parameters for the item feature vector distributions $\Theta = \{(\boldsymbol{\nu}_1, \boldsymbol{\Psi}_1), \ldots, (\boldsymbol{\nu}_N, \boldsymbol{\Psi}_N)\}$, σ_0, λ_p
  Initialization: $\mathbf{A} \leftarrow \lambda_p \mathbf{I}$
  $\boldsymbol{b} \leftarrow \mathbf{0}$
  for $t = 1, 2, 3, ..., T$ **do**
    Estimate $\boldsymbol{\mu}_{u,t} = \mathbf{A}^{-1}\boldsymbol{b}$
    Estimate $\boldsymbol{\Sigma}_{u,t} = \mathbf{A}^{-1}\sigma_0^2$
    Sample $\tilde{\boldsymbol{p}}_{u,t}$ from $\mathcal{N}(\boldsymbol{p}_{u,t}|\boldsymbol{\mu}_{u,t}, \boldsymbol{\Sigma}_{u,t})$
    Sample $\tilde{\boldsymbol{q}}_i$ from $\mathcal{N}(\boldsymbol{q}_i|\boldsymbol{\nu}_i, \boldsymbol{\Psi}_i)$for $\{i \in \{1, 2, \ldots, N\}\}$
    Select the arm $i^*(t) = \arg\max_i \tilde{\boldsymbol{p}}_u^T \tilde{\boldsymbol{q}}_i$
    Receive the reward $r_{u,i^*(t)}$
    Update $\mathbf{A} \leftarrow \mathbf{A} + \tilde{\boldsymbol{q}}_{i^*(t)} \tilde{\boldsymbol{q}}_{i^*(t)}^T$
    Update $\boldsymbol{b} \leftarrow \boldsymbol{b} + r_{u,i^*(t)} \tilde{\boldsymbol{q}}_{i^*(t)}$
  end for

to the conditional distributions for them. Then, both the expected user and the expected item feature vectors ($\boldsymbol{\mu}_u$ and $\boldsymbol{\nu}_i$) and their uncertainties ($\boldsymbol{\Sigma}_u$ and $\boldsymbol{\Psi}_i$) are obtained.

3.2.2 Thompson Sampling

Thompson sampling can be implemented according to the distributions while they are updated online whenever new ratings are collected by the system. However, in the ICF scenario, the distribution of the target user's feature vector is much more sensitive to her feedback on the items. On the item side, since each item has usually collected relatively sufficient ratings, it is not necessary to retrain its feature vector immediately after receiving any rating from the target user, and we choose to periodically retrain them. Therefore, we simply use the notation $\tilde{\boldsymbol{q}}_i$ to express a sampled item feature vector from the presently calculated item feature vector distribution. For the target user, its observation matrix grows each time, and its distribution can be described similarly conditioned on the observations:

$$\tilde{\boldsymbol{p}}_{u,t} \sim \mathcal{N}(\boldsymbol{p}_{u,t}|\boldsymbol{\mu}_{u,t}, \boldsymbol{\Sigma}_{u,t}), \tag{3.15}$$

where

$$\boldsymbol{\mu}_{u,t} = (\mathbf{D}_{u,t}^T \mathbf{D}_{u,t} + \lambda_p \mathbf{I})^{-1} \mathbf{D}_{u,t}^T \boldsymbol{r}_{u,t}, \tag{3.16}$$

$$\boldsymbol{\Sigma}_{u,t} = (\mathbf{D}_{u,t}^T \mathbf{D}_{u,t} + \lambda_p \mathbf{I})^{-1} \sigma_0^2 \ , \tag{3.17}$$

Similarly, $\mathbf{D}_{u,t}$ is the observation matrix with each row being the recommended item's feature vector; $\boldsymbol{\Sigma}_{u,t}$ is the uncertainty of the user feature vector at time t; and $\boldsymbol{r}_{u,t}$ is the column vector that contains all the observations until time t.

From Eq. (3.5), the Thompson sampling method with the PMF modeling suggests to choose the item with the highest value of the inner product of the sampled values, and Eq. (3.5) can be approximated as:

$$i^*(t)_{ts} = \arg\max_i \tilde{\boldsymbol{p}}_{u,t}^T \tilde{\boldsymbol{q}}_i, \tag{3.18}$$

where $\tilde{\boldsymbol{p}}_{u,t}$ is sampled from the estimated distribution in Eq. (3.15). This algorithm is described in Algorithm 3.1.

Algorithm 3.2: Linear UCB

Require: MAP solution of item feature vectors $\mathbf{Q} = \{\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_N\}$, σ_0, λ_p, and $\alpha \in \mathbb{R}_+$
  Initialization: $\mathbf{A} \leftarrow \lambda_p \mathbf{I}$
  $\boldsymbol{b} \leftarrow \mathbf{0}$
  for $t = 1, 2, 3, ..., T$ **do**
    Estimate $\boldsymbol{\mu}_{u,t} = \mathbf{A}^{-1}\boldsymbol{b}$
    Estimate $\boldsymbol{\Sigma}_{u,t} = \mathbf{A}^{-1}\sigma_0^2$
    Choose the item

$$i^*(t) = \arg\max_i \left((\boldsymbol{\mu}_{u,t})^T \boldsymbol{\nu}_i + \alpha ||\boldsymbol{\nu}_i||_{2,\boldsymbol{\Sigma}_{u,t}} \right)$$

    Receive the reward $r_{u,i^*(t)}$
    Update $\mathbf{A} \leftarrow \mathbf{A} + \boldsymbol{\nu}_{i^*(t)} \boldsymbol{\nu}_{i^*(t)}^T$
    Update $\boldsymbol{b} \leftarrow \boldsymbol{b} + r_{u,i^*(t)} \boldsymbol{\nu}_{i^*(t)}$
  end for

Thompson sampling enables exploration through the "width" of the distributions of the inner product of the user and the item feature vectors. The "width" further comes from both the uncertainties of the user and the item feature vectors. By this approach described above, the uncertainties of the user and the item feature vectors are considered on the same footing. However, considering ICF as a user-centric scenario (Figure 1.3), the obtained knowledge on the target user side may be much more important than that on the item side, especially when items have collected many ratings and thus are always already well-learnt. Therefore, in the following part, we adopt a biased view so that the item feature vectors are assumed to be well-learnt as the maximum a posteriori (MAP) solution $\boldsymbol{\nu}_i$ from the distributions obtained by PMF, and only the user feature vector distributions are maintained for the sampling process.

3.3 Item Selection via Confidence Bound

With the item feature vectors known and fixed, the reward in Eq. (3.2) tends to be a linear form with the item feature vectors as coefficients, and, the essence of the EE is to approach the user feature vector. Therefore, such a problem falls into the framework of *linear bandits* [29]. Linear upper confidence bound algorithm, and its variations are widely used for such problems. In this way, we take the MAP estimation of the item feature vectors $\boldsymbol{\nu}_i$ as the representatives of the items and assume them to be fixed.

In the following, linear and generalised linear UCB algorithms are presented for our problem respectively. A variation of ϵ-greedy algorithm is also provided for comparison.

3.3.1 Linear UCB

As mentioned above, assuming the item feature vectors as fixed, the reward function reduces to be linear in the item feature vectors, and the objective function in Eq. (3.1) is further written as

$$\boldsymbol{i}^*(\cdot) = \arg\max_{\boldsymbol{i}(\cdot)} \sum_{t=1}^{T} \mathbb{E}[r_{u,i(t)}] = \arg\max_{\boldsymbol{i}(\cdot)} \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{p}_u}[\boldsymbol{p}_u^T|t] \boldsymbol{\nu}_{i(t)}, \tag{3.19}$$

where $\mathbb{E}_{\boldsymbol{p}_u}[\boldsymbol{p}_u|t]$ can be estimated according to Eq. (3.16).

The expected user feature vector can be obtained according to Eq. (3.16). Now the uncertainty of the reward can be obtained as the estimated variance of the inner product of the user and item feature vectors

Algorithm 3.3: GLM-UCB

Require: MAP solution of item feature vectors $\mathbf{Q} = \{\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_N\}$, σ_0, λ_p, and $c \in \mathbb{R}_+$
  Initialization: $\mathbf{A} \leftarrow \lambda_p \mathbf{I}$
  for $t = 1, 2, 3, \ldots, T$ **do**
    Estimate $\hat{\boldsymbol{p}}_{u,t}$ by Eq. (3.25)
    Estimate $\boldsymbol{\Sigma}_{u,t} = \mathbf{A}^{-1}\sigma_0^2$
    Choose the item

$$i^*(t) = \arg\max_i \left(g(\hat{\boldsymbol{p}}_{u,t}^T \boldsymbol{\nu}_i) + c\sqrt{\log t} ||\boldsymbol{\nu}_i||_{2,\boldsymbol{\Sigma}_{u,t}} \right)$$

    Receive the reward $r_{u,i^*(t)}$
    Update $\mathbf{A} \leftarrow \mathbf{A} + \boldsymbol{\nu}_{i^*(t)} \boldsymbol{\nu}_{i^*(t)}^T$
  end for

$\boldsymbol{p}_u^T \boldsymbol{\nu}_i$, which comes from the uncertainty of the estimation in the user feature vector. The estimated variance is the 2-norm based on $\boldsymbol{\Sigma}_{u,t}$ (according to Eq. (3.17), but note that here the observation matrix is made up of the posterior feature vectors of the items):

$$||\boldsymbol{\nu}_i||_{2,\boldsymbol{\Sigma}_{u,t}} \equiv \sqrt{\boldsymbol{\nu}_i^T \boldsymbol{\Sigma}_{u,t} \boldsymbol{\nu}_i} \,. \tag{3.20}$$

According to [157], with the item feature vectors known and fixed, the expectation of the reward by choosing item i is bounded in the interval $\Theta_{i,t}$ with a probability at least $1 - \zeta$

$$\Theta_{i,t} = \left[(\boldsymbol{\mu}_{u,t})^T \boldsymbol{\nu}_i - \alpha ||\boldsymbol{\nu}_i||_{2,\boldsymbol{\Sigma}_{u,t}}, (\boldsymbol{\mu}_{u,t})^T \boldsymbol{\nu}_i + \alpha ||\boldsymbol{\nu}_i||_{2,\boldsymbol{\Sigma}_{u,t}} \right] \tag{3.21}$$

where $\alpha = 1 + \sqrt{\ln(2/\zeta)/2}$. The bounded interval motivates an UCB bandit algorithm, i.e., at each timestep, to choose the item with the highest upper confidence bound:

$$i^*(t)_l = \arg\max_i \left((\boldsymbol{\mu}_{u,t})^T \boldsymbol{\nu}_i + \alpha ||\boldsymbol{\nu}_i||_{2,\boldsymbol{\Sigma}_{u,t}} \right) \,. \tag{3.22}$$

This algorithm is given in Algorithm 3.2. This algorithm is proven to have a very tight regret bound of $\tilde{O}(\sqrt{T})$ [26].

As defined in Eq. (3.17), matrix $\boldsymbol{\Sigma}_{u,t}$ is a regularised fisher information matrix, measuring how much "information" is known about the user feature vector from the previously recommended items, given that the item feature vectors are known already. This means that, to recommend an item that maximises $||\boldsymbol{\nu}_i||_{2,\boldsymbol{\Sigma}_{u,t}}$ is to recommend an item that has been the least represented (understood) by the perviously recommended items.

3.3.2 Generalised Linear UCB

The problem can be also linked to the generalised linear bandit problem in [125], which gives a general solution generalised linear model bandit-upper confidence bound (GLM-UCB) if we assume the reward takes the following form

$$r_{u,i(t)} = g\left(\boldsymbol{p}_u^T \boldsymbol{q}_{i(t)}\right) + \xi, \tag{3.23}$$

where g is a monotonically increasing function which takes a linear or nonlinear form. Here we give two options of function g: a linear form suggested in Eq. (3.2), and a sigmoid form

$$g(\boldsymbol{p}_u^T \boldsymbol{q}_i) = \frac{1}{1 + \boldsymbol{e}^{-\boldsymbol{p}_u{}^T \boldsymbol{q}_i}} \ . \tag{3.24}$$

Similar to the derivations in LinUCB, here the item feature vector $\boldsymbol{q}_i$ is approximated by the MAP solution $\boldsymbol{\nu}_i$. On the other hand, we need to estimate the user feature vector according to the generalised linear model, which here we denote as $\hat{\boldsymbol{p}}_{u,t}$ (note that here the solution $\hat{\boldsymbol{p}}_{u,t}$ is no longer the MAP solution in Eq. (3.16) due to the nonlinear function g). In general, according to [125], the quasi-likelihood estimator $\hat{\boldsymbol{p}}_{u,t}$ of Eq. (3.23) is the solution of

$$\sum_{\tau=1}^{t-1} \left(r_{u,i(\tau)} - g(\hat{\boldsymbol{p}}_{u,t}^T \boldsymbol{\nu}_{i(\tau)}) \right) \boldsymbol{\nu}_{i(\tau)} = 0 \ . \tag{3.25}$$

Specifically, for a sigmoid form, it is estimated as

$$\sum_{\tau=1}^{t-1} \left(r_{u,i(\tau)} - \frac{1}{1 + \boldsymbol{e}^{-\hat{\boldsymbol{p}}_{u,t}^T \boldsymbol{\nu}_{i(\tau)}}} \right) \boldsymbol{\nu}_{i(\tau)} = 0 \ . \tag{3.26}$$

For a linear form, the estimate is the same as the MAP estimation of the user feature vectors Eq. (3.16).

The GLM-UCB algorithm follows a similar process as linear UCB, i.e., first $\hat{\boldsymbol{p}}_{u,t}$ is estimated, and the choice of the item is based on the estimated $\hat{\boldsymbol{p}}_{u,t}$ but with the exploration part added which is 2-norm based on $\boldsymbol{\Sigma}_{u,t}$ (Eq. (3.20)) multiplied by a factor $c\sqrt{\log t}$ [125]

$$i^*(t)_{gl} = \arg\max_i \left(g(\hat{\boldsymbol{p}}_{u,t}^T \boldsymbol{\nu}_i) + c\sqrt{\log t} ||\boldsymbol{\nu}_i||_{2,\boldsymbol{\Sigma}_{u,t}} \right) \ . \tag{3.27}$$

The GLM-UCB algorithm is illustrated in Algorithm 3.3. Note that the exploration term α is time-dependent:

$$\alpha = \alpha(t) = c\sqrt{\log t}, \tag{3.28}$$

where c is a constant with respect to t [125]. With term $c\sqrt{\log t}$, the decreasing trend of $||\boldsymbol{\nu}_i||_{2,\boldsymbol{\Sigma}_{u,t}}$ is weakened so that the exploration level is maintained to some extent. Using the conclusion from [125], GLM-UCB has a regret bound of $\tilde{O}(\sqrt{T})$.*

Just like the other index-based EE algorithms [29], these algorithms have a low computational complexity, which is $O(T^3 + K^2 N)$.

*The detailed form of the bound is looser than that of LinUCB but it is more general.

Algorithm 3.4: Linear ϵ-greedy

Require: MAP solution of item feature vectors $\mathbf{Q} = \{\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_N\}$, λ_p and $\epsilon \in [0, 1]$
Initialization: $\mathbf{A} \leftarrow \lambda_p \mathbf{I}$
$\boldsymbol{b} \leftarrow \mathbf{0}$
for $t = 1, 2, 3, ..., T$ **do**
Estimate $\boldsymbol{\mu}_{u,t} = \mathbf{A}^{-1}\boldsymbol{b}$
With probability $1 - \epsilon$ choose the item

$$i^*(t) = \arg\max_i \left((\boldsymbol{\mu}_{u,t})^T \boldsymbol{\nu}_i\right)$$

Otherwise choose an item randomly
Receive the reward $r_{u,i^*(t)}$
Update $\mathbf{A} \leftarrow \mathbf{A} + \boldsymbol{\nu}_{i^*(t)} \boldsymbol{\nu}_{i^*(t)}^T$
Update $\boldsymbol{b} \leftarrow \boldsymbol{b} + r_{u,i^*(t)} \boldsymbol{\nu}_{i^*(t)}$
end for

3.3.3 Linear ϵ-greedy

The linear ϵ-greedy algorithm is based on the greedy strategy under our setting, which can be described as

$$i^*(t)_g = \arg\max_i (\boldsymbol{\mu}_{u,t})^T \boldsymbol{\nu}_i \ . \tag{3.29}$$

where $\boldsymbol{\mu}_{u,t}$ and $\boldsymbol{\nu}_i$ are the MAP solutions for the PMF model. For simplicity, we simply refer to this strategy as greedy PMF, or simply PMF.

This greedy strategy is the myopic strategy that always picks the item leading to the highest expected reward based on current knowledge. Linear ϵ-greedy that we adopt here is the naive algorithm which chooses the greedy strategy with probability $1 - \epsilon$ and explores into random items with probability ϵ. The algorithm is described in Algorithm 3.4.

For the above algorithms, two factors contribute to the selection of the item: the exploitation factor suggested by the greedy algorithm Eq. (3.29) and the exploration factor which is controlled by parameters α, c and ϵ respectively. For each of the three algorithms, the larger the parameter is, the more emphasis is put onto the exploration effort accordingly.

3.4 Experiments

In this part, we show the results for three experiments. First, we test the performance of the proposed EE algorithms on cold-start users. We interactively provide recommendations to these users using both EE algorithms and myopic algorithms to compare the results. Then, we conduct experiments on warm-start users with interest changes, in order to test the proposed algorithms effectiveness of adapting the users' taste drifts. Finally, we use the proposed algorithms in a top-n recommendation scenario to test their ranking performances. Among these three experiments, the first one is directly related to the formulations in this chapter. We include the other two (warm-start with taste-drifting and ranking scenarios) in addition to the cold-start experiment to show the flexibility of our proposed algorithms.

3.4.1 Datasets

We base our experiments on three popular datasets MovieLens 100K, EachMovie and Netflix. The basic information of these three datasets is summarised in Table 3.2.

The MovieLens 100K dataset was collected by the GroupLens Research Project at the University of Minnesota [60]. The data was collected through the MovieLens website (`http://movielens.umn.edu`) during the seven-month period from 19 Sep 1997 through 22 Apr 1998. Users with less than 20 ratings were removed from this dataset. Aside from the rating information between users and items, this dataset also includes the date/time when the ratings were registered, the demographic information of the users and the genre information of the items.

The EachMovie dataset was collected by the DEC Systems Research Center (currently HP/Compaq Research) via its EachMovie recommendation service during an eighteen-month period [158]. Similar to MovieLens, this dataset also includes the date/time information, the demographic information of the users and the genre information of the items. We choose the above two datasets because they are quite popular among the research community of recommender systems and both of them contain genre information of movies that we need for experiments in Section 3.4.5 and 3.5.

The Netflix dataset was constructed to support participants in the Netflix Prize (`http://www.netflixprize.com`). It was collected between Oct 1998 and Dec 2005. Only the user ID, item ID, and the value/date of the rating are included for each rating, whereas no demographic information or genre information is provided for the users or the items. Compared with the other two datasets, this dataset covers the longest period and has the largest size. We use Netflix to test the performance of our algorithms in a larger scale.

Due to the interactive nature of our problem, an online experiment with true interactions from users would be ideal, but it is not always possible [26]. Instead, we follow an unbiased offline evaluation scheme for contextual-bandit algorithms according to [159]. In our setting, we assume that the ratings recorded in the datasets are users' instinctive actions, not biased by the recommendations provided by the system. In this way, the records can be treated as unbiased to represent the feedback in an interactive setting [17].

In order to better compare the results between the three datasets, we normalise the ratings of each of them into the common range $[-1, 1]$. Then we split the data into two user-disjoint sets: the training users and their ratings are used to train the parameters for the item distributions, as required in Thompson Sampling, and to obtain the MAP solutions of the item feature vectors, as required in UCB-based algorithms (Section 3.2.1). The item feature vector information is maintained as unchanged during the test phase when the test users go through the interactive recommendation process during T timesteps because the collected ratings from the target user have trivial effect on the item feature vector distributions. According to the purpose (whether to test the performance on cold-start users, or on warm-start users), we select test users based on different criteria, detailed in Section 3.4.4 and Section 3.4.5 respectively.

3.4.2 Compared Algorithms

The baselines include:

Table 3.2: Characteristics of the datasets (MovieLens 100K, EachMovie and Netflix).

Dataset	MovieLens	EachMovie	Netflix
ratings (all integral)	1 – 5	0 – 5	1 – 5
#users	943	72,916	480,189
#items	1,683	1,648	17,770
#ratings per user	106.04	38.56	209.25
#ratings per item	59.42	1706.30	5654.50
total #ratings	100,000	2,811,983	100,480,507

Popularity-based (Pop). The recommender system picks the most popular items to recommend to the target user.

Greedy PMF (PMF). This algorithm is built upon the MAP solution obtained from PMF. We regard it as the myopic algorithm in ICF because it is exploitation only. For each target user, the system needs to retrain the PMF model after each interaction.

Active Learning (AL). Active learning methods have been proposed for the cold-start problem [35]. The idea is to minimise the uncertainty in the model, so that the item with the highest uncertainty is selected to reduce uncertainty [107]. Here we adapt the active learning method to the interactive recommendation process, such that, in each interaction, the item with the highest uncertainty is selected according to the up-to-date knowledge.

Interview Process (Interview). In this process, in the each of the first 5 timesteps, the target user is provided with the most discriminative item at each timestep (as in the active learning process above). From timestep 6, the system shifts to the greedy strategy. This process is to mimic the shift in an interview process from a learning period to a recommending period after a few interview questions. Here we set the number of interview questions to be 5 according to [36] as it is argued that a depth beyond that can bring little accuracy gain.

Our proposed algorithms include the following:

Thompson Sampling (TS). This is Algorithm 3.1.

Linear UCB (LinUCB). This is Algorithm 3.2. α is used to tune this model.

Generalised Linear UCB (GLM). This is Algorithm 3.3. We set function g as a linear function, i.e., GLM-Lin, and a sigmoid function, i.e., GLM-Sig. c is used to tune this model.

Linear ϵ-greedy (ϵ-greedy). This is Algorithm 3.4. A tuning parameter ϵ is used to control the balance between exploitation and exploration.

In addition, we add a constraint for all the algorithms that the same item should not be repeatedly recommended as suggested in most previous ranking-oriented recommendation settings [144, 160].

3.4.3 Evaluation Measures

Three evaluation metrics are used to test the performance of the ICF tasks.

Cumulative Hit@T. A straightforward evaluation measure is the number of the positive ratings collected during the total T interactions

$$\text{hit@}T = \frac{1}{\#\text{users}} \sum_{\text{users}} \sum_{t=1}^{T} \theta_{\text{hit}} \ . \tag{3.30}$$

For both datasets, we define $\theta_{\text{hit}} = 1$ if the rating is 4 or above, and 0 otherwise, similar to the definition of positive ratings in previous work [161].

Cumulative Recall@T. We can also check the recall during T timesteps of the interactions:

$$\text{recall@}T = \frac{1}{\#\text{users}} \sum_{\text{users}} \sum_{t=1}^{T} \frac{\theta_{\text{hit}}}{\#\text{preferences}} . \tag{3.31}$$

Cumulative NDCG@n@T. For the case that multiple items are shown in one interaction, the ranking of the item listed is also important: it is more useful to have the highly relevant items appear earlier in the ranking list. We use the normalised discounted cumulative gain (NDCG@n) as the ranking measure

$$\text{NDCG@}n = \frac{1}{\text{IDCG@}n} \sum_{j=1}^{n} \frac{2^{r_j} - 1}{\log_2(j+1)}, \tag{3.32}$$

where r_j is the real rating of the item shown at ranking position j. $IDCG@n$ is the score of a perfect ranking algorithm, and thus normalises the score such that $0 \leq \text{NDCG@}n \leq 1$. Similar to the cumulative hit and recall, here the cumulative NDCG@n@T takes the sum over T timesteps and average over all test users

$$\text{NDCG@}n\text{@}T = \frac{1}{\#\text{users}} \sum_{\text{users}} \sum_{t=1}^{T} \text{NDCG@}n. \tag{3.33}$$

3.4.4 Cold-Start Cases

Test User Selection

In order to test the system's performance on cold-start users, we first select users with sufficient numbers of recorded ratings in order to test the performance. Here we randomly select 200 users with more than 120 ratings as the test users in order to obtain averaged result of them. We study up to $T = 120$ interactions to sufficiently cover the cold-start period. Then the parameters of item feature vector distributions are trained without these user's ratings according to Section 3.2.1.

Performance Comparison

To do this experiment, we apply each algorithm to obtain the corresponding item to recommend to the users at each timestep, update the user vectors according to Eq. (3.15) and repeat T steps. Optimally-tuned parameters have been adopted for each $T = 10, 20, 40, 80, 120$. Performances of proposed algorithms and the baselines for cold-start users are compared and summarised in Table 3.3. The best-performing algorithm is shown in boldface with * marking significant improvements by Wilcoxon signed-rank test. We chose the very conservative Wilcoxon signed-rank test as it calculates the differences between paired observations and can be used for both Gaussian and non-Gaussian data [162]. The row of improvement shows the increases brought by the best-performing algorithm compared to the greedy PMF strategy.

Table 3.3: Cold-start performance comparison on MovieLens, EachMovie, and Netflix dataset.

Dataset	**MovieLens 100K**				**EachMovie**				**Netflix**			
Measure	Cumulative Hit				Cumulative Hit				Cumulative Hit			
T	10	20	40	120	10	20	40	120	10	20	40	120
Pop	5.060	9.500	15.710	42.170	3.200	6.72	11.200	24.675	4.700	9.335	17.760	45.370
PMF	6.360	11.035	19.390	43.155	4.095	7.590	13.295	31.435	5.675	10.410	18.975	48.900
AL	**6.850**	11.095	18.905	41.785	3.180	6.865	13.270	29.965	7.060	12.595	18.695	21.780
Interview	6.800	**11.885**	20.495	45.745	3.180	6.865	13.270	30.660	3.825	9.165	18.720	52.310
TS	6.465	11.31	19.565	43.745	4.605	8.270	14.025	31.395	6.160	11.420	20.585	52.300
ϵ-greedy	6.375	11.060	19.440	43.180	4.660	8.430	14.315	32.270	5.725	10.545	19.205	49.780
LinUCB	**6.850**	11.835	20.820	46.455	4.610	8.175	14.280	33.590	**7.060**	12.735	**22.855**	56.490
GLM-Lin	**6.850**	11.835	20.820	**46.470**	**5.105**	**9.030**	**15.170**	**35.045**	**7.060**	**12.835**	22.830	**55.620**
GLM-Sig	**6.850**	11.830	**20.825**	46.445	4.765	8.435	14.305	32.605	**7.060**	12.710	22.780	56.275
Improvement	7.7%*	7.2%*	7.4%*	7.7%*	24.7%*	19.0%*	14.1%*	11.5%*	24.4%*	23.3%*	20.5%*	13.7%
Measure	Cumulative Recall				Cumulative Recall				Cumulative Recall			
T	10	20	40	120	10	20	40	120	10	20	40	120
Pop	0.047	0.088	0.144	0.376	0.025	0.053	0.087	0.194	0.015	0.029	0.055	0.139
PMF	0.058	0.099	0.171	0.369	0.037	0.066	0.111	0.246	0.018	0.033	0.059	0.149
AL	**0.064**	0.102	0.170	0.369	0.027	0.059	0.099	0.232	**0.022**	0.04	0.06	0.071
Interview	0.063	0.108	0.182	0.395	0.025	0.054	0.102	0.231	**0.022**	0.04	0.071	0.175
TS	0.060	0.104	0.175	0.380	0.037	0.066	0.109	0.240	0.020	0.036	0.064	0.160
ϵ-greedy	0.0578	0.100	0.172	0.372	0.040	0.071	0.118	0.261	0.020	0.037	0.067	0.173
LinUCB	**0.064**	**0.109**	**0.187**	**0.409**	0.038	0.068	0.115	0.251	**0.022**	0.04	**0.072**	**0.173**
GLM-Lin	**0.064**	**0.109**	**0.187**	**0.409**	**0.042**	**0.072**	**0.121**	**0.272**	**0.022**	**0.041**	0.071	0.171
GLM-Sig	**0.064**	**0.109**	**0.187**	**0.409**	0.038	0.067	0.113	0.252	**0.022**	0.04	0.071	**0.173**
Improvement	9.4%*	10.1%*	9.4%*	10.8%*	13.5%*	9.1%*	9.1%*	10.6%*	22.2%*	24.2%*	22.0%*	16.1%

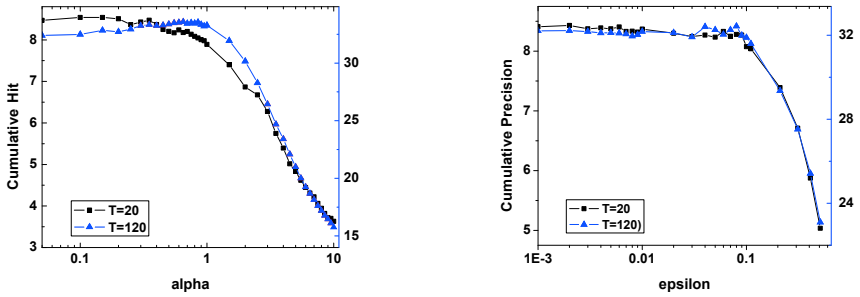
The observations can be summarised into the following points: (i) TS generally works better than the PMF or the ϵ-greedy algorithm. In most cases, TS also exceeds other baseline algorithms (according to cumulative hit). This means that the exploration by considering the uncertainties of the user and the items according to their probability distributions, is more promising than randomly conducting explorations. Nevertheless, TS fails to outperform the LinUCB or GLM algorithms. (ii) In almost all cases, LinUCB, GLM-Lin and GLM-Sig perform better than the baselines. In MovieLens and Netflix, the three algorithms have close performances; whereas in EachMovie, GLM-Lin outperforms all the baselines. The increase by the proposed EE algorithms compared to PMF is up to 7.7% on MovieLens, 24.7% on Eachmovie, and 24.4% on Netflix (according to the cumulative hit). All of the improvements (except one) are statistically significant. (iii) Among all the proposed EE algorithms, linear ϵ-greedy performs worst, but still better than PMF. This suggests that adding some level of exploration can always improve the pure exploitation strategy. (iv) PMF outperforms the popularity-based strategy Pop. (v) The Interview strategy performs better than the active learning strategy AL in the long run, because it shifts to exploitation after learning the user profile (5 timesteps). For all the three datasets, however, the Interview strategy is not as good as the algorithms which are proposed based on the ICF framework. (vi) Also note that the improvements by our algorithms on EachMovie and Netflix appear to be higher than on MovieLens. One explanation could be that MovieLens has been preprocessed: all the users with fewer than 20 ratings (which can be a large number of users) have been removed, and removing them (and simultaneously their ratings) can lead to less popular items being removed as well (when there is no rating left for them). Since more popular rather than less popular items remain in the dataset, it is less beneficial to explore, and eventually results in less significant improvements achieved on MovieLens.

There are two possible reasons that the UCB-based algorithms LinUCB, GLM-Lin and GLM-Sig outperform TS. First, the user uncertainties may play a much more important role in the ICF scenario. Consideration on item-side uncertainties may be helpful for learning the item feature vectors in the long run, but in this user-centric system, it may hamper the user experience. Second, compared with the UCB-based algorithms which explicitly pursue the highest possible performance for each item as their exploration strategy, TS involves considerations on both the positive and negative possible performances for each item. In addition, the sampling process itself imports the exploration instability. However, the UCB-based algorithms are built on the assumption that the item feature vectors are well-learnt. In the case of very limited available data and thus underestimated item feature vectors, it may be necessary to consider the uncertainty of item feature vectors. We leave this problem as our future work.

Impact of Trade-off Parameters

The algorithm-dependent parameters α, c, ϵ are used to balance between exploitation and exploration. Here we focus on the cumulative hit as the measure of performance, and investigate how the performance depends on these parameters, with respect to two horizons $T = 20$ and $T = 120$, shown in Figure 3.1. We show the impact of α for LinUCB as the representative of the UCB-based algorithms while other cases display similar trends.

We observe that when either α or ϵ (for either the case of $T = 20$ or $T = 120$) increases, the



(a) LinUCB (b) ϵ-greedy

Figure 3.1: Cumulative hit against parameter tuning (α for LinUCB and ϵ for ϵ-greedy) on EachMovie.

performance first increases, and then falls down. The peak performance corresponds to the optimal parameter which for $T = 20$ is smaller than that for $T = 120$. This is intuitively correct because more exploration is needed when a longer period is targeted. In practice, to choose T, we can make use of the statistics from the system record, such as the activity distribution of users within a certain period of time after the user's registration.

3.4.5 Warm-Start Cases with Taste Drift

Test User Selection

Through this experiment, we aim to answer the question of whether the algorithms are also applicable on warm-start users to follow up their interests throughout the interactions, especially when their tastes

Table 3.4: Performance comparison on warm-start users with taste drift on MovieLens and EachMovie.

Dataset	**MovieLens 100K**				**EachMovie**			
Measure	Cumulative Hit				Cumulative Hit			
T	60	80	100	120	60	80	100	120
Pop	16.025	18.420	20.490	22.775	19.526	20.437	21.416	22.447
PMF	18.620	21.290	24.060	25.980	19.447	22.453	25.458	28.353
TS	19.095	21.780	24.620	26.515	20.047	22.879	25.832	28.968
ϵ-greedy	18.995	21.72	24.535	26.480	19.984	22.904	25.974	28.805
LinUCB	**20.005**	22.875	**25.775**	27.780	20.205	23.137	26.221	29.247
GLM-Lin	19.895	**22.905**	25.665	27.775	**22.853**	**25.916**	**28.863**	**31.711**
GLM-Sig	20.000	22.835	25.760	**27.790**	20.437	23.358	26.279	29.284
Improvement	7.4%	7.6%*	7.1%	7.0%	17.5%*	15.4%*	13.4%*	11.8%*
Measure	Cumulative Recall				Cumulative Recall			
T	60	80	100	120	60	80	100	120
Pop	0.245	0.286	0.321	0.358	0.126	0.148	0.169	0.189
PMF	0.267	0.311	0.351	0.379	0.126	0.148	0.169	0.189
TS	0.272	0.314	0.360	0.388	0.129	0.149	0.170	0.192
ϵ-greedy	0.273	0.317	0.363	0.391	0.13	0.15	0.172	0.191
LinUCB	**0.291**	0.341	**0.389**	0.422	0.132	0.155	0.178	0.199
GLM-Lin	**0.291**	**0.342**	0.388	**0.424**	**0.156**	**0.180**	**0.204**	**0.223**
GLM-Sig	**0.291**	0.341	0.388	0.421	0.138	0.161	0.182	0.201
Improvement	9.0%	10.0%*	10.8%	11.9%	23.8%*	21.6%*	20.7%*	18.0%*

Table 3.5: Performance comparison for multiple-item recommendations by cumulative NDCG.

Dataset	**MovieLens**				**EachMovie**				**Netflix**			
Measure	NDCG@3		NDCG@5		NDCG@3		NDCG@5		NDCG@3		NDCG@5	
T	20	40	10	20	20	40	10	20	20	40	10	20
Pop	4.876	8.669	2.666	4.633	3.864	5.588	1.752	2.913	5.29	9.165	2.687	4.693
PMF	6.099	9.832	3.361	5.345	5.043	8.083	2.651	4.379	7.465	12.733	3.983	6.837
TS	6.195	9.912	3.393	5.452	5.167	8.320	2.678	4.431	7.962	13.887	**4.237**	7.363
ϵ-greedy	6.080	9.845	3.352	5.333	5.181	8.482	2.689	4.509	7.591	13.009	4.006	6.87
LinUCB	**6.391**	10.250	3.419	**5.519**	4.996	8.381	2.689	4.466	**8.113**	14.085	4.221	**7.376**
GLM-Lin	6.369	**10.253**	**3.427**	5.472	**5.367**	**8.862**	**2.815**	**4.719**	7.834	13.569	4.145	7.265
GLM-Sig	6.363	10.236	3.424	5.432	5.156	8.375	2.718	4.494	8.081	**14.094**	4.199	7.353
Improvement	4.8%	4.3%	2.0%*	3.3%*	6.4%	9.6%	6.2%*	7.7%*	8.7%*	10.7%*	6.4%*	7.9%*

are changing over time. To do this, we first divide the rating records of the users (whose ratings are more than 120) into two periods (set 1 and set 2). Then, we employ the genre information of the items as an indication of the user interest. That is, we calculate the cosine similarity between the genre vectors of the two periods. We choose the users with the smallest cosine similarity as an indication that they have significant interest drifts across the two time periods. All the other users with their ratings compose the training set. We only conduct experiments on the MovieLens and EachMovie datasets, as there is no movie genre information for the Netflix dataset.

Adaptability to Taste Drift

In order to test how the system can catch the users' taste drift, we conduct the empirical experiment as follows: for each user, in the first period with 60 interactions, we use set 1 as the groundtruth of the test users; and then, from the 61st interaction, the groundtruth is changed from set 1 to set 2 to simulate the process of the user's taste drift. Table 3.4 presents the results of our proposed algorithms compared to the baselines on the datasets, respectively. Because we focus on the performance when the user has changed the interest, only the results for $T \geq 60$ are shown.

From the results, it can be seen that the proposed algorithms outperform the baselines for both datasets. When compared with PMF, the improvement is up to 7.6% on the MovieLens dataset, and 17.5% on the EachMovie dataset. Among the proposed algorithms, LinUCB, GLM-Lin and GLM-Sig perform better than TS, which are similar to the results for the cold-start experiments.

3.4.6 Top-N Ranking Performance

We also conduct an experiment with multiple item slots at each interaction. The ranking-aware measure nNDCG is used to test the performance. The test users are the same as the ones in the cold-start setting. The only difference is that the number of interactions is reduced since the number of recommended items at each interaction increases. The results are shown in Table 3.5.

A similar trend is shown compared to the case of one item at each timestep: on MovieLens, either LinUCB or GLM-Lin performs the best, and on EachMovie, GLM-Lin always performs best. The results indicate that the algorithms still outperform the baselines in the multiple item setting. In addition, the performance on the NDCG measure suggests that our proposed algorithms are also capable regarding

Table 3.6: A case study of cold-start user #454 on Movielens. User feedback R: L-Like, D-Dislike, U-Unknown. Movie Genre Abbreviation: Ac-Action, Ad-Adventure, An-Animation, C-Comedy, CC-Children's Comedy, D-Drama, R-Romance, S-Scientific Fiction, T-Thriller, W-War.

T	R	Movies recommended by PMF	Genres	R	Movies recommended by LinUCB	Genres
1	L	Star Wars (1977)	Ac,Ad,R,S,W	L	Star Wars (1977)	Ac,Ad,R,S,W
2	L	Raiders of the Lost Ark (1981)	Ac,Ad	L	Raiders of the Lost Ark (1981)	Ac,Ad
3	L	Fargo (1996)	C,D,T	U	The Godfather (1972)	Ac,C,D
4	D	The Silence of the Lambs (1991)	D,T	D	The Silence of the Lambs (1991)	D,T
5	D	Dante's Peak (1997)	Ac,T	D	Return of the Jedi (1983)	Ac,Ad,R,S,W
6	U	Kika (1993)	D	D	The Empire Strikes Back (1980)	Ac,Ad,D,R,S,W
7	U	A Very Brady Sequel (1996)	C	L	Air Force One (1997)	Ac,T
8	U	Boomerang (1992)	C,R	U	Liar Liar (1997)	C
9	U	Black Sheep (1996)	C	U	Twelve Monkeys (1995)	D,S
10	L	The Saint (1997)	Ac,R,T	L	Contact (1997)	D,S
11	U	Kiss the Girls (1997)	C,D,T	D	Toy Story (1995)	An,CC
12	U	Batman (1989)	Ac,Ad,C,D	L	Braveheart (1995)	Ac,D,W
13	U	Matilda (1996)	CC	L	Titanic (1997)	Ac,D,R
14	D	Rock, The (1996)	Ac,Ad,T	L	Schindler's List (1993)	D,W
15	D	The Usual Suspects (1995)	C,T	L	The Shawshank Redemption (1994)	D

the CF ranking problems.

3.5 Case Studies

In order to better illustrate why the proposed algorithms outperform PMF, we present two case studies for a cold-start user and a taste-drift user respectively.

A Cold-start User Case

In Table 3.6, we present the first 15 sequentially recommended movies to a typical user #454 on Movielens, by PMF and LinUCB, and the corresponding feedback. From the results we can see that (i) LinUCB earns more "like" feedback and less "dislike" and "unknown" feedback. (ii) After the first three "likes", PMF keeps recommending action, crime and thriller movies, which is somewhat myopic. (iii) For LinUCB, after receiving the positive and negative feedback on action, war, thriller, and science fiction movies, it tries different genres such as drama, comedy and animation. After the next five interactions, LinUCB discovers the other interest in drama movies.

A Warm-start User with Taste Drift

In Figure 3.2, we show a typical taste-drift case of user #833 on Movielens. Specifically, 7 typical movie genres (out of 18) are involved here. The black bars show the user's taste drift by calculating the percentage difference of the normalised distributions on each genre between two time periods as in Section 3.4.5. The blue and orange bars show the percentage difference on each genre of the recommended items by LinUCB and PMF respectively. We see that LinUCB captures the user's taste drift in a way better than PMF: (i) for these genres, LinUCB captures the drift direction. For example, the user's interest in Comedy movies decreases (-4.3%[†]) between the two periods. LinUCB also recommends fewer (-3.2%)

[†]The percentage measures the change of the proportion of Comedy movies the user watched between the two periods.

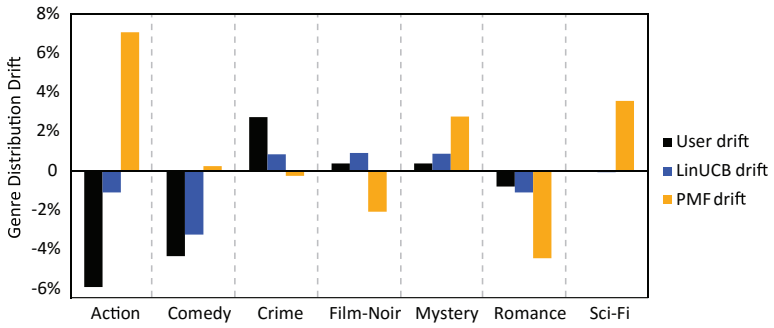


Figure 3.2: A case study on handling taste drift. Black, blue and orange histograms denote the changes in the genre distributions of items in the user groundtruth, recommended by LinUCB and recommended by greedy PMF respectively. LinUCB can partly catch the drift by adapting to the new genre distribution, but the greedy PMF approach cannot effectively reflect the taste drift in its recommendations.

Comedy movies, but PMF recommends more (+1.1%) Action movies to the user. (ii) For most genres, LinUCB to some extent captures the drift degree, e.g., the user has a 0.8% interest decrease on Romance movies and LinUCB also recommend 1.1% fewer Romance movies, but PMF dramatically decreases this type of movies by 4.4%.

3.6 Concluding Remarks

In this chapter, we introduced the ICF framework to solve the user cold-start problem. In this framework, two objectives, satisfying the user's information need and collecting the user's profile, are considered as an integrated goal of maximising the overall recommendation performance over a period of time.

Within the ICF framework, the PMF model is leveraged to capture the distributions of user and item feature vectors. Based on that, Thompson sampling and several EE algorithms are employed to balance between the exploitation and exploration aspects of the ICF problem. We conducted experiments in three situations: when a cold-start user joins the system, when a warm-start user's taste drifts, and when multiple items are recommended during each interaction. Throughout the experiments, we demonstrated that our proposed algorithms outperformed several strong baselines including the greedy PMF algorithm, the active learning approach and the interview process.

We theoretically focused on the case when only one item is recommended at each interaction. We also empirically addressed the case with multiple items at each interaction. However, to comprehensively consider multiple items, we need to have a more strict theoretical formulation of it, which leads to the research in the next chapter.

Chapter 4

Two-Stage Collaborative Filtering

In the previous chapter, we discussed the Interactive Collaborative Filtering (ICF) framework with a focus on the user cold-start problem. We defined the goal of the ICF framework as achieving maximal overall return over a period of time. We specifically considered the case where one item is recommended during each interaction with the user. Then, we related the problem to the multi-armed bandit problem, and solved it with various exploitation-exploration (EE) algorithms based on the matrix factorization (MF) model of collaborative filtering (CF).

In this chapter, we consider a two-stage recommendation process to address multiple recommendations during each interaction. The two-stage recommendation process can be used to tackle both user and item cold-start problems (see Figure 1.4). In the initial stage, we use a portion of recommendation allocations to estimate the new item's (user's) model (while also considering the new item (user)'s information need). After that, in the second stage, we use the remaining resources to make recommendations. Similar to ICF, the goal of this process is to maximise the overall feedback collected from the two stages. We focus on the item cold-start scenario in this chapter because the benefits of using a batch solution are more pronounced in this scenario as explained in Section 1.2.2. As the users and items can be modelled symmetrically [163, 30], the analysis can be easily applied to a user cold-start problem.

We first formulate the two-stage recommendation process into a partially observable Markov decision process (POMDP) to obtain its exact solution. Then, through an in-depth analysis of the POMDP value iteration solution, we identify that an exact solution can be abstracted as selecting resources that are not only highly relevant to the target according to the initial-stage information, but also highly correlated, either positively or negatively, with other *potential* resources for the next stage. With this finding, we propose an approximate solution to ease the intractability of the exact solution. Our initial results on synthetic data and the MovieLens 100K dataset confirm the performance gains and our theoretical analysis.

4.1 The Two-Stage Model

In this section, we formulate CF into the POMDP framework, which will lead us to the exact solution of our problem. A POMDP models a Markov decision process where the true current state of the system is partially unobservable [24]. In the scenario of the item cold-start recommendation, the true state is each

Table 4.1: Summary of key notations in Chapter 4.

Notation	Description
$\mathcal{U}$	The entire user set
$t \in \{1, 2\}$	The stage (timestep) of the process
m, n	The number of users to select at the initial stage and the second stage respectively
$\boldsymbol{u}, \boldsymbol{v}$	The users to choose in the initial stage and second stage respectively
$\setminus \boldsymbol{u}$	The users not selected in the initial stage, $\setminus \boldsymbol{u} = \mathcal{U} \setminus \boldsymbol{u}$
$\boldsymbol{R}$	The preferences (a random vector) of all users over the item under consideration
$\boldsymbol{R}_{\boldsymbol{u}}, \boldsymbol{R}_{\boldsymbol{v}}$	$\boldsymbol{R}$ partitioned by $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively
$\boldsymbol{r}_{\boldsymbol{u}}, \boldsymbol{r}_{\boldsymbol{v}}$	Feedback from $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively
$\boldsymbol{\theta}^{(t)}, \boldsymbol{\Phi}^{(t)}, \mathbf{C}^{(t)}$	The mean, covariance matrix, correlation matrix of $\boldsymbol{R}$ at time t (CU model)
$\rho_{i,j}^{(t)}$	Correlation between u and v at t
$\mathbf{P}$	The matrix with each row as a user vector (MF model)
$\boldsymbol{q}$	The target item's feature vector (MF model)
$\boldsymbol{\nu}^{(t)}, \boldsymbol{\Psi}^{(t)}$	The mean and covariance matrix of the item vector at time t (MF model)
T	The sampling number

user's genuine (potential) preference as to the new item, which is unknown for the users having not rated it. To model the decision process, we start with a correlated-user (CU) model as a probabilistic description of the memory-based models in CF [2, 56] and formulate it with POMDP. Then, we decompose the user-item rating matrix to gain its formulation in the domain of MF. We provide, for each model, the exact solution on how to select users optimally in order to collect maximal overall feedback from the users over two stages.

4.1.1 Correlated-User Model with POMDP

The CU model with POMDP (CU-POMDP) is depicted in Figure 4.1. Let us denote the available user pool as $\mathcal{U}$. For each new item that joins the system, the recommendation system should make the following decisions: in the initial stage, choose an initial m users to start with, collect their feedback, and update the system's belief state; and in the second stage, choose another n users to exploit the information gained from the initial stage. $N = m + n$ is the total number of users that the item is to be targeted to. For the reader's convenience, we provide a list of key notations used in this chapter in Table 4.1. We consider only one cold-start item, but the scenario is similar if multiple cold-start items are present.

Our goal is to find the optimal policy that can maximise the expected total ratings over two stages. To capture the relations between users' preferences, we model the preferences of all users, denoted by $\boldsymbol{R}$, to follow a multivariate Gaussian distribution

$$p^{(t)}(\boldsymbol{R}) \sim \mathcal{N}(\boldsymbol{\theta}^{(t)}, \boldsymbol{\Phi}^{(t)}),\ t \in \{1, 2\} \tag{4.1}$$

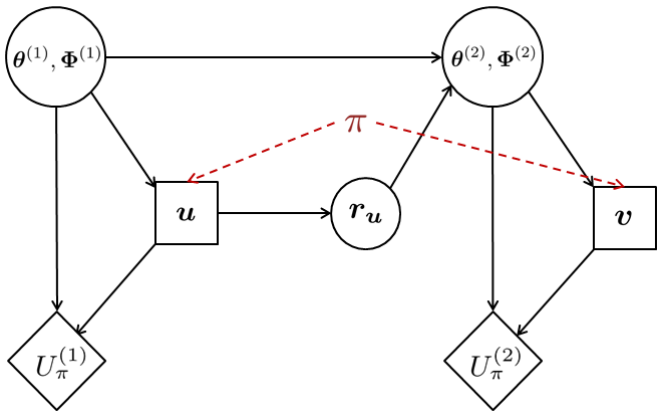


Figure 4.1: The two-stage CU-POMDP as illustrated by an influence diagram, with respect to the correlated-user model. Circular nodes are random variables and square nodes are the recommendation decision, and the rhombus nodes are the utility at each stage.

with its mean and covariance matrix as $\boldsymbol{\theta}^{(t)}$ and $\boldsymbol{\Phi}^{(t)}$. The distribution above is the system's belief over the true state $\boldsymbol{R}$ at each stage t, referred to as the belief state according to POMDP. By recommending the item to users and receiving their feedback, the belief state evolves from $p^{(1)}(\boldsymbol{R})$ to $p^{(2)}(\boldsymbol{R})$. Our problem is a POMDP because the true preferences $\boldsymbol{R}$ are unknown (or only partially known), but can be modelled through a distribution.

This model is non-trivial because it has utilised all user-user correlations via a multivariate Gaussian model. To obtain the belief state for the initial stage, we can impose an i.i.d. assumption on the users' preferences on different items. As such, $\boldsymbol{\theta}^{(1)}$ can be estimated by the users' mean ratings, and $\boldsymbol{\Phi}^{(1)}$ can be estimated by the user-user covariances on previously co-rated items. To emphasise the role of user-user correlation, in the following, we also make use of the following representation

$$\begin{aligned}\boldsymbol{\Phi}^{(1)} &= \mathrm{Dg}[\boldsymbol{\Phi}^{(1)}]^{1/2}\mathbf{C}^{(1)}\mathrm{Dg}[\boldsymbol{\Phi}^{(1)}]^{1/2} \\ &= \mathrm{diag}[\boldsymbol{\phi}^{(1)}]\mathbf{C}^{(1)}\mathrm{diag}[\boldsymbol{\phi}^{(1)}] \end{aligned} \tag{4.2}$$

where $\mathrm{Dg}(\boldsymbol{\Phi}^{(1)})$ denotes the diagonal matrix with the same diagonal elements of $\boldsymbol{\Phi}^{(1)}$, $\boldsymbol{\phi}^{(1)}$ denotes the vector formed by the users' standard deviations of ratings ($\boldsymbol{\phi}^{(1)} = \mathrm{diag}[\mathrm{Dg}^{1/2}(\boldsymbol{\Phi}^{(1)})]$), and $\mathbf{C}^{(1)}$ is the correlation matrix whose element $\rho_{u,v}^{(1)}$ is the correlation between user u and user v.

A policy π is defined to make the decision at each stage on the basis of the available information:

$$\boldsymbol{u} = \pi(\boldsymbol{\theta}^{(1)}, \boldsymbol{\Phi}^{(1)}, \mathcal{U}), \text{ and} \tag{4.3}$$

$$\boldsymbol{v} = \pi(\boldsymbol{\theta}^{(2)}, \boldsymbol{\Phi}^{(2)}, \mathcal{U}\backslash\boldsymbol{u}), \tag{4.4}$$

where we use vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ to denote the user selection decisions for the two stages respectively ($|\boldsymbol{u}| = m$ and $|\boldsymbol{v}| = n$). Similar to the last chapter, here we use the same constraint that the target item should not be recommended repeatedly to the same user. Therefore, the available user pool will be the remaining users $\mathcal{U}\backslash\boldsymbol{u}$ for the second stage. The total expected ratings collected at each stage is

the element-wise summation of the expected rating vector of each selection, which we refer to as reward $U_\pi^{(t)}$

$$U_\pi^{(1)} = \mathbb{E}^{(1)}[\mathbf{1}^T \boldsymbol{R}_{\boldsymbol{u}}], \tag{4.5}$$

$$U_\pi^{(2)} = \mathbb{E}^{(2)}[\mathbf{1}^T \boldsymbol{R}_{\boldsymbol{v}}]. \tag{4.6}$$

We use $\boldsymbol{R}_{\boldsymbol{u}}$ ($\boldsymbol{R}_{\boldsymbol{v}}$) to denote the random vector $\boldsymbol{R}$ partitioned by user selections $\boldsymbol{u}$ ($\boldsymbol{v}$). We will use the same partition rule throughout this chapter.

The objective is to find a policy of selecting users such that the expected *total* reward of the two stages are maximised

$$\pi^* = \arg\max_\pi \left(U_\pi^{(1)} + U_\pi^{(2)}\right). \tag{4.7}$$

Belief Update

Let us consider the problem in a reverse order. Suppose the system has already recommended the item to m users in the initial stage and received feedback $\boldsymbol{r}_{\boldsymbol{u}}$. Given the feedback, the system can update its belief state on the remaining users $\mathcal{U}\backslash\boldsymbol{u}$ (simplified as $\backslash\boldsymbol{u}$) by the conditional multivariate Gaussian distribution, conditioned on the observations

$$p^{(2)}(\boldsymbol{R}_{\backslash\boldsymbol{u}}) \sim \mathcal{N}(\boldsymbol{\theta}^{(2)}_{\backslash\boldsymbol{u}}, \boldsymbol{\Phi}^{(2)}_{\backslash\boldsymbol{u},\backslash\boldsymbol{u}}), \text{ where} \tag{4.8}$$

$$\boldsymbol{\theta}^{(2)}_{\backslash\boldsymbol{u}} = \boldsymbol{\theta}^{(1)}_{\backslash\boldsymbol{u}} + \boldsymbol{\Phi}^{(1)}_{\backslash\boldsymbol{u},\boldsymbol{u}}[\boldsymbol{\Phi}^{(1)}_{\boldsymbol{u},\boldsymbol{u}}]^{-1}(\boldsymbol{r}_{\boldsymbol{u}} - \boldsymbol{\theta}^{(1)}_{\boldsymbol{u}}) \tag{4.9}$$

$$\boldsymbol{\Phi}^{(2)}_{\backslash\boldsymbol{u},\backslash\boldsymbol{u}} = \boldsymbol{\Phi}^{(1)}_{\backslash\boldsymbol{u},\backslash\boldsymbol{u}} - \boldsymbol{\Phi}^{(1)}_{\backslash\boldsymbol{u},\boldsymbol{u}}[\boldsymbol{\Phi}^{(1)}_{\boldsymbol{u},\boldsymbol{u}}]^{-1}\boldsymbol{\Phi}^{(1)}_{\boldsymbol{u},\backslash\boldsymbol{u}}. \tag{4.10}$$

To gain insight with the view of correlated users, we reformulate the update functions with the correlation matrix $\mathbf{C}^{(1)}$ as follows. According to Eq. (4.2), we obtain

$$[\boldsymbol{\Phi}^{(1)}_{\boldsymbol{u},\boldsymbol{u}}]^{-1} = \text{diag}[\boldsymbol{\phi}^{(1)}_{\boldsymbol{u}}]^{-1}[\mathbf{C}^{(1)}_{\boldsymbol{u},\boldsymbol{u}}]^{-1}\text{diag}[\boldsymbol{\phi}^{(1)}_{\boldsymbol{u}}]^{-1}, \text{ and} \tag{4.11}$$

$$[\boldsymbol{\Phi}^{(1)}_{\backslash\boldsymbol{u},\boldsymbol{u}}] = \text{diag}[\boldsymbol{\phi}^{(1)}_{\backslash\boldsymbol{u}}]\mathbf{C}^{(1)}_{\backslash\boldsymbol{u},\boldsymbol{u}}\text{diag}[\boldsymbol{\phi}^{(1)}_{\boldsymbol{u}}]. \tag{4.12}$$

Substituting Eqs. (4.12) and (4.11) into (4.9) we further get

$$\boldsymbol{\theta}^{(2)}_{\backslash\boldsymbol{u}} = \boldsymbol{\theta}^{(1)}_{\backslash\boldsymbol{u}} + \text{diag}[\boldsymbol{\phi}^{(1)}_{\backslash\boldsymbol{u}}]\mathbf{C}^{(1)}_{\backslash\boldsymbol{u},\boldsymbol{u}}[\mathbf{C}^{(1)}_{\boldsymbol{u},\boldsymbol{u}}]^{-1}\text{diag}[\boldsymbol{\phi}^{(1)}_{\boldsymbol{u}}]^{-1}(\boldsymbol{r}_{\boldsymbol{u}} - \boldsymbol{\theta}^{(1)}_{\boldsymbol{u}}) \tag{4.13}$$

Particularly, if we assume equal rating variance for all users, and disregard the correlations among $\boldsymbol{u}$ such that $\mathbf{C}^{(1)}_{\boldsymbol{u},\boldsymbol{u}}$ becomes an identity matrix, then Eq. (4.13) reduces to a weighted summation of the observed ratings centred by their prior expectations $\boldsymbol{r}_{\boldsymbol{u}} - \boldsymbol{\theta}^{(1)}_{\boldsymbol{u}}$, with the weights as the correlations between unobserved users and observed users

$$\boldsymbol{\theta}^{(2)}_{\backslash\boldsymbol{u}} = \boldsymbol{\theta}^{(1)}_{\backslash\boldsymbol{u}} + \mathbf{C}^{(1)}_{\backslash\boldsymbol{u},\boldsymbol{u}}(\boldsymbol{r}_{\boldsymbol{u}} - \boldsymbol{\theta}^{(1)}_{\boldsymbol{u}}). \tag{4.14}$$

Eq. (4.14) looks very familiar to us because it simulates the popular memory-based (user-based) CF algorithm (see Eq. (2.2) in Chapter 2), which takes the neighbours' ratings regarding the target item, centres them by the mean ratings of the neighbours, and estimates the target user's preference regarding this item as their weighted summation [2], where Pearson correlation is commonly used to calculate the weights [107]. We thus see the user-based recommendation heuristic as an approximation of our CU model.

From the above formula we can see that: (i) by observing users $\boldsymbol{u}$ in the initial stage, the expectations of unobserved users are also updated; (ii) the covariances (correlations) between observed and unobserved users act as the bridge through which feedback from selected users can update our belief regarding other users.

Exact Solution

To obtain the exact solution, consider $V^*(\boldsymbol{\theta}^{(t)}, \boldsymbol{\Phi}^{(t)}, \mathcal{T})$ which is the maximally achievable expected total future reward with current information $\boldsymbol{\theta}^{(t)}$, $\boldsymbol{\Phi}^{(t)}$ and remaining steps ($\mathcal{T} = 1, 2$). With the updated belief according to Eq. (4.8) already *given*, the optimal expected reward for the second stage is simply a greedy approach:

$$\begin{aligned} V^*_{\text{CU}}(\boldsymbol{\theta}^{(2)}, \boldsymbol{\Phi}^{(2)}, 1) &= \max_{\pi} U^{(2)}_{\pi} \\ &= \max_{\boldsymbol{v} \subset \mathcal{U} \setminus \boldsymbol{u}} \mathbb{E}^{(2)}[\mathbf{1}^T \boldsymbol{R}_{\boldsymbol{v}}] \\ &= \max_{\boldsymbol{v} \subset \mathcal{U} \setminus \boldsymbol{u}} \mathbf{1}^T \boldsymbol{\theta}^{(2)}_{\boldsymbol{v}}. \end{aligned} \tag{4.15}$$

By working backwards the total maximal expected reward for two stages can be obtained as

$$\begin{aligned} & V^*_{\text{CU}}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\Phi}^{(1)}, 2) = \max_{\pi}(U^{(1)}_{\pi} + U^{(2)}_{\pi}) \\ =& \max_{\boldsymbol{u} \subset \mathcal{U}} \left(\mathbb{E}^{(1)}[\mathbf{1}^T \boldsymbol{R}_{\boldsymbol{u}} + V^*_{\text{CU}}(\boldsymbol{\theta}^{(2)}, \boldsymbol{\Phi}^{(2)}, 1)] \right) \\ =& \max_{\boldsymbol{u} \subset \mathcal{U}} \left(\mathbb{E}^{(1)}[\mathbf{1}^T \boldsymbol{R}_{\boldsymbol{u}}] + \int p^{(1)}(\boldsymbol{R}_{\boldsymbol{u}} = \boldsymbol{r}_{\boldsymbol{u}}) V^*_{\text{CU}}(\boldsymbol{\theta}^{(2)}, \boldsymbol{\Phi}^{(2)}, 1) d\boldsymbol{r}_{\boldsymbol{u}} \right). \end{aligned} \tag{4.16}$$

Substituting Eqs. (4.15) and (4.9) into (4.16) we reach the exact solution obtained by value iteration:

$$\begin{aligned} V^*_{\text{CU}}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\Phi}^{(1)}, 2) = \max_{\boldsymbol{u} \subset \mathcal{U}} \Bigg\{ & \overbrace{\mathbf{1}^T \boldsymbol{\theta}^{(1)}_{\boldsymbol{u}}}^{\text{exploitation}} + \\ & \underbrace{\int p^{(1)}(\boldsymbol{R}_{\boldsymbol{u}} = \boldsymbol{r}_{\boldsymbol{u}}) \max_{\boldsymbol{v} \subset \mathcal{U} \setminus \boldsymbol{u}} \left[\mathbf{1}^T \left(\boldsymbol{\theta}^{(1)}_{\boldsymbol{v}} + \boldsymbol{\Phi}^{(1)}_{\boldsymbol{v},\boldsymbol{u}} [\boldsymbol{\Phi}^{(1)}_{\boldsymbol{u},\boldsymbol{u}}]^{-1} (\boldsymbol{r}_{\boldsymbol{u}} - \boldsymbol{\theta}^{(1)}_{\boldsymbol{u}}) \right) \right] d\boldsymbol{r}_{\boldsymbol{u}}}_{\text{exploration}} \Bigg\}. \end{aligned} \tag{4.17}$$

Eq. (4.17) suggests that the merit of choosing users $\boldsymbol{u}$ at the initial stage lies in two components:

- **Exploitation**. It is the immediate expected reward, denoted by $\mathbf{1}^T \boldsymbol{\theta}^{(1)}_{\boldsymbol{u}}$, determined by the prior information on the users.
- **Exploration**. The exploration component shows how the feedback from users $\boldsymbol{u}$ can lead the system to find optimal selections with updated knowledge. Consider that the feedback deviates

from the prior information such that $(\boldsymbol{r_u} - \boldsymbol{\theta}_{\boldsymbol{u}}^{(1)}) \neq \mathbf{0}$, the updated belief state will then lead us to find users which bring "extra" returns via the term $\boldsymbol{\Phi}_{\boldsymbol{v},\boldsymbol{u}}^{(1)}[\boldsymbol{\Phi}_{\boldsymbol{u},\boldsymbol{u}}^{(1)}]^{-1}(\boldsymbol{r_u} - \boldsymbol{\theta}_{\boldsymbol{u}}^{(1)})$. No matter the deviation is positive or negative, we can always benefit from it by selecting corresponding optimal users in the second stage. As mentions above, this term relates to correlations between the users of the two stages. The larger the correlations are, the more the system can *gain* from the discrepancy between the observations and the prior information.

4.1.2 Matrix Factorization Model with POMDP

To gain insights from the formulation of latent factor models, consider MF with POMDP (MF-POMDP). For this purpose, we use the probabilistic model $\boldsymbol{R} = \mathbf{P}\boldsymbol{q} + \xi$ such that $\mathbf{P} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{|\mathcal{U}|})^T$ is a $|\mathcal{U}| \times K$ matrix containing the users' information, $\boldsymbol{q}$ is a K-dimensional item vector, and ξ is a random variable with zero mean and variance σ_0^2. If we assume fixed user vectors $\mathbf{P}$ and unknown item vector $\boldsymbol{q}$ (see Chapter 3), CU-POMDP is translated to a decision process under the belief state of the unobservable item vector (see Figure 4.2)

$$p^{(t)}(\boldsymbol{q}) \sim \mathcal{N}(\boldsymbol{\nu}^{(t)}, \boldsymbol{\Psi}^{(t)}), \tag{4.18}$$

where $\boldsymbol{\nu}^{(1)}$ and $\boldsymbol{\Psi}^{(1)}$ are the mean and covariance matrix of the item vector. The belief state over the item vector then determines the belief over the preferences of users

$$p^{(t)}(\boldsymbol{R}) \sim \mathcal{N}(\mathbf{P}\boldsymbol{\nu}^{(t)}, \mathbf{P}\boldsymbol{\Psi}^{(t)}\mathbf{P}^T + \sigma_0^2\mathbf{I}). \tag{4.19}$$

By observing users $\boldsymbol{u}$ with feedback $\boldsymbol{r_u}$ the belief state can be updated according to the Bayes rule

$$p^{(2)}(\boldsymbol{q}) \sim \mathcal{N}(\boldsymbol{\nu}^{(2)}, \boldsymbol{\Psi}^{(2)}), \text{ where} \tag{4.20}$$

$$\boldsymbol{\nu}^{(2)} = \boldsymbol{\nu}^{(1)} + \boldsymbol{\Psi}^{(1)}\mathbf{P}_{\boldsymbol{u}}^T(\mathbf{P}_{\boldsymbol{u}}\boldsymbol{\Psi}^{(1)}\mathbf{P}_{\boldsymbol{u}}^T + \sigma_0^2\mathbf{I})^{-1}(\boldsymbol{r_u} - \mathbf{P}_{\boldsymbol{u}}\boldsymbol{\nu}^{(1)}), \tag{4.21}$$

$$\boldsymbol{\Psi}^{(2)} = [(\boldsymbol{\Psi}^{(1)})^{-1} + \mathbf{P}_{\boldsymbol{u}}^T\mathbf{P}_{\boldsymbol{u}}/\sigma_0^2]^{-1}. \tag{4.22}$$

Thus,

$$\begin{aligned}\mathbb{E}^{(2)}(\boldsymbol{R}_{\setminus \boldsymbol{u}}|\boldsymbol{r_u}) &= \mathbf{P}_{\setminus \boldsymbol{u}}\boldsymbol{\nu}^{(2)} \\ &= \mathbf{P}_{\setminus \boldsymbol{u}}\boldsymbol{\nu}^{(1)} + \mathbf{P}_{\setminus \boldsymbol{u}}\boldsymbol{\Psi}^{(1)}\mathbf{P}_{\boldsymbol{u}}^T(\mathbf{P}_{\boldsymbol{u}}\boldsymbol{\Psi}^{(1)}\mathbf{P}_{\boldsymbol{u}}^T + \sigma_0^2\mathbf{I})^{-1}(\boldsymbol{r_u} - \mathbf{P}_{\boldsymbol{u}}\boldsymbol{\nu}^{(1)}).\end{aligned} \tag{4.23}$$

Comparing Eq. (4.23) with Eq. (4.9) we find a nice alignment between the two models. Actually, by dimension reduction the covariance between user u's and user v's ratings can be translated as

$$\Phi_{u,v}^{(1)} = \boldsymbol{p}_u^T\boldsymbol{\Psi}^{(1)}\boldsymbol{p}_v, \tag{4.24}$$

when σ_0^2 is very small compared to the covariance between the two users's true preferences ($\sigma_0^2 << \boldsymbol{p}_u^T\boldsymbol{\Psi}^{(1)}\boldsymbol{p}_v$). Eq. (4.24) has converted the statistical property (the covariance of preferences between the

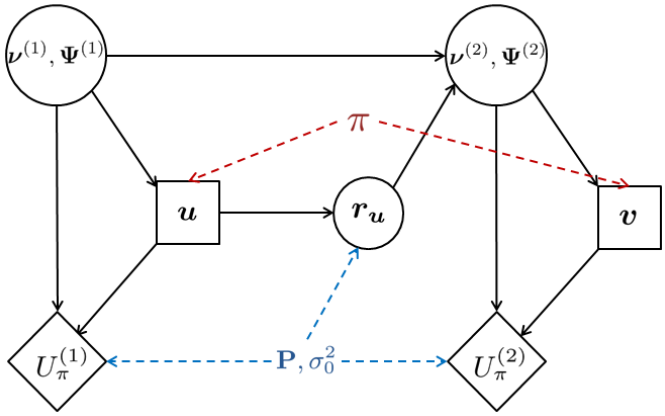


Figure 4.2: The two-stage MF-POMDP as illustrated by an influence diagram, with respect to the matrix factorization model.

two users) into the function of the feature vectors of the two users.

By the same token, we write the optimal value function for the MF-POMDP as

$$\begin{aligned} V^*_{\text{MF}}(\boldsymbol{\nu}^{(1)}, \boldsymbol{\Psi}^{(1)}, 2) = \max_{\boldsymbol{u}\subset\mathcal{U}} \Big\{ & \mathbf{1}^T \mathbf{P}_{\boldsymbol{u}} \boldsymbol{\nu}^{(1)} + \\ & \int p^{(1)}(\boldsymbol{R}_{\boldsymbol{u}} = \boldsymbol{r}_{\boldsymbol{u}}) \max_{\boldsymbol{v}\subset\mathcal{U}\setminus\boldsymbol{u}} \Big[\mathbf{1}^T \Big(\mathbf{P}_{\boldsymbol{v}} \boldsymbol{\nu}^{(1)} + \\ & \mathbf{P}_{\boldsymbol{v}} \boldsymbol{\Psi}^{(1)} \mathbf{P}_{\boldsymbol{u}}^T [\mathbf{P}_{\boldsymbol{u}} \boldsymbol{\Psi}^{(1)} \mathbf{P}_{\boldsymbol{u}}^T + \sigma_0^2 \mathbf{I}]^{-1} (\boldsymbol{r}_{\boldsymbol{u}} - \mathbf{P}_{\boldsymbol{u}} \boldsymbol{\nu}^{(1)}) \Big) d\boldsymbol{r}_{\boldsymbol{u}} \Big] \Big\}. \end{aligned} \tag{4.25}$$

4.1.3 A Toy Example

Let us look at a simple three-user case and its analytical solution. In this example, one user is selected in each stage. We base this example on the CU model so that the effect of user-user correlation can be illustrated more straightforwardly.

Suppose

$$\boldsymbol{\theta}^{(1)} = \begin{pmatrix} \theta_1^{(1)} \\ \theta_2^{(1)} \\ \theta_3^{(1)} \end{pmatrix}, \quad \boldsymbol{\Phi}^{(1)} = \begin{pmatrix} \Phi_{1,1}^{(1)} & \Phi_{1,2}^{(1)} & \Phi_{1,3}^{(1)} \\ \Phi_{2,1}^{(1)} & \Phi_{2,2}^{(1)} & \Phi_{2,3}^{(1)} \\ \Phi_{3,1}^{(1)} & \Phi_{3,2}^{(1)} & \Phi_{3,3}^{(1)} \end{pmatrix}.$$

Without loss of generality, we assume $\Phi_{1,3}^{(1)} > \Phi_{1,2}^{(1)} > \Phi_{2,3}^{(1)}$ (and ignore the case with equal covariance for now). Suppose user 1 is selected in the initial stage with the observation as r_1, the update for the second and the third users are,

$$\begin{aligned} \theta_2^{(2)}(r_1) &= \theta_2^{(1)} + \Phi_{2,1}^{(1)} (\Phi_{1,1}^{(1)})^{-1} (r_1 - \theta_1^{(1)}), \\ \theta_3^{(2)}(r_1) &= \theta_3^{(1)} + \Phi_{3,1}^{(1)} (\Phi_{1,1}^{(1)})^{-1} (r_1 - \theta_1^{(1)}). \end{aligned}$$

By introducing $z_1 = (r_1 - \theta_1^{(1)})/\sqrt{\Phi_{1,1}^{(1)}}$, the above updates become

$$\theta_2^{(2)}(z_1) = \theta_2^{(1)} + \Phi_{2,1}^{(1)}(\Phi_{1,1}^{(1)})^{-1/2} z_1,$$
$$\theta_3^{(2)}(z_1) = \theta_3^{(1)} + \Phi_{3,1}^{(1)}(\Phi_{1,1}^{(1)})^{-1/2} z_1.$$

We can see that both $\theta_2^{(2)}$ and $\theta_3^{(2)}$ are linear in z_1. The turning point between choosing user 2 and user 3 is obtained when the above two are equal to each other, which is at

$$d_1 = \frac{\theta_2^{(1)} - \theta_3^{(1)}}{\Phi_{3,1}^{(1)} - \Phi_{2,1}^{(1)}} \sqrt{\Phi_{1,1}^{(1)}}.$$

Since $\Phi_{3,1}^{(1)} > \Phi_{2,1}^{(1)}$, if $z_1 > d_1$, user 3 should be selected whereas if $z_1 < d_1$ user 2 should be selected in the second stage. Thus, the optimal reward when choosing user 1 at the initial stage is

$$\begin{aligned}
&V_{u=1}^*(\theta^{(1)}, \Phi^{(1)}, 2) = \\
&\quad \theta_1^{(1)} + \int p^{(1)}(r_1) \cdot \max_{v=2,3} \left(\theta_v^{(1)} + \Phi_{v,1}^{(1)} (\Phi_{1,1}^{(1)})^{-1} (r_1 - \theta_1^{(1)}) \right) dr_1 \\
&\quad = \theta_1^{(1)} + \int_{-\infty}^{d_1} p^{(1)}(z_1) \left[\theta_2^{(1)} + \Phi_{2,1}^{(1)} (\Phi_{1,1}^{(1)})^{-1/2} z_1 \right] dz_1 \\
&\qquad + \int_{d_1}^{\infty} p^{(1)}(z_1) \left[\theta_3^{(1)} + \Phi_{3,1}^{(1)} (\Phi_{1,1}^{(1)})^{-1/2} z_1 \right] dz_1 \\
&\quad = \theta_1^{(1)} + 1/2(\theta_2^{(1)} + \theta_3^{(1)}) + 1/2(\theta_2^{(1)} - \theta_3^{(1)}) \mathbf{erf}(\frac{d_1}{\sqrt{2}}) \\
&\qquad - \frac{1}{\sqrt{2\pi}} \frac{\Phi_{2,1}^{(1)} - \Phi_{3,1}^{(1)}}{\sqrt{\Phi_{1,1}^{(1)}}} \mathbf{e}^{-\frac{d_1^2}{2}}.
\end{aligned}$$

Similarly,

$$\begin{aligned}
&V_{u=2}^*(\theta^{(1)}, \Phi^{(1)}, 2) = \\
&\quad \theta_2^{(1)} + \int p^{(1)}(r_2) \cdot \max_{v=3,1} \left(\theta_v^{(1)} + \Phi_{v,2}^{(1)} (\Phi_{2,2}^{(1)})^{-1} (r_2 - \theta_2^{(1)}) \right) dr_2 \\
&\quad = \theta_2^{(1)} + \int_{-\infty}^{d_2} p^{(1)}(z_2) \left[\theta_3^{(1)} + \Phi_{3,2}^{(1)} (\Phi_{2,2}^{(1)})^{-1/2} z_2 \right] dz_2 \\
&\qquad + \int_{d_2}^{\infty} p^{(1)}(z_2) \left[\theta_1^{(1)} + \Phi_{1,2}^{(1)} (\Phi_{2,2}^{(1)})^{-1/2} z_2 \right] dz_2 \\
&\quad = \theta_2^{(1)} + 1/2(\theta_3^{(1)} + \theta_1^{(1)}) + 1/2(\theta_3^{(1)} - \theta_1^{(1)}) \mathbf{erf}(\frac{d_2}{\sqrt{2}}) \\
&\qquad - \frac{1}{\sqrt{2\pi}} \frac{\Phi_{3,2}^{(1)} - \Phi_{1,2}^{(1)}}{\sqrt{\Phi_{2,2}^{(1)}}} \mathbf{e}^{-\frac{d_2^2}{2}},
\end{aligned}$$

$$
\begin{aligned}
&V^*_{u=3}(\theta^{(1)},\Phi^{(1)},2) = \\
&\quad \theta_3^{(1)} + \int p^{(1)}(r_3)\cdot \max_{v=1,2}\left(\theta_v^{(1)} + \Phi_{v,3}^{(1)}(\Phi_{3,3}^{(1)})^{-1}(r_3-\theta_3^{(1)})\right) dr_3 \\
&\quad = \theta_3^{(1)} + \int_{-\infty}^{d_3} p^{(1)}(z_3)\left[\theta_2^{(1)} + \Phi_{2,3}^{(1)}(\Phi_{3,3}^{(1)})^{-1/2} z_3\right] dz_3 \\
&\qquad + \int_{d_3}^{\infty} p^{(1)}(z_3)\left[\theta_1^{(1)} + \Phi_{1,3}^{(1)}(\Phi_{3,3}^{(1)})^{-1/2} z_3\right] dz_3 \\
&\qquad = \theta_3^{(1)} + 1/2(\theta_1^{(1)} + \theta_2^{(1)}) + 1/2(\theta_2^{(1)} - \theta_1^{(1)})\mathbf{erf}(\frac{d_3}{\sqrt{2}}) \\
&\qquad - \frac{1}{\sqrt{2\pi}}\frac{\Phi_{2,3}^{(1)} - \Phi_{1,3}^{(1)}}{\sqrt{\Phi_{3,3}^{(1)}}}\mathbf{e}^{-\frac{d_3^2}{2}},
\end{aligned}
$$

where

$$
d_2 = \frac{\theta_3^{(1)} - \theta_1^{(1)}}{\Phi_{1,2}^{(1)} - \Phi_{3,2}^{(1)}}\sqrt{\Phi_{2,2}^{(1)}},\ d_3 = \frac{\theta_2^{(1)} - \theta_1^{(1)}}{\Phi_{1,3}^{(1)} - \Phi_{2,3}^{(1)}}\sqrt{\Phi_{3,3}^{(1)}}.
$$

Note that the above formula are not rotational symmetric due to the asymmetry caused by $\Phi_{1,3}^{(1)} > \Phi_{1,2}^{(1)} > \Phi_{2,3}^{(1)}$.

To illustrate the results, let us look at a numerical example according to the above solutions. Suppose

$$
\boldsymbol{\theta}^{(1)} = \begin{pmatrix} 3.2 \\ 2.5 \\ 3.5 \end{pmatrix},\ \boldsymbol{\Phi}^{(1)} = \begin{pmatrix} 1.6 & 0.25 & 1.6 \\ 0.25 & 3.2 & 0.20 \\ 1.6 & 0.20 & 3.5 \end{pmatrix}.
$$

The correlation matrix is thus

$$
\mathbf{C}^{(1)} = \begin{pmatrix} 1 & 0.11 & 0.68 \\ 0.11 & 1 & 0.06 \\ 0.68 & 0.06 & 1 \end{pmatrix}.
$$

When user 1 is selected at the initial stage:

$$
\begin{aligned}
\theta_2^{(2)}(r_1) &= \theta_2^{(1)} + \Phi_{2,1}^{(1)}(\Phi_{1,1}^{(1)})^{-1}(r_1 - \theta_1^{(1)}) \\
&= 2.5 + 0.25 \times (1.6)^{-1}(r_1 - 3.2), \\
\theta_3^{(2)}(r_1) &= \theta_3^{(1)} + \Phi_{3,1}^{(1)}(\Phi_{1,1}^{(1)})^{-1}(r_1 - \theta_1^{(1)}) \\
&= 3.5 + 1.6 \times (1.6)^{-1}(r_1 - 3.2).
\end{aligned}
$$

Therefore, when $r_1 < 2.01$ we should choose user 2 in the second stage whilst when $r_1 > 2.01$ we should choose user 3 (when $r_1 = 2.01$ choosing either will give the same expected reward in the second stage). The corresponding value function is

$$
\begin{aligned}
&V^*_{u=1}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\Phi}^{(1)}, 2) \\
&= \theta_1^{(1)} + \int p^{(1)}(r_1) \cdot \max_{j=2,3} \left(\theta_v^{(1)} + \Phi_{j,1}^{(1)} (\Phi_{1,1}^{(1)})^{-1} (r_1 - \theta_1^{(1)}) \right) dr_1 \\
&= 3.2 + \int_{-\infty}^{2.01} p^{(1)}(r_1)(2.5 + 0.25 \times (1.6)^{-1}(r_1 - 3.2)) dr_1 \\
&+ \int_{2.01}^{+\infty} p^{(1)}(r_1)(3.5 + 1.60 \times (1.6)^{-1}(r_1 - 3.2)) dr_1 \\
&\approx 6.80.
\end{aligned}
$$

Similarly, we can obtain the value functions for choosing user 2 and 3 at the initial stage

$$
\begin{aligned}
V^*_{u=2}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\Phi}^{(1)}, 2) &\approx 5.7, \\
V^*_{u=3}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\Phi}^{(1)}, 2) &\approx 6.77.
\end{aligned}
$$

And thus obtain the final value function

$$
V^*(\boldsymbol{\theta}^{(1)}, \boldsymbol{\Phi}^{(1)}, 2) = \max(6.80, 5.7, 6.77) = 6.80.
$$

We can see that the value function favours the first user at the first step, even though the prior information about the users favours the third user over the first user. Due to the fact that user 1 is highly correlated to user 3, and is more correlated with user 2 than user 3 is, choosing user 1 at the initial stage will enable the system to judge better in the second stage which results in a higher total expected reward over the two stages.

4.1.4 Computational Complexity

The exact solution of a finite-horizon POMDP has been proven to be PSPACE-complete [37]. In our case, the decision space at the initial stage is $C_m^{|\mathcal{U}|}$. For each decision, the m-dimensional observation space will be divided into $C_n^{|\mathcal{U}|-m}$ regions, each region corresponds to a (possibly) different optimal user combination to choose for the second stage. That is, the exact solution suggested by the value iteration algorithm requires going through all the possible decisions and all possible observations, which is intractable.

4.2 Approximation

To ease the intractability of the exact solution, we propose an approximation solution here, named guided exploitation-exploration (GEE). We provide its form for both the CU model and the MF model below.

4.2.1 Approximation for CU-POMDP

From Section 4.1.1, we have seen that the merit of selecting a group of users lies both in the immediate reward term (the exploitation part of Eq. (4.17)) and in how it can guide the system to find promising users in the next stage through the system update (the exploration part of Eq. (4.17)). However, when

Algorithm 4.1: CU-GEE by Sampling

Require: Prior mean ratings $\boldsymbol{\theta}^{(1)}$, covariance matrix $\boldsymbol{\Phi}^{(1)}$, GEE parameter λ, available users $\mathcal{U}$
  Initialise $\boldsymbol{u}^* \leftarrow \emptyset$
  for $t = 1 \dots T$ **do**
    Sample $\boldsymbol{u}_t$ ($|\boldsymbol{u}_t| = m$) from $\mathcal{U}$
    Calculate $V_{\boldsymbol{u}_t}^{\text{CU-GEE}}$ according to Eq. (4.29)
    if $V_{\boldsymbol{u}_t}^{\text{CU-GEE}}$ is the largest so far **then**
      Update $\boldsymbol{u}^* \leftarrow \boldsymbol{u}_t$
    end if
  end for

the decision of the initial stage is made, the system's belief state update is unknown before receiving any observations. To investigate the influence of selecting users $\boldsymbol{u}$ only (before making any observations), let us consider the conditional distribution of unselected users $\backslash\boldsymbol{u}$ over the selection of users $\boldsymbol{u}$, $p(\boldsymbol{R}_{\backslash\boldsymbol{u}}|\boldsymbol{u})$. Note that this conditional distribution is different from Eq. (4.8) because it is the distribution conditioned on the action $\boldsymbol{u}$ instead of the observations, as at the initial-decision stage these observations are still unknown.

Because the observations are not made yet, the expected feedback conditioned on the selection remains unchanged

$$\mathbb{E}[\boldsymbol{R}_{\backslash\boldsymbol{u}}|\boldsymbol{u}] = \boldsymbol{\theta}^{(1)}_{\backslash\boldsymbol{u}}. \tag{4.26}$$

However, its covariance changes according to the choice of $\boldsymbol{u}$:

$$\begin{aligned}\operatorname{Cov}[\boldsymbol{R}_{\backslash\boldsymbol{u}}|\boldsymbol{u}] &= \operatorname{Cov}\left[\boldsymbol{\theta}^{(1)}_{\boldsymbol{u}} + \boldsymbol{\Phi}^{(1)}_{\backslash\boldsymbol{u},\boldsymbol{u}}(\boldsymbol{\Phi}^{(1)}_{\boldsymbol{u},\boldsymbol{u}})^{-1}(\boldsymbol{R}_{\boldsymbol{u}} - \boldsymbol{\theta}^{(1)}_{\boldsymbol{u}})\right] \\ &= \boldsymbol{\Phi}^{(1)}_{\backslash\boldsymbol{u},\boldsymbol{u}}(\boldsymbol{\Phi}^{(1)}_{\boldsymbol{u},\boldsymbol{u}})^{-1}\operatorname{Cov}(\boldsymbol{R}_{\boldsymbol{u}})(\boldsymbol{\Phi}^{(1)}_{\boldsymbol{u},\boldsymbol{u}})^{-1}\boldsymbol{\Phi}^{(1)}_{\boldsymbol{u},\backslash\boldsymbol{u}} \\ &= \boldsymbol{\Phi}^{(1)}_{\backslash\boldsymbol{u},\boldsymbol{u}}(\boldsymbol{\Phi}^{(1)}_{\boldsymbol{u},\boldsymbol{u}})^{-1}\boldsymbol{\Phi}^{(1)}_{\boldsymbol{u},\backslash\boldsymbol{u}},\end{aligned} \tag{4.27}$$

where the last step is due to $\operatorname{Cov}(\boldsymbol{R}_{\boldsymbol{u}}) = \boldsymbol{\Phi}^{(1)}_{\boldsymbol{u},\boldsymbol{u}}$.

Therefore, with the initial-stage users as $\boldsymbol{u}$, the expected returns at the second stage by choosing users $\boldsymbol{v}$ are bounded by the interval $\Theta_{\boldsymbol{u},\boldsymbol{v}}$:

$$\begin{aligned}\Theta_{\boldsymbol{u},\boldsymbol{v}} = \Bigg[&\mathbf{1}^T\left(\boldsymbol{\theta}^{(1)}_{\boldsymbol{v}} - \lambda\cdot\operatorname{diag}\left[\operatorname{Dg}^{-\frac{1}{2}}\left(\operatorname{Cov}(\boldsymbol{R}_{\boldsymbol{v}}|\boldsymbol{u})\right)\right]\right), \\ &\mathbf{1}^T\left(\boldsymbol{\theta}^{(1)}_{\boldsymbol{v}} + \lambda\cdot\operatorname{diag}\left[\operatorname{Dg}^{-\frac{1}{2}}\left(\operatorname{Cov}(\boldsymbol{R}_{\boldsymbol{v}}|\boldsymbol{u})\right)\right]\right)\Bigg]\end{aligned} \tag{4.28}$$

with the probability at least $(1 - 2\mathrm{e}^{-\lambda^2/2})^n$ [164]*.

The GEE algorithm therefore optimistically assumes the highest return could be achieved within this interval [157]. And thus we choose the users $\boldsymbol{u}$ which can achieve the highest total ratings under this

*To be more exact, the conditional vector $\boldsymbol{R}_{\backslash\boldsymbol{u}}|\boldsymbol{u}$ is bounded in an ellipsoid. This form is obtained with an approximation of considering only the diagonal elements of $\operatorname{Cov}(\boldsymbol{R}_{\backslash\boldsymbol{u}}|\boldsymbol{u})$.

Algorithm 4.2: CU-GEE-I by Sampling

Require: Prior mean ratings $\boldsymbol{\theta}^{(1)}$, correlation matrix $\mathbf{C}^{(1)}$, GEE parameter λ', available users $\mathcal{U}$
  Initialise $\boldsymbol{u}^* \leftarrow \emptyset$
  for $t = 1 \dots T$ **do**
    Sample $\boldsymbol{u}_t$ ($|\boldsymbol{u}_t| = m$) from $\mathcal{U}$
    Calculate $V_{\boldsymbol{u}_t}^{\text{CU-GEE-I}}$ according to Eq. (4.32)
    if $V_{\boldsymbol{u}_t}^{\text{CU-GEE-I}}$ is the largest so far **then**
      Update $\boldsymbol{u}^* \leftarrow \boldsymbol{u}_t$
    end if
  end for

assumption

$$\begin{aligned}
&\pi_{\text{CU-GEE}}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\Phi}^{(1)}, \mathcal{U}) \\
&= \arg\max_{\boldsymbol{u}\subset\mathcal{U}} \left\{ \mathbf{1}^T \boldsymbol{\theta}_{\boldsymbol{u}}^{(1)} + \max_{\boldsymbol{v}\subset\mathcal{U}\setminus\boldsymbol{u}} \mathbf{1}^T \left(\boldsymbol{\theta}_{\boldsymbol{v}}^{(1)} + \right.\right. \\
&\quad \left.\left. \lambda \cdot \operatorname{diag}\left[\operatorname{Dg}^{-\frac{1}{2}} \left(\boldsymbol{\Phi}_{\boldsymbol{v},\boldsymbol{u}}^{(1)} (\boldsymbol{\Phi}_{\boldsymbol{u},\boldsymbol{u}}^{(1)})^{-1} \boldsymbol{\Phi}_{\boldsymbol{u},\boldsymbol{v}}^{(1)} \right) \right] \right) \right\}. \qquad (4.29)
\end{aligned}$$

This algorithm suggests that, in order to determine the users for stage one, we first calculate the immediate reward based on the prior information. Then we calculate the optimistic reward when acting optimally in the second stage. We call GEE *guided* as the initial-stage decision is optimistically guided by pseudo optimal user selections in the next stage. By inspecting into the next stage, we utilise the correlation between users of the two stages, which will be explained further in Section 4.2.1. To implement this algorithm, we can adopt a sampling-based method depicted in Algorithm 4.1.

Independent Intra-Stage User Assumption

To align our algorithm with the popular memory-based CF, we adopt the correlation function Eq. (4.2) and reformulate Eq. (4.29) as follows:

$$\begin{aligned}
&\boldsymbol{\Phi}_{\boldsymbol{v},\boldsymbol{u}}^{(1)} (\boldsymbol{\Phi}_{\boldsymbol{u},\boldsymbol{u}}^{(1)})^{-1} \boldsymbol{\Phi}_{\boldsymbol{u},\boldsymbol{v}}^{(1)} \\
&= \left[\operatorname{diag}(\boldsymbol{\phi}_{\boldsymbol{v}}^{(1)}) \mathbf{C}_{\boldsymbol{v},\boldsymbol{u}}^{(1)} \operatorname{diag}(\boldsymbol{\phi}_{\boldsymbol{u}}^{(1)})\right] \left[\operatorname{diag}^{-1}(\boldsymbol{\phi}_{\boldsymbol{u}}^{(1)}) (\mathbf{C}_{\boldsymbol{u},\boldsymbol{u}}^{(1)})^{-1} \operatorname{diag}^{-1}(\boldsymbol{\phi}_{\boldsymbol{u}}^{(1)})\right] \left[\operatorname{diag}(\boldsymbol{\phi}_{\boldsymbol{u}}^{(1)}) \mathbf{C}_{\boldsymbol{u},\boldsymbol{v}}^{(1)} \operatorname{diag}(\boldsymbol{\phi}_{\boldsymbol{v}}^{(1)})\right] \\
&= \operatorname{diag}(\boldsymbol{\phi}_{\boldsymbol{v}}^{(1)}) \mathbf{C}_{\boldsymbol{v},\boldsymbol{u}}^{(1)} (\mathbf{C}_{\boldsymbol{u},\boldsymbol{u}}^{(1)})^{-1} \mathbf{C}_{\boldsymbol{u},\boldsymbol{v}}^{(1)} \operatorname{diag}(\boldsymbol{\phi}_{\boldsymbol{v}}^{(1)}). \qquad (4.30)
\end{aligned}$$

Eq. (4.29) thus becomes

$$\begin{aligned}
&\pi_{\text{CU-GEE}'}(\boldsymbol{\theta}^{(1)}, \boldsymbol{\phi}^{(1)}, \mathbf{C}^{(1)}, \mathcal{U}) \\
&= \arg\max_{\boldsymbol{u}\subset\mathcal{U}} \left\{ \mathbf{1}^T \boldsymbol{\theta}_{\boldsymbol{u}}^{(1)} + \max_{\boldsymbol{v}\subset\mathcal{U}\setminus\boldsymbol{u}} \mathbf{1}^T \left(\boldsymbol{\theta}_{\boldsymbol{v}}^{(1)} + \right.\right. \\
&\quad \left.\left. \lambda \cdot \operatorname{diag}\left[\operatorname{Dg}^{-\frac{1}{2}} \left(\operatorname{diag}(\boldsymbol{\phi}_{\boldsymbol{v}}^{(1)}) \mathbf{C}_{\boldsymbol{v},\boldsymbol{u}}^{(1)} (\mathbf{C}_{\boldsymbol{u},\boldsymbol{u}}^{(1)})^{-1} \mathbf{C}_{\boldsymbol{u},\boldsymbol{v}}^{(1)} \operatorname{diag}(\boldsymbol{\phi}_{\boldsymbol{v}}^{(1)}) \right) \right] \right) \right\}. \qquad (4.31)
\end{aligned}$$

The term of $(\mathbf{C}_{\boldsymbol{u},\boldsymbol{u}}^{(1)})^{-1}$ in the above equation suggests us to diversify the items in the initial stage. Here in order to catch the more important relation between the two stages, we assume the initial-stage

Algorithm 4.3: MF-GEE by Sampling

Require: Prior mean $\boldsymbol{\nu}^{(1)}$ and covariance matrix $\boldsymbol{\Psi}^{(1)}$ of the target item feature vector, GEE parameter λ, available users $\mathcal{U}$
Initialise $\boldsymbol{u}^* \leftarrow \emptyset$
for $t = 1 \dots T$ **do**
  Sample $\boldsymbol{u}_t$ ($|\boldsymbol{u}_t| = m$) from $\mathcal{U}$
  Calculate $V_{\boldsymbol{u}_t}^{\text{MF-GEE}}$ according to Eq. (4.34)
  if $V_{\boldsymbol{u}_t}^{\text{MF-GEE}}$ is the largest so far **then**
    Update $\boldsymbol{u}^* \leftarrow \boldsymbol{u}_t$
  end if
end for

users $\boldsymbol{u}$ are independent of each other, which suggests an already-diversified user list. We will intensively discuss the diversification problem in the next two Chapters. In addition to the independent assumption, we also impose an equal variance assumption, i.e., all the users have the same variance ϕ'^2 (so $\text{diag}(\boldsymbol{\phi}_{\boldsymbol{v}}^{(1)}) = \phi'\mathbf{I}$). With the two assumptions, Eq. (4.31) can be further approximated to

$$\begin{aligned}&\pi_{\text{CU-GEE-I}}(\boldsymbol{\theta}^{(1)}, \mathbf{C}^{(1)}, \mathcal{U}) \\ &= \arg\max_{\boldsymbol{u}\subset\mathcal{U}} \left[\sum_{\alpha=1}^{m} \theta_{u_\alpha}^{(1)} + \max_{\boldsymbol{v}\subset\mathcal{U}\setminus\boldsymbol{u}} \sum_{\beta=1}^{n} \left(\theta_{v_\beta}^{(1)} + \lambda' \sqrt{\sum_{\alpha=1}^{m} (\rho_{u_\alpha,v_\beta}^{(1)})^2} \right) \right],\end{aligned} \tag{4.32}$$

where $\lambda' = \lambda\phi'$, and $\rho_{u_\alpha,v_\beta}^{(1)}$ is just the correlation between u_α and v_β according to the prior information. The effect of inter-stage user-user correlations is shown clearly in the above formula. According to Eq. (4.32), given the user selection at the initial stage $\boldsymbol{u}$, we can foresee the optimistic return in the next stage through highly expected values (via $\theta_{v_\beta}^{(1)}$) and also highly correlated users (via the term $\sqrt{\sum_{\alpha=1}^{m}(\rho_{u_\alpha,v_\beta}^{(1)})^2}$). Identifying these users then guides the system to determine the user selection $\boldsymbol{u}^*$.

The sampling method for this algorithm is illustrated in Algorithm 4.2.

4.2.2 Approximation for MF-POMDP

With the MF model, the conditional covariance matrix of $\boldsymbol{R}_{\setminus\boldsymbol{u}}$ given the user selection $\boldsymbol{u}$ is written as

$$\text{Cov}(\boldsymbol{R}_{\setminus\boldsymbol{u}}|\boldsymbol{u}) = \mathbf{P}_{\setminus\boldsymbol{u}}\boldsymbol{\Psi}^{(1)}\mathbf{P}_{\boldsymbol{u}}^T(\mathbf{P}_{\boldsymbol{u}}\boldsymbol{\Psi}^{(1)}\mathbf{P}_{\boldsymbol{u}}^T + \sigma_0^2\mathbf{I})^{-1}\mathbf{P}_{\boldsymbol{u}}\boldsymbol{\Psi}^{(1)}\mathbf{P}_{\setminus\boldsymbol{u}}^T. \tag{4.33}$$

Following the same reasoning as in Section 4.2.1, we give the formulation for the matrix factorization model

$$\begin{aligned}&\pi_{\text{MF-GEE}}(\boldsymbol{\nu}^{(1)}, \boldsymbol{\Psi}^{(1)}, \mathcal{U}) \\ &= \arg\max_{\boldsymbol{u}\subset\mathcal{U}} \left\{ \mathbf{1}^T\mathbf{P}_{\boldsymbol{u}}\boldsymbol{\nu}^{(1)} + \max_{\boldsymbol{v}\subset\mathcal{U}\setminus\boldsymbol{u}} \mathbf{1}^T \left(\mathbf{P}_{\boldsymbol{v}}\boldsymbol{\nu}^{(1)} + \lambda\cdot \right.\right. \\ &\left.\left. \text{diag}\left[\text{Dg}^{-\frac{1}{2}} \left(\mathbf{P}_{\boldsymbol{v}}\boldsymbol{\Psi}^{(1)}\mathbf{P}_{\boldsymbol{u}}^T(\mathbf{P}_{\boldsymbol{u}}\boldsymbol{\Psi}^{(1)}\mathbf{P}_{\boldsymbol{u}}^T + \sigma_0^2\mathbf{I})^{-1}\mathbf{P}_{\boldsymbol{u}}\boldsymbol{\Psi}^{(1)}\mathbf{P}_{\boldsymbol{v}}^T \right) \right] \right) \right\}\end{aligned} \tag{4.34}$$

The corresponding algorithm is shown in Algorithm 4.3.

Algorithm 4.4: MF-GEE-I by Sampling

Require: Prior mean $\boldsymbol{\nu}^{(1)}$ and the diagonal element of the covariance matrix $\boldsymbol{\psi}^{(1)}$ of the target item feature vector, GEE parameter λ, available users $\mathcal{U}$
Initialise $\boldsymbol{u}^* \leftarrow \emptyset$
for $t = 1 \dots T$ **do**
  Sample $\boldsymbol{u}_t$ ($|\boldsymbol{u}_t| = m$) from $\mathcal{U}$
  Calculate $V_{\boldsymbol{u}_t}^{\text{MF-GEE-I}}$ according to Eq. (4.35)
  if $V_{\boldsymbol{u}_t}^{\text{MF-GEE-I}}$ is the largest so far **then**
    Update $\boldsymbol{u}^* \leftarrow \boldsymbol{u}_t$
  end if
end for

Independent Intra-Stage User Assumption

With the MF model, in addition to the independent intra-stage user assumption which turns $\mathbf{P}_{\boldsymbol{u}}\boldsymbol{\Psi}^{(1)}\mathbf{P}_{\boldsymbol{u}}^T$ into a diagonal matrix, we may also assume independent latent dimensions such that the prior covariance matrix is diagonal: $\boldsymbol{\Psi}^{(1)} = \text{diag}^2[\boldsymbol{\psi}^{(1)}]$, where $\boldsymbol{\psi}^{(1)}$ are the standard deviations of latent dimensions. Eq. (4.34) can be further simplified as:

$$\pi_{\text{MF-GEE-I}}(\boldsymbol{\nu}^{(1)}, \boldsymbol{\psi}^{(1)}, \mathcal{U}) = \arg\max_{\boldsymbol{u}\subset\mathcal{U}} \Bigg\{ \sum_{\alpha=1}^{m} \boldsymbol{p}_{u_\alpha}^T \boldsymbol{\nu}^{(1)} + \max_{\boldsymbol{v}\subset\mathcal{U}\setminus\boldsymbol{u}} \sum_{\beta=1}^{n} \left(\boldsymbol{p}_{v_\beta}^T \boldsymbol{\nu}^{(1)} + \lambda \sqrt{\sum_{\alpha=1}^{m} \frac{(\boldsymbol{p}_{v_\beta}^T \text{diag}^2[\boldsymbol{\psi}^{(1)}] \boldsymbol{p}_{u_\alpha})^2}{\boldsymbol{p}_{u_\alpha}^T \text{diag}^2[\boldsymbol{\psi}^{(1)}] \boldsymbol{p}_{u_\alpha} + \sigma_0^2}} \right) \Bigg\}. \tag{4.35}$$

The corresponding algorithm is shown in Algorithm 4.4.

Particularly, when assuming $\boldsymbol{\psi}^{(1)} = \psi^{(1)}\mathbf{1}$, i.e., equal prior standard deviation (variance) along different dimensions, we gain the form

$$\pi_{\text{MF-GEE-II}}(\boldsymbol{\nu}^{(1)}, \psi^{(1)}, \mathcal{U}) = \arg\max_{\boldsymbol{u}\subset\mathcal{U}} \Bigg\{ \sum_{\alpha=1}^{m} \boldsymbol{p}_{u_\alpha}^T \boldsymbol{\nu}^{(1)} + \max_{\boldsymbol{v}\subset\mathcal{U}\setminus\boldsymbol{u}} \sum_{\beta=1}^{n} \left(\boldsymbol{p}_{v_\beta}^T \boldsymbol{\nu}^{(1)} + \lambda \sqrt{\sum_{\alpha=1}^{m} \frac{((\psi^{(1)})^2 \boldsymbol{p}_{v_\beta}^T \boldsymbol{p}_{u_\alpha})^2}{(\psi^{(1)})^2 \boldsymbol{p}_{u_\alpha}^T \boldsymbol{p}_{u_\alpha} + \sigma_0^2}} \right) \Bigg\}. \tag{4.36}$$

Actually, with such a spherical prior variance, Eq. (4.24) becomes $\Phi_{u,v}^{(1)} = (\psi^{(1)})^2 \boldsymbol{p}_u^T \boldsymbol{p}_v$, i.e., the covariance between u and v is proportional to the inner product of the user latent factors. Actually, with a spherical prior variance, the correlation between user u and v, $\rho_{u,v}$, is proportional to $\boldsymbol{p}_u^T \boldsymbol{p}_v$, corresponding to the MF obtained by a regularised linear regression estimation [107].

4.3 Comparisons to Other EE Methods

4.3.1 Comparison to Active Learning

As mentioned in Chapter 2, active learning (AL) methods have been adopted to handle cold-start problems in recommender systems [107, 35, 33, 108], which are also referred to as optimal design by statisticians [109]. AL uses a limited number of items (usually much smaller than the total number of available items) to present to the target user to review, and then learns the user's profile based on the users' feed-

back on these items. The criterion for selection is usually represented by a statistical measure such as achieving minimal mean squared error in the model estimation (A-optimality criterion) [34], minimal 2-norm of the inverse of the information matrix (E-optimality criterion) [33] or minimal determinant of resulting covariance matrix of the system (D-optimality criterion) [33]. This objective differs from our objective function, and thus leads to significant differences from our approach.

There are two main differences between AL and our GEE approach. First, AL techniques such as D-Optimal design [33], A-Optimal design [34] and their applications to the cold-start item problem have divided exploration and exploitation into two separate stages. In the exploration stage, a small number of training points are selected for the system to learn, and in the exploitation stage the gained information is fully exploited. However, the returns (or regrets) collected from the exploration stage are not considered. In other words, The objective is imposed onto only the exploitation stage, and thus the trade-off between exploration and exploitation is not modeled [107]. For example, in [34], a budget has been imposed on the number of users to select at the experimental stage, and these users' returns are excluded from the objective function.

Second, the goal of AL is usually measured statistically using a global criterion. The criterion can be, for example, (to minimise) the mean square error of the estimates [34], or, (to maximise) the differential Shannon information [33]. However, from Eqs. (4.17) and (4.25) and from the example, we can see that the exact solution is achieved by prioritising the learning process towards the promising users of the next stage. Therefore, it is not necessary to achieve a global optimum. On the contrary, GEE captures this feature and make decisions guided by potential users of the second stage.

4.3.2 Comparison to UCB methods

The EE problem has been intensively studied in the literature of multi-armed bandit problems, where an agent decides dynamically which arm to choose at each step bearing the objective to maximise the total reward collected during a period of time [29]. Gittins has provided an optimal solution under the condition that only one arm at a time can evolve [23], but this is intractable in practice. UCB seeks a bounded regret instead of optimality and is used to balance the exploitation and exploration in practice [29, 22, 126, 157]. In UCB, usually a decision is made based on both the expectation and uncertainty of the return of individual choices at each step. In Chapter 3, we proposed several UCB-based algorithms for a multiple-stage interactive recommendation process. And recently GP-UCB algorithms have also been applied to solve the user cold-start problems interactively in recommender systems [165].

Our approach differs from UCB approaches in the following ways. First, UCB-based approaches seek to limit the regret within a bound, but they do not model how the specific selection within the bound can influence the outcome. In other words, EE achieved by UCB is not guided by the potential rewarding choice of the following stage, but is rather to limit the regret of the current stage. Second, UCB-based approaches are usually achieved in a long-term and interactive process, and may not be suitable for the two-stage process. Conversely, our algorithms are derived directly from the exact solutions of POMDP. They have directly considered the effect that choosing the initial-stage users has on the potential returns from the second stage.

Table 4.2: Total reward compared using synthetic data.

Algorithm	$N = 10$	$N = 20$	$N = 30$	$N = 40$
Greedy	19.084	38.919	52.517	60.55
AL	18.953	37.719	52.655	62.537
UCB	19.568	39.903	54.632	63.959
GEE	21.238	43.151	59.315	69.198
Improvement	8.5%	8.1%	8.6%	8.2%

4.4 Experiment

In this section, we compare our proposed approximate solutions with several baseline methods. To understand the model further and verify our theoretical analysis, we first present the results on synthetic data, and then on a real dataset.

4.4.1 Synthetic Data Experiment

Synthetic Data Generation

First, we define a 5-dimensional latent space and randomly generate a multivariate Gaussian distribution as the prior information of the cold-start item. In detail, each dimension of the multivariate Gaussian mean vector is generated randomly according to $\mathcal{N}(0, 0.1)$, and each dimension's standard deviation is generated according to $\mathcal{N}(0, 1)$. Then we generate 50 cold-start items according to this randomly-generated distribution. Second, we generate 100 users' vectors according to $\mathcal{N}(\mathbf{0}, \mathbf{I})$ as the available user pool for the 50 cold-start items to target to. Their real ratings are then produced according to Eq. (4.19) with the noise's standard deviation as 0.5. As such, we can obtain a 100×50 rating matrix as the groundtruth. The true prior information is then provided for each compared algorithm to perform recommendations. Finally, the above process is repeated for a total of 30 times, each time with a different prior information of the cold-start items. The results are then averaged over the different trials.

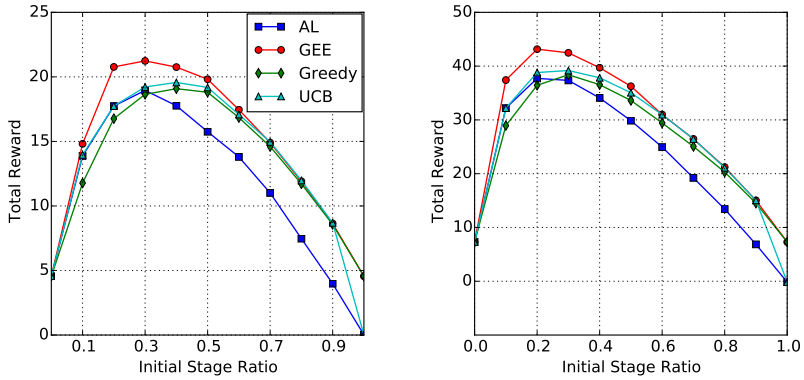
Compared Methods

We compare our proposed GEE algorithm to the following algorithms. (i) Greedy. Greedy method chooses the initial-stage users with the highest expected feedback. (ii) Active learning (AL). AL method chooses the users to minimise the uncertainty in the model, so that the users with the highest variances are chosen [35, 107]. (iii) Upper confidence bound (UCB). UCB method chooses the initial-stage users with the highest values calculated as the linear combination of the expected reward and the standard deviation [126]. All the algorithms select the second-stage users greedily after the system's state is updated with observations.

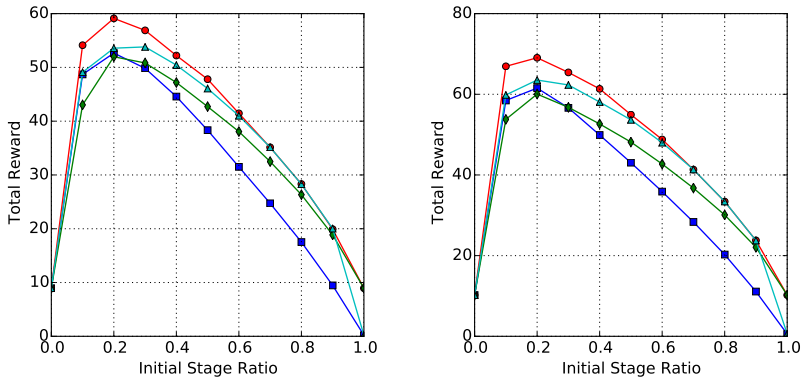
Results

The results are shown in Figure 4.3, with the evaluation measure as the total reward gained from the two stages. The result of the original GEE algorithm (Eq. (4.34)) is shown and we emphasise that the result of the GEE algorithm with the intra-stage independence assumption produces similar results.

From this figure, we can make the following observations. (i) For all the algorithms, the performance improves as m increases. This shows that by separating the recommendation process into two



(a) $N = 10$ (b) $N = 20$



(c) $N = 30$ (d) $N = 40$

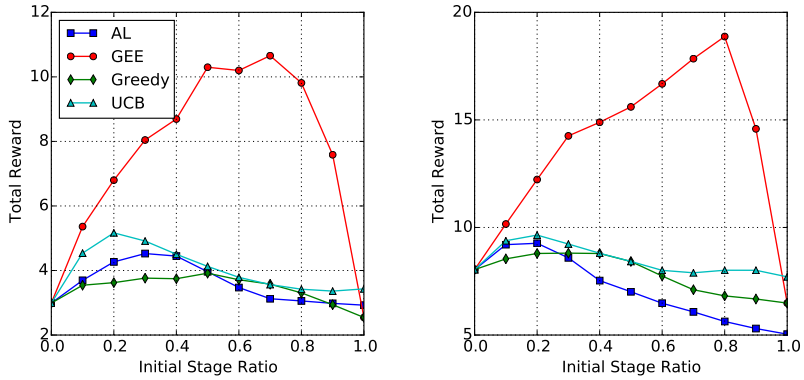
Figure 4.3: Total reward comparison of different algorithms on the synthetic data. The x-axis is m/N, the ratio of users to choose at the initial stage, and the y-axis is the total reward of both stages.

stages the performance can be greatly improved over a PRP-like once-for-all batch solution. (ii) For all the algorithms, the total reward increases more sharply than it drops after the performance peak. This phenomenon indicates that a small portion of allocation of users in the initial stage can significantly improve the overall performance. Note that in our synthetic data generation, we have used K=5, and the peak is also around $m = 5$. Therefore, the dimension of the latent factor model may be an indicator of the allocation ratio. The best result gained with optimal parameters of each algorithm is shown in Table 4.2.

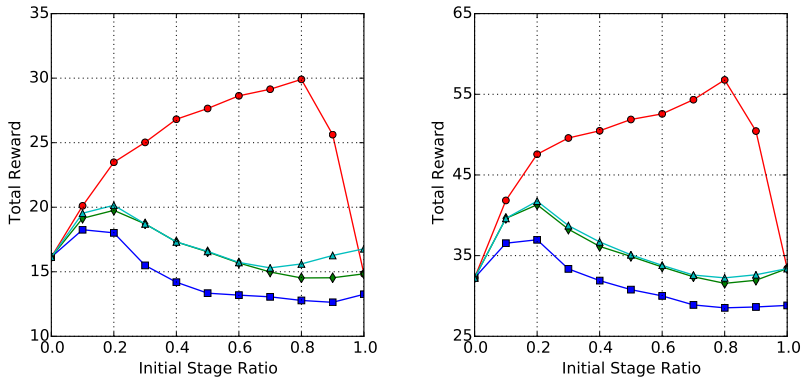
4.4.2 Experiments on the MovieLens Dataset

Experiment setup

As our study is a theoretical one, we use a relatively small research-based dataset MovieLens 100K, which has been described in Chapter 3. To conduct the experiment, we first divide the dataset into the training set and test set. For the sake of simulating cold-start item recommendations, we first randomly choose 200 items with sufficient numbers of ratings (at least 50) as the test cold-start items, and use their ratings as the groundtruth in the test dataset. The ratings between users and the remaining items are used to train the model. Similar to the synthetic data experiment, we compare our algorithms with Greedy, AL and UCB. After observing the feedback, the system updates according to the user-based



(a) $N = 10$ (b) $N = 20$



(c) $N = 40$ (d) $N = 80$

Figure 4.4: Total reward comparison of different algorithms on the MovieLens 100K data. The x-axis is m/N, the ratio of users to choose at the initial stage, and the y-axis is the total reward of both stages.

CF model suggested by Eq. (4.14). The results are evaluated by using both the total reward, and the total hit number – the total number of ratings equal or above 4 of the two stages (similar to the hit@2 as defined in the last chapter). To be consistent with what the user-based CF model suggests, we use the independent intra-user assumption for the GEE algorithm used.

Results

The results are shown in Figure 4.4, and Tables 4.3 and 4.4 with $N = 10, 20, 40$ and 80 respectively. Both the total reward and the total hit number measures are compared. Here the total hit number is defined as the total number of ratings collected which are 4 or above. We can see significant improvements over all four cases with the implementation of our algorithm. Similar to the synthetic experiment results, all algorithms show a peaking manner as m increases. From Tables 4.3 and 4.4 we can see that the improvements evaluated by using the total reward are even higher than the total hit number, which may be the result of targeting directly to the optimal reward in our objective function.

There is an apparent difference between the shapes of curves shown in Figure 4.4 and in Figure 4.3. We refer this different to the following two possible causes. (i) In a real recommendation system, the ratings' distributions may deviate from Gaussian which is used as the generative function for the synthetic

Table 4.3: Total reward compared on MovieLens.

Algorithm	$N = 10$	$N = 20$	$N = 40$	$N = 80$
Greedy	4.255	8.95	20.75	45.26
AL	4.705	9.91	21.715	41.665
UCB	5.38	10.2	21.715	45.26
GEE	12.125	19.48	31.05	60.97
Improvement	125.4%	91.0%	43.0%	34.7%

Table 4.4: Total hit number compared on MovieLens.

Algorithm	$N = 10$	$N = 20$	$N = 40$	$N = 80$
Greedy	0.845	1.745	4.045	8.73
AL	0.875	1.905	4.155	7.815
UCB	1.015	1.955	4.155	8.73
GEE	2.245	3.245	5.325	10.225
Improvement	121.2%	66.0%	28.2%	17.1%

test. However, even with a non-Gaussian dataset, the conclusion derived from our algorithm should still hold. (ii) In the experiments on MovieLens 100k, correlations between users are calculated based on the co-rated items between users. Since our algorithm considers both stages when selecting the initial-stage users, it automatically prefers those that have (either positive or negative) correlations with the potential second-stage users, to those whose ratings are not sufficient to make a correlation estimation. Thus this process naturally filters out those users who do not have sufficient ratings to have concurrent ratings with others. This may be the reason why the results continuously grow until the initial stage ratio gets very large in all panels of Figure 4.4, and yet, it reinforces our conclusion that we should choose the users with not only high expected ratings but also high correlations (positive or negative) with potential users to be selected in the second stage.

4.5 Concluding Remarks

In this chapter, we presented a two-stage CF process to address cold-start problems, with an item cold-start problem as a working example. We formulated the problem using a CU model and a PMF model, using POMDP in search of the exact solution for each. We found from analysing the exact solutions that the users to choose at the initial stage should be not only of high expected values, but also highly correlated with potential users in the next stage – a property that can guide the system to find promising users in the next stage. We proposed the approximate algorithm GEE based on this finding. And we conducted initial experiments using GEE and compared the results with several baseline algorithms on both a synthetic and a real dataset, which confirmed the effectiveness of our algorithm.

Since the algorithm is derived from approximating the exact solution based on a Gaussian model, the experimental results on the real dataset appear to be different from the synthetic data. However, the GEE algorithm prevails in both cases. This means that our conclusion – the users to choose in the initial

stage should be of both high expected value and high correlations with potential second-stage users holds regardless of the Gaussianity of data.

In the approximations, we especially focused on the inter-stage user correlations while assuming the intra-stage users are independent of each other (4.32). This assumption leads to significant simplifications of the GEE algorithms. In the next chapter, we will intensively discuss the effect of intra-stage correlations in regard to the diversification problem.

Chapter 5

Item Portfolio Diversification

In the previous two chapters, when considering multiple recommendations in one interaction step, we disregarded the correlations between the recommendations, resulting in several index-based algorithms in Chapter 3 (e.g., LinUCB, GLM-Lin and GLM-Sig), and several approximations (the proposed GEE algorithms) of the exact solution in Chapter 4. In this chapter, we specifically study the effect of correlations between items when multiple items are concerned.

When we consider correlations of items in a list, the diversification problem naturally emerges. We argue that the diversification level in a recommendation list should be adapted to the target user's individual situations and needs. For example, different users may have different ranges of interests – the preference of a highly focused user might include only few topics, whereas that of the user with broad interests may encompass a wide range of topics. Thus, the recommended items should be diversified according to the interest range of the target user. Also, the uncertainty of the estimated user preference model may vary significantly between users: different users may have provided different number of ratings – some have provided very few, such as cold-start users, whereas some have provide many. As a result, the recommended items should be diversified at a higher degree for the former, and a lower degree for the latter due to the different levels of risks in their user models. In general, the diversification should be tailored to each individual user's need.

In this chapter, we theoretically study the adaptive diversification problem. We start with commonly used latent factor models and reformulate them using the mean-variance analysis from the portfolio theory. The resulting latent factor portfolio (LFP) model captures the user's interest range and the uncertainty of the user preference by employing the distribution and variance of the learned user latent factors, respectively. Our mathematical derivation reveals that the need for diversification is not only due to the system's risk-aversion preference (non-adaptive), but, most importantly, due the target user's situation (adaptive). Our experiments confirm the theoretical insights and show that LFP succeeds in improving latent factor models by adaptively introducing recommendation diversity to fit the individual users' needs.

Though focused on the item diversification problem, due to the symmetric property of latent factor models, the arguments and methods in this chapter can be easily adapted to a user diversification scenario.

5.1 Uncertainty in Latent Factors

Similar to previous chapters, we adopt a latent factor model here, in which the rating $r_{u,i}$ from user u to item i can be expressed as the inner product of their latent factor vectors,

$$r_{u,i} = \boldsymbol{p}_u^T \boldsymbol{q}_i + \xi \tag{5.1}$$

where ξ is a noise factor with zero mean, $\boldsymbol{q}_i = (q_{i,1}, q_{i,2}, \ldots, q_{i,K})^T$ denotes the latent factor vector of the item, with each component $q_{i,k}$ as the extent to which the i-th item possesses the k-th latent factor, and $\boldsymbol{p}_u = (p_{u,1}, p_{u,2}, \ldots, p_{u,K})^T$ similarly denotes the latent factor vector of the user, with each component $p_{u,k}$ as the extent that user u is interested in the k-th latent factor. K is the number of latent factors that are employed in the model (the dimension of the user/item vector). For the reader's convenience, the key notations related to this chapter are summarised in Table 5.1.

Assuming that the representation of items is independent from individual users and that the latent item factors can be learned beforehand, in practice, we can regard the latent item factors as constants. Similar to the arguments in Chapter 3 and Chapter 4, we focus on the uncertainty of the target user factors (yet, the uncertainty of the item factors can be analogically derived). The expected value of a rating $r_{u,i}$ is thus expressed as:

$$\mathbb{E}(r_{u,i}) = \boldsymbol{q}_i^T \mathbb{E}(\boldsymbol{p}_u). \tag{5.2}$$

We also derive the variance of rating $r_{u,i}$ and the covariance between rating $r_{u,i}$ and $r_{u,i'}$ as follows:

$$\begin{aligned}
\mathrm{Var}(r_{u,i}) &= \mathbb{E}[r_{u,i} - \mathbb{E}(r_{u,i})]^2 \\
&= \mathbb{E}[\boldsymbol{q}_i^T (\boldsymbol{p}_u - \mathbb{E}[\boldsymbol{p}_u])]^2 \\
&= \mathbb{E}[\boldsymbol{q}_i^T (\boldsymbol{p}_u - \mathbb{E}[\boldsymbol{p}_u]) (\boldsymbol{p}_u - \mathbb{E}[\boldsymbol{p}_u])^T \boldsymbol{q}_i] \\
&= \sum_{k=1}^{K} q_{i,k}^2 \sigma_{u,k}^2, \qquad (5.3) \\
\mathrm{Cov}(r_{u,i}, r_{u,i'}) &= \mathbb{E}[(r_{u,i} - \mathbb{E}(r_{u,i})) (r_{u,i'} - \mathbb{E}(r_{u,i'}))] \\
&= \mathbb{E}[\boldsymbol{q}_i^T (\boldsymbol{p}_u - \mathbb{E}[\boldsymbol{p}_u]) (\boldsymbol{p}_u - \mathbb{E}[\boldsymbol{p}_u])^T \boldsymbol{q}_{i'}] \\
&= \sum_{k=1}^{K} q_{i,k} q_{i',k} \sigma_{u,k}^2, \qquad (5.4)
\end{aligned}$$

where $\sigma_{u,k}^2$ is the variance of the k-th latent factor of user u, i.e.,

$$\sigma_{u,k}^2 = \mathbb{E}[p_{u,k} - \mathbb{E}(p_{u,k})]^2. \tag{5.5}$$

The variance of each rating in terms of latent factors represents the uncertainty. Note that in the derivation of Eq. (5.3) and (5.4) we make use of the property that the user's interest in different latent factors are uncorrelated, i.e., $\mathbb{E}[(p_{u,k} - \mathbb{E}(p_{u,k}))(p_{u,l} - \mathbb{E}(p_{u,l}))] = 0, k \neq l$, which makes $\boldsymbol{p}_u$'s

Table 5.1: Summary of key notations in Chapter 5.

Notation	Description
K	The dimension of the latent space
N	The size of the recommendation list
$\boldsymbol{p}_u$, $\boldsymbol{q}_i$	User u's latent factor vector, item i's latent factor
$\sigma^2_{u,k}$	The variance of the user u's kth latent factor
$\mathcal{I}_u$	The set of items rated by user u
$j(n)$	The rank function that returns the item index at position n
$\boldsymbol{j} = (j(1), j(2), \ldots, j(N))$	The recommendation list
$P(\boldsymbol{j})$	The recommendation portfolio based on $\boldsymbol{j}$
w_n	The weight of the recommendation at position n
$R_{u,P(\boldsymbol{j})}$	The overall relevance of the recommendation list with respect to $P(\boldsymbol{j})$
$U[R_{u,P(\boldsymbol{j})}]$	u's utility over the portfolio $P(\boldsymbol{j})$
$U_n[R_{u,P(\boldsymbol{j})}]$	u's utility over the top n positions of the portfolio $P(\boldsymbol{j})$
b	The risk-reward trade-off parameter (system-level)

covariance matrix $(\boldsymbol{p}_u - \mathbb{E}[\boldsymbol{p}_u])\,(\boldsymbol{p}_u - \mathbb{E}[\boldsymbol{p}_u])^T$ a diagonal matrix $\text{diag}[\sigma^2_{u,1}, \sigma^2_{u,2}, \ldots, \sigma^2_{u,K}]$.

This property is a common assumption in latent factor models, in which each latent factor represents one aspect independent of all the others. We have two insights from the above formulation in Eqs. (5.3) and (5.4). First, as seen in Eq. (5.3), the variance (uncertainty) of the user preference score (the rating) is associated with the variance in the latent factors, indicating that taking into account the uncertainty in the latent factors could contribute to the modelling of the user preference. Second, as seen in Eq. (5.4), the covariance (proportionate to the correlation) between a user's preferences of two items is also associated to the variance of the latent factors, indicating that it is feasible to exploit the uncertainty of latent factors to regulate the recommended items in order to satisfy the user's demand of the diversity and coverage of recommended items.

Ideally, the variance of a latent user factor, e.g., $\sigma^2_{u,k}$, is estimated from a number of observations of $p_{u,k}$, which means that we need to sample the rated items from user u multiple times. However, this estimation could be infeasible in practice, since (1) multiple observations of user profiles are typically unavailable, thus requiring heuristic sampling strategy, and (2) it requires training the model multiple times according to different observations of user profiles, thus inflating the computational cost. We may also first model the user latent factors according to PMF to obtain the estimated variance from a probabilistic model. However, in order to address a wider category of latent factor models which may lack a probabilistic representation, we propose a heuristic calculation for the variance of each latent user factor, based on the latent factors of the items that have been rated by the user, as shown below:

$$\sigma^2_{u,k} = \frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} (p_{u,k} - q_{i,k})^2 \tag{5.6}$$

where $\mathcal{I}_u$ represents the set of items rated by user u, and $|\mathcal{I}_u|$ denotes its cardinality. Note that this

approximation satisfies our basic assumptions about the properties of the uncertainty. In the case that a user prefers two or more similar items, i.e., the items may be probably represented by some similar latent factors, the estimated variance with respect to those latent factors could be low. Conversely, if two rated items are quite different, i.e., the items may be probably represented by far different latent factors, the correspondent variance of the latent user factor could be high.

5.2 Latent Factor Portfolio Ranking

We will use vector $\boldsymbol{j}$ to denote the recommendation list. In our top-N recommendation task, we further denote $\boldsymbol{j} = (j(1), j(2), \ldots, j(N))$ where we have introduced a rank function $j(n)$ that returns the item index of the n-th item in the ranking list.

With weighting coefficients assigned to different ranking positions, denoted by w_n for position n, we define the list of recommendation items as a recommendation portfolio $P(\boldsymbol{j})$:

$$P(\boldsymbol{j}) = \{(j(1), w_1), (j(2), w_2), \ldots, (j(N), w_N)\}. \tag{5.7}$$

w_n is a monotonically decreasing function of the ranking position. The most common function for w_n is $w_n = 1/2^{n-1}$ [166], which is also used here.

We can then express the overall relevance (the user's preference over the ranked list) of the recommendation portfolio based on latent factors as below:

$$R_{u,P(\boldsymbol{j})} = \sum_{n=1}^{N} w_n r_{u,j(n)} = \sum_{n=1}^{N} w_n \sum_{k=1}^{K} q_{j(n),k} p_{u,k}, \tag{5.8}$$

where $R_{u,P(\boldsymbol{j})}$ denotes the overall relevance of the recommended list for user u.

5.2.1 From Factor Level to Rank Level

Taking into account Eqs. (5.2)$\sim$ (5.4), we obtain the expected value of the relevance of the ranked list as

$$\mathbb{E}[R_{u,P(\boldsymbol{j})}] = \sum_{n=1}^{N} w_n \sum_{k=1}^{K} q_{j(n),k} \mathbb{E}(p_{u,k}) \tag{5.9}$$

and the variance of the ranked list as:

$$\mathrm{Var}[R_{u,P(\boldsymbol{j})}] = \sum_{n=1}^{N} w_n^2 \sum_{k=1}^{K} q_{j(n),k}^2 \sigma_{u,k}^2 + \sum_{n=1}^{N} \sum_{\substack{m=1 \\ m \neq n}}^{N} w_n w_m \sum_{k=1}^{K} q_{j(n),k} q_{j(m),k} \sigma_{u,k}^2 \tag{5.10}$$

where, for the purpose of readability, we skip the detailed derivation from the topic variance to the rank list variance (see Appendix A to this Chapter). Note that the uncertainty of the recommendation list is represented by the variance in terms of latent factors. There are two insights from the two terms in Eq. (5.10).

The first term indicates that the uncertainty of a recommendation list is also top-biased. In other words, if the variance of a latent user factor, e.g., $\sigma_{u,k}^2$, is given, then the latent factor of top-ranked items

would have a larger influence on the uncertainty of the recommendation list than that of low-ranked items. In this sense, in order to reduce the uncertainty of the recommendation list, we need to rank an item relatively higher if its latent factor, e.g., $q_{j(n),k}$, is large and the variance of the corresponding latent user factor, i.e., $\sigma^2_{u,k}$, is low.

The second term indicates that the relative rank positions of any two items in the recommendation list influence the overall uncertainty. For example, if the variance of a latent user factor is large and the user has shown interest in an item whose corresponding latent factor is also large, then ranking another item with also a large corresponding latent factor at the higher position leads to larger uncertainty. Conversely, if the variance of a latent user factor is small, then ranking the two items higher would not lead to a large increase of the overall uncertainty. Note that the variances of all the latent user factors need to be taken into account for an aggregated impact on the overall uncertainty. In summary, it is evident that by exploiting the uncertainty of the latent factors, recommendation diversification can be attained adaptively.

5.2.2 Sequential Ranking

Following the portfolio theory of information retrieval (IR), we can attain an optimal recommendation list by taking into account the trade-off between the mean relevance of the recommended items and the corresponding variance. As a result, the utility function is expressed as:

$$U[R_{u,P(\boldsymbol{j})}] = \mathbb{E}[R_{u,P(\boldsymbol{j})}] - b\,\mathrm{Var}[R_{u,P(\boldsymbol{j})}] \tag{5.11}$$

in which b is a risk-reward trade-off parameter. As we shall see later, parameter b is a system level parameter and does not contribute to the adaptive adjustment of the diversification level. Instead, the diversification level in the ranked list will automatically be adjusted according to the uncertainty of the user factors and their distributions (in other words, it relies on how much we understand the target user from the provided ratings). By maximising this utility function, an optimal ranking can be achieved, which attains an optimal mean-variance balance. Here, we adopt the sequential ranking algorithm as used in [167] to solve the optimisation problem in Eq. 5.11. To do this, first we define the utility function of the top n positions as U_n. And then we can obtain the final item ranking rule at rank n as:

$$\begin{aligned} j^*(n) &= \arg\max_{j(n)} \{\Delta U_n(R_{u,P(\boldsymbol{j})})\} \\ &= \arg\max_{j(n)} \{U_n(R_{u,P(\boldsymbol{j})}) - U_{n-1}(R_{u,P(\boldsymbol{j})})\} \\ &= \arg\max_{j(n)} \Big\{ \sum_{k=1}^{K} \Big(q_{j(n),k} p_{u,k} - b w_n \sigma^2_{u,k} q^2_{j(n),k} - 2b\sigma^2_{u,k} \sum_{m=1}^{n-1} w_m q_{j(n),k} q_{j(m),k} \Big) \Big\}. \end{aligned} \tag{5.12}$$

Here, again, for readability, we leave the exact derivation to Appendix B to this Chapter. We have also dropped w_n from Appendix B since it is a constant for rank n.

Two Levels of Diversification

We call the above formulation *latent factor portfolio ranking*, since both the mean and the variance are defined based on latent factors of users and items, as indicated by the summation over the factor space k. The most important characteristic of LFP is the introduction of the variances of the latent factors $\sigma^2_{u,k}$, which introduces the adaptation. Combining $\sigma^2_{u,k}$ and b, topic diversity in the ranked list is adjusted in two levels:

- At the *system level*, the need for diversification is due to the risk-averse behaviour of the system, and it is adjusted at parameter b. As shown in [25], the risk-averse behaviour is query (user profile)-independent and related to the utility of the system, defined by the used IR metric.
- At the *user profile level*, the need for diversification is related to the level of absolute certainty about the latent topics that the target user is interested in. The uncertainty is represented by the variances $\sigma^2_{u,k}$ of the latent factors in the formula. Combined with b, it adaptively balances the mean and reward trade-off in the user profile level, thus adapting the topic diversification to each individual user's needs.

Comparison to Previous Work

From Eq. (5.12), we observe that: on one hand, compared to the latent factor models (e.g., [100] and [30]), LFP ranks an item at position n based on not only its rating predictions, i.e., the first term in Eq. (5.12), but also its uncertainty in terms of the latent item factors, i.e., the second term, and the correlation between this item and the items ranked before it, i.e., the third term. Therefore, we can regard LFP as a general extension of latent factor models by introducing the trade-off with respect to the uncertainty of recommended items. Note that in our derivation we only consider the uncertainty coming from user latent factors. One could also consider the randomness of both the user factors and items factors simultaneously; however, that is not the focus of this work, and we have therefore left its discussions to future work.

It is also of interest to specifically compare the formula in Eq. (5.12) to other adaptive diversification methods in text retrieval. In [49], the diversification trade-off of an unseen query was obtained by mapping it to the known queries whose optimal diversification level is known a priori. By contrast, our method is fully unsupervised and the diversification level is naturally adapted to the latent topics that the target user is interested in and also to how many of them we have already obtained in the ranked list. As shown by the first term of Eq. (5.12), an item is promoted if it has the same topic as the user's. However its rank score will be penalised if the same topic has already appeared in the lower ranks (see the product $q_{j(n),k}q_{j(m),k}$ in the third term). In [50], an intent-aware search result diversification method was proposed, where essentially the study was focused on the first term in our formula. In that approach, query aspect intents are classified into two categories (factors); informational and navigational, and a machine learning algorithm is used to rank documents with respect to the categories.

The other branch of research related to our work exploits portfolio theory for various information retrieval and recommendation tasks. The importance of such approaches has recently been underlined in

a talk by Resnick [168], who projects the usefulness and the necessity of portfolio theory in personalised systems. The application of portfolio theory in information retrieval and recommendation was first proposed by Wang et al. [167, 25]. Recent increasing attention to exploiting principles in economics for IR [155] may also fall under the same direction.

In the original portfolio retrieval formulation, the uncertainty about the overall relevance of a ranked list is linked to the co-variances between individual documents (relevancies) [167, 25]. However, as they are conditioned on a given query or user profile, exactly, how to obtain such a co-variance matrix remains an open question. In practice, the covariance between two relevance scores is approximated by the covariance between their document term occurrences or user ratings. Computationally, this approach is expensive because every document or item pair needs to be considered. In this chapter, we solve this issue by providing a better explanation of the correlation: document or items are correlated because of their underlying topics and latent factors. As shown in Eq. (5.12), LFP ranks items by taking into account item correlations based on their latent factors/topics, i.e., the products between item factors, which are not exploited in the original model. For example, when ranking a movie in position n, LFP (in the case of $b > 0$) would prefer a movie with a genre (assumed to be represented by a latent factor) that was not contained in the movies ranked before position n, in order to maximise Eq. (5.12). In this sense, we can regard LFP as a general extension of the original retrieval model, where when K = 1, LFP returns to the original retrieval model in [25].

5.3 Experimental Evaluation

In this section we present a series of experiments to evaluate the proposed adaptive diversification method. We specifically focus on the following aspects: (1) As discussed, user tastes have different scope and coverage, reflected by their rated items: some are more specific, while others have wider interests. The question is whether our method could adapt the diversification level to the taste of each user. (2) The number of rated items provided by users varies. As a result, we have different accuracy and uncertainty about the users' "true" taste. We intend to investigate whether our method could adjust the diversification level of the ranked list to this uncertainty. (3) If we consider the overall recommendation quality as an aggregated effect from both the relevance and the diversity, whether LFP could improve the overall recommendation quality.

5.4 Configurations

5.4.1 Dataset

The publicly available dataset MovieLens 1M is used in our experimental evaluation. The data set contains 1M ratings (scale 1–5) from about 6K users on about 3.7K movies items. We use this dataset instead of a larger dataset Netflix because Netflix does not have genre information. The data sparseness is 95.5%. Each user in the dataset has at least 20 ratings. In addition, the genre information of movies is provided. There are in total 18 genres, and the average number of genres per movies is 1.62. Note that our focus in this chapter is not on the performance comparison against the state-of-the-art baselines, but on investigating how the proposed method could diversify recommendation results for different types of user profiles (tastes). The choice of a moderate size dataset enables an efficient exploration of experimental

results under various settings.

We randomly separate users into two subsets, i.e., a training set containing 80% of the users, and a test set containing 20% of the users. For each user in the test set, we randomly select different user profile lengths (UPL), i.e., the number of rated items, together with the ratings in the training set for training the latent factor models, and use the remaining user rated items as the basis for evaluation.

5.4.2 Evaluation Metrics

In our experiments, we adopt Mean Average Precision (MAP) to measure the effectiveness of the ranked recommendation list. The average precision (AP) of a size-N list for the corresponding user is defined as

$$\text{AP} = \frac{\sum_{n=1}^{N} \text{P@}n}{N}, \tag{5.13}$$

where precision at n, P@n, is defined as the ratio of relevant items at rank n.

And MAP is its average over all users

$$\text{MAP} = \frac{1}{\#\text{users}} \sum_{\text{users}} \text{AP}. \tag{5.14}$$

In order to calculate MAP, we set the relevance threshold as rating 4. In other words, we regard items with ratings equal to or larger than 4 as relevant.

For the investigation of the trade-off (and combination) of the relevance and the diversity in Section 5.5.4, we utilise αNDCG, which summarises both the diversity and the quality of a ranked list [48]. We use movie genres as "nuggets" in calculating αNDCG. The exact definition of αNDCG is expressed below:

$$\alpha\text{NDCG@}N = \frac{\alpha\text{DCG@}N}{\alpha\text{IDCG@}N} \tag{5.15}$$

in which

$$\alpha\text{DCG@}N = \sum_{n=1}^{N} \frac{\sum_{l=1}^{L} J_{nl}^{u} (1-\alpha)^{\beta_{l,n-1}^{u}}}{log_2(1+n)} \tag{5.16}$$

J_{nl}^{u} is an indicator function that is equal to $r_{u,j(n)}$ (the rating of the n-th movie in the list for user u), if the n-th movie in the recommendation list of user u contains genre l, otherwise 0. $\beta_{l,n-1}^{u}$ denotes the number of movies ranked up to position $n-1$ that contain genre l in the recommendation list for user u. Therefore, with more movies ranked up to position $n-1$ that have already contained genre l, the less is $(1-\alpha)^{\beta_{l,n-1}^{u}}$, meaning that recurrence of the same genre is punished. α is a constant set to control the magnitude of the penalty for the redundancy of the recommended items. The value of α can be within the range $[0, 1]$, in which the higher value indicates the larger penalty. In our experiments, we use $\alpha = 0.5$ as a moderate choice for measuring diversity. αIDCG@N denotes the highest possible value of αDCG@N in the case that the top N recommendation list contains "ideally" diversified relevant items. Thus, αNDCG is normalised to be [0,1]. Note that αNDCG depends on both the movie ratings and

genres, representing a suitable metric for our purpose of evaluating the trade-off between the relevance and the diversity. Since we particularly focus on the top-ranked items in recommender systems, we use $N = 5$ in the experiments.

To solely measure the recommendation diversity in a ranked list, we also introduce a simple diversity measure called DNG@N. This measures the number of genres in the top-N ranked list. The number is discounted according to the position of the corresponding movie in order to consider the rank bias. Specifically, we define DNG@N as:

$$\text{DNG@}N = \sum_{n=1}^{N} w_n G(n) \tag{5.17}$$

in which $G(n)$ denotes the number of genres that the n-th movie has and that are not included in the top $n-1$ movies. w_n is a discount factor that is set as $w_n = 1/2^{n-1}$. Similar to αNDCG, we focus on the evaluation with $N = 5$. The reported DNG is an average across all the test users.

5.4.3 User Latent Factor Models

There are various ways of obtaining latent factors in Eq. (5.1), either by non-probabilistic approaches [30, 53, 57, 1] or from a probabilistic viewpoint [31, 100]. In our experiment, we choose a basic latent factor model from each of the two categories: PureSVD and PMF. The details of PureSVD have been introduced in Chapter 2. As for PMF, we use a variation of it which estimates the latent factor of users and items according to the following objective function:

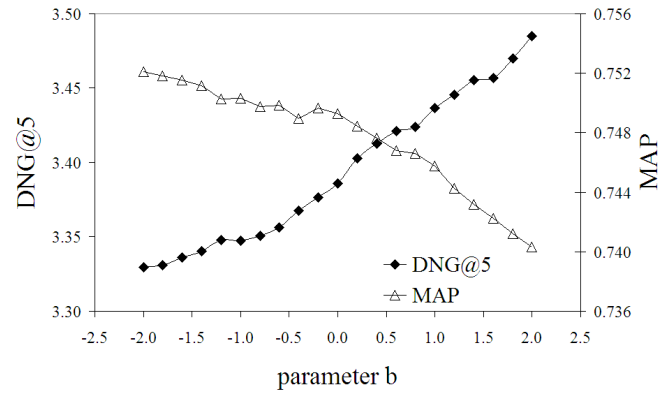
$$\mathbf{P}, \mathbf{Q} = \arg\min_{\mathbf{P},\mathbf{Q}} \frac{1}{2} \sum_u \sum_i \delta_{u,i} \left(r_{u,i} - g(\boldsymbol{p}_u^T \boldsymbol{q}_i)\right)^2 + \frac{\lambda_p}{2} \sum_u \|\boldsymbol{p}_u\|_F^2 + \frac{\lambda_q}{2} \sum_i \|\boldsymbol{q}_i\|_F^2 , \tag{5.18}$$

where $\delta_{u,i}$ is an indicator function equal to 1 when the rating $r_{u,i}$ is available and 0 otherwise. $||\cdot||_F^2$ denotes the Frobenius norm. The latent factors of users and items are learned from the user-item ratings (normalised to $[0, 1]$), and the magnitude of latent factors are penalised in order to alleviate overfitting. We introduce $g(x)$, a logistic function, i.e., $g(x) = 1/(1 + \mathbf{e}^{-x})$, which serves to bound the range of the inner product of latent factors. A simplification is usually made to set $\lambda = \lambda_p = \lambda_q$, which is also used here. In the following, we will use PureSVD or PMF to denote the latent factor model only, and use PureSVD+LFP and PMF+LFP to denote the corresponding LFP methods achieved based on the latent factor model.

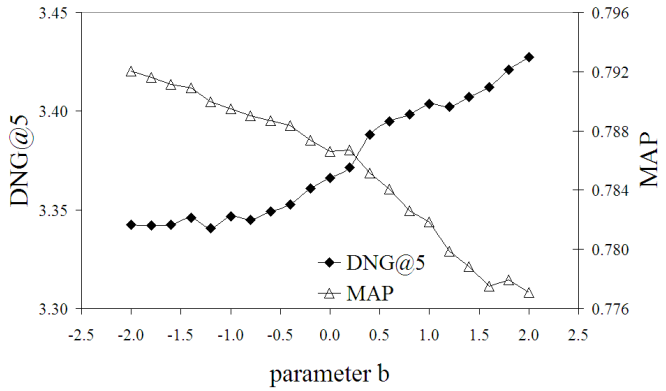
5.5 Results and Analysis

5.5.1 System-Level Diversity

As discussed, our LFP model implies that the need for diversification in a ranked list comes from two levels. Our first experiment is to investigate the system level diversity, which is controlled by the parameter b in Eq. (5.12). In our experiment, we use the training set to train the latent factor models, and for each user in the test set, we randomly select 2 rated items, i.e., User Profile Length (UPL=2), as user



(a) PureSVD+LFP



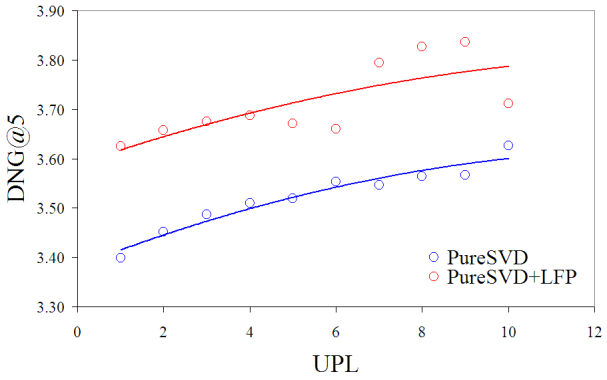
(b) PMF+LFP

Figure 5.1: The system level diversity: the impact of parameter b on DNG@5 and MAP. (a) When PureSVD is the used for obtaining the latent factors. (b) When PMF is the used for obtaining the latent factors.

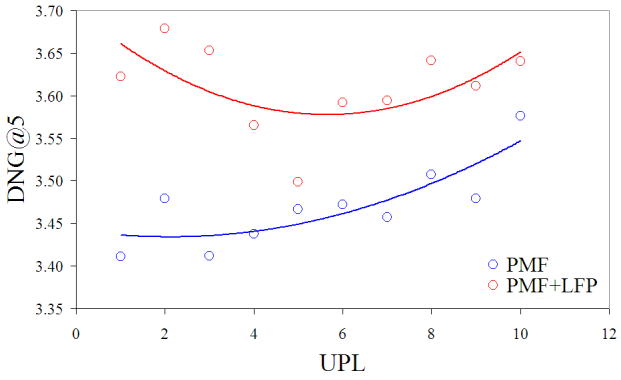
profiles, and use the remaining rated items as ground truth. By varying the value of parameter b in LFP, we evaluate its influence on the recommendation performance of different latent factor models, which is shown in Figure 5.1. As can be seen, for both PureSVD+LFP and PMF+LFP, the diversity measure DNG@5 generally increases as the value of b in LFP increases, while MAP decreases. Note that the baseline latent factor models are equivalent to the case when we set $b = 0$. The figures show that a positive value of b could contribute to diversifying the recommendation results, and that the magnitude of diversification is controlled by its value. However, a positive value of b could reduce the MAP, indicating that it may degrade the end-user satisfaction when the results are over-diversified. This observation is consistent with the study in text retrieval in [25]. Because the parameter is a constant across target users, it serves to adjust the diversity of recommendation in a system level, and its optimal value depends on the used evaluation metrics (in other words, the utility of the recommendation system).

5.5.2 Adaptive Diversity: the Number of Rated Items

We have discussed at the beginning of this chapter that the observed numbers of rating items are different across users. As illustrated in Figure 1.5, the number of user rated items influences the uncertainty of the learned latent user factors – the more information we have about the user, the less uncertain our model



(a) PureSVD vs. PureSVD+LFP



(b) PMF vs. PMF+LFP

Figure 5.2: The diversity that depends on the target user profiles: the number of rated items. (a) Comparison between PureSVD and PureSVD+LFP. (b) Comparison between PMF and PMF+LFP.

is about her hidden taste. In the following experiment, we evaluate the impact of the model uncertainty on the diversity of recommended items, where the model uncertainty is indicated by the number of rated items provided in the user profile. We generate the user profile length (UPL) from 1 to 15, and randomly select the rated items as user profile items. As in our dataset, each user has at least rated 20 items. Setting UPL up to 15, we ensure that there are at least 5 rated items per user used for testing. The results are shown in Figure 5.2.

From the figures, we observe that the LFP models PureSVD+LFP and PMF+LFP succeed in consistently increasing the diversity of recommendation results for their basic models PureSVD and PMF. Note that the increases are all statistically significant according to the Wilcoxon signed rank significance test with $p < 0.01$. This indicates that LFP could effectively capture the uncertainty of latent factors and use it to diversify recommendation results.

In addition, the diversity achieved by both our LFP models PureSVD+LFP and PMF+LFP and the basic latent factor models PureSVD and PMF, generally increases as the users rate more items. At a first glimpse, the result seems to contradict the understanding that adding ratings in the user profile would reduce the uncertainty of the user model and thus the need for diversifying the results. A closer look, however, suggests that this is intuitively correct because when there are few rated items known

Table 5.2: The diversity adapted to the target user profiles: the range of interests.

	UPL=2		UPL=3	
	Focused	Broad	Focused	Broad
PureSVD	3.148	3.419	3.174	3.386
PureSVD+LFP	3.293	**3.641**	3.304	**3.608**
p-value	0.096	0.000*	0.026	0.000*
PMF	3.353	3.400	3.373	3.432
PMF+LFP	3.415	**3.494**	3.449	**3.570**
p-value	0.335	0.006*	0.411	0.002*

from the users, the recommended items could be strongly biased toward the few known items, and thus less diversified, whereas when more rated items are known from the users, the recommended items could be more likely to cover different aspects of user interest, and are thus more diversified.

Finally and most importantly, we also observe that the increase of diversity introduced by LFP generally decreases as the number of user-rated items increases. As shown, the diversity increase brought by PureSVD+LFP compared to PureSVD tends to be constant as the UPL increases, and the diversity increase achieved by PMF+LFP compared to PMP+LFP tends to be smaller as the UPL increases. When the UPL is small, LFP automatically provides a relatively larger increase of diversity against the basic latent factor models. In other words, when the users rated only a few items, such as a cold-start user, the basic latent factor models tend to recommend items based on highly uncertain latent factors. LFP addresses the risk of the basic latent factor models by providing more diversified results.

5.5.3 Adaptive Diversity: the Range of Interests

We now focus on the evaluation by considering the users with different ranges of interests. To make our study focused and controlled, we are particularly interested in two types of user profiles: the users who rated movies with the same genre (denoted as the "Focused" type), and the users who rated movies with non-overlapped genres (denoted as the "Broad" type). The first type of user profiles represents a typical situation in which the target user has a specific range of interests and as a result, the diversification is less required, while the second type represents the opposite situation in which diversification is more desired. Also as demonstrated in the previous section, LFP could be most beneficial for diversifying recommendation results for the users who only rated a limited number of items. For this reason, we fix UPL (User Profile Length) to 2 and 3 in this investigation. For each UPL, we classify a user into the "Focused" type if all her rated items contain the same genre, and into the "Broad" type if all rated items contain genres different from each other.

The results are shown in Table 5.2, from which we draw two observations. First, for both the basic latent factor models PureSVD and PMF, diversity of the "Focused" type is lower than that of the "Broad" type for most of the cases. This result is in accordance with our understanding, as discussed in Section 1.2.3, that the commonality of the items in the user profile has a significant impact on the

Table 5.3: Example recommendation results for the two different types of user profiles. We refer "Ac" to *Action*, "Ad" to *Adventure*, "C" to *Comedy*, "D" to *Drama*, "H" to *Horror*, "R" to *Romance*, "SF" to *Sci-Fi*, "T" to *Thriller*, and "W" to *War*. PureSVD+LFP is used.

(a) An example of a *focused* user profile.

Type	*Focused*	
Profile	Chariots of Fire (D) Erin Brockovich (D)	
Rank	PureSVD	PureSVD+LFP
1	Second Best(D)	Second Best(D)
2	Saving Private Ryan(Ac,D,W)	North by Northwest(D,T)
3	North by Northwest(D,T)	The Truman Show(D)
4	The Truman Show(D)	Saving Private Ryan(Ac,D,W)
5	Jakob the Liar(D)	Jakob the Liar(D)
DNG@5	2.25	1.75

(b) An example of a *broad* user profile.

Type	*Broad*	
Profile	American Pie (C) The Blair Witch Project (H)	
Rank	PureSVD	PureSVD+LFP
1	Big Daddy(C)	Big Daddy(C)
2	Bowfinger(C)	The Mask of Zorro(Ac,Ad,R)
3	Parasite(H,SF)	Baby Geniuses(C)
4	Baby Geniuses(C)	Parasite(H,SF)
5	The Mask of Zorro(Ac,Ad,R)	Bowfinger(C)
DNG@5	1.69	2.75

user need for diversification. In the case of the "Focused" type of user profiles, the latent user factors are learned from the items that have the same or similar topics (in this case genres), and thus the latent factors of those movies could be similar. As a result, the uncertainty and variance of the latent user factors could be low, and those latent user factors would promote recommending movies with the same or similar genres as the movies that the user has already watched, i.e., a less diversified recommendation. By contrast, a more diversified recommendation would be promoted to the "Broad" type of user profiles.

Second, we observe that for both UPL=2 and UPL=3, either PureSVD+LFP or PMF+LFP has achieved a significant increase of diversity for the "Broad" type of user profiles, while introducing a slight change of diversity for the "Focused" type. This result indicates that LFP could effectively exploit the distribution of latent user factors to adaptively determine the level of diversification. This is further illustrated by the example in Table 5.3. We clearly see that LFP automatically adjusts the diversity of recommendations according to the different range of interests learned from the user profiles.

5.5.4 Combining Relevance and Diversity

Our final experiment investigates how LFP can benefit the end-user satisfaction by considering both the relevance and diversity of recommended items. We test the recommendation performance in terms of

Table 5.4: Relevance (measured by MAP) vs. diversity (measured by DNG@n and αNDCG@n).

(a) Relevance vs. diversity (PureSVD).

	MAP	DNG@5	αNDCG@5
UPL=2			
PureSVD	0.749	3.483	0.845
PureSVD+LFP ($b = -1$)	**0.751**	3.341	0.833
PureSVD+LFP ($b = 1$)	0.738	**3.645**	**0.858**
UPL=5			
PureSVD	0.764	3.533	0.854
PureSVD+LFP ($b = -1$)	**0.772**	3.417	0.841
PureSVD+LFP ($b = 1$)	0.741	**3.683**	**0.866**
UPL=10			
PureSVD	0.769	3.555	0.857
PureSVD+LFP ($b = -1$)	**0.774**	3.456	0.845
PureSVD+LFP ($b = 1$)	0.745	**3.706**	**0.870**

(b) Relevance vs. diversity (PMF).

	MAP	DNG@5	αNDCG@5
UPL=2			
PMF	0.787	3.412	0.846
PMF+LFP ($b = -1$)	**0.791**	3.361	0.839
PMF+LFP ($b = 1$)	0.758	**3.500**	**0.864**
UPL=5			
PMF	0.807	3.515	0.863
PMF+LFP ($b = -1$)	**0.814**	3.415	0.851
PMF+LFP ($b = 1$)	0.763	**3.611**	**0.874**
UPL=10			
PMF	0.818	3.538	0.865
PMF+LFP ($b = -1$)	**0.826**	3.462	0.855
PMF+LFP ($b = 1$)	0.774	**3.610**	**0.872**

relevance, as measured by MAP, and the performance in terms of diversity, as measured by DNG@5, under two different settings of parameter b in LFP, i.e., $b = -1$ and 1. Note that as shown in Section 5.5.1, a positive value of b tends to increase the recommendation diversity, while decreasing the recommendation relevance. Opposite results can be observed in the case of a negative value of b used in LFP. The results are shown in Table 5.4. We first observe that LFP could improve the relevance of recommendations for the users who are risk-loving. When $b = -1$, MAP is improved by both PureSVD+LFP and PMF+LFP. This result indicates that in the case where LFP increases the variance (thus similarity) of the latent item factors among the recommended items, it could contribute to improving the relevance of the recommendation. The empirical result is also consistent with the statistical mean-variance analysis of MAP conducted in [153]. Second, LFP could attain a trade-off between the recommendation relevance and the diversity. As can be seen, when $b = 1$, both PureSVD+LFP and PMF+LFP achieve diversified recommendation results shown in the improved DNG@5, while degrading the relevance performance as measured by MAP. But as a whole, αNDCG is substantially improved, indicating that the degraded

relevance could payoff for the overall quality of the recommendation that takes into account both the relevance and the diversity. Although here αNDCG could only serve as an approximation of the end-user satisfaction for the recommended items, the results are evident in that we can use LFP to adjust the trade-off between the relevance and the diversity of recommendations from latent factor models.

5.6 Concluding Remarks

In this chapter, we proposed a new recommendation framework LFP specially for adaptively diversifying multiple recommendation results for individual users. We exploited the variance of the latent user factors to capture the range of user interests and uncertainty of the user profiles, and to use them as the basis for indicating users' needs for diversity. Through our experiments, we demonstrated LFP's effectiveness for adapting result diversification to the users' needs without accessing explicit item properties. In addition, we also showed that LFP is capable of effectively adjusting the trade-offs between the relevance and the diversity of recommended items, and thus could further contribute to the overall recommendation quality.

From the analysis, especially, in Section 5.5.2, we have demonstrated that the diversification need in regard to a user with only a few rated items is especially high. This situation aligns with our discussions that a diversified recommendation list for a cold-start user could be especially beneficial. The results presented in this Chapter can be easily adapted to an item cold-start situation by swapping the user and item latent factors in all the presented equations. In addition, in the item cold-start scenario, we should assume the users' latent factors are fixed and model the target item's latent factors with distributions. The results will be a diversified user list adapted to the item's specific characteristics.

Chapter Appendices

A. Topical vs. Rank Variances

We present the detailed derivation of $\mathrm{Var}(R_{u,P(\boldsymbol{j})})$ in Eq. (5.10) below. Let us start with

$$\mathrm{Var}[R_{u,P(\boldsymbol{j})}] = \mathbb{E}[R_{u,P(\boldsymbol{j})} - \mathbb{E}(R_{u,P(\boldsymbol{j})})]^2.$$

Taking into account Eq. (5.8), we have:

$$\begin{aligned}
\mathrm{Var}[R_{u,P(\boldsymbol{j})}] =& \mathbb{E}[\sum_{n=1}^{N} w_n \sum_{k=1}^{K} q_{j(n),k} p_{u,k} - \sum_{n=1}^{N} w_n \sum_{k=1}^{K} q_{j(n),k} \mathbb{E}(p_{u,k})]^2 \\
=& \mathbb{E}[\sum_{n=1}^{N} w_n \sum_{k=1}^{K} q_{j(n),k} \left(p_{u,k} - \mathbb{E}(p_{u,k})\right)]^2 \\
=& \mathbb{E}[\sum_{n=1}^{N} w_n^2 \sum_{k=1}^{K} \sum_{l=1}^{K} q_{j(n),k} q_{j(n),l} (p_{u,k} - \mathbb{E}(q_{u,k}))(p_{u,l} - \mathbb{E}(p_{u,l})) \\
&+ \sum_{n=1}^{N} \sum_{\substack{m=1 \\ m \neq n}}^{N} w_n w_m \times \sum_{k=1}^{K} \sum_{l=1}^{K} q_{j(n),k} q_{j(m),l} (p_{u,k} - \mathbb{E}(p_{u,k}))(p_{u,l} - \mathbb{E}(p_{u,l}))].
\end{aligned}$$

Using the property as in Eq. (5.5), we obtain:

$$\begin{aligned}\mathrm{Var}[R_{u,P(\boldsymbol{j})}] =& \sum_{n=1}^{N} w_n^2 \sum_{k=1}^{K} q_{j(n),k}^2 \mathbb{E}[(p_{u,k} - \mathbb{E}(p_{u,k}))^2] \\ &+ \sum_{n=1}^{N} \sum_{\substack{m=1 \\ m\neq n}}^{N} w_n w_m \sum_{k=1}^{K} q_{j(n),k} q_{j(m),k} \mathbb{E}[(p_{u,k} - \mathbb{E}(p_{u,k}))^2].\end{aligned}$$

The Eq. (5.10) is obtained as above by with the definition of $\sigma_{u,k}^2$.

B. Sequential Ranking

The detailed derivation of $\Delta U_n(R_{u,P(\boldsymbol{j})})$ in Eq. (5.12) is given below.

$$\begin{aligned}&\Delta U_n(R_{u,P(\boldsymbol{j})}) \\ =&U_n(R_{u,P(\boldsymbol{j})}) - U_{n-1}(R_{u,P(\boldsymbol{j})}) \\ =& \sum_{m=1}^{n} w_m \sum_{k=1}^{K} q_{j(m),k} p_{u,k} - b\Big(\sum_{m=1}^{n} w_m^2 \sum_{k=1}^{K} q_{j(m),k}^2 \sigma_{u,k}^2 + \sum_{m=1}^{n} \sum_{\substack{l=1 \\ l\neq m}}^{n} w_m w_l \sum_{k=1}^{K} q_{j(m),k} q_{j(l),k} \sigma_{u,k}^2\Big) \\ &- \Big(\sum_{m=1}^{n-1} w_m \sum_{k=1}^{K} q_{j(m),k} p_{u,k} - b\Big(\sum_{m=1}^{n-1} w_m^2 \sum_{k=1}^{K} q_{j(m),k}^2 \sigma_{u,k}^2 + \sum_{m=1}^{n-1} \sum_{\substack{l=1 \\ l\neq l}}^{n-1} w_m w_l \sum_{k=1}^{K} q_{j(m),k} q_{j(l),k} \sigma_{u,k}^2\Big)\Big) \\ =& w_n \sum_{k=1}^{K} q_{j(n),k} p_{u,k} - b\Big(w_n^2 \sum_{k=1}^{K} q_{j(n),k}^2 \sigma_{u,k}^2 \\ &+ \sum_{m=1}^{n} \sum_{\substack{l=1 \\ l\neq m}}^{n} w_m w_l \sum_{k=1}^{K} q_{j(m),k} q_{j(l),k} \sigma_{u,k}^2 - \sum_{m=1}^{n-1} \sum_{\substack{l=1 \\ l\neq m}}^{n-1} w_m w_l \sum_{k=1}^{K} q_{j(m),k} q_{j(l),k} \sigma_{u,k}^2\Big) \\ =& w_n \sum_{k=1}^{K} q_{j(n),k} p_{u,k} - b\Big(w_n^2 \sum_{k=1}^{K} q_{j(n),k}^2 \sigma_{u,k}^2 + \sum_{m=1}^{n-1} w_n w_m \sum_{k=1}^{K} q_{j(n),k} q_{j(m),k} \sigma_{u,k}^2 \\ &+ \sum_{m=1}^{n-1} \sum_{\substack{l=1 \\ l\neq m}}^{n} w_m w_l \sum_{k=1}^{K} q_{j(m),k} q_{j(l),k} \sigma_{u,k}^2 - \sum_{m=1}^{n-1} \sum_{\substack{l=1 \\ l\neq m}}^{n-1} w_m w_l \sum_{k=1}^{K} q_{j(m),k} q_{j(l),k} \sigma_{u,k}^2\Big).\end{aligned}$$

Combining the last two terms, we have:

$$\begin{aligned}\Delta U_n(R_{u,P(\boldsymbol{j})}) =& w_n \sum_{k=1}^{K} q_{j(n),k} p_{u,k} - b\Big(w_n^2 \sum_{k=1}^{K} q_{j(n),k}^2 \sigma_{u,k}^2 \\ &+ \sum_{m=1}^{n-1} w_n w_m \sum_{k=1}^{K} q_{j(n),k} q_{j(m),k} \sigma_{u,k}^2 \\ &+ \sum_{l=1}^{n-1} w_n w_l \sum_{k=1}^{K} q_{j(n),k} q_{j(l),k} \sigma_{u,k}^2\Big).\end{aligned}$$

Note that in above m and l are interchangeable. Thus, we have:

$$\Delta U_n(R_{u,P(\boldsymbol{j})}) = w_n \sum_{k=1}^{K} q_{j(n),k} p_{u,k} - b\Big(w_n^2 \sum_{k=1}^{K} q_{j(n),k}^2 \sigma_{u,k}^2 + 2 \sum_{m=1}^{n-1} w_n w_m \sum_{k=1}^{K} q_{j(n),k} q_{j(m),k} \sigma_{u,k}^2\Big).$$

Swapping the summation order over space m and k in the last term, we obtain Eq. (5.12):

$$\Delta U_n(R_{u,P(\boldsymbol{j})}) = w_n \sum_{k=1}^{K} \Big(q_{j(n),k} p_{u,k} - b w_n \sigma_{u,k}^2 q_{j(n),k}^2 - 2b\sigma_{u,k}^2 \sum_{m=1}^{n-1} w_m q_{j(n),k} q_{j(m),k} \Big).$$

Chapter 6

Risk-Hedging Diversification

In the previous chapter, we focused on the diversification problem and proposed latent factor portfolio to achieve adaptive diversification in personal recommendation. This chapter aims to relate the portfolio recommendation back to the temporal recommendation process. We argue that the recommended items could be decided jointly with the past preference records of the users, instead of being optimised alone. This argument is demonstrated by means of the risk-hedging joint portfolio diversification algorithms proposed in this chapter.

Since the algorithm proposed in this chapter is motivated by the concept from finance, we adopt a new dataset for our analysis – CrunchBase*, an online platform that records the investment activities of individuals and investors in regard to startups in the US high-tech sectors. The problem is to generate recommendations of promising investment opportunities tailored to the information need of individual venture capital firms, who may have different approaches for making investment decisions according to their financial situations, investment styles and investment expectations. This problem, on one hand, is similar to a traditional recommendation problem as the dataset also comprises two components: the venture capital firms and the startups, together with the recorded investment behaviours (such as investment amount and date), making it possible for us to explore the patterns of the investment behaviours using techniques such as collaborative filtering (CF). As such, we can view the proposed recommender system as a novel quantitative solution for the venture finance screening process. On the other hand, however, the dataset of venture finance investment also presents some unique properties when compared to a traditional movie/music rating dataset, such as its sparseness and its unique categorical distribution.

With the above considerations, in this chapter, we first provide a brief introduction to the venture capital investment screening process and the characteristics of the considered dataset. Then, we propose the joint portfolio recommendation solution, followed by experimental evaluations.

6.1 Venture Finance Background

Venture finance refers to the financing of private companies through the use of venture capital, a form of private equity, a medium to long-term form of finance provided in return for an equity stake in potentially high growth companies. VC has five main characteristics [169]: is a financial intermediary; invests only

*http://www.crunchbase.com/, the details of the dataset will be described in Section 6.3

in private companies; takes an active role in monitoring and helping portfolio companies; primary goal is to maximise financial return by exiting investments through sale or an initial public offering (IPO); invests to fund the internal growth of companies.

Early-stage investment is typified by venture capital firms (VCs) who deploy capital towards high-risk ventures. It is a key driving force of technological innovation and is vitally important to the wider economy, especially in high-growth and hi-tech industries, such as life sciences, clean-tech and information technology. Traditionally, investment opportunities are either referred or identified through manual technology scans [170]. The main stages of an investor's decision making process involve deal origination, screening, evaluation, structuring and post investment activities. These stages align with those identified by other research into VC investment [171]. In recent years the traditional venture financing landscape has also shown signs of evolving. Some commentators [172] depict an industry "trifurcating" with (i) top-tier firms, e.g., Sequoia Capital, (ii) incubators and accelerators, e.g., Y Combinator, and, finally, (iii) firms that are taking a more quantitative approach to funding, e.g., Correlation Ventures. There is potentially a fourth factor in the emergence of entirely new funding sources such as "crowdfunding" which generally operate through online platforms, e.g., AngelList. Shifts towards more quantitative and data driven approaches along with new opportunities for online private investment provide additional impetus and scope for applying data mining and intelligent recommendations to this domain [173].

6.2 Recommendation for Venture Finance

Whilst there have been some applications of recommender systems to the broader domain of finance, including microfinance [174], there has seemingly been few previous academic research in applying such techniques directly to venture finance. To our knowledge, our work in [173] is the first and the only one that has studied CF on venture finance recommendation. However, in [173], we only showed some empirical results of a direct application of recommendation algorithms to venture finance, lacking a more sophisticated consideration or adjustment of recommendation methods to the unique domain, where the risk is a major concern. It is worth mentioning that [175] also considered risk in recommendation optimisation for a P2P lending investment recommendation problem. However, the authors failed to address the correlations between investments or to analyse the investors' risk-averse levels, making it significantly different from our method.

The domain of investment recommendation shows some special features compared with traditional applications of recommender systems (e.g., for movies and music). First, modeling and controlling risk for an investment portfolio is more essential for making investment recommendations. However, existing work on recommender system applications in venture finance has largely disregarded the risk factor, such as [173], only the similarity between new investment opportunities and VCs' holding investments were explored. Promoting similar opportunities may be attractive to the VCs at the first sight, but such similarity-based methods fail to catch VCs' underlying main investment intention, which is to examine how well the new investment will fit into the current investment portfolio to hedge the risk and increase the return [176].

Second, recommending jointly-diversified items can be especially beneficial in the investment re-

Table 6.1: Summary of key notations in Chapter 6.

Notation	Description
u	The VC (user) under consideration
n	The size of the recommendation list
$\boldsymbol{i}$	The startups (items) that have been invested by u
$\boldsymbol{j}$	The startups to recommend to u, $\|\boldsymbol{j}\| = n$
$\mathcal{I}, \mathcal{I}^u, \mathcal{I}^c$	The entire available startup set, the available startup set to recommend to u ($\mathcal{I}^u = \mathcal{I} \backslash \boldsymbol{i}$), the candidate startup set for u
$J(\boldsymbol{j})$	The joint portfolio based on the startup set $\boldsymbol{i}$ and $\boldsymbol{j}$
$\boldsymbol{w_i}$, $\boldsymbol{w_j}$	The corresponding weights of $\boldsymbol{i}$ and $\boldsymbol{j}$ in the portfolio respectively
$R_{u,J(\boldsymbol{j})}$	The overall relevance of the joint portfolio $J(\boldsymbol{j})$ to u
$U[R_{u,J(\boldsymbol{j})}]$	u's utility over the joint portfolio $J(\boldsymbol{j})$
b	u's risk-averse level
$\boldsymbol{p}_u, \boldsymbol{q}_i$	The feature vector of VC u, the feature vector of startup i, calculated as the MAP solution of the PMF model
N	The size of the candidate startup set, $\|\mathcal{I}^c\| = N$
T	The sampling number
J_g	The joint portfolio based on $\boldsymbol{i}$ and the candidate startup set $\mathcal{I}^c$

commendation problem. It is because a VC's decision on further investments does not necessarily indicate terminations of previous holding investments, but adding them into the existing portfolio. This requires us to diversify the *joint* investment portfolio including both the VCs' holding investments and potential future ones, instead of diversifying the recommendation list alone as commonly studied [177, 47].

Finally, the CrunchBase dataset is much sparser than the traditional rating-based recommendation dataset (see detailed dataset description in the next section), because each VC usually only invests in a small number of investments. Meanwhile, a VC usually only focuses on a few industry categories, therefore, it is infeasible to employ a topic diversification method [46, 178] to explicitly diversify the items.

6.3 The CrunchBase Dataset

CrunchBase is a repository of startup companies, individual partners, and financial institutes focusing on the US high-tech sectors [179]. With its self-description as a "free database of technology companies, people and investors that anyone can edit", CrunchBase maintains the investment events between investors (including financial institutes and individual partners) and investment opportunities (usually startup companies) associated with the total amount of raised funding[†] and time. According to [173], financial organisations and individual partners are significantly different in their investment behaviours. Thus in search of consistent properties, in this work we focus on only the financial organisations. We

[†]It is the total raised money in one round for a startup instead of indicated for each funding party. Therefore we choose not to use the funding amount information in this work.

crawled the CrunchBase data from its official API[‡] in May 2014. In total, we collected 62,926 investment events between 7,706 VCs and 18,026 startups from 1987 to 2014. We publicise the dataset online for research use[§].

By comparing the statistics between CrunchBase and the MovieLens 1M dataset, we have identified several unique characteristics. First, as shown in Figure 6.1, VCs in the CrunchBase dataset tend to invest in a small number of industry categories, whereas users in the MovieLens dataset tend to rate a variety of movies, which often span more than 15 different genres. The reasons could be that, on one hand, VCs' investment numbers in CrunchBase are generally much lower than the user rating numbers in MovieLens, due to the severe sparsity of the CrunchBase dataset (detailed below); on the other hand, VCs may be cautious in investing in unfamiliar industry categories to avoid risk.

Second, the CrunchBase dataset is much sparser than conventional recommendation data. The rating ratio of the MovieLens 1M dataset is about 4.46%, and it is 1.17% for another well-known movie-rating dataset Netflix. These ratios are already very low, but the observed investment ratio of CrunchBase is even lower: only 0.045%, about $1/97$ of MovieLens 1M's and $1/25$ of Netflix's. Such sparsity is reasonable since private investment activity is not as commonplace as simply watching movies. Also, the final investment decision will require the consent of both the company and VCs and usually involves a lengthy due diligence process [180].

We argue that, the venture capital investment recommendation problem can especially benefit from joint portfolio diversification, due to its following characteristics revealed by the CrunchBase data:

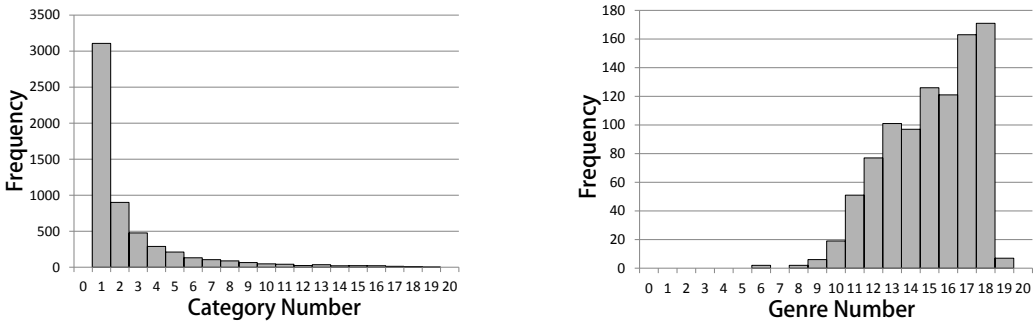
- From Figure 1.6, VCs tend to invest in opportunities with risk concerns rather than pure recommendations based on similarity.
- VCs usually cannot make extremely large numbers of investments. For each new investment opportunity, the VC may consider how it can fit into its holding portfolio. This requires us to optimise the portfolio including both the invested startups and those to be recommended together.
- VCs normally focus on a small number of industry categories, unlike the wide range of genres in users' movie watching behaviours. This suggests that we cannot simply use a topic-diversification method commonly used for recommendation list diversification [46].

6.4 Methodology

6.4.1 Problem Formulation

Let us denote a VC (venture capital firm) as u and the available startup (investment opportunity) pool as $\mathcal{I}$. For the reader's convenience, we list key notations of this chapter in Table 6.1. Suppose that VC u has already invested in m startups from the pool, and the recommender system is to seek another n startups from the pool for this VC to invest in. Without loss of generality, we denote the m holding investments (startups that the VC has already invested in) as $\boldsymbol{i} = (i_1, i_2, \ldots, i_m)$, and denote the startups

[‡]CrunchBase API: `http://developer.crunchbase.com`
[§]http://www0.cs.ucl.ac.uk/staff/w.zhang/cb.html



(a) CrunchBase (b) MovieLens

Figure 6.1: The category/genre distributions of the CrunchBase and the MovieLens datasets.

to recommend as $\boldsymbol{j} = (j_1, j_2, \ldots, j_n)$ where $\boldsymbol{j} \subset \mathcal{I} \backslash \boldsymbol{i}$. We will also refer to the available startup set for VC u as $\mathcal{I}^u$ ($\mathcal{I}^u = \mathcal{I} \backslash \boldsymbol{i}$) in the following.

Since for user u, $\boldsymbol{i}$ is known and fixed, we simply use $J(\boldsymbol{j})$ to denote the joint portfolio (without explicitly showing the involvement of $\boldsymbol{i}$ in this notation), which is a linear combination of the $m + n$ items (m invested startups $\boldsymbol{i}$ and n recommendations $\boldsymbol{j}$) with normalised weights:

$$J(\boldsymbol{j}) = \{(i_1, w_{i_1}), \ldots (i_m, w_{i_m}), (j_1, w_{j_1}), \ldots (j_n, w_{j_n})\} \tag{6.1}$$

where $\sum_{\alpha=1}^{m} w_{i_\alpha} + \sum_{\beta=1}^{n} w_{j_\beta} = 1$. Here the weights stand for the estimated importance of each startup in the portfolio and will finally determine the ranking of the recommendation list [177].

We further denote VC u's overall preference on the joint portfolio $J(\boldsymbol{j})$ as $R_{u,J(\boldsymbol{j})}$, which is a weighted linear combination of the preferences on its component, as will be discussed later. According to probabilistic matrix factorization (PMF), $R_{u,J(\boldsymbol{j})}$ can be modelled as a random variable [31]. The utility function $U[R_{u,J(\boldsymbol{j})}]$ based on the random variable $R_{u,J(\boldsymbol{j})}$ is defined as a trade-off between the expected reward $\mathbb{E}[R_{u,J(\boldsymbol{j})}]$ and the associated risk. The risk is usually defined as the variance of the reward $\mathrm{Var}[R_{u,J(\boldsymbol{j})}]$ [152, 25, 177]. In the risk-averse case, it is subtracted from the expected reward $\mathbb{E}[R_{u,J(\boldsymbol{j})}]$ to form the utility function.

The objective function is thus to find n startups (with rankings) to recommend to u so that u's utility over the joint portfolio is optimised:

$$\boldsymbol{j}^* = \underbrace{\arg\max_{\boldsymbol{j}}}_{\text{startup selection}} \underbrace{\left[\max_{\boldsymbol{w_i}, \boldsymbol{w_j}} \left(\mathbb{E}[R_{u,J(\boldsymbol{j})}] - b\,\mathrm{Var}[R_{u,J(\boldsymbol{j})}]\right)\right]}_{\text{portfolio optimisation}}. \tag{6.2}$$

Here we have already used vectors $\boldsymbol{w_i}$ and $\boldsymbol{w_j}$ to denote the weight vector of startups $\boldsymbol{i}$ and startups $\boldsymbol{j}$ respectively. Different from Chapter 5 where the weights for corresponding ranking positions are predefined and fixed, here in the investment scenario, the weights should be modifiable, which further determine the ranking of the n recommended startups. b is the VC's risk-averse level. A higher b means that the VC is more risk-averse and more willing to sacrifice the expected reward to hedge the risk. It can be optimised globally (for all the VCs) or personally (adapted for each individual VC) from the data. We will show in the experiment section how b is determined and calibrated from the data.

We can see that there are two sub-problems in the objective function:

- **Portfolio optimisation**: given a candidate recommendation set of startups $\boldsymbol{j}$, to find the optimally allocated weights to maximise the utility of the joint portfolio $J(\boldsymbol{j})$. This part is discussed in Section 6.4.2.

- **Startup selection and ranking**: given a pool of available startups $\mathcal{I}^u$, to select a subset $\boldsymbol{j} \subseteq \mathcal{I}^u$ ($|\boldsymbol{j}| = n$) to form the joint portfolio $J(\boldsymbol{j})$. This part is discussed in Section 6.4.3.

6.4.2 Portfolio Optimisation

We first focus on the portfolio optimisation problem:

$$\max_{\boldsymbol{w_i}, \boldsymbol{w_j}} U[R_{u,J(\boldsymbol{j})}] = \max_{\boldsymbol{w_i}, \boldsymbol{w_j}} \mathbb{E}[R_{u,J(\boldsymbol{j})}] - b \operatorname{Var}[R_{u,J(\boldsymbol{j})}]. \tag{6.3}$$

To further simplify the notations, we will use $\boldsymbol{w}$ as the concatenation $(\boldsymbol{w_i}, \boldsymbol{w_j})$ in the following. We will also denote the startups contained in the joint portfolio as $\boldsymbol{\kappa}$ which is a concatenation $(\boldsymbol{i}, \boldsymbol{j})$. A startup in the joint portfolio is thus denoted by a single symbol κ ($\kappa \in \boldsymbol{\kappa}$). Now, the optimisation problem is simplified as $\max_{\boldsymbol{w}} \mathbb{E}[R_{u,J(\boldsymbol{j})}] - b \operatorname{Var}[R_{u,J(\boldsymbol{j})}]$. Here we allow flexible weights $\boldsymbol{w_i}$ of existing startups in the portfolio optimisation process as we assume the VC can adjust their importance and priorities.

Portfolio-Level Preference

As mentioned before, we associate weights as the importance of startups in the portfolio. We also define the ranking order of startups by the importance (weight) order among all recommended items. Now that the problem is translated into a ranking problem, and thus we adopt a generalised definition of weight which can be either positive or negative [177]. An advantage of this treatment also lies in its analytical solution for weight optimisation.

As mentioned before, a VC u's preference on a portfolio is a random variable $R_{u,J(\boldsymbol{j})}$, which is a linear combination of the preference random variables of individual startups denoted by $r_{u,\kappa}$:

$$R_{u,J(\boldsymbol{j})} = \sum_{\kappa} w_\kappa r_{u,\kappa} = \boldsymbol{w}^T \boldsymbol{r}, \tag{6.4}$$

where $\boldsymbol{r}$ is the vector representation of the VC's preferences of startups in the portfolio. By denoting the mean and variance of the preference $r_{u,\kappa}$ as $\mu_{u,\kappa}$ and $\sigma^2_{u,\kappa}$, the expectation and variance of $R_{u,J(\boldsymbol{j})}$ are calculated as:

$$\mathbb{E}[R_{u,J(\boldsymbol{j})}] = \sum_{\kappa} w_\kappa \mathbb{E}[r_{u,\kappa}] = \boldsymbol{w}^T \boldsymbol{\mu}, \tag{6.5}$$

$$\operatorname{Var}[R_{u,J(\boldsymbol{j})}] = \sum_{\kappa} \sum_{\kappa'} w_\kappa w_{\kappa'} \operatorname{Cov}(\kappa, \kappa') = \boldsymbol{w}^T \boldsymbol{\Sigma} \boldsymbol{w}. \tag{6.6}$$

Here we have used $\boldsymbol{\mu}$ to denote the vector of the preference expectations, and $\boldsymbol{\Sigma}$ to denote the covariance matrix whose (κ, κ')-th element is given by the covariance $\operatorname{Cov}(\kappa, \kappa') = \sigma_{u,\kappa} \rho_{\kappa,\kappa'} \sigma_{u,\kappa'}$,

where $\rho_{\kappa,\kappa'}$ is the correlation between startup κ and κ' [152, 25] and can be estimated via industry category overlap [173] or latent factor vector cosine [177].

We follow [177] and use the PMF model discussed in Chapter 3 to obtain the probabilistic representations of the VC-startup preferences. Assuming that uncertainty in the preference originates from the uncertainty of the user latent factor estimation similar to Chapter 3, we can estimate the expectation and variance of $r_{u,\kappa}$ as follows:

$$\mu_{u,\kappa} = \mathbb{E}[\boldsymbol{p}_u]^T \boldsymbol{q}_\kappa, \tag{6.7}$$

$$\sigma^2_{u,\kappa} = \boldsymbol{q}_\kappa^T \operatorname{Cov}[\boldsymbol{p}_u] \boldsymbol{q}_\kappa. \tag{6.8}$$

Here $\boldsymbol{p}_u$ and $\boldsymbol{q}_\kappa$ can be estimated by the maximum a posteriori (MAP) solution of $p(r_{u,i}|\boldsymbol{p}_u, \boldsymbol{q}_i, \sigma_0^2) = \mathcal{N}(r_{u,i}|\boldsymbol{p}_u^T \boldsymbol{q}_i, \sigma_0^2)$ (see Chapter 3).

Portfolio Weight Optimisation

Integrating Eqs. (6.5) and (6.6) into Eq. (6.3), we translate the portfolio optimisation problem into the portfolio weight optimisation problem:

$$\max_{\boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{\mu} - b \boldsymbol{w}^T \boldsymbol{\Sigma} \boldsymbol{w}, \tag{6.9}$$

which is a standard quadratic optimisation problem. In the case that $\boldsymbol{w}$ can take any value in $\mathbb{R}^{m+n}$, the analytic solution can be written as [177]

$$\boldsymbol{w}_M = \frac{\begin{vmatrix} 1 & \mathbf{1}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ \mu_p & \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \end{vmatrix} \boldsymbol{\Sigma}^{-1}\mathbf{1} + \begin{vmatrix} \mathbf{1}^T\boldsymbol{\Sigma}^{-1}\mathbf{1} & 1 \\ \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\mathbf{1} & \mu_p \end{vmatrix} \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}{\begin{vmatrix} \mathbf{1}^T\boldsymbol{\Sigma}^{-1}\mathbf{1} & \mathbf{1}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\mathbf{1} & \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \end{vmatrix}}, \tag{6.10}$$

where

$$\mu_p = \frac{1 - b\theta_2}{2b\theta_1}, \tag{6.11}$$

and

$$\theta_1 = \frac{(x\boldsymbol{\mu} - y\mathbf{1})^T\boldsymbol{\Sigma}^{-1}(x\boldsymbol{\mu} - y\mathbf{1})}{(xz - y^2)^2},$$

$$\theta_2 = \frac{2(x\boldsymbol{\mu} - y\mathbf{1})^T\boldsymbol{\Sigma}^{-1}(z\mathbf{1} - y\boldsymbol{\mu})}{(xz - y^2)^2}, \tag{6.12}$$

$$\tag{6.13}$$

Algorithm 6.1: Sampling-based Startup Selection (Sampling)

Require: VC u, current invested startups $i_1, \ldots, i_m$, candidate startup set $\mathcal{I}^c$, risk-averse parameter b, recommendation size n, utility function U, sampling number T
Initialise $\boldsymbol{j}^* \leftarrow \emptyset$
for $t = 1 \ldots T$ **do**
  Sample $\boldsymbol{j}_t = (j_1, j_2, \ldots, j_n)$ from $\mathcal{I}^c$
  Build joint portfolio $J(\boldsymbol{j}_t)$ based on the joint set $(i_1, \ldots, i_m, j_1, j_2, \ldots, j_n)$ via Eq. (6.1)
  Calculate the maximum utility $U[R_{u,J(\boldsymbol{j}_t)}]$ via Eq. (6.3)
  if $U[R_{u,J(\boldsymbol{j}_t)}]$ is the largest utility so far **then**
    Update $\boldsymbol{j}^* \leftarrow \boldsymbol{j}_t$
  end if
end for
return Rank list $\boldsymbol{j}^*$

where $x = \mathbf{1}^T\boldsymbol{\Sigma}^{-1}\mathbf{1}$, $y = \mathbf{1}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} = \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\mathbf{1}$, and $z = \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$.

Without loss of generality we assume an optimised portfolio is ranked according to the item weight such that

$$w_{i_1} > w_{i_2} > \cdots > w_{i_m}, \text{ and } w_{j_1} > w_{j_2} > \cdots > w_{j_n},$$

i.e., elements in $\boldsymbol{i}$ and $\boldsymbol{j}$ are ranked by their importance.

6.4.3 Startup Selection and Ranking

In Section 6.4.2, we discussed the model to estimate the maximum investment utility $U[R_{u,J(\boldsymbol{j})}]$ given the recommended startups $\boldsymbol{j}$. In this section, we discuss the algorithms to efficiently find the optimal recommendation set $\boldsymbol{j}$ from a large candidate corpus $\mathcal{I}^u$.

Considering the fact that the possible startup combination space is extremely large ($C_n^{|\mathcal{I}^u|}$), we need to first reduce the candidate set by pre-selecting a size-N candidate startup set $\mathcal{I}^c \subseteq \mathcal{I}^u$ ($|\mathcal{I}^c| = N$) with the highest expected preferences $\mu_{u,\kappa}$ estimated from PMF (Eq. (6.7)). Then within the candidate set $\mathcal{I}^c$ we determine the final ranked list of startups $\boldsymbol{j}$. All of our proposed algorithms share the procedure of first choosing the size-N candidate set and then determining the final size-n ranked recommendations.

With the candidate set $\mathcal{I}^c$ we propose the following 5 different algorithms to find the optimal selections and their ranking.

Startup Selection by Sampling

A straightforward solution is to use a sampling method to approximate the optimal solution, which greatly reduces the computational cost. The details are presented in Algorithm 6.1. By sampling n-sized startup combinations among the N candidates for T times and picking the combination with the highest utility, we can get a globally $1/T$ best combination in expectation. As $T \to C_n^{|\mathcal{I}^c|}$, the performance of the sampling-based method will converge to the globally optimal solution, i.e., the portfolio $J(\boldsymbol{j}^*)$ leading to the highest utility $U[R_{u,J(\boldsymbol{j}^*)}]$.

Startup Selection by Individual Score Ranking

This is a simple ranking algorithm that ranks the startup utility by considering individual startups joining the current portfolio. We denote the joint portfolio including one candidate startup j as $J(j)$, and the

Algorithm 6.2: Individual Startup Selection (Idv)

Require: VC u, current invested startups $i_1, \ldots, i_m$, candidate startup set $\mathcal{I}^c$, risk-averse parameter b, recommendation size n, utility function U
for each candidate startup j in $\mathcal{I}^c$ **do**
Build joint portfolio $J(j)$ based on the joint set $(i_1, \ldots, i_m, j)$ via Eq. (6.1)
Calculate the maximum utility $U[R_{u,J(j)}]$ via Eq. (6.3)
end for
return Rank list $\boldsymbol{j}^*$ of n startups with highest $U[R_{u,J(j)}]$

Algorithm 6.3: Sequential Startup Selection (Seq)

Require: VC u, current invested startups $i_1, \ldots, i_m$, candidate startup set $\mathcal{I}^c$, risk-averse parameter b, recommendation size n, utility function U
Initialise startup set $\boldsymbol{j}^* \leftarrow \emptyset$
for $l = 1 \ldots n$ **do**
Select the optimal startup j_l^* in $\mathcal{I}^c$ such that

$$j_l^* = \arg\max_{j_l \in \mathcal{I}^c} U[R_{u,J(\boldsymbol{j}^*, j_l)}]$$

where the joint portfolio $J(\boldsymbol{j}^*, j_l)$ is built based on startups $\{i_1, \ldots, i_m, j_1^*, j_2^*, \ldots, j_{l-1}^*, j_l\}$ via Eq. (6.1)
$\boldsymbol{j}^* \leftarrow \boldsymbol{j}^* \cup \{j_l^*\}$
$\mathcal{I}^c \leftarrow \mathcal{I}^c \backslash j_l^*$
end for
return Rank list $\boldsymbol{j}^*$ with the selection order

maximum utility $U[R_{u,J(j)}]$ with j will act as the ranking score of j. Based on the score of each candidate startup, we can rank them and choose the top-n startups with the highest scores. This procedure is given in Algorithm 6.2.

As we can see, Algorithm 6.2 is quite straightforward: selecting each startup based on the utility it brings. However, this algorithm fails to consider the correlation among the n recommended startups.

Sequential Startup Selection

Similar to Chapter 5, we select the startups incrementally to approximate the optimal solution with a large computational cost reduced. For each iteration, in a greedy fashion, we select one startup which can bring the highest increase in the utility function when being added into the current portfolio. This procedure is described in Algorithm 6.3.

Sequential methods have been adopted also in webpage ranking [25]. Though it is a greedy method, it has shown high efficiency and good empirical performances.

Startup Selection by Weight Ranking

With the candidate startup set $\mathcal{I}^c$, we can build a portfolio J_g with all the candidate startups $\mathcal{I}^c$ and the invested startups $\boldsymbol{i}$. Then we can apply the portfolio optimisation according to Eq. (6.3) to obtain the optimal weights for all the candidates. We rank their weights and select the top n. This algorithm is illustrated in Algorithm 6.4.

This algorithm takes into account the relationship between each pair of candidate startups in $\mathcal{I}^c$. However, by selecting the top n candidates with the highest portfolio weights, the resulting portfolio is

Algorithm 6.4: Weight-based Startup Selection (Weight)

Require: VC u, current invested startups $i_1, \ldots, i_m$, candidate startup set $\mathcal{I}^c$, risk-averse parameter b, recommendation size n, utility function U
Build the portfolio J_g based on the joint set $\boldsymbol{g} = \{i_1, \ldots, i_m\} \cup \mathcal{I}^c$ via Eq. (6.1)
Calculate the optimal weights $\boldsymbol{w_g}$ via Eq. (6.3)
Sort the candidate startups by their weight in $\boldsymbol{w_g}$
return Rank list $\boldsymbol{j}^*$ of n startups with highest weights

Algorithm 6.5: Filtering-based Startup Selection (Filtering)

Require: VC u, current invested startups $i_1, \ldots, i_m$, candidate startup set $\mathcal{I}^c$, risk-averse parameter b, recommendation size n, utility function U
while $|\mathcal{I}^c| > n$ **do**
  Build joint portfolio J_g based on the joint startup set $\boldsymbol{g} = \{i_1, \ldots, i_m\} \cup \mathcal{I}^c$ via Eq. (6.1)
  Obtain the optimal weights $\boldsymbol{w_g}$ via Eq. (6.3)
  Obtain j_f with the lowest weight in $\boldsymbol{w_g}$
  Update $\mathcal{I}^c \leftarrow \mathcal{I}^c \backslash j_f$
end while
return Rank list of startups in $\mathcal{I}^c$ by the optimal weights

already different from the global portfolio J_g. In other words, the top n candidates are selected based on a globally learnt weight ranking rather than being a direct optimisation on the joint portfolio with only these n candidates added, which is a discrepancy.

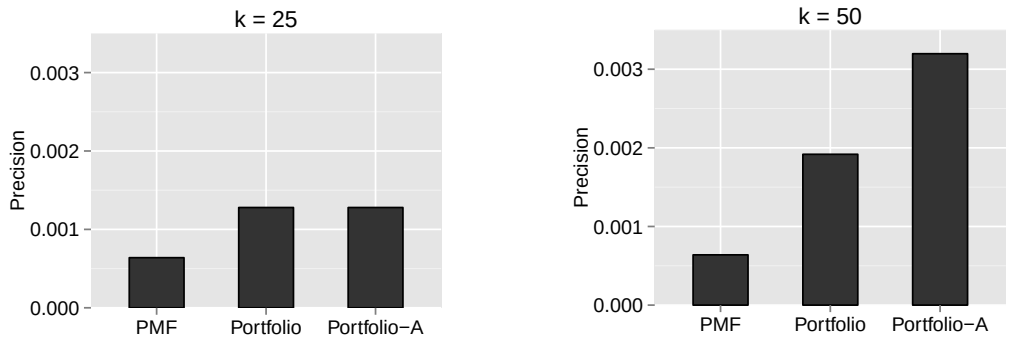
Startup Selection by Weight Filtering

Here we implement a backward sequential method shown in Algorithm 6.5. In each iteration, we build the global portfolio J_g based on the invested startups and the startups in the candidate set, optimise the portfolio to obtain the optimal weights according to Eq. (6.3), and remove the candidate startup with the lowest weight from the candidate startup set. This process iterates until the resulting candidate startup set shrinks to the size of n. Similar to the weight ranking algorithm, the weight filtering algorithm is also based on the weights obtained by optimising the portfolio constructed by the overall startup set rather than the selected subset, and thus suffers from the same discrepancy as the weight ranking algorithm.

6.4.4 Adaptive Risk-Averse Level

With different industry category focuses and investment strategies, different VCs may have different risk-averse levels, represented as the parameter b in our model Eq. (6.2). All the above discussed algorithms take b as a model parameter, yet b can also be learnt for each VC and thus the portfolio can be optimised in a personalised manner.

In order to adaptively learn this parameter for each VC, we conduct a cross validation on the training data, tune the parameter b_u for each VC u, and pick its optimal value for each VC which maximises the startup ranking evaluation measure (e.g., NDCG) on the validation data. Then the learnt b_u for VC u is used in the test phase.



(a) k = 25 (b) k = 50

Figure 6.2: The recommendation precision compared between PMF, the portfolio-based algorithm denoted by Portfolio and the adaptive portfolio-based algorithm denoted by Portfolio-A, when $n = 1$. $k = 25$ is used in (a) and $5 = 50$ in (b) for training the PMF model.

6.5 Experiments

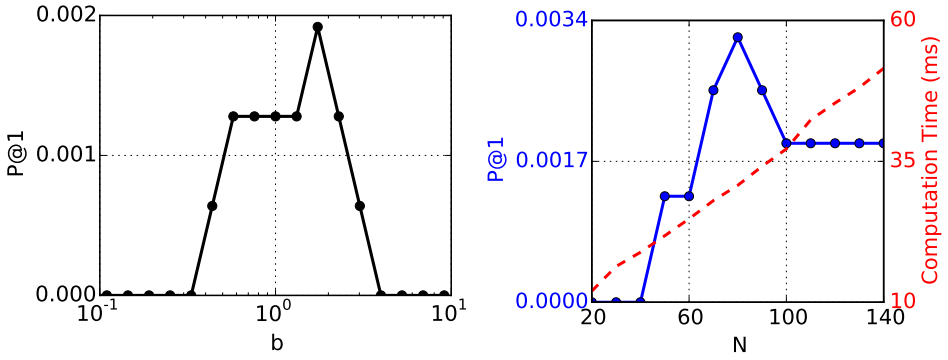
After describing the experiment setup in Section 6.5.1, we present the experimental results in three parts. (i) In Section 6.5.2, we focus on the case of recommending the next startup, i.e., $n = 1$. With only one startup to be recommended, the correlation between the recommended startup and the existing portfolio plays the key role in the decision process. (ii) In Section 6.5.3, we study the cases where multiple recommendations are made, i.e., $n = 3, 5, 10$. In these cases, not only the correlation between each of the new items and the existing investments, but also correlations among the recommended ones are important. (iii) In Section 6.5.4, we further perform a statistical data analysis on the optimal risk-averse level b among the VCs.

6.5.1 Experimental Setup

Data Processing

As described in Section 6.3, we base our experiments on the CrunchBase dataset that we collected. We first divide the CrunchBase dataset into training set and test set with 2:1 ratio for each VC according to investment time. Splitting this way, the total investment number is 69,422 in the training set and 24,138 in the test set.

We label a recorded investment from a VC to a startup as 1, i.e., a positive observation. Since it is a one-class training data [181], we follow [181] to perform a user-oriented negative item sampling process, i.e., for each VC, we sample the same number of negative data points as its observed positive points and label them with 0. We train the PMF model to obtain the latent factors for the VCs and startups as well as the probabilistic representation of the VC latent factors (as discussed in Section 6.4.2). Note that our focus is not on the performance comparison against the state-of-the-art recommendation methods, but on investigating how the proposed portfolio-based algorithms can *improve* the recommendation results. The choice of PMF enables a coherent view of the effectiveness of the proposed method as it enables pure model-based mean/variance/covariance estimation for building portfolios.



(a) Risk-averse level b (b) Candidate set size N

Figure 6.3: Performance against (a) risk-averse level b and (b) candidate set size N, when $n = 1$. The computation time is calculated as per test VC.

Compared Algorithms

Three types of state-of-the-art algorithms are compared: the conventional recommendation algorithms, portfolio-based algorithms and adaptive-b portfolio-based algorithms. As described in Section 6.4.3, we always first determine a candidate item set $\mathcal{I}^c$ obtained as the top-N items from PMF, before applying any item selection and ranking algorithm.

Random sampling (Random). As a baseline, we compare our results with randomly-chosen n startups from the candidate set.

PMF. PMF method directly gives the top-n startup determined by the MAP estimation obtained from the PMF.

Portfolio-based methods. These methods include Sampling, Sequential Selection (Seq), Individual Score Ranking (Idv), Weight Ranking (Weight), and Weight Filtering (Filtering). Details of each algorithm are described in Section 6.4.3.

Adaptive-b portfolio-based methods. These methods adopt a personalised risk-averse level b for each VC, as described in Section 6.4.4. We denote them with '-A' following the algorithm's name.

Evaluation Measures

As the task falls into the category of top-N recommendation based on implicit data, we use the similar evaluation method described in Chapter 3 to evaluate the recommendation performances with the following ranking evaluation measures: Precision (P@n), Normalised Discounted Cumulative Gain (NDCG@n) [166], and Mean Reciprocal Rank (MRR@n) [182]. For each algorithm, we calculate the recommendation performances (with respect to these three measures) in regard to each test VC, then average for all test VCs to get the average performances.

6.5.2 Next Startup Recommendation

In this subsection, we focus on the case of $n = 1$, i.e., only one startup is recommended for each test VC. In this case, Sampling, Idv, and Seq are essentially the same, denoted as Portfolio. We compare Portfolio and its adaptive-b version Portfolio-A with the baseline algorithm PMF. As P@1, NDCG@1 and MRR@1 provide exactly the same result in the case $n = 1$, we only use P@1 as the measure here.

Figure 6.2 shows the result comparison between PMF, Portfolio and the adaptive-b version

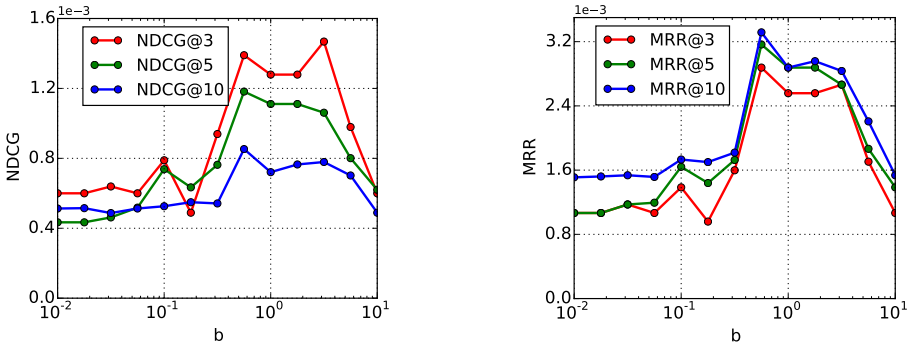
Table 6.2: Performance comparison by different algorithms. The improvement(-A) is calculated from the best Portfolio(-A) algorithm over PMF for each measure. (All numbers except percentages are in the unit of 0.001.)

n	@3			@5			@10		
Measure / Rec	P	NDCG	MRR	P	NDCG	MRR	P	NDCG	MRR
Random	0.746	0.849	1.662	0.665	0.646	1.452	0.659	0.647	1.859
PMF	0.853	0.829	1.492	0.895	0.87	1.939	0.703	0.748	2.288
Sampling	1.492	1.729	3.41	1.279	1.235	3.218	0.959	0.955	3.177
Seq	1.279	1.429	2.771	1.151	1.126	2.931	0.831	0.871	2.931
Idv	1.279	1.239	2.451	1.151	1.105	2.771	0.959	0.987	2.937
Weight	1.066	1.279	2.558	0.767	0.946	2.558	0.703	0.741	2.835
Filtering	1.066	1.35	2.771	0.895	0.927	2.398	0.767	0.781	2.394
Improvement	74.9%	108.6%	128.6%	42.9%	42.0%	66.0%	36.4%	32.0%	38.9%
Sampling-A	5.968	6.126	11.509	5.243	5.395	12.479	3.964	4.112	12.293
Seq-A	1.705	1.879	3.73	1.662	1.74	3.986	1.087	1.299	4.276
Idv-A	1.705	1.768	3.41	1.662	1.616	3.89	1.087	1.304	4.452
Weight-A	1.066	1.279	2.558	0.767	1.008	2.685	0.703	0.786	2.92
Filtering-A	1.705	1.807	3.41	1.407	1.391	3.325	1.023	1.163	3.773
Improvement-A	599.6%	639.0%	671.4%	485.8%	520.1%	543.6%	463.9%	449.7%	437.3%

Portfolio-A, for different latent space dimensions ($k = 25$ and $k = 50$). The candidate size N and the risk-averse level b are both tuned to optimal to obtain the Portfolio performance.

From Figure 6.2, we have the following observations. (i) For both cases ($k = 25$ and $k = 50$), Portfolio and Portfolio-A perform significantly better than PMF. (ii) Comparing between $k = 25$ and $k = 50$, the performance of PMF keeps unchanged, whilst the performance improvements by Portfolio and Portfolio-A are even higher when $k = 50$. (iii) In the case of $k = 50$, Portfolio is outperformed by Portfolio-A. These facts show the effectiveness of our proposed algorithms over PMF, indicating that recommendations with risk concerns are superior in catching the VCs' investment behaviour, and different VCs have different risk-averse levels. We will extend these discussions in Section 6.5.4.

In Figure 6.3, we show the effect of parameters b and candidate size N. From Figure 6.3(a), we can see that the performance peaks when the global risk-averse level $b = 1$. The global b reflects the overall risk-averse level for all test VCs, and the peak value indicates that a certain risk-averse level optimally catches the VCs' overall investment behaviour and leads to the best recommendation result. In Figure 6.3(b), we show the effect of tuning the candidate size N with the corresponding computational time shown as a reference. We can see that when the candidate set gets larger, the performance first increases and then drops to a lower level. It indicates that though an increasing candidate size N adds more options for the algorithm to choose, an oversized candidate set may also mislead the algorithm due to overfitted estimation of latent factors from PMF. The computation time increases linearly as the candidate set enlarges, so one may find a trade-off between the candidate size N and the computation speed as desired.



(a) NDCG@n (b) MRR@n

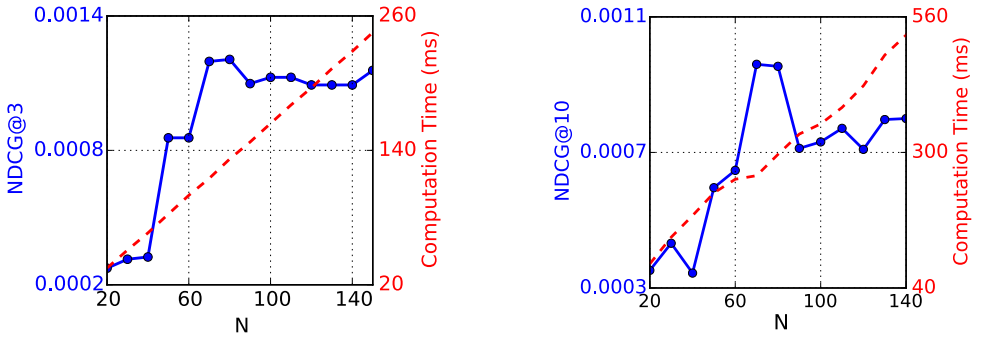
Figure 6.4: Performance against risk-averse level b, evaluated by (a) NDCG@n, and (b) MRR@n.

6.5.3 Next Top-n Startup Recommendation

Here we present the results for multiple item recommendations. In this task, ranking measures NDCG@n and MRR@n are also used in addition to P@n. In Table 6.2, we compare the results between baseline algorithms Random and PMF, portfolio-based algorithms and adaptive-b portfolio-based algorithms. All the (hyper-)parameters are optimised with cross-validation. Because of the severe sparsity of the dataset, the numbers are all very small, nevertheless, the comparison of numbers and the improvements are still meaningful. From this table, we can make the following observations. (i) All the proposed algorithms have great improvements over the results of PMF for all three measures (with a few exceptions for Weight and Filtering), showing the effectiveness of our algorithms in a multiple item recommendation task generally. (ii) Among the (non-adaptive) portfolio-based methods, Sampling, Seq and Idv perform better than Filtering or Weight. This fact indicates that top-down algorithms like Filtering and Weight, which filter out items according to the *direct* portfolio optimisation weight for the *overall* joint portfolio (invested startups plus *all* the candidate startups), do not work as well as the group-selection-based Sampling, or the bottom-up Seq and Simple. This is due to the discrepancy between the weights learnt by a global optimisation and the weights learnt directly for the chosen group, as mentioned in Section 6.4.3. (iii) Adaptive-b portfolio-based algorithms perform better than non-adaptive ones, showing that each VC's risk-averse level is indeed different, so by adaptively fitting the VC's own risk-averse level, the performance can be further improved. (iv) Sampling(-A) outperforms all other algorithms. Again we ascribe its superior performances to its group-selection nature, as according to Eq. (6.2), a group selection method can achieve the best results. The other methods Seq(-A), Idv(-A), Weight(-A) and Filtering(-A) are further approximations than Sampling to approach the exact solution. (iv) Among the two baselines, PMF performs better than Random, indicating the effectiveness of the PMF model to catch the latent factors of VCs and startups.

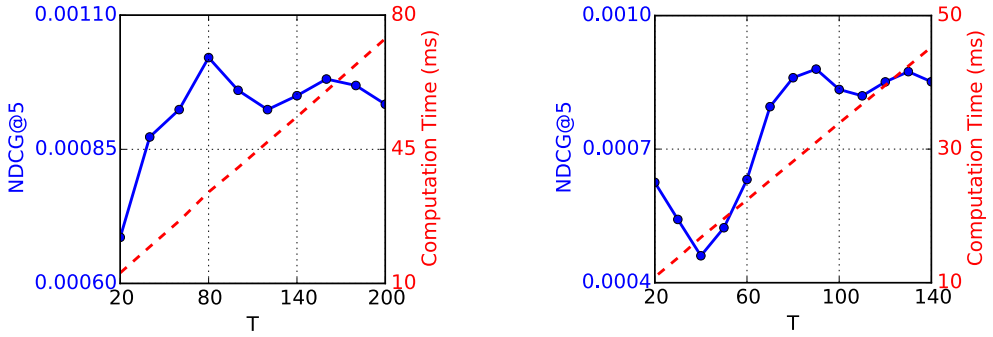
Parameter Tuning

In Figure 6.4 we present the influence of b evaluated by NDCG and MRR, with different $n = 3, 5, 10$. From Figure 6.4, we can see that for each n and each measure, the performance has a peak around $b = 1$, which is consistent with the case of P@1 in Section 6.5.2. Furthermore, comparing different top-n tasks, as n increases, NDCG@n decreases whilst MRR@n increases. This can be explained by the sparsity of



(a) $n = 3$ (b) $n = 10$

Figure 6.5: Impact of candidate size N on performance and computational time when (a) $n = 3$ and (b) $n = 10$. The computation time is calculated as per test VC.



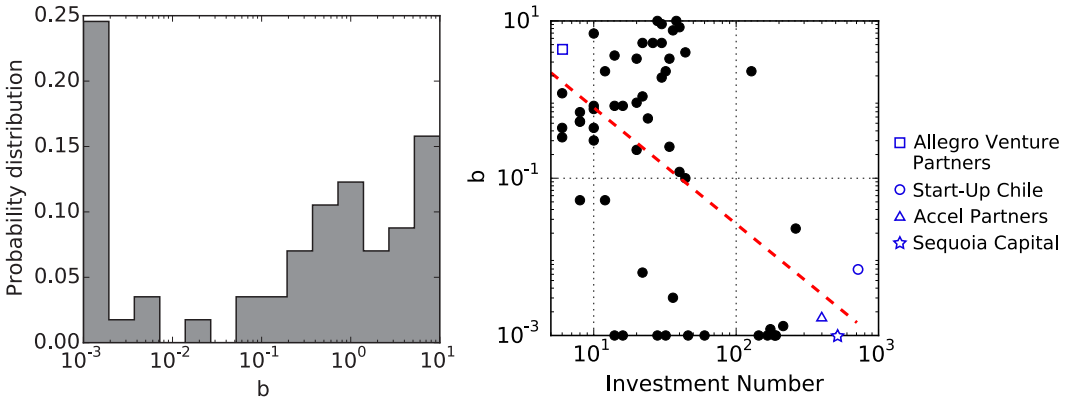
(a) Sampling (b) Sampling-A

Figure 6.6: Impact of sampling number T on the performance of (a) Sampling and (b) Sampling-A. The computation time is calculated as per test VC.

the dataset. When only a small number of recommendations are made (e.g., 3), only a smaller number of VCs are provided with the correct recommendations within the recommendation list. Whereas when the number of recommendations is enlarged (e.g., 10), more users are provided with correct recommendation within the longer recommendation list. According to the definition of MRR [182], only the first correct recommendation counts. Thus, the result of MRR always increases with n in this case. On the other hand, NDCG considers the whole ranking list in a discounted manner, and, due to the sparseness of the dataset, it naturally decreases as n increases.

In Figure 6.5, we plot the influence of the candidate size N for the algorithm Seq, when $n = 3$ and $n = 10$. We can see that the performance first increases as the candidate size gets larger, then slightly drops after peaking around $N = 70$. This result may be due to the overfitting of PMF as mentioned before. Meanwhile, we plot the computation time for each N accordingly. We can see the computation time increases linearly with the candidate size N. Similar to the case when $n = 1$, we may trade off some performance for the computation speed by choosing a smaller candidate set than optimal.

In Figure 6.6, we plot the influence of the sampling number T in Algorithm 6.1 on the performances for both the non-adaptive and the adaptive-b cases. We can see that the performance peaks around $T = 120$ for both cases. The decrease of performance after the peak in Figure 6.6(b) may also be caused by the overfitting of the PMF model. Again, for the sampling method, we may also seek a trade-off



(a) b distribution (b) Correlation

Figure 6.7: Data analysis on the personalised b. (a) Distribution of personalised b. (b) Correlation between VC's investment number and b for majority VCs.

between the ranking performance and efficiency by tuning the sampling number T.

6.5.4 Risk-Averse Level Analysis

In Figure 6.7(a), we plot the distribution of b optimised for individual VCs. We can see that VCs generally form two clusters: a risk-sensitive group whose risk-averse levels b are larger than 0.1 and a risk-neutral group whose risk-averse levels are much smaller. We also have interesting findings on the relationship between the number of investments made by a VC and its optimal risk-averse level b, shown in Figure 6.7(b). Here we applied a log-scale on the investment number, because the VCs' investment activity distribution is power-law [173]. We can find that on the log-log plot, there is a slight negative correlation between the two: companies holding a large number of investments tend to be more risk-neutral, whilst companies with smaller investment scales tend to have higher risk-averse levels (more risk-sensitive). By inspecting company names, we can find some of the largest VCs in the world, such as Start-Up Chile, Sequoia Capital and Accel Partners, fall in the category of the risk-neutral group, whereas smaller VCs, such as Allegro Venture Partners, are more risk-averse. These companies are tagged on Figure 6.7(b) for reference. These observations coincide with the intuition that the fewer investments held by a VC, the more careful it should be in making new investments, whereas, for a VC with a great number of investments, the risk may have already been diversified in its holding portfolio, and thus there is less risk concerns in making new investments compared to smaller VCs.

6.6 Concluding Remarks

In this chapter, we proposed a portfolio optimisation framework to solve the information filtering problem in venture finance, specifically by optimising the joint portfolio of VC's holding investments and potential investment opportunities. We exploited the variance defined on latent factors using a probabilistic matrix factorisation model, and optimised the joint portfolio towards a trade-off between expected preference and uncertainty. We divided the problem into two connected sub-problems including an item selection problem and a portfolio optimisation problem, and proposed five different algorithms to solve it. Through the experiments, we demonstrated significant improvement by using our portfolio-based algorithms and adaptive-b portfolio-based algorithms, compared with a direct PMF approach. In addition,

we discussed the influence of the risk-averse level b, and conducted a data analysis over the distribution of risk-averse levels among the VCs.

Though we have shown significant improvements by our method, the analysis was based on a dataset related to finance. Further analysis is to be carried in traditional recommendation scenarios. We leave this part to future work.

Chapter 7

Conclusions and Future Plan

This thesis presented an interactive recommendation process for solving cold-start problems, in which the two goals – learning and recommendation – are integrated in order to maximise the overall performance during a period of time. The consequence of such integration is an exploitation-exploration trade-off: during each interaction stage, the recommendations serve as both the information source for the system to learn about the new user (item) and the information source to satisfy the need of the user (item). We therefore need to choose the recommendations intelligently to balance between the two. We formulated the problem using the multi-armed bandit and POMDP, and discussed both the situation of single-item and that of multiple-item recommendations at each interaction stage. The case of multiple item recommendations further leads to discussions on item diversification and risk-aware recommendation problems. Further, in this section, we discuss several possible future directions of research following this thesis.

7.1 Thesis Contributions

We formulated the proposed objective and discussed several related interconnected aspects based on the framework in this thesis, including the exploitation-exploration trade-off, resource allocation, and diversification.

In Chapter 3, we studied a sequential interactive recommendation process, in which one item is shown in each interaction round. We related the exploitation-exploration trade-off in this scenario with a multi-armed bandit problem in which a large number of arms are present. The presence of a large number of available items first required us to find a low-dimensional feature space. We utilised alternative least squares to conduct PMF in order to obtain the probabilistic representation of the feature vectors. Next, we used Thompson sampling to achieve EE by using sampled feature vectors (through their probabilistic distributions) in the decision making process. Then, we assumed that the corresponding uncertainties of ratings come entirely from the users, leading to a series of linear-bandit algorithms. In empirical studies, we demonstrated that the developed algorithms lead to significant performance improvements over several strong baselines for dealing cold-start problems, including interview processes, active learning and greedy selection. In addition, we showed that the proposed exploitation-exploration algorithms can also automatically adapt to users' taste drifts during the interactions.

In Chapter 4, we extended the interactive recommendation process to consider multiple recommendations in each interaction for two consecutive stages. We used the item cold-start scenario as a working example. With the proposed setting, the cold-start recommendation problem was transformed into a resource allocation problem. We argued that in order to achieve an optimised overall performance over two stages, we should not focus exclusively on exploration at the first stage. Instead, the users allocated to the first stage should also contribute to the overall utility. We formulated the problem with POMDP to obtain its exact solution. Both a multivariate Gaussian model and a matrix factorization model were used. We found that, the initial-stage users should have high expected returns according to the prior information; and, at the same time, they should also be highly correlated with potential selections in the next stage. The second aspect enables the system to best utilise the deviated feedback from prior information in the second stage, and so led to a guided EE process. Based on this point, we then proposed an approximate algorithm GEE, which adopts the following process: first, the pseudo-selections of the second stage are determined optimistically, and then the initial-stage users are determined. The effectiveness of the proposed algorithms was confirmed through both simulated experiments and experiments on the MovieLens dataset.

In Chapter 5, we focused intensively on the item diversification. Our argument was that the diversification of items should be achieved with regard to risk-awareness during the recommendation process. The risk refers to the uncertainty in predicting user-item preferences and originates from the uncertainty in learnt latent feature vectors of users. We defined the uncertainty of a user latent factor as a function of the number and range of the items rated by the users in the past. Either a fewer number of available ratings, or a wider variety of rated items can lead to a higher level of uncertainty in one or multiple components of the user's latent feature factors, and eventually result in a higher risk in the rating prediction of items. As the risk in predicting different items can be correlated, reducing the risk using a portfolio of items was possible. We utilised the concept of portfolio theory from economics and proposed a portfolio diversification ranking algorithm. Our algorithm LFP captured two levels of diversification: the system-level diversity tuned by an external parameter and the personal-level diversity adjusted adaptively according to the latent feature vectors. Through our experiments, we demonstrated the effectiveness of LFP for adapting result diversification to the users' needs without accessing to explicit item properties. In addition, we also showed that LFP is capable of effectively adjusting the trade-off between the relevance and the diversity of recommended items, and thus could further contribute to the overall recommendation quality.

In Chapter 6, we related the diversification algorithm proposed in Chapter 5 back to the interactive recommendation process considering the case of the venture capital investment opportunity recommendation with the CrunchBase dataset. We proposed a joint portfolio optimisation process, arguing that optimising the future recommendations together with the user's past (holding) investments can lead to an offset of the risk inherited in the user's holding portfolio. We divided the problem into two connected sub-problems including an item selection problem and a portfolio optimisation problem, and proposed five different algorithms to solve it. Through the experiments, we demonstrated significant improvement

by using our algorithms, compared to a direct PMF approach. In addition, we discussed the influence of the risk-averse level, and conducted a data analysis over the distribution of risk-averse levels among the users.

Bringing the above aspects together, we can see that *uncertainty* plays an important part for both EE and diversification aspects. In EE, uncertainty in prediction requires us to explore and enables us to learn from feedback, and, in diversification, uncertainty in feature vectors requires us to diversify the recommendations in a list.

7.2 Future Work

There are various directions into which the work described in this thesis can be extended. We have identified three potential topics. The first regards the cold-start problem, the second focuses on the role of "uncertainty", and the third is concerned with the goal of the interactive recommendation process discussed in the thesis.

Preference Prediction between New Users and New Items

Existing work has utilised content information in assisting the recommendation process where both the users and items are new. For example, in [113] the authors studied the prediction task between new users and new items. They, however, used the user demographic information and item content features, and thus fell out of the scope of CF. To our knowledge, there has been no previous CF focused research to discuss the prediction problem between new users and new items.

This problem could be potentially tackled using an interactive process similar to that described in this thesis. In Chapter 3, we have discussed Thompson Sampling which may be useful to solve this special case within the ICF framework. In Thompson Sampling, both the users and items are represented by probability distributions and thus, if we assign prior distributions to both the new items and the new users we could use Thompson Sampling directly. There are also other potential methods, for example, we could alternate between a new user problem and a new item problem, i.e., alternate between a recommendation algorithm for cold-start users and a recommendation algorithm for cold-start items. A more theoretical solution could be established using POMDP with a full probabilistic description of all the users and items (including cold-start ones) for an exact solution.

Unifying EE and Diversification

In this thesis, we have discussed the role of correlations between users/items for both the interactive recommendation task and the diversification problem. In Chapter 4, we intensively focused on the correlations between the users of the two stages, while in Chapter 5 and Chapter 6 we focused on the correlations of the users recommended at the same time. The two different aspects, however, can be related to each other which we did not discuss. On one hand, in the EE task, the goal of achieving maximal utility over a period of time naturally requires both exploitation and exploration of each recommendation stage, and (if multiple items are included in each stage) the diversification of items (see Section 4.2.1 in Chapter 4); however, on the other hand, portfolio theory achieves diversification with the explicit goal of balancing between the utility and the uncertainty (see Eq. 5.11) which differs from the maximal utility

goal. The question is, whether these two goals are the same, and, if not, whether it is possible to address them both within one goal. This problem may be answered by formulating the exact solution to carefully analyse the diversification aspect inherit in the EE task and compare it to the diversification achieved by the portfolio theory.

Goal-Driven Exploitation-Exploration

The above-mentioned problem also leads us to consider a goal-driven EE problem, where the goal is to explicitly target to an evaluation metric other than the utility itself. For example, by setting the goal as maximising the overall utility while minimising the overall risk, we can explicitly bring diversity of items into the process. This will be beneficial for both the learning process and for serendipity considerations. We can also use goals such as the collective NDCG or collective MRR over stages to promote ranking in each recommendation list.

As seen in Chapter 4, by integrating the goal into the POMDP framework, the exact solution can be obtained using value iteration. The Thompson Sampling method mentioned in Chapter 3 can also be utilised to obtain an approximate solution to the problem. As the goal differs from that of a multi-armed bandit, index-based MAB methods may not be directly applicable.

Bibliography

[1] Riccardo Bambini, Paolo Cremonesi, and Roberto Turrin. A recommender system for an iptv service provider: a real large-scale production environment. In *Recommender Systems Handbook*, pages 299–331. Springer, 2011.

[2] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 230–237, 1999.

[3] Nicholas J Belkin and W Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[4] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[5] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[6] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to Recommender Systems Handbook*. Springer, 2011.

[7] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[8] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: state of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[9] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.

[10] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[11] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:4, 2009.

[12] Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. Transfer learning in collaborative filtering for sparsity reduction. In *Proceedings of the 24th AAAI Conference on Artifical Inteligence*, volume 10, pages 230–235, 2010.

[13] 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, 2002.

[14] Mustansar Ali Ghazanfar and Adam Prugel-Bennett. A scalable, accurate hybrid recommender system. In *Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on*, pages 94–98, 2010.

[15] 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, 2014.

[16] 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, 2013.

[17] 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*, pages 315–324, 2011.

[18] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. On bootstrapping recommender systems. In *Proceedings of the 19th ACM International Conference on Information and knowledge management*, pages 1805–1808, 2010.

[19] Jingwei Xu, Yuan Yao, Hanghang Tong, Xianping Tao, and Jian Lu. Ice-breaking: mitigating cold-start recommendation problem by rating comparison. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3981–3987, 2015.

[20] Jun Wang, Stephen Robertson, Arjen P de Vries, and Marcel JT Reinders. Probabilistic relevance ranking for collaborative filtering. *Information Retrieval*, 11(6):477–497, 2008.

[21] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the 5th ACM Conference on Recommender systems*, pages 109–116, 2011.

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

[23] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[24] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[25] Jun Wang and Jianhan Zhu. Portfolio theory of information retrieval. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in information retrieval*, pages 115–122, 2009.

[26] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World wide web*, pages 661–670, 2010.

[27] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pages 817–824, 2008.

[28] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[29] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003.

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

[31] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[32] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

[33] Neil Rubens, Ryota Tomioka, and Masashi Sugiyama. Output divergence criterion for active learning in collaborative settings. *IPSJ Online Transactions*, 2:240–249, 2009.

[34] 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, 2015.

[35] Abhay S Harpale and Yiming Yang. Personalized active learning for collaborative filtering. In *Proceedings of the 31st annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 91–98, 2008.

[36] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *Proceedings of the 4th ACM international conference on Web Search and Data Mining*, pages 595–604, 2011.

[37] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 1987.

[38] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Implicit user modeling for personalized search. In *Proceedings of the 14th ACM International Conference on Information and knowledge management*, pages 824–831, 2005.

[39] Xiaoran Jin, Marc Sloan, and Jun Wang. Interactive exploratory search for multi page search results. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 655–666, 2013.

[40] Weinan Zhang, Shuai Yuan, and Jun Wang. Optimal real-time bidding for display advertising. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1077–1086, 2014.

[41] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual International ACM SIGIR Conference on Research and Development in information retrieval*, pages 335–336, 1998.

[42] Cheng Xiang Zhai, William W Cohen, and John Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 10–17, 2003.

[43] Harr Chen and David R Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *Proceedings of the 29th annual International ACM SIGIR Conference on Research and Development in information retrieval*, pages 429–436, 2006.

[44] Neil Hurley and Mi Zhang. Novelty and diversity in top-n recommendation–analysis and evaluation. *ACM Transactions on Internet Technology*, 10(4):14, 2011.

[45] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 210–217, 2010.

[46] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, pages 22–32, 2005.

[47] Mi Zhang and Neil Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 123–130, 2008.

[48] Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 659–666, 2008.

[49] Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. Selectively diversifying web search results. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1179–1188, 2010.

[50] Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. Intent-aware search result diversification. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 595–604, 2011.

[51] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[52] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM Conference on Recommender Systems*, pages 39–46, 2010.

[53] Miklós Kurucz, András A Benczúr, and Károly Csalogány. Methods for large scale svd with missing values. In *Proceedings of KDD Cup and Workshop*, volume 12, pages 31–38, 2007.

[54] Shuang-Hong Yang, Bo Long, Alexander J Smola, Hongyuan Zha, and Zhaohui Zheng. Collaborative competitive filtering: learning recommender using context of user choice. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 295–304, 2011.

[55] 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.

[56] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[57] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434, 2008.

[58] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of Machine Learning Research*, 3:993–1022, 2003.

[59] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[60] Bradley N Miller, Istvan Albert, Shyong K Lam, Joseph A Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 263–266, 2003.

[61] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[62] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[63] 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, 1998.

[64] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*, pages 107–144. Springer, 2011.

[65] Jon Herlocker, Joseph A Konstan, and John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval*, 5(4):287–310, 2002.

[66] 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, 2001.

[67] 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, 1995.

[68] Maurice George Kendall. Rank correlation methods. 1948.

[69] Alejandro Bellogin and Javier Parapar. Using graph partitioning techniques for neighbour selection in user-based collaborative filtering. In *Proceedings of the 6th ACM Conference on Recommender Systems*, pages 213–216, 2012.

[70] Hyeong-Joon Kwon, Tae-Hoon Lee, and Kwang-Seok Hong. Improved memory-based collaborative filtering using entropy-based similarity measures. In *Proceedings of the 2009 International Symposium on Web Information Systems and Applications (WISA09)*, pages 29–34, 2009.

[71] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 158–167, 2000.

[72] Yi Zhang and Jamie Callan. Maximum likelihood estimation for filtering thresholds. In *Proceedings of the 24th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 294–302, 2001.

[73] Fidel Cacheda, Víctor Carneiro, Diego Fernández, and Vreixo Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web*, 5(1):2, 2011.

[74] Adele E Howe and Ryan D Forbes. Re-considering neighborhood-based collaborative filtering parameters in the context of new data. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 1481–1482, 2008.

[75] Kai Yu, Anton Schwaighofer, Volker Tresp, Xiaowei Xu, and H-P Kriegel. Probabilistic memory-based collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):56–69, 2004.

[76] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 501–508, 2006.

[77] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 95–104, 2007.

[78] Badrul M Sarwar, Joseph A Konstan, Al Borchers, Jon Herlocker, Brad Miller, and John Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Proceedings of the 1998 ACM Conference on Computer supported cooperative work*, pages 345–354, 1998.

[79] Nathaniel Good, J Ben Schafer, Joseph A Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, and John Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the 16th National Conference on Artificial Intelligence and 11th Conference on Innovative Applications of Artificial Intelligence*, pages 439–446, 1999.

[80] Geoffrey I Webb, Michael J Pazzani, and Daniel Billsus. Machine learning for user modeling. *User Modeling and User-Adapted Interaction*, 11(1-2):19–29, 2001.

[81] Lyle H Ungar and Dean P Foster. Clustering methods for collaborative filtering. In *AAAI Workshop on Recommendation Systems*, volume 1, pages 114–129, 1998.

[82] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *5th IEEE International Conference on Data Mining*, pages 4–pp, 2005.

[83] Badrul M Sarwar, George Karypis, Joseph Konstan, and John Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the 5th International Conference on Computer and Information Technology*, volume 1, 2002.

[84] Gediminas Adomavicius and Alexander Tuzhilin. Using data mining methods to build customer profiles. *Computer*, (2):74–82, 2001.

[85] Bamshad Mobasher. Data mining for web personalization. In *The Adaptive Web*, pages 90–135. Springer, 2007.

[86] Michelle Keim Condliff, David D Lewis, David Madigan, and Christian Posse. Bayesian mixed-effects models for recommender systems. In *ACM SIGIR99 Workshop on Recommender Systems: Algorithms and Evaluation*, volume 15, 1999.

[87] Han-Saem Park, Ji-Oh Yoo, and Sung-Bae Cho. A context-aware music recommendation system using fuzzy bayesian networks with utility theory. In *Fuzzy Systems and Knowledge Discovery*, pages 970–979. Springer, 2006.

[88] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[89] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, pages 791–798, 2007.

[90] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, 1999.

[91] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th annual International ACM SIGIR Conference on Research and Development in informaion retrieval*, pages 259–266, 2003.

[92] Alexandrin Popescul, David M Pennock, and Steve Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pages 437–444, 2001.

[93] Benjamin M Marlin. Modeling user rating profiles for collaborative filtering. In *Advances in Neural Information Processing Systems*, page None, 2003.

[94] Michael W Berry. Large-scale sparse singular value computations. *International Journal of Supercomputer Applications*, 6(1):13–49, 1992.

[95] Simon Funk. Netflix update: Try this at home, 2006.

[96] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, volume 2007, pages 5–8, 2007.

[97] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Major components of the gravity recommendation system. *ACM SIGKDD Explorations Newsletter*, 9(2):80–83, 2007.

[98] Markus Weimer, Alexandros Karatzoglou, and Alex Smola. Adaptive collaborative filtering. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 275–282, 2008.

[99] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *8th IEEE International Conference on Data Mining*, pages 263–272, 2008.

[100] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

[101] Stanley Loh, Fabiana Lorenzi, Roger Granada, Daniel Lichtnow, Leandro Krug Wives, and José Palazzo Moreira de Oliveira. Identifying similar users by their scientific publications to reduce cold start in recommender systems. In *Proceedings of the 5th International Conference on Web Information Systems and Technologies*, pages 593–600, 2009.

[102] 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.

[103] Cane Wing-ki Leung, Stephen Chi-fai Chan, and Fu-lai Chung. An empirical study of a cross-level association rule mining approach to cold-start recommendations. *Knowledge-Based Systems*, 21(7):515–529, 2008.

[104] 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.

[105] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K Lam, Sean M McNee, Joseph A Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, pages 127–134, 2002.

[106] 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.

[107] Neil Rubens, Dain Kaplan, and Masashi Sugiyama. Active learning in recommender systems. In *Recommender Systems Handbook*, pages 735–767. Springer, 2011.

[108] Neil Rubens and Masashi Sugiyama. Influence-based collaborative active learning. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 145–148, 2007.

[109] Genichi Taguchi. *Introduction to quality engineering: designing quality into products and processes*. 1986.

[110] R. Jin and L. Si. A bayesian approach toward active learning for collaborative filtering. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 278–285, 2004.

[111] A. Gunawardana and C. Meek. Tied boltzmann machines for cold start recommendations. In *Proceedings of the 2008 ACM Conference on Recommender systems*, pages 19–26, 2008.

[112] Li-Tung Weng, Yue Xu, Yuefeng Li, and Richi Nayak. Exploiting item taxonomy for solving cold-start problem in recommendation making. In *20th IEEE International Conference on Tools with Artificial Intelligence, ICTAI'08.*, volume 2, pages 113–120, 2008.

[113] Seung-Taek Park and Wei Chu. Pairwise preference regression for cold-start recommendation. In *Proceedings of the 3rd ACM Conference on Recommender Systems*, pages 21–28, 2009.

[114] Stephen E Robertson. The probability ranking principle in ir. *Journal of Documentation*, 1977.

[115] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 2nd edition, 1979.

[116] Joseph John Rocchio. Relevance feedback in information retrieval. 1971.

[117] Jinxi Xu and W Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, 1996.

[118] Yuanhua Lv and ChengXiang Zhai. Positional relevance model for pseudo-relevance feedback. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 579–586, 2010.

[119] T.H. Cormen, CE Leiserson, RL Rivest, and C. Stein. Greedy algorithms. *Introduction to Algorithms*, 2001.

[120] Richard Weber et al. On the gittins index for multiarmed bandits. *The Annals of Applied Probability*, 2(4):1024–1033, 1992.

[121] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

[122] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2005.

[123] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[124] M. Dudik, D. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, and T. Zhang. Efficient optimal learning for contextual bandits. *arXiv preprint arXiv:1106.2369*, 2011.

[125] Sarah Filippi, Olivier Cappe, Aurélien Garivier, and Csaba Szepesvári. Parametric bandits: The generalized linear case. In *Advances in Neural Information Processing Systems*, pages 586–594, 2010.

[126] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[127] Wei Li, Xuerui Wang, Ruofei Zhang, Ying Cui, Jianchang Mao, and Rong Jin. Exploitation and exploration in a performance based contextual advertising system. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 27–36, 2010.

[128] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[129] Martin L Puterman and Moon Chirl Shin. Modified policy iteration algorithms for discounted markov decision problems. *Management Science*, 24(11):1127–1137, 1978.

[130] Andrey Kolobov. Planning with markov decision processes: An ai perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–210, 2012.

[131] Arthur Guez, David Silver, and Peter Dayan. Scalable and efficient bayes-adaptive reinforcement learning based on monte-carlo tree search. *Journal of Artificial Intelligence Research*, pages 841–883, 2013.

[132] Jilles S Dibangoye, Christopher Amato, and Arnoud Doniec. Scaling up decentralized mdps through heuristic search. *arXiv preprint arXiv:1210.4865*, 2012.

[133] John Rust. Structural estimation of markov decision processes. *Handbook of Econometrics*, 4(4), 1994.

[134] Sven Koenig and Reid Simmons. Xavier: A robot navigation architecture based on partially observable markov decision process models. *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, pages 91–122, 1998.

[135] Georgios Theocharous, Khashayar Rohanimanesh, and Sridhar Maharlevan. Learning hierarchical observable markov decision process models for robot navigation. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 511–516, 2001.

[136] Esther Levin, Roberto Pieraccini, and Wieland Eckert. Using markov decision process for learning dialogue strategies. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 1, pages 201–204, 1998.

[137] Esther Levin, Roberto Pieraccini, and Wieland Eckert. Learning dialogue strategies within the markov decision process framework. In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 72–79, 1997.

[138] Qinru Qiu and Massoud Pedram. Dynamic power management based on continuous-time markov decision processes. In *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*, pages 555–561, 1999.

[139] Guy Shani, Ronen I Brafman, and David Heckerman. An mdp-based recommender system. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 453–460, 2002.

[140] Tariq Mahmood and Francesco Ricci. Learning and adaptivity in interactive recommender systems. In *Proceedings of the 9th International Conference on Electronic commerce*, pages 75–84, 2007.

[141] Tariq Mahmood and Francesco Ricci. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM Conference on Hypertext and hypermedia*, pages 73–82, 2009.

[142] Craig Boutilier. A pomdp formulation of preference elicitation problems. In *Proceedings of the 18th National Conference on Artificial Intelligence and 14th Conference on Innovative Applications of Artificial Intelligence*, pages 239–246, 2002.

[143] Finale Doshi and Nicholas Roy. The permutable pomdp: fast solutions to pomdps for preference elicitation. In *Proceedings of the 7th International joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 2008.

[144] Nathan N Liu and Qiang Yang. Eigenrank: a ranking-oriented approach to collaborative filtering. In *Proceedings of the 31st annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 83–90, 2008.

[145] Sean M McNee, John Riedl, and Joseph A Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, 2006.

[146] Gulden Uchyigit and Matthew Y Ma. *Personalization techniques and recommender systems*, volume 70. World Scientific, 2008.

[147] Li Hang. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1854–1862, 2011.

[148] George Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the 10th International Conference on Information and Knowledge Management*, pages 247–254, 2001.

[149] Ernesto Diaz-Aviles, Lucas Drumond, Lars Schmidt-Thieme, and Wolfgang Nejdl. Real-time top-n recommendation in social streams. In *Proceedings of the 6th ACM Conference on Recommender Systems*, pages 59–66, 2012.

[150] Shengbo Guo and Scott Sanner. Probabilistic latent maximal marginal relevance. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 833–834, 2010.

[151] Hal R Varian. Economics and search. In *SIGIR Forum*, volume 33, pages 1–5, 1999.

[152] Harry Markowitz. Portfolio selection*. *The Journal of Finance*, 7(1):77–91, 1952.

[153] Xiangyu Wang and Mohan Kankanhalli. Portfolio theory of multimedia fusion. In *Proceedings of the International Conference on Multimedia*, pages 723–726, 2010.

[154] Marc Sloan and Jun Wang. Dynamical information retrieval modelling: a portfolio-armed bandit machine approach. In *Proceedings of the 21st International Conference Companion on World Wide Web*, pages 603–604, 2012.

[155] Leif Azzopardi. The economics in interactive information retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 15–24, 2011.

[156] Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *International Joint Conference on Artificial Intelligence*, volume 99, pages 688–693, 1999.

[157] Thomas J Walsh, István Szita, Carlos Diuk, and Michael L Littman. Exploring compact reinforcement-learning representations with linear regression. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 591–598, 2009.

[158] Paul McJones. Eachmovie collaborative filtering data set. *DEC Systems Research Center*, 249, 1997.

[159] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, pages 297–306, 2011.

[160] Markus Weimer, Alexandros Karatzoglou, Quoc Viet Le, and Alex Smola. Maximum margin matrix factorization for collaborative ranking. *Advances in Neural Information Processing Systems*, 2007.

[161] Xavier Amatriain, Josep M Pujol, Nava Tintarev, and Nuria Oliver. Rate it again: increasing recommendation accuracy by user re-rating. In *Proceedings of the 3rd ACM Conference on Recommender Systems*, pages 173–180, 2009.

[162] Jean Dickinson Gibbons and Subhabrata Chakraborti. *Nonparametric statistical inference*. Springer, 2011.

[163] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unified relevance models for rating prediction in collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 26(3):16, 2008.

[164] Frederi G Viens. Steins lemma, malliavin calculus, and tail bounds, with application to polymer fluctuation exponent. *Stochastic Processes and their Applications*, 119(10):3671–3698, 2009.

[165] Hastagiri P Vanchinathan, Isidor Nikolic, Fabio De Bona, and Andreas Krause. Explore-exploit in top-n recommender systems via gaussian processes. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 225–232, 2014.

[166] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[167] Jun Wang. Mean-variance analysis: A new document ranking theory in information retrieval. In *Advances in Information Retrieval*, pages 4–16. Springer, 2009.

[168] P. Resnick. Personalized filters yes; bubbles no. http://presnick.livejournal.com/21239.html, July 2011.

[169] Andrew Metrick and Ayako Yasuda. *Venture capital and the finance of innovation, 2nd edition*. John Wiley and Sons, Inc, 2008.

[170] Tyzoon T Tyebjee and Albert V Bruno. A model of venture capitalist investment activity. *Management Science*, 30(9):1051–1066, 1984.

[171] Vance H Fried and Robert D Hisrich. Toward a model of venture capital investment decision making. *Financial Management*, pages 28–37, 1994.

[172] Scott Anthony. Is Venture Capital Broken?, 2012.

[173] Thomas Stone, Weinan Zhang, and Xiaoxue Zhao. An empirical study of top-n recommendation for venture finance. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 1865–1868, 2013.

[174] Tarun Bhaskar and Gopi Subramanian. Loan recommender system for microfinance loans: Increasing efficiency to assist growth. *Journal of Financial Services Marketing*, 15(4):334–345, 2011.

[175] Hongke Zhao, Le Wu, Qi Liu, Yong Ge, and Enhong Chen. Investment recommendation in p2p lending: A portfolio perspective with risk management. In *2014 IEEE International Conference on Data Mining*, pages 1109–1114, 2014.

[176] John Hull. *Risk Management and Financial Institutions,+ Web Site*, volume 733. John Wiley & Sons, 2012.

[177] Weinan Zhang, Jun Wang, Bowei Chen, and Xiaoxue Zhao. To personalize or not: A risk management perspective. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 229–236, 2013.

[178] Rong Hu and Pearl Pu. Enhancing recommendation diversity with organization interfaces. In *Proceedings of the 16th International Conference on Intelligent User Interfaces*, pages 347–350, 2011.

[179] Oliver T Alexy, Joern H Block, Philipp Sandner, and Anne LJ Ter Wal. Social capital of venture capitalists and start-up funding. *Small Business Economics*, 39(4):835–851, 2012.

[180] Bent Flyvbjerg. Quality control and due diligence in project management: Getting decisions right by taking the outside view. *International Journal of Project Management*, 31(5):760–774, 2013.

[181] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *The 8th IEEE International Conference on Data Mining*, pages 502–511, 2008.

[182] Ellen M Voorhees et al. The trec-8 question answering track report. In *The 18th Text Retrievl Conference Proceedings*, volume 99, pages 77–82, 1999.