Functional Matrix Factorizations for Cold-Start Recommendation

Ke Zhou
College of Computing
Georgia Institute of Technology
Atlanta, GA 30032
kzhou@gatech.edu

Shuang-Hong Yang
College of Computing
Georgia Institute of Technology
Atlanta, GA 30032
shy@gatech.edu

Hongyuan Zha
College of Computing
Georgia Institute of Technology
Atlanta, GA 30032
zha@cc.gatech.edu

ABSTRACT

A key challenge in recommender system research is how to effectively profile *new* users, a problem generally known as *cold-start* recommendation. Recently the idea of progressively *querying* user responses through an *initial interview* process has been proposed as a useful new user preference elicitation strategy. In this paper, we present *functional matrix factorization* (fMF), a novel cold-start recommendation method that solves the problem of initial interview construction within the context of learning user and item profiles. Specifically, fMF constructs a decision tree for the initial interview with each node being an interview question, enabling the recommender to query a user adaptively according to her prior responses. More importantly, we associate latent profiles for each node of the tree — in effect restricting the latent profiles to be a function of possible answers to the interview questions — which allows the profiles to be gradually refined through the interview process based on user responses. We develop an iterative optimization algorithm that alternates between decision tree construction and latent profiles extraction as well as a regularization scheme that takes into account of the tree structure. Experimental results on three benchmark recommendation data sets demonstrate that the proposed fMF algorithm significantly outperforms existing methods for cold-start recommendation.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Experimentation

Keywords

Recommender systems, Cold-start problem, Collaborative filtering, Decision tree, Functional matrix factorization

1. INTRODUCTION

Recommendation systems have become a core component in today's online business world. While E-commerce giants such as Amazon have been greatly benefitted from the abilities of their online recommenders in effectively delivering the *right* items to the *right* people, online startups are increasingly becoming aware of this secret and aggressively sharpening their weapons in order to compete. A key challenge for building an effective recommender system is the well-known cold-start problem — *how to provide recommendations to new users?* While existing *collaborative filtering* (CF) approaches to recommendation perform quite satisfactorily for warm-start users (e.g. users purchasing a lot from an online retailer), it could fail completely for fresh users, simply because the system knows very little about those users in terms of their preferences.

Providing effective cold-start recommendations is of fundamental importance to a recommender system. Offering bad recommendations could risk severely the webshop's revenue, market share and overall reputations, because disappointed customers may turn to other competitors and never come back. This problem is more acute for newly-launched recommender systems (e.g. online startups) because their existing user base is small and attracting more new customers are crucial to their survival. Even for relatively mature recommender systems, a vast majority of the (potential) customers are actually cold-start users – in principle, the population of customers typically follows a power-law distribution w.r.t. the participation frequency such that most customers lie in the long-tail part.

A natural approach to solving the cold-start problem is to elicit new user preferences by query users' responses progressively through an initial interview process [19]. Specifically, at the visit of a new user, the recommender initiates several trials. At each trial, the recommender provides a seed item as a question and ask the user for her opinion; based on the user's responses to the questions, the recommender gradually refines its characterization (i.e. profile) of the user so that it can provide more satisfactory recommendations to the user in the future. A good initial interview process should not be time-consuming: the recommender can only ask a very limited number of questions or otherwise the user will become impatient and leave the system. Equally important, the process should also be effective, i.e. the answers collected from the user should be useful for constructing at least a rough profile for the user. It has been convincingly argued that an effective model for designing the interview process should ask the interview questions adaptively, taking into account the user's responses when asking the next question in the interview process, and decision trees have been found to work especially well for this purpose [7, 19, 20]. As an illustration of movie recommendation, in an interview process depicted in Figure 1, the system asks a new user the question "Do you like the movie Lethal Weapon 4?". The user is allowed to answer with either "Like", "Dislike" or "Unknown". The recommender refines its impression about the user and then direct the user to one of the child nodes and then presents the next interview question according to her previous response. For example, if a user chooses "Like" for "Lethal Weapon 4?" at the root node of Figure 1, she will be directed to the left child node and will be asked the question "Do you like the movie Taxi Driver?"

In this paper, we propose *functional matrix factorization* (fMF), a novel method for the construction of such interview decision trees. We argue that it is more effective to combine the matrix factorization model for collaborative filtering and the decision-tree based interview model into a single framework. Our proposed method is based on the low rank factorization of the incomplete user-item rating matrix with an important twist: it restricts the user profiles to be a *function* of the answers to the interview questions in the form of a decision tree, thus the name functional matrix factorization. The function — playing the role of the initial interview — is in the form of a decision tree with each node being an interview question [7, 20]. Both the decision tree and the item profiles need to be learned; for that we propose an iterative optimization algorithm that alternates between decision tree construction and latent profiles extraction.

Our proposed method tends to explore more effectively the correlation between different items. In particular, the low dimensional user profiles at each node of the decision tree and the item profiles adapt to each other to better fit the training data. Thus, correlation between different items is captured by the low dimensional profiles. As a result, the prediction of ratings can be enhanced by making use of the ratings of different items. Experimental results on three benchmark recommendation data sets, the MovieLens data set, the EachMovie data set and the Netflix data set, demonstrate that the proposed fMF algorithm significantly outperforms existing methods in cold-start recommendation.

Outline: In Section2, we briefly review existing studies for cold-start collaborative filtering. In Section 3, we first introduce matrix factorization for collaborative filtering and then present functional matrix factorization for constructing the interview process for cold-start collaborative filtering by restricting the user profiles to be a function in the form of a decision tree. The learning algorithm based on alternative optimization is then proposed with detailed derivations. Then, we evaluate the proposed method on three data sets and analyze the results in Section 4. Finally we conclude our work and present several future directions in Section 5.

2. RELATED WORK

2.1 Collaborative Filtering

Many studies on recommender systems have been focused on collaborative filtering approaches. These methods can be categorized into memory-based and model-based. The reader is referred to the survey papers [1, 26] for a comprehensive summary of collaborative filtering algorithms.

Recently, matrix factorization becomes a popular direction for collaborative filtering [12,14,16,21,24]. These methods are shown to be effective in many applications. Specifically, matrix factorization methods usually seek to associate both users and items with latent profiles represented by vectors in a low dimension space that can capture their characteristics. In [22], a convex relaxation for low rank matrix factorization is proposed and applied to collaborative filtering. A probabilistic model for extracting low dimensional profiles is studies in [12], in which latent variables are introduced to capture the users and items and the EM algorithm is used to estimate the parameters. More recently, fueled by the Netflix competition, several improvements have been proposed including the use of regularized SVD [16], and the idea of matrix factorization combined with neighborhood-based methods [14].

2.2 Cold-Start Collaborative Filtering

There have been several studies on the elicitation strategies for new user preferences. The work [18] surveys several guidelines for user preference elicitation. Several models such as [9, 19, 20] constructed the interview process with a static seed set selected based on measures such as informativeness, popularity and coverage. The recent work of [7] proposed a greedy seed selection algorithm by optimizing the prediction performance. Such seed-based methods are not completely satisfactory because the seeds are selected statically in batch, and they do not reflect the user responses during the interview process. In [20], while discussing the IGCN algorithm, it was mentioned the idea of adaptively selecting the interview questions by fitting a decision tree to predefined user clusters. In particular, each node represents an interview question and the user is directed to one of the child nodes according to her response to the parent question. In [8], the authors proposed an algorithm that fits the decision to the users' ratings. This seems to provide a more disciplined approach than that based on the predefined user clusters discussed in IGCN. Our proposed framework goes one step further by integrating the decision tree construction into the matrix factorization framework of collaborative filtering. We should also mention that the decision tree structure used in those methods exhibit interesting resemblance to the Bayesian network approach in [5].

A complimentary line of research for solving the cold-start problem is to utilize the features of users or items. The content features can be used to capture the similarities between users and items and thus reduce the amount of data required to make accurate predictions [2, 3, 10, 15, 23, 25]. For example, the work of [2] utilizes features of items and users as the prior distribution for latent profiles in matrix factorization. Our method is based on the interview process and does not solely rely on features of users and items.

We also want to mention active learning for collaborative filtering [4,11,13,17]. These methods usually select questions that are optimal with respect to some selection criteria, such as the expected value of information or the distance to the true user model. These methods generally are not suitable for interview process since the selection process usually involves optimizing the selection criteria, which is not efficient enough for online interview.

3. FUNCTIONAL MATRIX FACTORIZATION

In this section, we describe the *functional matrix factor-*

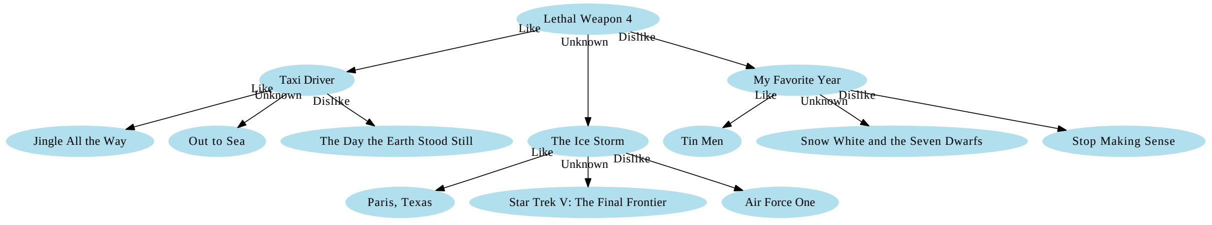

Figure 1: An example for decision trees: Top three levels of a model of depth 6 for MovieLens data set

ization (fMF) method for cold-start collaborative filtering which explores the well-known matrix factorization methods for constructing the interview process. The key innovation is that we parameterize user profiles to be a function of the responses to the possible questions of the interview process and use matrix factorization to compute the profiles.

3.1 Low-Rank Matrix Factorization

Before describing fMF, we first review the matrix factorization method for collaborative filtering. Let r_{ij} denote the rating of user i for item j, where $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, M$. In practice, only a small subset of ratings are observed, denoted by $\mathcal{K} = \{r_{ij} \,|\, (i,j) \in O\}$. The goal of collaborative filtering is to predict the unknown ratings based on ratings in $\mathcal{K}$. Collaborative filtering exploits a basic heuristic that similar users tend to rate similar items in a similar way. One important class of methods for collaborative filtering are based on matrix factorization [12, 24]. Specifically, we associate a K-dimensional vector $u_i \in \mathbb{R}^K$ for each user i and $v_j \in \mathbb{R}^K$ for each item j. The vectors u_i and v_j are usually called user profiles and item profiles since they are intended to capture the characteristics of users and items. The rating r_{ij} of user i for item j can be approximated by a similarity function of u_i and v_j in the low dimensional space, for example, $r_{ij} = u_i^T v_j$.

Given the set of known ratings $\mathcal{K}$, the parameters u_i and v_j can be estimated through fitting the training data by solving the following optimization problem:

$$\min_{u_i, v_j} \sum_{\{(i,j)\} \in O} (r_{ij} - u_i^T v_j)^2.$$

Regularization terms such as the Frobenius norms on the profile vectors can be introduced to avoid overfitting. The problem can be solved by existing numerical optimization methods such as alternating minimization and stochastic gradient descent. In our implementation, we use the alternating optimization for its amenability for the cold-start settings. Specifically, the optimization process performs the following two updates alternatively.

First, for $i = 1, 2, \ldots, N$, minimizing with respect to u_i with all $u_j, j \neq i$ and all v_j fixed:

$$u_i = \operatorname*{argmin}_{u_i} \sum_{(i,j) \in O} (r_{ij} - u_i^T v_j)^2,$$

which is a linear regression problem with squared loss. The closed form solution can be expressed as

$$u_i = \left(\sum_{(i,j) \in O} v_j v_j^T \right)^{-1} \left(\sum_{(i,j) \in O} r_{ij} v_j \right)$$

Then, for $j = 1, 2, \ldots, M$, minimizing with respect to v_j with all $v_i, i \neq j$ and all u_i fixed:

$$v_j = \operatorname*{argmin}_{v_j} \sum_{(i,j) \in O} (r_{ij} - u_i^T v_j)^2,$$

which is also a linear regression problem with similar closed-form solution.

3.2 Functional Matrix Factorization

Now we consider constructing the interview process for cold-start collaborative filtering. Assume that a new user registers at the recommendation system and nothing is known about her. To capture the preferences of the user, the system initiates several interview questions to query the responses from the user. Based on the responses, the system constructs a profile for the user and provides recommendations accordingly.

In the plain matrix factorization model described in Section 3.1, the user profile u_i is estimated by optimizing the ℓ_2 loss on the history ratings r_{ij}. This model does not directly apply to cold-start settings because no rating is observed for the new user prior to the interview process. To build user profiles adaptively according to the user's responses in the course of the interview process, we propose to parameterize the user profile u_i in such a way that the profile u_i is tied to user i's responses in the form of a function, thus the name functional matrix factorization (fMF). More precisely, assume there are P possible *interview questions*. We assume that an answer to a question takes value in the finite set $\{0, 1, \text{Unknown}\}$, representing "Dislike", "Like" and "Unknown", respectively. Furthermore, let a_i denote the P-dimensional vector representing the answers of user i to the P questions. And we tie the profile to the answers by assuming $u_i = T(a_i)$, where T is a function that maps the responses a_i to the user profile $u_i \in \mathbb{R}^k$. To make recommendations for user i, we simply use $r_{ij} = v_j^T T(a_i)$.

Our goal is to learn both T and v_j from the observed ratings $\mathcal{K}$. To this end, substituting $u_i = T(a_i)$ into the low-rank matrix factorization model, we have the following optimization problem:

$$T, V = \operatorname*{argmin}_{T \in \mathcal{H}, V} \sum_{(i,j) \in O} (r_{ij} - v_j^T T(a_i))^2 + \lambda \|V\|^2, \quad (1)$$

where $V = (v_1, \ldots, v_M)$ is the matrix of all item profiles, $\mathcal{H}$ is the space from which the function $T(a)$ is selected and the second term is the regularization term.

Several issues need to be addressed in order to construct the interview process by the above functional matrix factorization. First, the number of all possible interview questions can be quite large (e.g. up to millions of items in movie recommendation); yet a user is only patient enough to answer a few interview questions. Second, the interview process should be adaptive to user's responses, in other words, a follow-up question should be selected based on the user's responses to the previous questions. Therefore, the selection process should be efficient to generate interview questions in real time after the function $T(a)$ is constructed. In addition, since we allow a user to choose "Unknown" to the interview questions, we need to deal with such missing values as well.

Following prior works of [8,20], we use a ternary decision tree to represent $T(a)$. Specifically, each node of the decision tree corresponds to an interview question and has three child nodes. When the user answers the interview question, the user is directed to one of its three child nodes according to her answer. As a result, each user follows a path from the root node to a leaf node during the interview process. A user profile is estimated at each leaf node based on the users' responses, i.e., $T(a)$. The number of interview questions presented to any user is bounded by the depth of the decision tree, generally a small number determined by the system. Also, non-responses to a question can be handled easily in the decision tree with the introduction of a "Unknown" branch.

3.3 Alternative Optimization

The objective function defined in Eq. (1) can be optimized through an alternating minimization process. Specifically, we alternate between the following two steps:

1. Given $T(a)$, we can compute v_j by regularized least square regression. Formally, for each j, we find v_j such that
$$v_j = \underset{v_j}{\operatorname{argmin}} \sum_{(i,j)\in O} (r_{ij} - v_j^T T(a_i))^2 + \lambda \|v_j\|^2.$$
This problem has a closed-form solution given by
$$v_j = \left(\sum_{(i,j)\in O} T(a_i)T(a_i)^T + \lambda I \right)^{-1} \left(\sum_{(i,j)\in O} r_{ij} T(a_i) \right), \quad (2)$$
where I is the identity matrix of appropriate size.
2. Given v_j, we try to fit a decision tree T such that
$$T = \underset{T\in\mathcal{H}}{\operatorname{argmin}} \sum_{(i,j)\in O} (r_{ij} - T(a_i)^T v_j)^2. \quad (3)$$

A critical challenge is that the number of possible trees grows exponentially with the depth of the trees, which can be extremely large even for trees of limited depth. It is therefore computationally extensive to obtain a global optimal solution for the above. We address this problem by proposing an efficient greedy algorithm for finding an approximate solution.

3.4 Decision Tree Construction

Traditional decision tree algorithms such as C4.5 or CART [6] usually minimize objective functions such as classification error and square loss for regression. In our scenario, the objective function is defined in Eq. (3), and we now describe how to build a decision tree in a greedy and recursive fashion to minimize it. Specifically, at each node, we select the best interview question by optimizing the objective defined in Eq. (3) based on the response to this question; the decision tree then splits the user set into three subsets corresponding to the child nodes (i.e. "Like", "Dislike" and "Unknown"). We carry out this procedure recursively until the tree reaches certain depth. In our experiments, the depth is usually set to a small number between 3 and 7.

Formally, starting from the root node, the set of users at current node are partitioned into three disjoint subsets $R_L(p)$, $R_D(p)$ and $R_U(p)$ corresponding to "Like", "Dislike" and "Unknown" of their responses to the interview question p:

$$\begin{aligned} R_L(p) &= \{i | a_{ip} = \text{“Like”}\}, \\ R_D(p) &= \{i | a_{ip} = \text{“Dislike”}\}, \\ R_U(p) &= \{i | a_{ip} = \text{“Unknown”}\}. \end{aligned}$$

To find the optimal question p that leads to the best split, we minimize the following objective:

$$\min_p \sum_{i\in R_L(p)} \sum_{(i,j)\in O} (r_{ij} - u_L^T v_j)^2 + \sum_{i\in R_D(p)} \sum_{(i,j)\in O} (r_{ij} - u_D^T v_j)^2 + \sum_{i\in R_U(p)} \sum_{(i,j)\in O} (r_{ij} - u_U^T v_j)^2, \quad (4)$$

where u_L, u_D and u_U are the optimal profiles for users in the child nodes corresponds to the answers of "Like", "Dislike" and "Unknown", respectively:

$$\begin{aligned} u_L &= \underset{u}{\operatorname{argmin}} \sum_{i\in R_L(p)} \sum_{(i,j)\in O} (r_{ij} - u^T v_j)^2, \\ u_D &= \underset{u}{\operatorname{argmin}} \sum_{i\in R_D(p)} \sum_{(i,j)\in O} (r_{ij} - u^T v_j)^2, \\ u_U &= \underset{u}{\operatorname{argmin}} \sum_{i\in R_U(p)} \sum_{(i,j)\in O} (r_{ij} - u^T v_j)^2. \end{aligned}$$

There also exist closed-form solutions to u_L, u_D and u_D as follows:

$$u_L = \left(\sum_{i\in R_L(p)} \sum_{(i,j)\in O} v_j v_j^T \right)^{-1} \left(\sum_{i\in R_L(p)} \sum_{(i,j)\in O} r_{ij} v_j \right), \quad (5)$$

$$u_D = \left(\sum_{i\in R_D(p)} \sum_{(i,j)\in O} v_j v_j^T \right)^{-1} \left(\sum_{i\in R_D(p)} \sum_{(i,j)\in O} r_{ij} v_j \right), \quad (6)$$

$$u_U = \left(\sum_{i\in R_U(p)} \sum_{(i,j)\in O} v_j v_j^T \right)^{-1} \left(\sum_{i\in R_U(p)} \sum_{(i,j)\in O} r_{ij} v_j \right). \quad (7)$$

After the root node is constructed, its child nodes can be constructed in a similar way, recursively. We summarize our algorithms for functional matrix factorization and decision tree construction in Algorithm 1 and Algorithm, 2 respectively.

Implementation and Computational Complexity. The time complexity for computing v_j with Equation (2) is $O(MK^3)$, where M is the number of items and K is the dimension of the latent space. In order to compute u_L, u_D and u_U at each split, we need to compute the inverse of a square matrix of size K, which takes $O(K^3)$ time. The brute-force approach for generating the matrix itself requires $O(UMK^2)$ time. In order to reduce the time complexity, we observe that the coefficient matrix can be computed based on the sum of ratings, the sum of squares of ratings and the number of ratings for each item within time $O(MK^2)$. With a similar method proposed in [8], the time complexity of computing these statistics of all items is $O(\sum N_i^2)$ for each level of the decision tree, where N_i is the number of ratings by the user i. Thus, the computation complexity for constructing

Algorithm 1 Alternating Optimization for Functional Matrix Factorization

```
Require: The training data $\mathcal{K} = \{r_{ij} \mid (i,j) \in O\}$.
Ensure: Estimated decision tree $T(a)$ and item profiles $v_j, j = 1, 2, \ldots, M$.
1: Initialize $v_j$ randomly for $j = 1, \ldots, M$.
2: while not converge do
3:    Fit a decision tree $T(a)$ using Algorithm 2.
4:    Update $v_j$ with Equation (2).
5: end while
6: return $T(a)$ and $v_j, j = 1, 2, \ldots, M$.
```

Algorithm 2 Greedy Decision Tree Construction

```
1: function FitTree(UserSet)
2: // UserSet: the set of users in current node.
3: Calculate $u_L$, $u_D$, $u_U$ by Equation (5), (6) and (7) for $p = 1, \ldots, P$.
4: Compute the split criteria $L_p$ by Equation (4) for $p = 1, \ldots, P$.
5: Find the best interview question $p = \text{argmax}_p L_p$.
6: Split user into three groups $R_L(p)$, $R_D(p)$ and $R_U(p)$.
7: if square error reduces after split and depth < maxDepth then
8:    Call FitTree($R_L(p)$), FitTree($R_D(p)$) and FitTree($R_U(p)$) to construct the child nodes.
9: end if
10: return $T(a)$.
11: end function
```

the decision tree is $O(D\sum_i N_i^2 + LMK^3 + LM^2K^2)$, where D is the depth of the tree and L represents the number of nodes in the tree. The computation time can be reduced by selecting a subset of items based on some criteria such as the rating variance, and using only the selected items as candidates for the interview questions.

3.5 Hierarchical Regularization

In the above section, u_L, u_D and u_U for each node are estimated by linear regression. When the amount of training data is small, the estimate may overfit the training data. The problem becomes more severe especially for the nodes close to the leaves of the decision tree, because as the split process progresses, users belonging to those nodes become fewer and fewer. To avoid overfitting, regularization terms need to be introduced. Although traditional ℓ_2 regularization can be applied in this case [16], but such approach does not take into account the rich structure of the decision tree.

Here, we propose to apply hierarchical regularization that utilize the structure of the decision tree. Specifically, we shrink the coefficient u of a node toward the one at its parent node. For example:

$$u_L = \operatorname*{argmin}_u \sum_{i \in R_L(p)} \sum_{(i,j) \in O} (r_{ij} - u^T v_j)^2 + \lambda_h \|u - u_C\|^2,$$

where u_C is the estimation at the current node and u_L is the estimation at its child node corresponding to the answer "Like" and the parameter λ_h controls the trade-off between training error and regularization. We use $\lambda_h = 0.03$ in our experiments which seems to give good results. The similar idea using parent node as the prior for the child node is also studied in [8], but the ratings for the parent node is used to improve the prediction in the child node.

Table 1: Statistics for Data Sets

Dataset	No. Users	No. Items	No. Ratings
MovieLens	6040	3900	1,000,209
EachMovie	72,916	1628	2,811,983
Netflix	480,189	17,770	104,706,033

4. EXPERIMENTS

In order to evaluate the proposed method for cold-start collaborative filtering, we carry out a set of controlled experiments on three widely used benchmark data sets: MovieLens, EachMovie and Netflix.

4.1 Data sets and Evaluation Metrics

We first briefly describe the data sets.

- The MovieLens[1] data set contains 3900 movies, 6040 users and about 1 million ratings. In this data set, about 4% of the user-movie dyads are observed. The ratings are integers ranging from 1 (bad) to 5 (good).
- The EachMovie data set, collected by HP/Compaq Research, contains about 2.8 million ratings for 1628 movies by 72,916 users. The ratings are integers ranging from 1 to 6.
- The Netflix[2] is one of the largest test bed for collaborative filtering. It contains over 100 million ratings and was split into one training and one test subsets. The training set consists of 100,480,507 ratings for 17,770 movies by 480,189 users. The test set contains about 2.8 million ratings. All the ratings are ranged from 1 to 5.

The performance of an collaborative filtering algorithm will be evaluated in terms of the widely used root mean square error (RMSE) measure, which is defined as follows

$$\text{RMSE} = \left(\frac{1}{|\mathcal{T}|} \sum_{(i,j) \in \mathcal{T}} (r_{ij} - \hat{r}_{ij})^2 \right)^{1/2},$$

where $\mathcal{T}$ represents the set of test ratings, r_{ij} is the ground-truth values for movie j by user i and $\hat{r}_{ij}$ is the predicted value by a collaborative filtering model. We can also normalize the error of each user by the number of their ratings. Our initial results show that the two metrics are consistent in general, so we only report RMSE in our experiments.

4.2 Deriving Interview Responses from User Ratings

The responses a_i of the interview process for user i is required in order to train and evaluate the interview process. Following the standard settings [8,20], we restrict the format of interview questions to be "Do you like item j?" For example, for movie recommendation system, we can ask questions like "Do you like the movie Independence Day?" Then, we can infer the responses for existing users according to their ratings for the corresponding items. For instance, with ratings in 1-5 scale, we assume that the responses a_{ij} of user i to the question "Do you like item j?" can be inferred from her rating r_{ij} as follows:

$$a_{ij} = \begin{cases} 0, & \text{if } (i,j) \in O \text{ and } r_{ij} \le 3 \\ 1, & \text{if } (i,j) \in O \text{ and } r_{ij} > 3 \\ \text{"Unknown"}, & \text{if } (i,j) \notin O \end{cases}$$

[1]http://www.grouplens.org/node/73
[2]http://www.netflixprize.com/

Intuitively, we assume that the user will response 0 (*dislike*) and 1 (*like*) for items she rates with ≤ 3 rating or > 3 rating. If the user did not rate an item selected for the interview process, we assume that she will respond with *unknown*. For EachMovie data set, we use a similar method except setting the threshold to be 4 since its ratings range from 1 to 6.

4.3 Experiment Design

We seek to address the following questions:

1. For new users, how does the proposed algorithm perform? In particular, how effective is the interview process? Moreover, how does the proposed algorithm perform compared to the baseline algorithms?
2. We utilize the ratings from users to simulate their responses to the interview questions in our evaluation. How do missing values in user ratings impact the performance of the proposed algorithm?
3. How does fMF perform in warm-start settings compared to traditional collaborative filtering methods such as plain matrix factorization?
4. How do the parameters impact the performance of the proposed model?

4.4 Cold-Start Performance

We first evaluate the performance of fMF in cold-start settings. For each data set, we split the users into two disjoint subsets, the training set and the test set, containing 75% and 25% users, respectively. The users in the training set are assumed to be warm-start users whose ratings are known by the system. We learn the models and construct the interview process based on these training users. In contrast, the users in the test set are assumed to be cold-start users. The ratings of each user in the test set are partitioned into two sets: the first set is called the *answer set* which is used to generate the user responses in the interview process while the second set is called the *evaluation set* which is used to evaluate the performance after the interview process. The sizes of the answer set and the evaluation set are 75% and 25% of the rated items of each user, respectively. Note that the interview process typically includes only a small number (≤ 7 in our experiments) of questions although the answer set contains 75% of the ratings when deriving the responses for the cold-start users. The evaluation process is summarized in Figure 2.

We compare the performance with two baseline methods, named as Tree and TreeU and briefly described as follows:

- Tree: this is the method proposed in the very recent work of [8]. The method learns a decision tree for initial interview without using latent user/item profiles. It fits the tree based on the ratings of all the items in the inventory. Therefore, at the leaf node, it needs to estimate M ratings for all the M items in the system.

- TreeU: this method learns decision tree and matrix factorization through two separate steps. It first estimates the user profiles and item profiles through plain matrix factorization described in Section 3.1; then, a decision tree is constructed separately using the algorithm of [8] to fit the latent user profiles. The model predicts ratings by using the user profiles from the decision tree and the item profiles from matrix factorization.

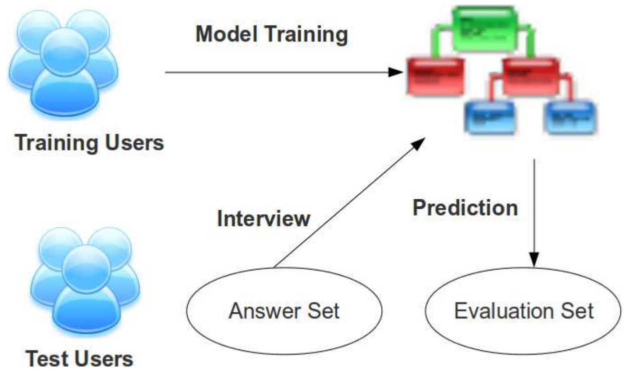

Figure 2: Evaluation process for cold-start users

Our proposed fMF algorithm differs from these two algorithms in that it is a natural integration of both decision tree and matrix factorization.

We use the following parameter settings: For TreeU, we use as default the regularization weight of $\lambda = 0.01$. For Tree and fMF, we apply 4-fold cross validation to determine the parameters.

The results on MovieLens, EachMovie and Netflix data sets are reported in Table 2, Table 3 and Table 4, respectively. Our first observation is that, as expected, the performance is gradually improved (i.e. the RMSE decreases) as the number of interview questions increases. This is true for all three methods, suggesting that these three algorithms are all capable of refining user preference through the interview process. Therefore, all the three methods can be applied to solve the cold-start problem.

Comparing the performance of fMF and Tree, we can see that fMF consistently outperforms Tree in all the three data sets. The improvements are significant according to t-test with significance level $p = 0.05$. This observation illustrate that the interview processes learned by fMF is more effective than those by Tree. We attribute this to the fact that fMF naturally integrates the matrix factorization model into the interview process for cold-start collaborative filtering. In particular, fMF inherits the ability of matrix factorization models in *collaboratively* uncovering the user-item ratings. In contrast, Tree assumes that users/items are independent from one another, and therefore cannot capture the vital collaborative effect.

We can also see that TreeU does not work well, too. Recall that TreeU is a two-stage method — It first extracts the user profiles and item profiles using plain matrix factorization, and then constructs a decision tree on the answers of the interview questions. A possible reason for why it does not perform well is that the user profiles and item profiles obtained from the matrix factorization in warm-start setting usually capture the refined preferences of users and refined characteristics of items. However, in cold-start setting, we are only allowed to ask users a few questions so that the model usually captures only very coarse interests of users. As a result, TreeU usually fails to fit the user profiles from matrix factorization accurately. Thus, its prediction accuracy is relatively low. On the other hand, our method estimates the decision tree and item profiles simultaneously in a combined optimization process. Thus, the user profiles obtained by decision tree and the item profiles can adapt to each other and improve the prediction accuracy.

To provide more comprehensive views of the interview pro-

Table 2: RMSE on MovieLens data set for cold-start users with respect to the number of interview questions

No. Questions	3	4	5	6	7
fMF	0.9509	0.9480	0.9437	0.9429	0.9410
Tree	0.9767	0.9683	0.9615	0.9512	0.9523
TreeU	0.9913	0.9887	0.9789	0.9690	0.9657

Table 3: RMSE on EachMovie data set for cold-start users with respect to the number of interview questions

No. Questions	3	4	5	6	7
fMF	1.2853	1.2717	1.2715	1.2691	1.2673
Tree	1.2989	1.2842	1.2800	1.2779	1.2750
TreeU	1.3134	1.3009	1.2928	1.2907	1.2905

Table 4: RMSE on Netflix data set for cold-start user with respect to the number of interview questions

No. Questions	3	4	5	6	7
fMF	0.9205	0.9189	0.91582	0.9123	0.9121
Tree	0.9320	0.9252	0.9239	0.9228	0.9219
TreeU	0.9422	0.9382	0.9359	0.9322	0.9323

Table 5: Examples of interview process with 6 questions

(a) Interview process

Round	Question	Response
1	Being John Malkovich	Dislike
2	Armageddon	Like
3	Maid in Manhattan	Dislike
4	Reservoir Dogs	Like
5	Collateral Damage	Dislike
6	2 Fast 2 Furious	Unknown

(b) Top-5 recommendations

Rank	Movie Title
1	Lord of the Rings: The Return of the King
2	Lord of the Rings: The Two Towers
3	Mobile Suit Gundam: Char's Counterattack
4	Star Wars: Episode V: The Empire Strikes Back
5	Raiders of the Lost Ark

cess, we further look into particular cases. In Table 5 and Table 6, we present the interview process for users in Netflix data set as well as the top-5 recommendations for them after the interview process. From Table 5, we can see that the user chooses "Like" on the movie Armegeddon and Reservoir Dogs, which belong to Fiction and Adventure movies. The recommendation includes Lord of the Rings series and Star Wars. They are quite related to the movie Armegeddon based on their genres. Similarly, the interview process of Table 6 shows that the user likes Drama and Romance movies and the top recommendations for her contain both those two types of movies. Those results illustrate that the recommendations generated by fMF are indeed reasonable.

4.5 The Impact of Non-responses

In our experiments, we utilize the ratings of users to simulate their responses to the interview questions as described in Section 4.2 since the users' responses to the interview questions are not available for the benchmark data sets. In particular, we assume that the user will respond "Unknown" to an interview question if she does not rate the corresponding item. This assumption, however, might be inaccurate in practice. For example, a user's rating to an item might be missing simply because she does not have time to rate it. In this case, the user may actually responds with "Like" or "Dislike" rather than "Unknown" if she is asked to respond to the interview questions. As a result, the missing values in data sets may introduce bias in the training and interview process.[3]

Here, we explore how the missing ratings influence the initial interview process. We investigate the performance of the proposed method on users with different numbers of ratings. To this end, we sort users in the test set by the their numbers of ratings in descending order and then plot, in Figure 3, the performance measured by RMSE with respect to the fraction of users with the most ratings. We can see that the RMSE increases when more users with few ratings are included, which indicates that the performance for users with more ratings is better then the ones with less ratings. This is because that the users with more ratings are less likely to select "Unknown" in our simulation. Thus, they are less likely prone to the influences by the bias introduced by the missing values in the training data.

With this basic understanding, we carry out a set of experiments to investigate the impact of missing values in training

Table 6: Examples of interview process with 6 questions

(a) Interview process

Round	Question	Response
1	Being John Malkovich	Like
2	Armageddon	Dislike
3	What Women Want	Dislike
4	The Royal Tenenbaums	Unknown
5	Pretty Woman	Like
6	Cats: The Ultimate Edition	Unknown

(b) Top-5 recommendations

Rank	Movie Title
1	The Shawshank Redemption
2	Schindler's List
3	Sex and the City: Season 5
4	Sex and the City: Season 4
5	The Usual Suspects

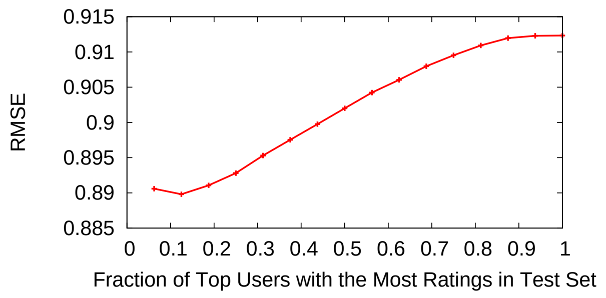

Figure 3: Performance measured by RMSE on Netflix data set with respect to the fraction of users with the most known ratings.

[3]The bias in training data seems to have been largely ignored in previous work.

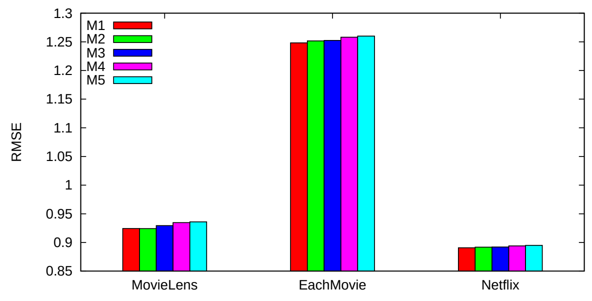

Figure 4: Performance measured by RMSE on three data sets with respect to the fraction of users with the most known ratings.

data. For each of the three data sets, we sort users in training set in descending order. Then, the users in the test set (labeled by MT) is sampled from top 20% users with the most ratings. We only consider the users with sufficiently large number of ratings since we would like to rule out the impact of bias during the interview process and focus on the missing values for training users. We generate five training sets, namely M1 to M5, from the rest of the users in the training set, including the top 20%, 40%, 60%, 80% and 100% of the users that are not selected in the test set, respectively. Again, 75% ratings for each user in MT is used to simulate the interview process and the performance is evaluated on the remaining 25% ratings.

We perform experiments by training the proposed models on M1 to M5 and evaluate them on MT for all three data sets. The results are shown in Figure 4. It can be observed from Figure 4 that the prediction error measured by RMSE increases when more users with few training data are included in the training set. This observation suggests that the performance are indeed affected by the bias introduced by the missing values in training set.

4.6 Warm-Start Performance

In order to answer the third question proposed in Section 4.3, we evaluate the proposed algorithm in warm-start settings and show the relative performance between the cold-start methods and warm-start methods. In particular, we consider the matrix factorization method described as follows:

- MF: The matrix factorization method described in Section 3.1 is included for comparison. We include ℓ_2 regularization to avoid overfitting with $\lambda = 0.01$. The method is not designed for the cold-start problem and thus it requires the user ratings in order to construct user profiles.

We include this method to present a concrete comparisons of the relative performance of the proposed cold-start method and the warm-start methods that are widely studies in previous researches. We compare the proposed algorithm (fMF) with all three methods for warm-start settings: Tree, TreeU and MF. To this end, we perform experiments with different depths of decision trees and report the RMSE of the all the methods over three data sets. For MovieLens and EachMovie data set, we randomly split the ratings for each users into training set and test set and perform 4-fold cross validation. For Netflix data set, we follow the popular evaluation protocol on this data set. Therefore, we train our models on the training set and report the performance on the test set. The performance measured by RMSE with respect to the depths of decision trees is reported in Table 7, Table 8 and Table 9.

We can observe that all the cold-start methods perform worse than the matrix factorization method. This observation is consistent with our expectation because all the three cold-start algorithms are constrained in some ways in order to deal with cold-start users — the goal of the cold-start algorithms is to provide cold-start users with reasonable predictions within a few quick interview questions. For example, the fMF model is restricted in its maximum depth. We should expect comparable performance, if this constraint is eliminated.

As an empirical validation, we further carry out experiments on the MovieLens data set by fitting the decision trees in our model with relatively larger depth. We note that the decision trees of very large depth correspond to interview processes with many questions, which are not proper for initial interview process for cold-start users since users are typically not willing to answer many questions. However, we can show that the performance of the proposed method can be quite close to the matrix factorization method if we relax the constraint and allow the model to use decision trees with large depth. We report the RMSE of fMF with respect to the depth of the decision tree in Figure 5. We also include Tree and MF for comparison. We can see that the RMSE of fMF monotonically decreases as the depth of decision trees increases. Moreover, its performance can be quite close to the matrix factorization method (MF). When the depth of the trees grows, the number of leaf nodes increases. Therefore, there are only a few users at each node. Thus, the user profiles estimated by fMF can be quite close to those by MF. We conclude that our method can be a reasonably good method of general collaborative filtering as well. On the other hand, we can see that the performance of Tree is much worse than MF even with a large number of questions. Moreover, the gap between fMF and Tree becomes larger when the depth of the decision tree grows. This is because Tree predicts the ratings of different items independently at each leaf node, which is not a good model for collaborative filtering in general. We also perform similar experiments with the cold start setting. In this case, the RMSE of fMF decreases with the depth of the decision tree when the depth is not very large (e.g. ≤ 14). We emphasize that the depth of the decision tree is usually set to be a small number since we can not ask a lot of questions to users during the interview process.

4.7 Impact of Model Parameters

There are several parameters that affect the performance of the proposed model. In this section, we carry out experiments to investigate the impacts of these parameters. We only report the results on MovieLens data set as the observations are similar when other data sets are used. By default, the cold-start setting described in Section 4.4 is applied unless otherwise stated.

The parameter K controls the dimension of the user profiles and item profiles for the matrix factorization model. We fix the depth of the decision tree to be 6, and report the performance obtained with different K. The results are depicted in Figure 6. Particularly, as K increases, the factorization model becomes more and more flexible, as a results,

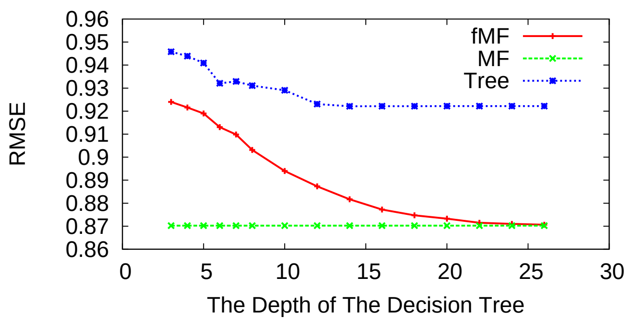

Figure 5: RMSE of fMF, Tree and MF on MovieLens data set with respect to the number of interview questions

Table 7: RMSE on MovieLens data set in warm-start setting

No. Questions	3	4	5	6	7
fMF	0.9240	0.9216	0.9190	0.9134	0.9098
Tree	0.9458	0.9439	0.9409	0.9321	0.9329
TreeU	0.9906	0.9850	0.9771	0.9706	0.9728
MF	0.8702				

Table 8: RMSE on EachMovie data set in warm-start setting

No. Questions	3	4	5	6	7
fMF	1.2548	1.2499	1.2449	1.2366	1.2226
Tree	1.2709	1.2614	1.2664	1.2570	1.2549
TreeU	1.28742	1.2813	1.2790	1.2772	1.2751
MF	1.1790				

Table 9: RMSE on Netflix data set in warm-start setting

No. Questions	3	4	5	6	7
fMF	0.9770	0.9749	0.9721	0.9715	0.9703
Tree	0.9772	0.9761	0.9726	0.9717	0.9713
TreeU	0.9914	0.9879	0.9842	0.9792	0.9752
MF	0.9399				

the RMSE first decreases, and reaches the optima around $K = 20$; thereafter, the model becomes increasingly overparameterized and the performance in turn starts to degrades. For example, the performance with $K = 30$ is worse than the that with $K = 20$. In principle, the optimal K should provide the best trade-off between fitting bias and model complexity.

One of our contributions is that we propose to use hierarchical regularization to avoid overfiting. The impact of the hierarchical regularization is controlled by the parameter λ_h. We evaluate fMF with different values of λ_h on MovieLens data set. The performance obtained by ℓ_2 regularization is also reported for comparison. We can see from Figure 7 that the hierarchical regularization always performs better than the basic ℓ_2 regularization. This observation suggest that hierarchical regularization, by exploiting the structure of the decision tree, provides better regulatory to the functional matrix factorization model.

Another parameter of interest is the regularization weight λ for the item profiles v_j. We vary the value of λ and report the performance measured by RMSE in Figure 8. We can see

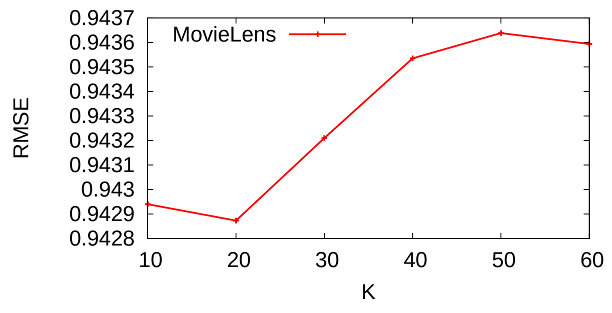

Figure 6: Performance measured by RMSE with respect to different values of K on MovieLens data set. The performance is reported by setting the depth of the decision tree to be 6.

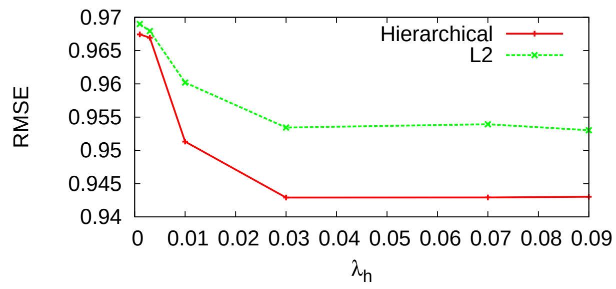

Figure 7: The performance of hierarchical regularization. The performance is reported by setting the depth of the decision tree to be 6.

that the optimal performance is achieved when a moderate λ is used. Especially, the model has a high risk of overfitting if λ is too small.

The learning algorithm of the proposed model is an iterative process. In Figure 9, we plot the both training and test performance measured by RMSE with respect to the number of iterations. We can see that the algorithm converges very quickly, usually within 5 iterations.

5. CONCLUDING REMARKS

The main focus of this paper is on the cold-start problem in recommender systems. We have presented the functional matrix factorization, a framework for simultaneously learning decision tree for initial interview and latent factors for user/item profiling. The proposed fMF algorithm seamlessly integrates matrix factorization for collaborative filtering and decision-tree based interview into one unified framework by reparameterizing the latent profiles as a function of user responses to the interview questions. We have established learning algorithms based on alternating minimization and demonstrated the effectiveness of fMF on real-world recommendation benchmarks.

The current fMF model is based on a basic matrix factorization formulation and does not take into account the content features such as the demographical information of users. For future work, we plan to investigate how to utilize such content features to further enhance the performance of fMF in cold-start recommendations. Moreover, we also plan to conduct in-depth examination on the missing value problem, which seems to introduce considerable bias to the learning process as revealed by our experiments.

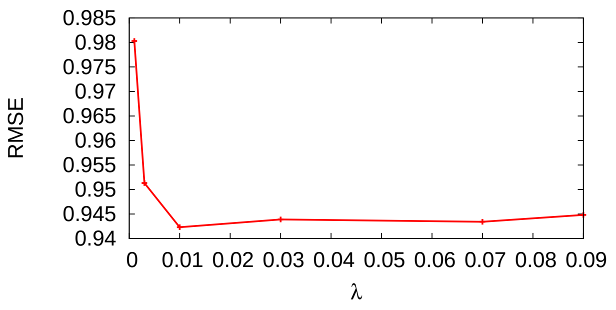

Figure 8: Performance measured by RMSE with respect to different values of λ on MovieLens data set.

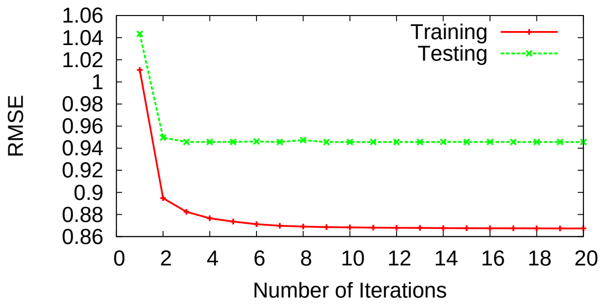

Figure 9: Performance measured by RMSE with respect to the number of iterations on MovieLens data set.

6. ACKNOWLEDGEMENTS

We would like to thank the reviewers for their constructive and insightful comments. Part of this work was supported by NSF Grant IIS-1049694, a Yahoo! faculty grant and 111 Project B07022.

7. REFERENCES

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

[2] D. Agarwal and B.-C. Chen. Regression-based latent factor models. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, page 19, 2009.

[3] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. *Twenty-first international conference on Machine learning - ICML '04*, page 9, 2004.

[4] C. Boutilier, R. Zemel, and B. Marlin. Active collaborative filtering. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 98–106. Citeseer, 2003.

[5] J. Breese, D. Heckerman, C. Kadie, and Others. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[6] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth and Brooks, Monterey, CA, 1984.

[7] N. Golbandi, Y. Koren, and R. Lempel. On bootstrapping recommender systems. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1805–1808. ACM, 2010.

[8] N. Golbandi, Y. Koren, and R. Lempel. Adaptive Bootstrapping of Recommender Systems Using Decision Trees. *WSDM*, 2011.

[9] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[10] 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, New York, New York, USA, 2008. ACM.

[11] A. Harpale and Y. 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. ACM, 2008.

[12] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, Jan. 2004.

[13] 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. AUAI Press, 2004.

[14] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):1–24, 2010.

[15] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. *Proceedings of the third ACM conference on Recommender systems - RecSys '09*, page 21, 2009.

[16] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, volume 2007, pages 5–8. Citeseer, 2007.

[17] D. Pennock, E. Horvitz, S. Lawrence, and C. Giles. Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *Proceedings of the 16th conference on uncertainty in artificial intelligence*, pages 473–480. Citeseer, 2000.

[18] P. Pu and L. Chen. User-involved preference elicitation for product search and recommender systems. *AI Magazine*, 29(4):93, 2009.

[19] A. Rashid, I. Albert, D. Cosley, S. Lam, S. McNee, J. Konstan, and J. 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. ACM, 2002.

[20] A. M. Rashid, G. Karypis, and J. Riedl. Learning Preferences of New Users in Recommender Systems: An Information Theoretic Approach. *SIGKDD Workshop on Web Mining and Web Usage Analysis*, 22, Oct. 2008.

[21] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized Markov chains for next-basket recommendation. *Proceedings of the 19th international conference on World wide web - WWW '10*, page 811, 2010.

[22] J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 713–719. ACM, 2005.

[23] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '02*, page 253, 2002.

[24] N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. *Advances in neural information processing systems*, 17:1329–1336, 2005.

[25] D. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *Proceedings of the 18th international conference on World wide web*, pages 111–120. ACM, 2009.

[26] X. Su and T. M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 2009(Section 3):1–20, 2009.