N-dimensional Markov random field prior for cold-start recommendation

Furong Peng a, Jianfeng Lu a,*, Yongli Wang a, Richard Yi-Da Xu b, Chao Ma a, Jingyu Yang a

a Nanjing University of Science and Technology, P.R. China
b University of Technology, Sydney, Australia

ABSTRACT

A recommender system is a commonly used technique to improve user experience in e-commerce applications. One of the popular recommender methods is Matrix Factorization (MF) that learns the latent profile of both users and items. However, if the historical ratings are not available, the latent profile will draw from a zero-mean Gaussian prior, resulting in uninformative recommendations. To deal with this issue, we propose using an n-dimensional Markov random field as the prior of matrix factorization (called mrf-MF). In the Markov random field, the attribute (such as age, occupation of users and genre, release year of items) is considered as the site and the latent profile, the random variable. Through the prior, new users or items will be recommended according to its neighbors. The proposed model is suitable for three types of cold-start recommendation: (1) recommend new items to existing users; (2) recommend new users for existing items; (3) recommend new items to new users. The proposed model is assessed on two movie datasets, Movielens 100K and Movielens 1M. Experimental results show that it can effectively solve each of the three cold-start problems and outperforms several matrix factorization based methods.

1. Introduction

The sheer volume of information on the internet often makes it extremely difficult to obtain useful information. To deal with this problem, a recommender system is proposed that advises users of interesting information such as movies, books, music, bookmarks, CDs, news, images and TV programs [1–4]. When customers browse products online, many famous companies provide a recommendation service to save customers’ time. In the Amazon online store, the system recommends books, music and other products. A bookmark sharing social website, Delicious recommends bookmarks according to users’ favored URLs. With increasing demand for recommendation technology, scientists and engineers are paying more attention to recommendation systems and great progress has been made in the field.

In the past few decades, many recommendation algorithms have been proposed [3–6]. They are generally divided into three categories: content-based, collaborative filtering (CF) and hybrid methods. The content-based method mainly focuses on the item’s content. The profile of users is built based on previously rated items. If the content of candidate items is more likely to fit the profile of users then the candidate is more likely to be recommended to them. CF methods such as nearest neighbor search, singular value decomposition and matrix factorization technique [4], recommend items to a user if similar users have interacted with those items. Only historical actions are used in this kind of method. Therefore, CF is widely applied and is very hotly investigated. Usually, CF outperforms the content-based method since the content-based method cannot recommend items with which the user have never interacted. The content-based method performs better than CF for new items (no historical action available) such as the news, books and music. To combine the advantages of both methods, the hybrid method is proposed to deal with new items and to recommend ones outside user’s history of interaction [5].

In a real application, it is common to encounter a situation in which there are new users/items with no historical ratings. Recommending to an inexperienced user is called the user cold-start problem. Similarly, recommending a new item is called the item cold-start problem. Apart from these two situations, we have a third, called the system cold-start, in which the system recommends new items to new users. The three scenarios are illustrated in Fig. 2. In this work, a hybrid method is proposed to deal with all three scenarios (Table 1).
The user attributes (e.g. age, gender and occupation) and item attributes (e.g. genre and release year) are utilized to deal with the cold-start problems. Based on probabilistic matrix factorization (PMF) [7], which factorizes the rating matrix into two low rank matrices: user and item profiles, we add an n-dimensional Markov random field (MRF) prior to the latent profile (called mrf-MF). With the MRF prior, the new users/items can obtain latent profiles from their neighbors. The advantage of MRF prior is that the mapping from the attributes to profiles is implicit and flexible. Through constraining the similarity of profiles according to the attributes, the mapping function is implicitly defined. Therefore, the mapping function is capable of non-linear situations. The MRF is usually stable which will be demonstrated in our experiments. The overview of the proposed model is illustrated in Fig. 1.

The user attributes (e.g. age, gender and occupation) and item attributes (e.g. genre and release year) are utilized to deal with the cold-start problems. Based on probabilistic matrix factorization (PMF) [7], which factorizes the rating matrix into two low rank matrices: user and item profiles, we add an n-dimensional Markov random field (MRF) prior to the latent profile (called mrf-MF). With the MRF prior, the new users/items can obtain latent profiles from their neighbors. The advantage of MRF prior is that the mapping from the attributes to profiles is implicit and flexible. Through constraining the similarity of profiles according to the attributes, the mapping function is implicitly defined. Therefore, the mapping function is capable of non-linear situations. The MRF is usually stable which will be demonstrated in our experiments. The overview of the proposed model is illustrated in Fig. 1.

The proposed mrf-MF model is a single model which can deal with all mentioned cold-start scenarios as if the attributes of user/item are acquirable. Compared to other complex models [8–10], mrf-MF is more elegant and easier to implement. In this work, we compare our method with several single models in all cold-start scenarios. The results demonstrate the superior performance of our algorithm on movie datasets. The proposed model can obtain optimal performance in large parameter spaces, and then greedy strategies are proposed to search for the optimal parameters. By the alternating least squares (ALS) solution, mrf-MF can converge in tens of iterations.

The main contributions of this work are listed as follows:

- The Markov random field is extended to be n-dimensional and is applied to matrix factorization (mrf-MF) to deal with three cold-start problems.
- An efficient ALS-like algorithm, which can converge in tens of iterations, is proposed to solve mrf-MF model.
- Efficient greedy strategies are proposed to search the optimal hyper parameters of mrf-MF.

The rest of this paper is organized as follows. In Section 2, we review related work about cold-start recommendations. In Section 3.3, we introduce the basic probabilistic matrix factorization method and then present the proposed mrf-MF model for cold-start scenarios. The algorithm to learn the mrf-MF model is also described in this section. Then, we evaluate the proposed model in Section 5. We conclude this paper and present a more challenging task as future work in Section 6. Finally, the acknowledgment is listed in Section 7.

2. Related work

In this section, we briefly review several recommender algorithms that can handle different cold-start problems.

Park et al. [11] proposed an agent technique called naive filterbots that generate ratings on behalf of users based on the
attributes of both users and items. Seven different global filterbots were proposed to improve the performance of item-based and user-based CF recommenders in cold-start scenarios. The improvement of naive filterbots is weak because the generated ratings cannot reflect the user’s attitude to items. Park et al. [12] extended this work and proposed a pairwise regression model that uses a joint profile of the user-item pair for regression. The joint profile, which is generated by the outer product of the attributes of user and item in pairs, is used to fit ratings. Through a pairwise loss function trick, the number of training samples increases from $O(n)$ to $O(n^2)$. Although the model is a linear regression, the increase of features and samples slows down the speed of the training process. As shown in Section 5, the linear regression is a bottleneck and limits the accuracy of the results. They also present a comprehensive experimental setting that splits the dataset into four parts as shown in Fig. 2, in which all cold-start scenarios are included. One part is used for training and the other three parts are used for item, user, and system cold-start evaluation, respectively. We set our experiments in a similar manner.

Agarwal and Chen [13] proposed a regression based matrix factorization model (called rLRF), which maps the latent profile to the attribute of users/items. A Monte Carlo Expectation Maximization (MCEM) algorithm was implemented to solve this model. However, as used in rLRF, the linear regression between attributes and latent profiles could not be guaranteed. Following this work, a general matrix factorization framework was proposed by Zhang et al. [14]. Multiple regression functions could be added in this framework, such as regression tree and lasso. MCEM was also applied to simulate the model. To cope with cold-start items with rich content, such as news and webpages, Agarwal and Chen [15] proposed a variant of the topic model (called fLDA). fLDA considers the content of item as words and learns topics over words as the latent profile of an item. The model is also optimized by the MCEM method. Experimental results showed that the fLDA is superior to the rLRF model on the item cold-start scenarios. As the MCEM needs to generate at least thousands of samples of latent profiles, the convergence speed is very slow.

Gantner et al. [16] proposed an attribute mapping model that extends Steffen et al.’s Bayesian personalized ranking [17] to handle cold-start scenarios. The model constructs the latent profile of both users and items based on matrix factorization. To cope with the cold-start problem, the attributes of users/items are linearly mapped to latent profiles. The number of training samples is quadratic of the observed values because of pair-wise loss function. To speed-up the training phase, stochastic gradient descent (SGD) was applied. According to their experience, the model can get a better result if the training is divided into two separate phases: constructing the latent profile and learning the mapping coefficients. However, it is difficult for this model to converge to a small training error. The linear mapping assumption also affects its accuracy. The interactive model is a new topic to deal with the user cold-start scenario. Zhou et al. [18] proposed functional Matrix Factorization (fMF) to build the profile of cold-start users. A decision tree is constructed to select representative items as interview questions. New users can produce their profile by answering the questions on the tree. Sun et al. [19] extended this work and proposed a multiple-question decision tree model. Compared with the fMF model, each node of the tree contains multiple questions, so that the new user has the opportunity to select an interesting one to answer. Zhang et al. [20] proposed a semi-supervised discriminative selection (SSDS) model to select the representative items as the interview questions. In the model, ratings generated by interviewing new users with those items will be used as historical information to make recommendations. However, in real interview processing, users may change their attitude affected by their emotions and environment.

Various pattern recognition techniques and machine learning methods are applied in cold-start recommendation. Houlsby et al. [21] proposed an active learning method with robust ordinal matrix factorization for user/item cold-start. Active learning is used to select typical items as in fMF and then an ordinal matrix factorization technique is applied to learn the latent profile. Lika et al. [10] combined some well-known classifiers and nearest neighbor based CF to cope with the user cold-start problem. In their experiments, the demographic information was applied to train classifiers and predict the user’s category. After the new user’s category was obtained, the rating was estimated by their similar users in the same category. Savesski and Mantrach [22] proposed a matrix factorization model, called Local Collective Embedding (LCE), that collectively decomposes the rating matrix and attribute information while local structure is preserved. In the model, a common latent profile $W$ is constructed to decompose both item attributes and ratings, but the user cold-start scenario is not considered.

Transfer learning is also applicable in cold-start recommendation. An auxiliary domain with dense interaction records is utilized to enrich a new user’s informations [23–26]. The new user’s information in the auxiliary domain is considered as the prior to make recommendation. However, it is common that the new user does not exist in the auxiliary domain. To handle this situation, a method to transfer knowledge without requiring overlapping users between domains was proposed [27,28]. Specifically, the collective knowledge such as codebook [27,29], shared latent-feature [30] and domain correlation [31] were used to tackle cold-start. However, the knowledge transferring method aims to alleviate the sparsity problem but is not able to deal with new users and items. This is because the transferred knowledge for all new users is the same and therefore the same non-personalized results will be recommended to all new users.

From the above review, we can see that the CF technique is very important in the recommendation. Especially, the matrix factorization defines the user/item's latent profile, which can be used to build a new model for the cold-start problem. In this work, we suppose that the users/items have a similar latent profile if similar attributes are found. To implement this assumption, an $n$-dimensional MRF prior incorporating with matrix factorization is proposed. Compared to the linear regression [13] and tree based models [14], the advantage of MRF prior is that the mapping from attributes to latent profile is implicitly learned and is capable of non-linear situations. In our experiments, we also found that the mf-MF is intensive to parameters.

3. Preliminaries

First, the definition of the cold-start problem is presented in this section. Then, we introduce the matrix factorization model, the foundation of our model, as well as Markov random field, which is the important prior we considered.

3.1. Problem definition

In a recommender system, there are usually two types of entity: user and item. We define user set as $U = \{U_i\}^n_{i=1}$ and item set as $I = \{I_j\}^m_{j=1}$, where $n$ and $m$ are numbers of users and items, respectively. In the following model description, we use $i$ for user’s index and $j$ for items index. Ratings are defined as a matrix $R \in \mathbb{R}^{n \times m}$ whose entry $R_{ij}$ is user $i$’s rating on item $j$. The rating set $(R_{ij})$ is composed of two parts: observed values $(R_{obs})$ and unobserved values $(R_{unobs})$, where $(R_{obs}) = (R_{ij}) \cup (R_{unobs})$ and $\emptyset = (R_{unobs}) \cap (R_{obs})$. $\mathbb{I}_{ij}$ is used to indicate if $R_{ij}$ is observed. Each user $i$ has its own attributes $\Omega_i \in \mathbb{R}^{d_U}$ and item $j$ has $\Theta_j \in \mathbb{R}^{d_I}$, where $d_U$ is the number of user’s attributes and $d_I$ is that of item’s. The task is to recommend the top-$n$ favorite items for each user in three different cold-start scenarios: new items are recommended to existing users in the item cold-start scenario; existing items are recommended to new
users in the user cold-start scenario; recommend new items to new users in the system cold-start scenario.

**Problem 1 (User cold-start).** Given \((R_{ij}^u), \Omega, \Theta\) for all existing users and items \(\{U, I\}\), and a new user \(i\) who has attribute \(\Theta_i\), recommend a list of existing items to the new user \(i\).

**Problem 2 (Item cold-start).** Given \((R_{ij}^u), \Omega, \Theta\) for all existing users and items \(\{U, I\}\), and a set of new items \(I'\) which have attributes \(\Theta\), recommend a list of new items to an existing user \(i\).

**Problem 3 (System cold-start).** Given \((R_{ij}^u), \Omega, \Theta\) for all existing users and items \(\{U, I\}\), a set of new items \(I'\) that have attributes \(\Theta\) and a new user \(i\) who has attribute \(\Theta_i\), recommend a list of new items to the new user \(i\).

3.2. Matrix factorization

Matrix factorization (MF) usually refers to the probabilistic matrix factorization model proposed by Ruslan and Andriy [7]. The model tries to factorize the rating matrix \(R \in \mathbb{R}^{m \times n}\), which is very sparse and contains a large number of unobserved entries, into two low rank matrices: one is the user latent profile \(U \in \mathbb{R}^{k \times m}\) and the other is the item latent profile \(I \in \mathbb{R}^{k \times n}\), where \(k\) is the dimensionality of latent space. The random variable \(R_{ij}\) is defined as

\[
R_{ij} = U_i^T I_j + \epsilon
\]  
(1)

where \(\epsilon \sim \mathcal{N}(0, \sigma^2)\) is a zero-mean Gaussian distribution. The number of parameters in the profile of users and items is \(k \times (n + m)\) in total which is such a large number that it is easy to over-fit the data. Therefore, the zero-mean Gaussian distribution is used as the prior of \(U\) and \(I\) to resist over-fitting:

\[
U_i \sim \mathcal{N}(0, \sigma_1^2) \\
I_j \sim \mathcal{N}(0, \sigma_2^2)
\]

where \(\sigma_1\) and \(\sigma_2\) represent users’ and items’ uncertainty, respectively. Combining the prior and likelihood, we can get the following loss function:

\[
L_{mf} = \frac{1}{2} \sum_{i,j=1}^{n,m} \left( U_i^T I_j - R_{ij} \right)^2 + \frac{1}{2} \sum_{i=1}^{n} ||U_i||^2_2 + \frac{1}{2} \sum_{j=1}^{m} ||I_j||^2_2,
\]  
(2)

where \(\lambda_1 = \sigma^2 / \sigma_1^2\), \(\lambda_2 = \sigma^2 / \sigma_2^2\) and \(\| \cdot \|_k\) is the Frobenius norm. The second and third terms are called regularization terms and \(\lambda_1\) and \(\lambda_2\) are regulation coefficients. In the cold-start scenario, new users or items do not appear in the first term. As a consequence, they will only take the zero-mean Gaussian distribution. That is to say all the new users or items will have the same latent profile: zeros. Undeniably no useful items will be effectively recommended. To make the MF technique be applicable in cold-start scenarios, we adopt the attributes of both user and item to build their prior in place of the zero-mean Gaussian. Before introducing the extended model, we briefly go through the Markov random field.

3.3. Markov random field

Markov random field (MRF) has achieved many successes in computer vision, particularly in image reconstruction, image segmentation and 3D vision [32]. It is defined on a set of sites \(S = \{s\}\), each of which is related to others via a neighborhood system \(N = \{N(t) \mid t \in S\}\). \(N(t)\) is the neighbor of site \(t\) and has the following properties:

- site \(t\) does not belong to its neighbor: \(t \notin N(t)\)
- if \(t\) belongs to \(o\)’s neighbor then \(o\) belongs to \(t\)’s: \(o \in N(t) \Leftrightarrow o \in N(t)\)

The sites \(S\) and the neighborhood system \(N\) comprise the Markov network. On the sites \(S\), we define a set of random variables \(X = \{X_t\}\), each of which takes a value \(x_t\) in \(\Omega\). The joint distribution \(p(x = x)\) is the probability measurement for a configuration \(x\) for the given Markov network. \(L^S\) is the configuration space and \(\sum_x p(x = x) = 1\). \(x\) is said to be a Markov random field if \(X\) on the sites \(S\) satisfy the following two conditions:

- Positivity: \(p(X = x) > 0\)
- Markovianity: \(p(X_t = x_t | X_{\delta(t)} = x_{\delta(t)}) = p(X_t = x_t | X_{N(t)} = x_{N(t)})\)

where \(S(t)\) are all sites except \(t\) and \(N(t) = \{N(0) \mid 0 \in N(t)\}\). In the application of image analysis, the sites are defined as a square lattice which corresponds to the location of pixels. In our cold-start scenarios, the sites are defined on attributes. However, the dimension of attributes is larger than 2 which is the case in image processing. Therefore, we need to adjust the Markov model to adapt to our specific situation. This will be illustrated in the next section.

4. The proposed model

We describe the proposed model in this section. First, we introduce the \(n\)-dimensional Markov random fields, followed by the mrf-MF model and its optimizing method. Finally, we illustrate how the method can make recommendations in all three cold-start scenarios.

4.1. \(n\)-dimensional Markov random field

In the MRF model, each site is located in a lattice. Since it is very common that users/items have more than two attributes, it demands a neighbor system with a high dimensional topology structure. We first define a specific neighbor system for attributes and then the conditional distribution of latent profiles.

A neighbor system for MF, called fix-radius nearest neighbors, can be defined as \(F_i = \{o : D(\Omega_i, \Omega_o) < r\}\), where \(D(\cdot, \cdot)\) is a distance measurement and \(r\) is the radius to select the scope of neighbors. If the distance between \(o\)’s location and \(r\) is less than \(r\), then \(o\) and \(r\) are neighbors to each other. The neighbor system is widely used in image processing where the pixel is considered as the site. As people with different attributes may have different interests, the radius should be as small as possible. To remove the users/items with different attributes in the neighbors at the extreme, we modify \(F_i\) to be a \(k\) nearest neighbor system \(K_{\{\{k\} \mid \text{top}\} k \text{ nearest sites to } t}\). In this case, if the attribute is owned by more than \(k\) users/items, then only the users/items with the same attribute will be involved in neighbors.

Based on the \(k\) nearest neighbor system \(K_{\{\{k\} \mid \text{top}\} k \text{ nearest sites to } t}\), the conditional distribution of latent profiles of users/items in \(n\)-dimensional Markov random field is defined as follows:

\[
P_m(U_i | U_{\neq i}) = P_m\left(U_i | U_{\neq i}, \{k^{\text{th}}_{\{\{j\} \mid \text{top}\} j \text{ nearest sites to } t}\} > 0\right)
\]  
(3)

\[
P_m(I_j | I_{\neq j}) = P_m\left(I_j | I_{\neq j}, k^{\text{th}}_{\{\{j\} \mid \text{top}\} j \text{ nearest sites to } t}\} > 0\right)
\]  
(4)

where \(U_{\neq i}\) is the user set except \(i\), \(I_{\neq j}\) is the item set except \(j\), \(k^{\text{th}}_{\{\{j\} \mid \text{top}\} j \text{ nearest sites to } t}\) is used to indicate user’s nearest neighbors: User \(i\) is the neighbor of user \(j\), if \(k^{\text{th}}_{\{\{j\} \mid \text{top}\} j \text{ nearest sites to } t}\) = 1, otherwise not. The item’s index \(k^{\text{th}}_{\{\{j\} \mid \text{top}\} j \text{ nearest sites to } t}\) has a similar meaning. All the neighbors are measured in their attribute space by Euclidean distance. We suppose the condition distribution of each user is Gaussian. The Gaussian mean is given by the average of all users in the neighbor. Then, the distribution of user \(i\) can be represented as

\[
P_m(U_i | U_{\neq i}) = \frac{1}{\sqrt{2\pi}^{dn}} \exp\left(-\frac{1}{2}(U_i - \bar{U}_i)^T \Sigma_i^{-1}(U_i - \bar{U}_i)\right)
\]  
(5)
where $\hat{U}_j = \sum \frac{k_i^j u_j^i}{k_i}$ and $\Sigma_{ij} = I \times \sigma_i^2$ is a diagonal matrix. $k_i$ is the number of neighbors. $\sigma_i$ is a hyper parameter to control the uncertainty of the prior. The larger the variance, the higher the probability that $U_i$ is different from $\bar{U}_i$. Similar to Eq. (5), we can formulate the prior probability as 

$$P_m(U_i|l_{-i}) = \frac{1}{\sqrt{(2\pi)^d \Sigma_i}} \exp \left( -\frac{1}{2} (U_i, -\bar{U}_i)^T \Sigma_i^{-1} (U_i, -\bar{U}_i) \right)$$

(6)

where $\bar{U}_i = \sum \frac{k_i^j u_j^i}{k_i}$, $\Sigma_i = I \times \sigma_i^2$ and $k_i$ is the number of neighbors.

4.2. N-dimensional MRF collaborative filtering

The latent profile determines the attitude of the user to an item: if the latent profiles of both users and items are very similar, then the response is very strong, otherwise weak. This derives the basic model of matrix factorization:

$$P(R_{ij}|U_i, I_j) = N(R_{ij}; U_i^T I_j, \sigma)$$

(7)

We could easily calculate the probability of ratings when the profiles $U$ and $I$ are given. To cope with cold-start, we define the $n$-dimensional Markov random field as the prior of users and items. The distribution of ratings is reformulated as

$$P(R, U, I | \Omega, \Theta) \propto \prod_{i=1}^{n} \prod_{j=1}^{m} \prod_{l \neq i} P(R_{ij}^l | U_i, I_j) P(U_i | \Omega) P(I_j | \Theta)$$

(8)

where

$$P(U_i | \Omega) = \prod_{j=1}^{m} P(U_i | l_{-i}^{(j)^c}, l_{ij}^{(j)})$$

(9)

$$P(l_{ij}^{(j)}) = \prod_{j=1}^{m} P(l_{ij} | l_{ij}^{(j)} > 0)$$

(10)

$$P_m(\cdot)$$ denotes the conditional Markov random field probability.

Then, the probability of unobserved ratings $R_{ij}^{(j)}$ are conditioned on both the observed ratings and the attributes

$$P(R_{ij}^{(j)} | R_{-ij}^{(j)}, \Omega, \Theta) = \int P(R_{ij}^{(j)} | U_i, I_j, R_{-ij}^{(j)}) P(R_{-ij}^{(j)} | \Omega, \Theta) dU_i dI_j$$

(11)

In this equation, the first two terms are defined by Gaussian distribution and the last two terms are priors from attributes. Computing the distribution is very difficult due to the high dimensionality of $(U, I)$ and the Markov random field prior. We adopt the point estimation to approximate this distribution.

The posterior distribution of $(U, I)$ is defined as

$$P(U, I | l_{-i}^*, \hat{U}_i, R^+, \hat{U}_i, R^+ \hat{U}_i) = \frac{P(R^+, U, I | l_{-i}^*, \hat{U}_i, R^+, \hat{U}_i, R^+ \hat{U}_i)}{\int P(R^+, U, I | l_{-i}^*, \hat{U}_i, R^+, \hat{U}_i, R^+ \hat{U}_i) dU_i dI_j}$$

(12)

where the denominator of Eq. (12) is constant with respect to $(U, I)$. Maximizing it can be formulated as

$$\{U_i, I_j^* = \arg \max_{(U, I)} \prod_{i=1}^{n} \prod_{j=1}^{m} P(R_{ij}^+ | U_i, I_j) P_m(U_i) P_m(I_j)$$

(13)

Lemma 1 (Conditional dependence of user's profile). Given the latent profile of items $I$, a user $U_i$ only depends on the interacted items, her/his neighbors, and the users who take him/her as the neighbors. That is $P(U_i | l_{-i}^*, R^+) = P(U_i | \hat{U}_i, I_j, R^+)$ where $\hat{U}_i = |U_i| \{ i \in K_i \} \text{ or } i \in K_i^d$, $K_i^d = \{ i | \hat{U}_i^j = 0 \}$, $A_i = |I_i| \{ I_j = 1 \}$ and $R^+ = \{ R_{ij}^+ | I_j = 1 \}$

Lemma 2 (Conditional dependence of item's profile). Given the latent profile of users $U$, an item $I_j$ only depends on the interacted users, its neighbors, and the items which take it as the neighbors. That is $P(I_j | U_i, l_{-j}^*, R^+) = P(I_j | \hat{U}_j, U_i, R^+ \hat{U}_j)$ where $\hat{U}_j = |I_j| \{ j \in K_j \} \text{ or } j \in K_j^d$, $K_j^d = \{ j | \hat{U}_j^i = 0 \}$, $B_j = |U_j| \{ U_i = 1 \}$ and $R^+ = \{ R_{ij}^+ | U_i = 1 \}$

Proof. The joint probability is

$$P(R^+, U, I | \Omega, \Theta) = \prod_{i=1}^{n} \prod_{j=1}^{m} P(R_{ij}^+ | U_i, I_j) P(U_i | \Omega) P(I_j | \Theta)$$

(14)

By integrating $U_i$ out, we can get

$$P(R^+, U, I | \Omega, \Theta) = \prod_{i=1}^{n} \prod_{j=1}^{m} P(R_{ij}^+ | U_i, I_j) P(U_i | \Omega)$$

(15)

$$C(U_i, A_i) = \prod_{j=1}^{m} P(R_{ij}^+ | U_i, I_j) = \prod_{i \in A_i} P(R_{ij}^+ | U_i, A_i)$$

(16)

$$Q(U_i, \hat{U}_i) = \prod_{i \in A_i} P_m(U_i | \hat{U}_i)$$

(17)

Then, the MAP of $U_i$ is

$$P(U_i | l_{-i}^*, R^+) = \frac{P(R^+, U_i, I_j^* | \Omega, \Theta)}{\int P(R^+, U_i, I_j^* | \Omega, \Theta) dU_i}$$

(18)

In Eq. (18), the random variables that $U_i$ does not depend on are canceled by division. Lemma 1 is proved. By exchanging $U$ and $I$, Lemma 2 can be proved in a similar way.

Theorem 1 (Conditional logarithmic concavity). (1) Given all the dependence of profile $U_i$, optimizing the logarithm form of MAP with respect to $U_i$ is a concave problem. (2) Given all the dependence of profile $I_j$, optimizing the logarithm form of MAP with respect to $I_j$ is a concave problem.

Proof. From Lemma 1, any user $U_i$ only depends on $A_i, \hat{U}_i, I_j$, and $R^+$. The posterior distribution of $U_i$ is

$$P(U_i | \hat{U}_i, A_i, R^+) = \frac{P(R_{ij}^+ | U_i, A_i) P(U_i | \hat{U}_i)}{\int P(R_{ij}^+ | U_i, A_i) P(U_i | \hat{U}_i) dU_i}$$

(19)

Since $U_i$ in the denominator of Eq. (19) is integrated out, maximizing the posterior is equal to maximize $L(U_i) = P(R_{ij}^+ | U_i, A_i) P(U_i | \hat{U}_i)$.

(20)

By substituting Eqs. (5), (6), (16) and (17) into Eq. (20), we can get

$$L(U_i) = \prod_{j \in A_i} \exp \left( -\frac{1}{2\sigma_i^2} (u_i - \hat{U}_i)^T (u_i - \hat{U}_i) \right)$$

(21)

where $C_i$ is some constant that does not depend on $U_i$. The logarithm of $L(U_i)$ is a concave function, that is, maximizing the logarithm form of MAP with respect to $U_i$ is a concave problem. By the similar reason, the logarithm of MAP with respect to $I_j$ is a concave problem.

From Theorem 1, we can see that when all other variables are fixed, $U_i | l_{-i}$ has an analytical solution. Therefore, an alternating iterative optimization algorithm, alternating least squares (ALS), can be applied. Usually the negative logarithm form of MAP is
applied for optimization, which can be reformulated as
\[
L = -\sum_i \sum_j \log(P(R_{ij}^U | U_i]) - \log(P_m(U)) - \log(P_m(I))
\]  
(22)

4.3. ALS-like optimization

Let the hyper parameter be \( \Gamma = \{d, k_U, k_I, \lambda_U, \lambda_I\} \), \( \lambda_U = \sigma^2/\sigma_{U}^2 \), \( \lambda_I = \sigma^2/\sigma_{I}^2 \), which is fixed when optimizing mrf-MF. The iterating processes can be conducted by iteratively solving one variable while others are fixed. First we derive the solution for \( U \) when others are fixed. \( I \) can also be solved with \( U \) fixed. The two steps are iterated until converged. The learning phase is illustrated in Algorithm 1.

4.3.1. Solving \( U \) with \( I \) fixed

In this model, we fix \( U \) to minimize the loss function with respect to \( U \). According to Lemma 1, optimizing \( L \) with respect to \( U \) can be represented as
\[
L = -\sum_i \sum_j \log(P(R_{ij}^U | U_i]) - \log(P_m(U_i | U_j)) - \sum_p \log(P_m(U_p | U_j)) + C,
\]
(23)

where \( C \) is some constant that does not depend on \( U \). The third term of Eq. (23) represents the relationship between user \( i \) and its neighbors. Substituting Eqs. (5) and (7) into the loss function, we get
\[
L = \frac{1}{2} \sum_i \sum_j (R_{ij} - U_i^T I_j)^2 + \frac{1}{2} \sum_p \| U_p - \sum_{p'} = \frac{1}{k_U} K_{ij}^U(p,p')U_{p'} \|^2_F + C.
\]

(24)

We see that minimizing the loss function with respect to \( U \) is a convex problem which has an analytical solution. Its derivation is
\[
\frac{\partial L}{\partial U} = \sum_i \sum_j (U_i^T I_j - R_{ij}) \frac{\partial}{\partial U} \left( \sum_{p'} = \frac{1}{k_U} K_{ij}^U(p,p')U_{p'} \right)
\]
\[
= \sum_i \sum_j (U_i^T I_j - R_{ij}) \frac{\partial}{\partial U} \frac{1}{k_U} K_{ij}^U(p,p')U_{p'}
\]
\[
= \sum_i \sum_j (U_i^T I_j - R_{ij}) \frac{\partial}{\partial U} \frac{1}{k_U} K_{ij}^U(p,p')U_{p'} + \lambda_U \left( U_i - \sum_{p'} = \frac{1}{k_U} K_{ij}^U(p,p')U_{p'} \right)
\]
\[
- \lambda_I \sum_p \left( U_p - \sum_{p'} = \frac{1}{k_U} K_{ij}^U(p,p')U_{p'} \right) K_{ij}^U(p, i)
\]

(25)

Let \( \frac{\partial L}{\partial U} = 0 \) and then we can get the analytical solution which is
\[
U_i = A_{ij}^U - B_{ij}^U
\]

(26)

where
\[
A_{ij}^U = \sum_j 1_j I_j^T + \lambda_U \frac{1}{k_U} K_{ij}^U(p, i)^2
\]
\[
B_{ij}^U = \sum_j 1_j R_j^T + \lambda_I \frac{1}{k_U} K_{ij}^U(p, i)U_{p'}
\]
\[
- \lambda_I \sum_p \left( U_p - \sum_{p'} = \frac{1}{k_U} K_{ij}^U(p,p')U_{p'} \right) K_{ij}^U(p, i)
\]
\[
S_{ij}^U(i) = \sum_p K_{ij}^U(p, i)U_p
\]

4.3.2. Solving \( I \) with \( U \) fixed

Analogously, we can solve one item's latent profile \( I_j \) by fixing \( U \) and \( I_{-j} \), which is also a convex problem. The solution is
\[
I_j = A_{ij}^I - B_{ij}^I
\]

(27)

where
\[
A_{ij}^I = \sum_i 1_i U_i^T + \lambda_I \frac{1}{k_I} K_{ij}^I(p, j)^2
\]
\[
B_{ij}^I = \sum_i 1_i R_i^T + \lambda_U \frac{1}{k_I} K_{ij}^I(p, j)U_{p'}
\]
\[
- \lambda_U \sum_p \left( U_p - \sum_{p'} = \frac{1}{k_I} K_{ij}^I(p,p')U_{p'} \right) K_{ij}^I(p, j)
\]
\[
S_{ij}^I(j) = \sum_p K_{ij}^I(p, j)U_p
\]

4.3.3. Computing complexity analysis

In each iteration of Algorithm 1, there are two different parameters to optimize \( \{U, I\} \). First we analyze the complexity of computing \( U \). In Eq. (26), the expression \( \sum_{p'=1}^m K_{ij}^U(i, p') \) can be finished in \( k \) times by the index technique, because there are only \( k \) elements in \( K_{ij}^U(i, p) \), and all others are 0. The computing cost of Eq. (26) arises from two parts: \( A_{ij}^U \) and \( B_{ij}^U \). Let the average number of ratings generated by users be \( N \) and the average number of ratings related to items be \( M \). We can get the cost to compute \( B_{ij}^U \) as \( O((N+d+kd+kd)=O(N+d+kd+kd)) \) and \( A_{ij}^U \) as \( O(N+d+kd) \). The cost to compute the inverse \( A_{ij}^{-1} \) is \( O(d^3) \). The total cost to compute \( U_i \) is \( O((N+d+kd+kd)=O(N+M+Md^2)(n+m)(n+m)d^2) \). Compared to the basic matrix factorization with a zero Gaussian prior, the increased cost is \( O(n+m)kd \). The ALS-like solution is so efficient that it can converge in tens of iterations which will be illustrated in Section 5.

4.3.4. Strategy for adjusting hyper parameters

In order to simplify the description, we let \( \lambda = \lambda_U = \lambda_I \) and \( k = k_U = k_I \). We discuss the effect of hyper parameters and give the suggestion on how to tune them as follows.

\( \lambda \) is the parameter to control the importance of MRF prior. When \( \lambda \) is set too large, the variance in Eqs. (5)–(8) becomes very small. Because the neighbors contain noise, the small variance will make noise to be trusted as data. Consequently, the performance will be affected. If \( \lambda \) is set too small, then all the neighbors are not trusted and MRF does not constrain the model in the training phase. The model is degenerated to use k-nn mapping to generate the latent profiles of cold-start users/items. The k-nn is a lazy algorithm that usually has a large prediction error. The error will be added to that of the matrix factorization model in the prediction phase. Hence, it will have a low performance. By experiments (see Section 5.4), we found that when \( \lambda \) increases from 0 to its optimal value, the performance increases very fast. Based on this observation, we suggest that it is useful to tune \( \lambda \) by increasing from 0 until the performance stops increasing.

\( d \) is the dimensionality of the latent profile. Usually, a small value of \( d \) will generate an under-fitting model and a large value will output an over-fitting model. In mrf-MF, when \( d \) takes a large value, the model
can resist over-fitting by MRF prior. The performance remains stable over a large range. From our test (see Section 5.4), it is shown that when $d$ increases from 0 to its optimal value, the performance increases very fast. After $d$ becomes larger than its optimal value, the model remains stable. However, the computing cost depends on $d$: A large value of $d$ will slow down the training speed. We recommend tuning the parameter $d$ by increasing from 5 until the performance stops increasing or the performance becomes acceptable. It is a trade-off between accuracy and computing cost.

$k$ is the number of neighbors in the MRF. A large value will include more uncorrelated users/items in neighbors and add more noise. A small value will make the prior unstable. Also, the latent profile of new item is given by

$$I_{new} = \sum_{j} k_{I_{new}}^j f_i^{(0)}$$

where $k_{I_{new}}^j (\theta_{new})$ is the $k_i$ nearest neighbors of $I_{new}$ and $k_{I_{new}}^j$ is the $k_j$ nearest neighbors of $I_{new}$

The method to make a recommendation for the new user/item is given in Algorithm 2. In the predicting algorithm, the rating is generated by the product of user’s and item’s profiles. The profile of new user is not learned in the learning phase and will be given by Eq. (32) or (33). When the user is already in the training set, then the profile learned in the learning phase is taken. The profile from the learning phase contains not only the MRF prior but also the rating preference, which describes more personal information. The item takes the same manner to generate its profile. In the system cold-start scenario, as both user and item are new, their profile can only be generated from the Markov field prior. This is the most difficult situation, because none of their historical ratings are available.

**Algorithm 2.** Cold-start recommending.

**Input:**
- The item candidate set $\tilde{I}$ and their attributes $\tilde{\Gamma}$ to recommend
- A user’s attribute $\Theta_i$
- Then number of items to recommend $N$

**Output:**
- A list of items;
  1. if user $i$ is new then
  2. $U_i^* \leftarrow U_i$
  3. else
  4. $U_i^* \leftarrow U_i$
  5. end if
  6. for all $j$ in $\tilde{I}$ do
  7. if item $j$ is new then
  8. $I_j^* \leftarrow I_j$
  9. else
  10. $I_j^* \leftarrow I_j$
  11. end if
  12. $R_i^* \leftarrow U_i^* I_j^*$
  13. end for
  14. Sort $R_i^*$ by descent
  15. return Top $N$ items by $R_i^*$

5. Experiments

In this section, we conduct experiments on two recommendation datasets, Movielens 100K and Movielens 1M, to demonstrate the performance of the proposed model. Firstly, we introduce the datasets and experiment setup. Secondly, the methods for comparison are listed. Thirdly, we report the experimental results. Finally, we analyze the effect of the hyper parameters on the proposed model.

---

5.1. Experimental datasets and setup

To compare the performance of algorithms in cold-start scenarios, we chose two datasets, Movielens 100K and Movielens 1M, which are summarized in Table 2. In the two datasets, the attributes of both users and items are available, such that we can evaluate algorithms in all three cold-start scenarios. The user's attributes include 20 different occupations plus an 'unknown'. The male and female users were considered and the 'unknown' users were removed. The genre of movies includes 17 different types. In Table 2, the density indicates the sparsity of the rating matrix. Our task was to predict the favor order of the items for testing users, which is also called the Top-N recommendation task. In our experiments, we evaluated the algorithm with the top 10 recommended items.

Ten-fold cross validation was used as the evaluation strategy. The users and items were divided into 10 parts. For each validation, 1 part was denoted as new users/items and the rest were existing users/items. Existing users and items, whose ratings were observed, were used to train the model. The existing users and new items were used for the evaluation of item cold-start recommendations. The new users and existing items are used for user cold-start recommendations. The most difficult scenario, system cold-start recommendation, was evaluated by new users and items. In the testing scenarios, all the ratings were used for evaluation and none were available in training phase.

Top-N recommendation metrics, which measure the result based on how many related items were retrieved for each user, were applied to evaluate the performance of algorithms. Discount cumulative gain (DCG) [33] is one of the metrics used to measure the quality of ranking. The recommendation results for a specific user can be viewed as ranking results: the items should be in the top rank that the user most likes. The DCG may vary from 0 to a large value. To make the value locate in the range of [0, 1], the DCG is normalized by its ideal value. The normalized DCG (nDCG) of user \( i \) is defined as

\[
\text{nDCG}_i = \frac{\text{DCG}_i^p}{\text{IDCG}_i^p} = \frac{1}{\log_2(j+1)} \sum_{j=1}^{R_i} \frac{2^{R_i} - 1}{\log_2(j+1)}
\]

where \( p \) denotes the number of recommended items; \( R_i \) is the real rating of user \( i \) on \( l \)-th item in the recommendation list and \( \text{IDCG}_i^p \) is the best DCG value for user \( i \). The larger nDCG the algorithm gets, the better performance the algorithm achieves. For each algorithm, the average nDCG over all users is listed in the experimental results.

Precision, Recall and F1 score were also used for evaluation. Precision describes how many relevant items are in the recommended lists. Higher precision indicates less non-relevant items in the recommendation list. Recall is the fraction of relevant items to the successfully recommended items. Low recall means only small a number of relevant items is recommended. When a recommender has very high precision, it usually obtains low recall, and vice versa. The high-effective model should get a high score in both precision and recall. F1 score is a measurement that combines precision and recall, which is commonly used as the final measurement. All three metrics are defined as

\[
\begin{align*}
\text{Precision} & = \frac{|\text{relevant items} \cap \text{recommended items}|}{|\text{recommended items}|} \\
\text{Recall} & = \frac{|\text{relevant items} \cap \text{recommended items}|}{|\text{relevant items}|} \\
F1 & = 2 \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\end{align*}
\]

The range of these scores is [0, 1]. 0 indicates the worst situation and 1 represents the best result. In our experiments, the recommendation list for each user was cut off at the 10th rank, that is, we only considered the performance on the top 10 recommendation results, because most people just browse a few recommendations in real situations.

5.2. Methods for comparison

Three baseline recommendation algorithms and 4 different recommenders for cold-start were implemented for recommendations. Baseline methods included Random Recommender (RanRec), Most Popular (MP) and Vibes Affinity (VA) [34,12]. RanRec, which is a baseline algorithm to validate if a recommender works, recommends items by random guess. MP ranks the items based on their popularity, which is not a personalized recommendation algorithm. As only the items' popularity is considered, MP could only be applied in the user cold-start scenario. In practice, this algorithm is usually hard to beat. VA defines some filterbots to generate rates for new users based on their attributes, which can be found in Ref. [34].

Pairwise Preference Regression (PPR) [12] is a pair-wise regression based model which can deal with various cold-start scenarios. In this model, some filterbots are applied to create new features for both users and items. User's and item's features (including existing attributes and created features) are coupled by the outer product to represent the user-item pair. The corresponding rating is considered as a target and the linear model is used to learn the regression coefficients. In the predictions phase, only the features and learned coefficients are used to make recommendation and the observed ratings for existing users/items are ignored. In our model, the profile involves the ratings of existing users/items which make the recommendation more precise.

The Regression-based Latent Factor Model (RLFM) [13] proposes to fit a latent profile to the corresponding attributes by linear regression, where the latent profile is learned by factorizing the rating matrix. The profile of a new user/item, which is unavailable in the cold-start scenario, can be predicted by the trained regression model. In the prediction phase, the profile of warm-start users/items naturally involves their historical ratings such that the profile contains the personal preference or item's characteristic. The original solution to RLFM is MCEM which is very slow. Similar to the proposed method, RLFM was optimized by the alternating least squares strategy in our experiments. RLMF was extended to nonlinear regression by Zhang et al. [14] who proposed a general matrix factorization model with flexible regression priors. In their general models, several multivariate regression methods are taken into account, such as lasso (lasso-MF), Random Forest (rf-MF) and Classification and Regression Tree (dr-MF). The MCEM algorithm is applied to learn their models. For

<table>
<thead>
<tr>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic statistical information of datasets.</td>
</tr>
<tr>
<td>Datasets</td>
</tr>
<tr>
<td>Movielens 100K</td>
</tr>
<tr>
<td>Movielens 1M</td>
</tr>
</tbody>
</table>
the purpose of efficiency, those models were solved by an alternating least squares algorithm in our experiment. We found that the performance of lasso-MF is worse than the random recommendation because the attributes of both users and items are already very sparse. The lasso-MF is not reported in our results.

5.3. Experimental results

To report the best performance of each algorithm, we search the parameter by 10-fold cross validation. The dimensionality is searched from 5 to 30 steps by 5. It will decrease the performance, when the dimensionality becomes larger than 30. The regularization coefficient $\lambda$ is selected from $\{10, 50, 100, 200, 300\}$. For simplicity, the regularization coefficient of both users and items are set to be equal, i.e. $\lambda = \lambda_U = \lambda_L$. In our algorithm, the year and age is scaled by 0.1 to reduce the impact. In Table 3, we report the dimensionality of matrix factorization based methods when they achieve the best performance. It can be seen that most methods can achieve their best performance in low dimension which is different from warm-start recommendation [7]. This is because there is very little information available in the cold-start scenario. A high dimensional latent profile will model a lot of noise and affect the performance.

The experimental results in all cold-start scenarios are reported in Table 5 (Movielens 100K) and Table 6 (Movielens 1M). Results in all scenarios were generated by the same trained model for each algorithm. The score of NDCG, Precision@10, Recall@10 and F1@10 in three different cold-start scenarios are given in these result tables. A higher score indicates better performance. Ratings greater than or equal to 4 were marked as positive and others were negative when calculating Precision, Recall and F1 scores. The four metrics in all scenarios were roughly agreed, i.e., the better performed algorithm will gain a higher score in all metrics. All experiments were conducted on a workstation equipped with a $24 \times 2.30$ GHz CPU and 64 GB RAM. The algorithms reported in our experiment are all implemented in python. The computation time of all algorithms on their best performance over datasets is reported in Table 4. As the baseline methods, RanRec, MP, and VA, are rule-based and almost do not need time to train, we do not report their time. From Table 4, it can be seen that the proposed mrf-MF is efficient. Its computing time was less than that of dt-MF, rf-MF and PPR for the two datasets. Although mrf-MF cost a little more time than RLFM, our method achieved the best performance. The features of PPR in the user scenario and item/system scenario were different, and so its time in different scenarios is reported separately. PPR was the most time consuming because of its pairwise trick.

From Tables 5 and 6, we can see that the proposed method almost achieves the best performance in all scenarios. RanRec recommends items randomly, so any method that cannot at least outperform this is nonsense. By giving its result, we can see how far we can improve upon simply guessing at random. As in the PPR

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Dimensionality of latent profile for each algorithm on its best performance over datasets.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>Movielens 100K</td>
</tr>
<tr>
<td>RLFM</td>
<td>5</td>
</tr>
<tr>
<td>rf-MF</td>
<td>10</td>
</tr>
<tr>
<td>dt-MF</td>
<td>5</td>
</tr>
<tr>
<td>mrf-MF$^a$</td>
<td>5</td>
</tr>
</tbody>
</table>

$^a$ Proposed method.

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Time consumption of all algorithms on their best performance over datasets.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>Movielens 100K (s)</td>
</tr>
<tr>
<td>PPR (user)</td>
<td>2735</td>
</tr>
<tr>
<td>PPR (item/system)</td>
<td>1392</td>
</tr>
<tr>
<td>RLFM</td>
<td>149</td>
</tr>
<tr>
<td>rf-MF</td>
<td>976</td>
</tr>
<tr>
<td>dt-MF</td>
<td>265</td>
</tr>
<tr>
<td>mrf-MF$^a$</td>
<td>228</td>
</tr>
</tbody>
</table>

$^a$ Proposed method.

<table>
<thead>
<tr>
<th>Table 5</th>
<th>Results on Movielens 100K.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold-start scenarios</td>
<td>Algorithms</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>User Cold-start (Recommend existing items to new users)</td>
<td>RanRec</td>
</tr>
<tr>
<td></td>
<td>MP</td>
</tr>
<tr>
<td></td>
<td>VA</td>
</tr>
<tr>
<td></td>
<td>PPR</td>
</tr>
<tr>
<td></td>
<td>RLFM</td>
</tr>
<tr>
<td></td>
<td>rf-MF</td>
</tr>
<tr>
<td></td>
<td>dt-MF</td>
</tr>
<tr>
<td></td>
<td>mrf-MF$^a$</td>
</tr>
</tbody>
</table>

$^a$ Proposed method.

| Item Cold-start (Recommend new items to existing users) | RanRec | 0.8169 | 0.0032 | 0.6357 | 0.0115 | 0.6552 | 0.0167 | 0.6217 | 0.0090 |
| | MP | 0.8292 | 0.0158 | 0.6526 | 0.0179 | 0.6730 | 0.0165 | 0.6383 | 0.0111 |
| | VA | 0.8230 | 0.0094 | 0.6449 | 0.0137 | 0.6645 | 0.0148 | 0.6305 | 0.0079 |
| | PPR | 0.8525 | 0.0103 | 0.6714 | 0.0204 | 0.7832 | 0.0204 | 0.6475 | 0.0141 |
| | RLFM | 0.8495 | 0.0076 | 0.6204 | 0.0258 | 0.7945 | 0.0235 | 0.6842 | 0.0158 |
| | rf-MF | 0.8517 | 0.0114 | 0.6103 | 0.0215 | 0.7891 | 0.0226 | 0.6469 | 0.0130 |
| | dt-MF | 0.8598 | 0.0080 | 0.6213 | 0.0219 | 0.7832 | 0.0225 | 0.6494 | 0.0126 |
| | mrf-MF$^a$ | 0.8568 | 0.0080 | 0.6213 | 0.0219 | 0.7832 | 0.0225 | 0.6494 | 0.0126 |

| System Cold-start (Recommend new items to new users) | RanRec | 0.8169 | 0.0201 | 0.6331 | 0.0288 | 0.6541 | 0.0407 | 0.6214 | 0.0279 |
| | MP | 0.8315 | 0.0020 | 0.6561 | 0.0396 | 0.6735 | 0.0401 | 0.6415 | 0.0322 |
| | VA | 0.8271 | 0.0150 | 0.6381 | 0.0347 | 0.6590 | 0.0453 | 0.6260 | 0.0343 |
| | PPR | 0.8483 | 0.0180 | 0.3964 | 0.0303 | 0.7807 | 0.0237 | 0.6274 | 0.0214 |
| | RLFM | 0.8457 | 0.0148 | 0.6099 | 0.0520 | 0.7849 | 0.0456 | 0.6429 | 0.0310 |
| | rf-MF | 0.8468 | 0.0092 | 0.6035 | 0.0405 | 0.7789 | 0.0312 | 0.6336 | 0.0246 |
| | dt-MF | 0.8593 | 0.0076 | 0.6096 | 0.0322 | 0.7716 | 0.0407 | 0.6365 | 0.0325 |

The score in boldface is the best in each group.

$^a$ Proposed method.
method, attributes directly regressed to ratings. The assumption that attributes can be linearly mapped to ratings is too strong and affects the performance. In both datasets and all scenarios, PPR did not achieve as high a score as matrix factorization based methods. VA’s performance was always lower than the matrix factorization based methods. This is because VA is a static method to make recommendations and cannot deal with the cold-start problem flexibly. MP recommended the most popular items, which got relatively good scores. However, it is not a personalized algorithm, such that the performance cannot get better than personalized ones. The remaining methods are matrix factorization based. RLFM got better performance than dt-MF and rf-MF in item and system cold-start scenarios on Movielens 100K. The reason is that when the training data is small, the linear method can outperform the non-linear methods. On Movielens 1M where more training data are available, dt-MF and rf-MF got better performance. Comparing rf-MF and dt-MF, we can see that the decision tree method performed better in all cold-start scenarios. However, they always try to optimize the mapping from attributes space to latent profile space. The regression error and matrix factorization error are added to the effect on performance. The proposed method just uses MRF priors to constrain the latent profile and learns the mapping function implicitly. Through MRF prior, the latent profiles whose owners have similar attributes are constrained to be similar in the matrix factorization optimizing phase. In the prediction phase, the profile of new items/users can be obtained from its neighbors. As a result, the implicit mapping function has the capability of non-linear situations. Therefore, the mrf-MF model outperformed the above mentioned methods.

5.4. Hyper Parameter Analysis

In the mrf-MF model, there are three hyper parameters to tune: the dimensionality of latent profile $d$, the regularization coefficient of MRF $\lambda = \lambda_0 = \lambda_1$, the number of neighbors $k = k^u = k^d$. The analysis was conducted in the user cold-start scenario on the Movielens 100K dataset and is indicated by NDGG.

The dimensionality of latent profile $d$ is a key parameter to control the freedom of the latent profile. When $d$ is very large, the model will over-fit. If $d$ is too small, under-fitting happens. Usually, a prior, zero mean Gaussian for instance, is added to alleviate the over-fitting [7]. In our model, the MRF prior can make the model robust to $d$. We analyzed the effect of $d$ in our model by fixing $\lambda = 50$ and $k = 100$. In Fig. 3, the nDCG score over $d$ is reported. We can see that, when $d \geq 3$ the model outputs a very stable result. When $d < 3$, there was not enough freedom of latent profile to describe the user/item so the performance decreased. To save computation, we can choose a smaller $d$, if the performance is acceptable.

The regularization weight $\lambda$ is the parameter to control the strength of MRF constraint. Small values of $\lambda$ reduces the dependence of prior and when $\lambda = 0$ the prior is removed. Without MRF constraint, the predicted result is generated by $k$ nearest neighbor regression. When $\lambda$ is very large, the weight of noise in neighbors will be increased. In our experiments, we found that a very strong constraint decreased the performance. Fig. 4 shows the performance changing over $\lambda$ where $d = 50$ and $k = 100$. When $\lambda > 50$ the performance began to slowly decrease. When $\lambda$ increased from 0 to 10, the performance significantly improved, which means the MRF prior was effective. Based on this observation, we suggest choosing $\lambda$ by increasing from 0.

The size of neighbors, $k$, controls the range of neighbors in the Markov network. Small ranges will generate an unstable model and large ranges increase the computation cost and includes more noise in neighbors. When large numbers of uncorrelated users/items are included, the error in learned latent profile increases. To demonstrate the effect of $k$, the nDCG over $k$ is illustrated in Fig. 5, where $d = 5$ and $\lambda = 50$. $k$ greater than 600 is not reported due to the high computation cost. When decreasing from 100 to 1, as the model becomes unstable, the nDCG score begins to decrease. The parameter $k$ from 100 to 300 will output the best performance. In practice, when the performance is acceptable, we choose the smaller $k$ to save computation.

From the above analysis, it can be seen that the proposed model can achieve its optimal performance in a large range of

<table>
<thead>
<tr>
<th>Table 6</th>
<th>Results on Movielens 1M.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold-start scenarios</td>
<td>Algorithms</td>
</tr>
<tr>
<td>User Cold-start (Recommend Existing items to new users)</td>
<td>RanRec</td>
</tr>
<tr>
<td></td>
<td>MF</td>
</tr>
<tr>
<td></td>
<td>VA</td>
</tr>
<tr>
<td></td>
<td>PPR</td>
</tr>
<tr>
<td></td>
<td>RLFM</td>
</tr>
<tr>
<td></td>
<td>rf-MF</td>
</tr>
<tr>
<td></td>
<td>dt-MF</td>
</tr>
<tr>
<td></td>
<td>mrf-MF*</td>
</tr>
<tr>
<td>Item Cold-start (Recommend new items to Existing users)</td>
<td>RanRec</td>
</tr>
<tr>
<td></td>
<td>VA</td>
</tr>
<tr>
<td></td>
<td>PPR</td>
</tr>
<tr>
<td></td>
<td>RLFM</td>
</tr>
<tr>
<td></td>
<td>rf-MF</td>
</tr>
<tr>
<td></td>
<td>dt-MF</td>
</tr>
<tr>
<td></td>
<td>mrf-MF*</td>
</tr>
<tr>
<td>System Cold-start (Recommend new items to new users)</td>
<td>RanRec</td>
</tr>
<tr>
<td></td>
<td>VA</td>
</tr>
<tr>
<td></td>
<td>PPR</td>
</tr>
<tr>
<td></td>
<td>RLFM</td>
</tr>
<tr>
<td></td>
<td>rf-MF</td>
</tr>
<tr>
<td></td>
<td>dt-MF</td>
</tr>
<tr>
<td></td>
<td>mrf-MF*</td>
</tr>
</tbody>
</table>

The score in boldface is the best in each group.

* Proposed method.
parameter spaces. In practice we can initially choose some small value of $d$, $k$ and $\lambda$ and then greedily increase them until the performance begins to decrease or be acceptable.

5.5. Speed of convergence

We evaluated the speed of convergence on Movielens 100K. Fig. 6 shows the speed of the convergence. Fig. 7 illustrates the nDCG score over iterations. From the two figures, we can see that when the loss value decreases very quickly, the nDCG score also increases very quickly. The model can reach its optimal value in several iterations. In all the experiments, we set the number of iterations to be 50.

6. Conclusion and future work

In this paper, we have presented a new matrix factorization based method with an $n$-dimensional Markov random field prior (mrf-MF) to cope with three types of cold-start problem. The proposed method can deal with all cold-start scenarios by training a single model. Experimental results on movie datasets demonstrate that the proposed method can achieve better performance when compared to several matrix-based methods and regression based methods. The proposed model can obtain its optimal performance in a large range of parameter spaces. The best parameters can be searched by greedily increasing them until the performance begins to decrease.

In the real world, there is another cold-start scenario: very sparse training data may be encountered when a new system is built or when the number of active users is very small. In this scenario, the non-linear models and the proposed model cannot achieve as high a performance as the linear method. How to build a reliable model for this situation will be studied in our future work.

As can be seen in Section 5, the time required for the Movielens 1M dataset was nearly half an hour. It will be valuable to extend this model to be a more scalable model such as Refs. [35,36].
Acknowledgment

The authors would like to thank the reviewers for their valuable comments and suggestions. Thanks to Rodney Berry and Antarlovich Michael for their proofreading. This work was supported in part by the National Natural Science Foundation of China (NSFC) (Nos. 61232001, 91220301, 61170035 and 61272420), by Jiangsu Natural Science Foundation (No. BK20131351), by the Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions, by the 111 Project (No. B13022), and by Six talent peaks project in Jiangsu Province.

References


Furong Peng received the B.Sc. degree in 2009 in computer science of Nanjing University of Science and Technology, China. He is currently a Ph.D. candidate in the Computer Science and Engineering Department of Nanjing University of Science and Technology, China. His research interests include data mining, machine learning, and recommendation system.

Yongli Wang received the Ph.D. degree from the Southwest Jiaotong University in 2006. He is currently a Professor in the Department of Computer Science and Engineering at Nanjing University of Science and Technology, China. He is the author or coauthor of more than 60 papers in database and leading pattern recognition journals and conferences. His research includes sensor networks, data streams management, mobile computing, data mining, cyber-physical system, and health care monitoring, and is supported by the NSFC and other agencies. For his research activities, he also spent extended periods of time at Drexel...
Richard Yi-Da Xu received the B.Eng. degree in computer engineering from the University of New South Wales, Sydney, NSW, Australia, in 2001, and the Ph.D. degree in computer sciences from the University of Technology at Sydney (UTS), Sydney, NSW, Australia, in 2006. He is currently a Senior Lecturer with the School of Computing and Communications, UTS. His current research interests include machine learning, computer vision, and statistical data mining.

Chao Ma received the B.S. degree in electronic business and M.S. degree in computer application technology from Nanjing Agriculture University. He is currently a Ph.D. candidate in the School of Computer Science and Engineering of Nanjing University of Science and Technology, China. His research interests include object tracking and data-dependent hashing technology.

Jingyu Yang received the B.S. degree in computer science from Nanjing University of Science and Technology (NUST), Nanjing, China. From 1982 to 1984, he was a visiting scientist at the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign. From 1993 to 1994, he was a Visiting Professor at the Department of Computer Science, Missouri University. In 1998, he acted as a Visiting Professor at Concordia University in Canada. He is currently a Professor in the Department of Computer Science at NUST. He is the author of more than 500 scientific papers in computer vision, pattern recognition, and artificial intelligence. He has won more than 20 national and provincial awards. His current research interests are in the areas of pattern recognition, robot vision, image processing, data fusion, and artificial intelligence.