Recommendation Algorithms For Implicit Information

Xinxin Bai*, Jinlong Wu*†, Haifeng Wang*, Jun Zhang*, Wenjun Yin*, Jin Dong*
*IBM Research - China, Beijing, China
Email: {baixx, whf, zjuncrl, yinwenj, dongjin}@cn.ibm.com
†Peking University, Beijing, China
Email: wujinlong1984@gmail.com

***Abstract*—Collaborative filtering (CF) methods are popular for recommender systems. In this paper we focus on exploring how to use implicit and hybrid information to produce efficient recommendations. We suggest a new similarity measure and rating strategy for neighborhood models, and extend original matrix factorization (MF) models to explore implicit information more efficiently . By the mean time, We extend the new MF models to integrate user or item features and obtain a new hybrid model and a corresponding algorithm. Finally we compare our new models with some well known models in our experiments.**

***Keywords*-recommender system, hybrid algorithm, implicit rating, collaborative filtering, matrix factorization**

I. Introduction

With recommender systems being more and more popular in modern society, many companies like *Amazon* and *Netflix* are making a great deal of effort to establish and improve their own commercial recommender systems.

Recommender system first achieves user preferences by analyzing their histories of usage behaviors, then uses the resulting models to produce personalized recommendations for the users. For example, an online store can learn customer preferences according to ratings they provided for products (these ratings are usually called *explicit ratings*), and what products they bought, added to the cart, and browsed (ratings obtained from these information are usually called *implicit ratings*) [1], [2]. In the light of different ways to produce recommendations, recommender systems are usually classified into three types [3], [1]:

- **Content-based filtering (CBF)**
  CBF methods provide recommendations based on features of users and items which are usually created according to his(her) consuming items. Recommendations to one user are those items whose features best match those of the target user. CBF has some main drawbacks: (1) CBF depends on preprocessing features of items, which usually are not easy to obtain. (2) CBF usually recommends to one user items which are similar with ones he(she) already consumed.
- **Collaborative filtering (CF)**
  Unlike CBF ,CF methods do not require any explicit feature of items or users. They only rely on past behavior of users, which might cause explicit or implicit ratings of users to items. Hence CF is domain free, and can circumvent difficulties faced by CBF. But at the same time CF introduces some new difficulties: (1) Cold start: CF can rarely produce effective recommendations to a new user. Similar difficulty exists for a new item. (2) Scalability: CF systems may produce recommendations to millions of users from tens of thousands of items which are challenging for memory of computers.
- **Hybrid of CBF and CF**
  Hybrid filtering methods combine CBF and CF methods in order to overcome the drawbacks of both CBF and CF and benefit from their advantages. Three main different ways to combine CBF and CF are suggested [3]: weighted combinaton , mixed combination and sequential combination.

So far most of published work focus on exploring how to use explicit ratings, especially after Netflix distributed the Netflix Prize competition [4] in October, 2006. However, only implicit information is available to construct CF systems in some practical circumstances [5]. Supermarkets store plenty of transaction data, but usually obtain little direct information about whether one user likes one product or not. Some work have been published to investigate implicit information, such as [5], [2].Compared with explicit information, implicit information exhibits distinct characteristics [2]. Values of explicit information usually represent the level of preferences, but values of implicit information usually represent frequencies of user actions. For instance, rating 1 and 5 respectively represent "totally dislike" and "quite like" in the Netflix Prize data set. However, in our supermarket data set, ratings represent how many times an item are bought by a user. More detailed distinctions between explicit and implicit information can be found in [2].

In this paper we first explore some methods to analyze implicit information. The implicit information is how many times a customer bought a product in two months. Apart from the direct purchase history, several features for each user are extracted by experts according to the purchase history of the user. We then propose a hybrid model to integrate all known information and then extend the method suggested by Hu et al. [2] to solve the hybrid model efficiently.

II. Preliminaries

A. Notation

Let U and I be the numbers of users and items respectively. Denote the associated value of user u and item i as r_{ui}. For data sets with explicit information, r_{ui} represents the rating of user u to item i. For data sets with implicit information, r_{ui} usually reveals observations for user actions [2]. In this paper , we simply call r_{ui} the rating of user u to item i consistently. Algorithms which consider ungiven ratings of user-item pairs as missing data don't work for implicit information since there is no negative information in this situation [2]. Hence, here we set r_{ui} to be 0 if no action happened between user u and item i, as suggested in [2]. In our supermarket data set, $r_{ui} = 0$ means that user u did not purchase item i.

Denote the training set by $\mathcal{P}$. Without confusion, we also use $\mathcal{P}$ to express the set of all user-item pairs with non-zero ratings on the training set, namely, $\mathcal{P} = \{(u,i)|r_{ui} > 0\}$. Denote $\{i|(u,i) \in \mathcal{P}\}$ by $\mathcal{P}_u$ and $\{u|(u,i) \in \mathcal{P}\}$ by $\mathcal{P}^i$. We usually use bold letters to express tensors, e.g., bold lowercases for vectors, bold capitals for matrices. For example, $\mathbf{R} = (r_{ui}) \in \mathbb{R}^{U\times I}$ represents the rating matrix. The hyper parameters of models are expressed by Greek letters, such as α, β and so forth.

B. Recommendation framework and quality metric

We always construct our models in the training set, and evaluate their effectiveness in the test set. After models are constructed, predictions for all items are acquired from the models, and then items with highest predictions are recommended to users.

In practical applications, the size of the recommendation list is usually restricted by practical situations. So we simply use precision to evaluate the recommendation quality and fix the size of the recommendation list to 10, since the purpose of the paper is to compare different models for recommendations but not to construct a practical recommender system.

III. Neighborhood models

Neighborhood (also called k Nearest Neighbors, or kNN for short) models are utilized to obtain recommendations since the inception of recommender systems [6], [1], [3], and still popular nowadays because they are easy to understand and realize. KNN consist of three steps:

1) **similarity formation** This step finishes the calculation of similarity between users (user-based kNN) or items (item-based kNN). Some widely used similarity measures include *Pearson correlation* and *Cosine*. For example, cosine similarity between user u and v is calculated as follows:

$$S^{\mathrm{c}}_{uv} = \frac{\sum_{i\in\mathcal{P}_{uv}} r_{ui}r_{vi}}{\sqrt{\sum_{i\in\mathcal{P}_{uv}} r^2_{ui} \sum_{i\in\mathcal{P}_{uv}} r^2_{vi}}} \quad , \tag{1}$$

where $\mathcal{P}_{uv}$ is defined as the set of items which were rated by user u and v, namely, $\mathcal{P}_{uv} = \mathcal{P}_u \cap \mathcal{P}_v$.

2) **neighbor selection** For user-based kNN models, users with the highest similarity with the target user form his(her) neighborhood. Item-based kNN models form the target item's neighborhood similarly. This technique is usually called *best-n-neighbors* [1], [3]. Usually kNN models share the same neighbor selection strategy.
3) **prediction generation** The final step forms the predictive ratings of the target user to all items, and return items with the highest ratings as the recommended items for the target user. One of the popular and effective strategies to obtain the prediction for an item is to use the frequency that the item was purchased by the neighbors of the target user, that is,

$$\hat{r}_{ui} = \sum_{v\in\mathcal{N}_u} \delta_{vi} \quad , \tag{2}$$

where

$$\delta_{vi} = \begin{cases} 1\ , & \text{if } r_{vi} > 0 \\ 0\ , & \text{if } r_{vi} = 0 \end{cases} \tag{3}$$

and $\mathcal{N}_u$ is the neighborhood of user u selected in the previous step.

In many practical situations, it is reasonable to consider the importance of items when similarity is calculated.Therefore we suggest to use the following *weighted cosine* similarity measure:

$$S^{\mathrm{wc}}_{uv} = \frac{\sum_{i\in\mathcal{P}_{uv}} c_i r_{ui}r_{vi}}{\sqrt{\sum_{i\in\mathcal{P}_{uv}} c_i r^2_{ui} \sum_{i\in\mathcal{P}_{uv}} c_i r^2_{vi}}} \quad , \tag{4}$$

where c_i is the weight associated with item i. In our experiments, we take

$$c_i = \log\left(\frac{U}{|\mathcal{P}^i| + 1}\right) \quad , \tag{5}$$

which is motivated by the popular *term frequency-inverse document frequency (tf-idf)* weight in text mining [7]. The choice of c_i in (5) produces larger weights for items which were purchased less frequently. It works well for our supermarket data set.

We also use the shrinkage technique proposed for sparse explicit ratings in [8], since typically a user purchases a small proportion of products in the supermarket data set. Hence finally we calculate the similarity between user u and v as follows:

$$S^{\mathrm{swc}}_{uv} = \frac{|\mathcal{P}_{uv}|}{|\mathcal{P}_{uv}| + \gamma} \cdot S^{\mathrm{wc}}_{uv} \quad , \tag{6}$$

where γ is the shrinkage parameter, which can be determined by cross validation.

The prediction strategy in (2) totally ignores rating values which may brings much information about the user's interest to the products . Here we suggest a new strategy to consider the above factor,

$$\hat{r}_{ui} = \sum_{v\in\mathcal{N}_u} \delta_{vi} \log(r_{vi} + 1) \quad . \tag{7}$$

More details for predictions by using kNN models can be found in section VI.

IV. Matrix Factorization Models

A. Matrix Factorization For Explicit Ratings

Matrix Factorization (MF) models attract much attention recently ,and one typical example is their impressive results for the Netflix Prize problem [9], [10], [11], [12]. MF models assume that each user's preferences can be depicted by a small number of factors, and each item's characteristics can also be expressed by the same number of factors. The user and item factors are achieved by solving the following minimization problem:

$$\min_{\mathbf{X},\mathbf{Y}} \sum_{(u,i)\in\mathcal{P}} \left[(r_{ui} - \mathbf{x}_u^T\mathbf{y}_i)^2 + \lambda(\| \mathbf{x}_u \|_2^2 + \| \mathbf{y}_i \|_2^2)\right] \quad , \quad (8)$$

where $\mathbf{x}_u$ and $\mathbf{y}_i$ are user u's and item i's factor vectors of length K, and λ is the regularization parameter. $\mathbf{X} \in \mathbb{R}^{U\times K}$ is defined as $(\mathbf{x}_1, \ldots, \mathbf{x}_U)^T$, and $\mathbf{Y} \in \mathbb{R}^{I\times K}$ is defined similarly. *Stochastic gradient descent* [9] or *alternate least squares* [13] are usually used to solve the above unconstrained optimization problem.

B. Weighted Matrix Factorization For Implicit Ratings

As we stated in section II-A, MF for explicit ratings (8) only uses given ratings of user-item pairs and ignores all ungiven ratings when it is constructed. Ungiven ratings are replaced with rating 0 in implicit CF problems ,and non-zero ratings deliver more information about users and items than zero ratings do. So models should present different weights for them. We use variable c_{ui} to represent the weight of rating r_{ui}. According to the previous discussion, our final choice for c_{ui} is

$$c_{ui} = 1 + \alpha_1 \log\left(1 + \frac{r_{ui}}{\alpha_2}\right) + \alpha_3\delta_{ui}\log\left(\frac{U}{|\mathcal{P}^i| + 1}\right) \quad , \quad (9)$$

where δ_{ui} is defined in (3), and α_k $(k = 1, 2, 3)$ are the hyper parameters, which will be determined by *Nelder-Mead simplex* method [14] in our subsequent experiments. Note that $c_{ui} = 1$ if $r_{ui} = 0$, which is important to design scalable algorithms for our new models.

We introduce a new variable p_{ui} which is more extended than r_{ui} to represent the preference of user u to item i. Our choice for p_{ui} is:

$$p_{ui} = \beta_1 \log\left(1 + \frac{r_{ui}}{\beta_2}\right) \quad , \quad (10)$$

where β_1 and β_2 are the hyper parameters. In our experiments, we simply take $\beta_1 = \beta_2 = 1$. Note that $p_{ui} = 0$ if $r_{ui} = 0$.

Integrating the comprehensive considerations above, we revise the original MF model in (8) to obtain our new objective function:

$$F(\mathbf{X},\mathbf{Y}) = \sum_{u=1}^{U}\sum_{i=1}^{I}\big[c_{ui}(p_{ui} - \mathbf{x}_u^T\mathbf{y}_i)^2 + \lambda_1 \| \mathbf{x}_u \|_2^2 + \lambda_2 \| \mathbf{y}_i \|_2^2\big], \quad (11)$$

where λ_1 and λ_2 are the regularization parameters. We call the new model the *Weighted Matrix Factorization (WMF)*. The same model framework but with different types of c_{ui} and p_{ui} is also proposed in [2] to analyze implicit information.

Minimizing the objective function of WMF (11) will lead to the computational complexity of at least $O(UI)$, which is unfeasible for large data sets. Fortunately, alternate least squares method can be utilized here to achieve the computational complexity which is linear to $|\mathcal{P}|$ if we notice that $c_{ui} = 1$ and $p_{ui} = 0$ if $r_{ui} = 0$ [2].

Alternate least squares method first fixes item factor matrix $\mathbf{Y}$ and updates user factor matrix $\mathbf{X}$, then fixes $\mathbf{X}$ and updates $\mathbf{Y}$. The whole loop is repeated until the stopping criterion is satisfied. In the following we introduce the update of $\mathbf{X}$ in detail.

When updating the factor vector of user u, we differentiate F with respect to $\mathbf{x}_u$ and set the derivative to 0, and obtain

$$\mathbf{x}_u = (\mathbf{Y}^T\mathbf{C}_u\mathbf{Y} + \lambda_1 I\mathbf{I}_d)^{-1}(\mathbf{Y}^T\mathbf{C}_u\mathbf{p}_u) \quad , \quad (12)$$

where $\mathbf{C}_u = \text{diag}(c_{u1}, \ldots, c_{uI})$, $\mathbf{p}_u = (p_{u1}, \ldots, p_{uI})^T$ and $\mathbf{I}_d$ is the unit matrix. By some calculations, the whole user factor matrix $\mathbf{X}$ can be updated in time $O(K^2|\mathcal{P}| + K^3U)$, which is linear to the size of $\mathcal{P}$.

The update of the item factor matrix $\mathbf{Y}$ can be derived symmetrically. The new $\mathbf{y}_i$ can be obtained as follows:

$$\mathbf{y}_i = (\mathbf{X}^T\mathbf{C}^i\mathbf{X} + \lambda_2 U\mathbf{I}_d)^{-1}(\mathbf{X}^T\mathbf{C}^i\mathbf{p}^i) \quad , \quad (13)$$

where $\mathbf{C}^i = \text{diag}(c_{1i}, \ldots, c_{Ui})$, $\mathbf{p}^i = (p_{1i}, \ldots, p_{Ui})^T$. The update of $\mathbf{Y}$ can be finished in time $O(K^2|\mathcal{P}| + K^3 I)$. Hence the computational complexity of the whole updates in one loop of the alternate least squares method is $O(K^2|\mathcal{P}| + K^3(U+I))$.

After the user and item factor matrices $\mathbf{X}$ and $\mathbf{Y}$ are achieved, WMF first produces the predictive values for all the user-item pairs by

$$\hat{p}_{ui} = \mathbf{x}_u^T\mathbf{y}_i \ , \quad (u = 1, \ldots, U;\ i = 1, \ldots, I) \ , \quad (14)$$

then N items with the highest predictive values associated with a user are recommended to the user.

C. Weighted Matrix Factorization With Biases

MF models usually benefit much from user and item biases in explicit CF problems [9]. Here we add biases to our WMF model, and obtain the new objective function:

$$G(\mathbf{X},\mathbf{Y},\mathbf{b},\mathbf{d}) = \sum_{u=1}^{U}\sum_{i=1}^{I}\big[c_{ui}(p_{ui} - b_u - d_i - \mathbf{x}_u^T\mathbf{y}_i)^2 + \lambda_1 \| \mathbf{x}_u \|_2^2 + \lambda_2 \| \mathbf{y}_i \|_2^2 + \lambda_3 b_u^2 + \lambda_4 d_i^2\big], \quad (15)$$

where $\mathbf{b} = (b_1, \ldots, b_U)^T$, $\mathbf{d} = (d_1, \ldots, d_I)^T$ and λ_j $(j = 1, 2, 3, 4)$ are the regularization hyper parameters. We refer to the new model as *BWMF*.

Similar to WMF, we can obtain the new $\mathbf{x}_u$ by

$$\mathbf{x}_u = (\mathbf{Y}^T\mathbf{C}_u\mathbf{Y} + \lambda_1 I\mathbf{I}_d)^{-1}(\mathbf{Y}^T\mathbf{C}_u(\mathbf{p}_u - b_u\mathbf{e} - \mathbf{d})) \quad , \quad (16)$$

where $\mathbf{e}$ is a vector with all the elements equal to 1. $\mathbf{X}$ can still be updated in time $O(K^2|\mathcal{P}| + K^3U)$.

Symmetrically we can obtain the new $\mathbf{y}_i$ by

$$\mathbf{y}_i = (\mathbf{X}^T\mathbf{C}^i\mathbf{X} + \lambda_2 U\mathbf{I}_d)^{-1}(\mathbf{X}^T\mathbf{C}^i(\mathbf{p}^i - \mathbf{b} - d_i\mathbf{e})) \quad . \tag{17}$$

Similarly we can update $\mathbf{Y}$ in time $O(K^2|\mathcal{P}| + K^3 I)$.

The update of the user bias b_u can be obtained by

$$b_u = \frac{\mathbf{e}^T\mathbf{C}_u(\mathbf{p}_u - \mathbf{d} - \mathbf{Y}\mathbf{x}_u)}{\mathbf{e}^T\mathbf{C}_u\mathbf{e} + \lambda_3 I} \quad . \tag{18}$$

. Hence the complexity of updating all $\mathbf{b}_u$'s is $O(K|\mathcal{P}|)$ when we precompute $\mathbf{e}^T\mathbf{d}$ and $\mathbf{e}^T\mathbf{Y}$.

Symmetrically the update of the item bias d_i can be obtained by

$$d_i = \frac{\mathbf{e}^T\mathbf{C}^i(\mathbf{p}^i - \mathbf{b} - \mathbf{X}\mathbf{y}_i)}{\mathbf{e}^T\mathbf{C}^i\mathbf{e} + \lambda_4 U} \quad . \tag{19}$$

Similarly the complexity of updating all $\mathbf{d}_i$'s is $O(K|\mathcal{P}|)$ when we precompute $\mathbf{e}^T\mathbf{b}$ and $\mathbf{e}^T\mathbf{X}$.

Hence the complexity of all the updates in one loop of BWMF is $O(K^2|\mathcal{P}| + K^3(U + I))$ if we precompute $\mathbf{e}^T\mathbf{b}$, $\mathbf{e}^T\mathbf{d}$, $\mathbf{X}^T\mathbf{e}$, $\mathbf{Y}^T\mathbf{e}$, $\mathbf{X}^T\mathbf{b}$, $\mathbf{Y}^T\mathbf{d}$, $\mathbf{X}^T\mathbf{X}$ and $\mathbf{Y}^T\mathbf{Y}$ before the corresponding updates begin.

After the model parameters are achieved, BWMF first produces the predictive values for all the user-item pairs by

$$\hat{p}_{ui} = b_u + d_i + \mathbf{x}_u^T\mathbf{y}_i \ , \quad (u = 1, \ldots, U;\ i = 1, \ldots, I) \ , \tag{20}$$

then N items with the highest predictive values associated with a user are recommended to the user.

V. A Hybrid model of implicit and feature information

In the following we introduce the method of incorporating user features into BWMF. The incorporation of item features can be obtained symmetrically.

Denote the feature vector of user u by $\mathbf{f}_u = (f_{u1}, \ldots, f_{uL})^T$, where L is the number of user features. $\mathbf{F} \in \mathbb{R}^{U\times L}$ is defined as $(\mathbf{f}_1, \ldots, \mathbf{f}_U)^T$. We expand BWMF to incorporate the user features and obtain the new objective function:

$$\begin{aligned} &H(\mathbf{X}, \mathbf{Y}, \mathbf{W}, \mathbf{b}, \mathbf{d}) \\ &= \sum_{u=1}^{U}\sum_{i=1}^{I}\big[c_{ui}(p_{ui} - b_u - d_i - \mathbf{x}_u^T\mathbf{y}_i - \mathbf{f}_u^T\mathbf{w}_i)^2 \\ &\quad + \lambda_1 \parallel \mathbf{x}_u \parallel_2^2 + \lambda_2 \parallel \mathbf{y}_i \parallel_2^2 + \lambda_3 b_u^2 + \lambda_4 d_i^2 + \lambda_5 \parallel \mathbf{w}_i \parallel_2^2\big], \end{aligned} \tag{21}$$

where $\mathbf{w}_i$ is the regression coefficient vector associated with item i, and λ_j $(j = 1, 2, 3, 4, 5)$ are the regularization hyper parameters. $\mathbf{W} \in \mathbb{R}^{I\times L}$ is defined as $(\mathbf{w}_1, \ldots, \mathbf{w}_I)^T$. We refer to the new model as *HBWMF.*

Analogously alternate least squares method can be used to solve HBWMF efficiently if we use the previous tricks: pre-computation and dividing c_{ui} into two parts. The complexity of all the updates in one loop of HBWMF is $O((K+L)^2|\mathcal{P}| + K^3(U + I) + L^2U + L^3 I)$. The detailed updates of the model parameters are shown in Algorithm 1.

Algorithm 1 (HBWMF): Choose the number of factors K, and the regularization hyper parameters λ_j $(j = 1, 2, 3, 4, 5)$. Initialize the item factor matrix $\mathbf{Y}$, the item coefficient matrix $\mathbf{W}$, and the bias vectors $\mathbf{b}$ and $\mathbf{d}$, e.g. sampling the values from a uniform distribution or simply setting all of them (except $\mathbf{Y}$) to 0. Perform the following steps iteratively:

1) Precompute $\mathbf{e}^T\mathbf{d}$, $\mathbf{Y}^T\mathbf{e}$, $\mathbf{Y}^T\mathbf{d}$, $\mathbf{W}^T\mathbf{e}$, $\mathbf{Y}^T\mathbf{W}$ and $\mathbf{Y}^T\mathbf{Y}$.
2) For user u $(u = 1, \ldots, U)$, do:
   a) update user factor vector $\mathbf{x}_u$ by:

$$\begin{aligned}\mathbf{x}_u = &(\mathbf{Y}^T\mathbf{C}_u\mathbf{Y} + \lambda_1 I\mathbf{I}_d)^{-1} \\ &\cdot (\mathbf{Y}^T\mathbf{C}_u(\mathbf{p}_u - b_u\mathbf{e} - \mathbf{d} - \mathbf{W}\mathbf{f}_u)) \ ;\end{aligned} \tag{22}$$

   b) update user bias b_u by:

$$b_u = \frac{\mathbf{e}^T\mathbf{C}_u(\mathbf{p}_u - \mathbf{d} - \mathbf{Y}\mathbf{x}_u - \mathbf{W}\mathbf{f}_u)}{\mathbf{e}^T\mathbf{C}_u\mathbf{e} + \lambda_3 I} \ . \tag{23}$$

3) Precompute $\mathbf{e}^T\mathbf{b}$, $\mathbf{X}^T\mathbf{e}$, $\mathbf{X}^T\mathbf{b}$, $\mathbf{F}^T\mathbf{e}$, $\mathbf{F}^T\mathbf{b}$, $\mathbf{X}^T\mathbf{F}$, $\mathbf{X}^T\mathbf{X}$ and $\mathbf{F}^T\mathbf{F}$.
4) For item i $(i = 1, \ldots, I)$, do:
   a) update item factor vector $\mathbf{y}_i$ by:

$$\begin{aligned}\mathbf{y}_i = &(\mathbf{X}^T\mathbf{C}^i\mathbf{X} + \lambda_2 U\mathbf{I}_d)^{-1} \\ &\cdot (\mathbf{X}^T\mathbf{C}^i(\mathbf{p}^i - \mathbf{b} - d_i\mathbf{e} - \mathbf{F}\mathbf{w}_i)) \ ;\end{aligned} \tag{24}$$

   b) update item coefficient vector $\mathbf{w}_i$ by:

$$\begin{aligned}\mathbf{w}_i = &(\mathbf{F}^T\mathbf{C}^i\mathbf{F} + \lambda_5 U\mathbf{I}_d)^{-1} \\ &\cdot (\mathbf{F}^T\mathbf{C}^i(\mathbf{p}^i - \mathbf{b} - d_i\mathbf{e} - \mathbf{X}\mathbf{y}_i)) \ ;\end{aligned} \tag{25}$$

   c) update item bias d_i by:

$$d_i = \frac{\mathbf{e}^T\mathbf{C}^i(\mathbf{p}^i - \mathbf{b} - \mathbf{X}\mathbf{y}_i - \mathbf{F}\mathbf{w}_i)}{\mathbf{e}^T\mathbf{C}^i\mathbf{e} + \lambda_4 U} \ . \tag{26}$$

5) Stop if some given stopping criterion is satisfied by using the updated parameters.

After the model parameters are achieved, HBWMF first produces the predictive values for all the user-item pairs by

$$\hat{p}_{ui} = b_u + d_i + \mathbf{x}_u^T\mathbf{y}_i + \mathbf{f}_u^T\mathbf{w}_i, \ (u = 1, \ldots, U;\ i = 1, \ldots, I), \tag{27}$$

then N items with the highest predictive values associated with a user are recommended to the user.

VI. Experimental Study

A. Data Description

All of our experiments are based on a data set from a supermarket. The data set comprises of $13,025$ users' two-month purchase data in the supermarket, which contains $9,850$ different items. Each value (r_{ui}) in the data set represents the number of times a user purchased an item. We remove the data of users who purchased less than 10 different items and users who purchased more than 100 different items. Since users who purchased less than 10 different items provide too little information,and users who purchased more than 100 different items in two months are usually small traders.

After the above cleaning for the data set, the number of users decreases to $8,638$ and the number of user-item pairs with non-zero values decreases from $274,925$ to $244,967$. Then the whole data set is split into training set and test set randomly

TABLE I
AVERAGE RECOMMENDATION PRECISIONS OF THE KNN MODELS IN 5 RANDOM SPLITS OF THE WHOLE DATA SET. DIFFERENT KNN MODELS USE DIFFERENT SIMILARITY MEASURES AND RATING STRATEGIES TO OBTAIN RECOMMENDATIONS. THE NUMBER OF NEIGHBORS IS CHOSEN TO BE 150, WHICH PRODUCES THE BEST RECOMMENDATION ACCURACIES FOR ALL THE KNN MODELS. THE SUPERSCRIPTS "S", "C" AND "W" STAND FOR "SHRUNK", "COSINE" AND "WEIGHTED" RESPECTIVELY. THE SUBSCRIPT "F" INDICATES THE CORRESPONDING SIMILARITY IS CALCULATED BASED ON THE USER FEATURES.

similarity measure	rating strategy	shrinkage parameter	precision
$S_{\mathrm{f}}^{\mathrm{sc}}$	(7)	$\gamma = 2$	0.2158
S^{c}	(2)	-	0.2328
S^{wc}	(2)	-	0.2337
S^{swc}	(7)	$\gamma = 1$	0.2343
$0.35S_{\mathrm{f}}^{\mathrm{sc}} + 0.65S^{\mathrm{swc}}$	(7)	-	0.2376

for five times to eliminate the impact of random split on the final recommendation precisions of models. Without specific declaration, the recommendation precisions of models in this section represent the average precisions of the corresponding models by randomly splitting the whole data set five times. For example, the training set comprises of $131,364$ non-zero values of user-item pairs, and the test set comprises of $130,856$ non-zero values of user-item pairs[1] in our first random split.

Apart from the direct purchase data set above, 12 features for each user are extracted by experts according to the purchase history of the user. Typically one user has one or two non-zero feature values. These user features are used to construct the hybrid model and further improve the recommendation accuracy.

B. Results

1) Results of kNN Models: At first we implement the kNN models with different similarity measures ((1) and (6)) and rating strategies ((2) and (7)). When the conventional cosine measure (1) is used to compute similarity between two users, and the most-frequently-purchased-product strategy (2) is used to produce ratings and then recommendations, kNN acquires the average recommendation precision of 0.2328 in our 5 random splits of the whole data set if the number of neighbors is chosen to be 150. The numbers of neighbors for all kNN models are chosen to obtain the best recommendation accuracies respectively. The average precision increases to 0.2343 with the same number of neighbors, if we use the weighted cosine similarity (6) and the log-rating strategy (7), where the shrinkage hyper parameter $\gamma = 1$. Our new similarity measure and rating strategy improve the recommendation accuracy.

For comparison we also use user features to achieve recommendations. The feature variables are first normalized to the same range $[0, 1]$, and then used to calculate the similarity between different users. We use the cosine measure with the shrinkage technique (and the shrinkage parameter $\gamma = 2$) because of the sparsity of the feature values. The final average recommendation precision is 0.2158 (the number of neighbors is still 150), which is much lower than 0.2343.

[1]Note that $131,364 + 130,856 > 244,967$ because some original values larger than 1 are divided into two positive integer values, which are put into the training and test sets respectively.

TABLE II
VALUES OF THE HYPER PARAMETERS IN DIFFERENT MODELS, WHICH ARE OBTAINED BY USING NELDER-MEAD SIMPLEX METHOD AND THE DATA SETS FROM OUR FIRST RANDOM SPLIT. THE NUMBERS OF THE MODEL FACTORS K ARE ALL FIXED TO 30, AND THE MAXIMUM NUMBERS OF ITERATIONS FOR THE NELDER-MEAD SIMPLEX METHOD ARE ALL SET TO 30 WHEN THEY ARE TRAINED.

model	$\alpha_1, \alpha_2, \alpha_3$	$\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$
HKV	$2.3251, 1.0307, -$	$0.0044, 0.0081, -, -, -$
WMF	$0.1957, 0.9098, 1.4464$	$0.0077, 0.0075, -, -, -$
BWMF	$0.0610, 0.9101, 1.5434$	$0.0084, 0.0074, 0.0060, 0.0124, -$
HBWMF	$0.1247, 0.8602, 1.5719$	$0.0085, 0.0076, 0.0077, 0.0208, 0.0020$

To improve the accuracy of our new kNN model ((4)+(7)), we linearly combine its similarity and the similarity calculated from the features. The combining proportion is determined by *grid search* from 1.0 to 0.0 with step size 0.05 in order to achieve the best recommendation accuracy. Finally we obtain the optimal combining proportion of 0.65, and the average recommendation precision increases from 0.2343 to 0.2376 with the same number of neighbors. We refer to the model as *hybrid kNN*.

The average recommendation precisions of the above kNN models and their parameter settings are also shown in Table I detailedly.

2) Results of Matrix Factorization Models: As we stated before, the same model framework (11) is also proposed in [2], but with the different choices for c_{ui}:

$$c_{ui} = 1 + \alpha_1 \log\left(1 + \frac{r_{ui}}{\alpha_2}\right)$$

where α_1 and α_2 are the hyper parameters, which will be determined by Nelder-Mead simplex method in our experiments, and p_{ui}:

$$p_{ui} = \delta_{ui}$$

We refer to their model as *HKV*, named after the first capitals of the last names of the authors [2].

To compare HKV with our new MF models, we first obtain their hyper parameters by applying Nelder-Mead simplex method to the resulting data sets from our first random split respectively, with the numbers of factors K fixed to 30. The maximum numbers of iterations for the Nelder-Mead simplex method are both limited to 30. Then these obtained hyper parameters are used in all the data sets from the 5 random splits, since it is too expensive computationally to run the Nelder-Mead simplex method in each random split. The resulting values of the hyper parameters for the models are listed in Table II.

The final recommendation results of HKV and WMF are shown in Figure 1 when the number of factors K is 30 and the data sets from the first random split are used. From Figure 1, HKV obtains the highest precision of 0.2352 when the

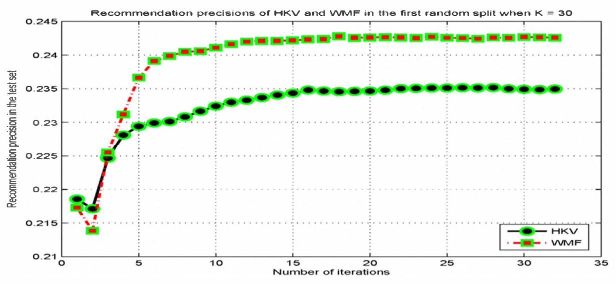

Fig. 1. Recommendation precisions of HKV and WMF when the number of factors K is 30 and the data sets from the first random split are used. Their hyper parameters α_k's and λ_l's, which can be found in Table II, are chosen by using Nelder-Mead simplex method to obtain the best recommendation accuracies respectively. The horizontal axis represents the number of iterations, and the vertical axis represents the recommendation precision within 10 recommended products in the test set.

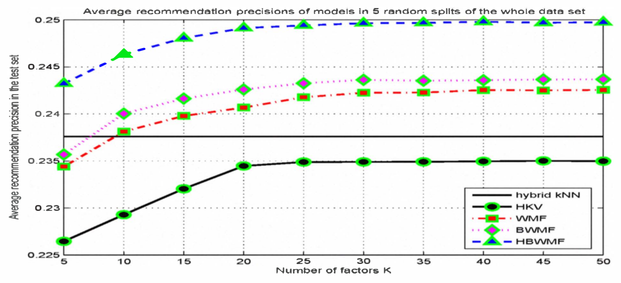

Fig. 2. Average recommendation precisions of various models with different numbers of factors K in 5 random splits of the whole data set. The hyper parameters of each model (except hybrid kNN) are determined by applying Nelder-Mead simplex method to the data sets from the first random split respectively when K is fixed to 30. The obtained hyper parameters are then used for all K's.

number of iterations is 27, and WMF obtains the highest precision of 0.2428 when the number of iterations is 17. WMF produces significantly better recommendation accuracy in the supermarket data set.

Average recommendation precisions of HKV and our matrix factorization (MF) models with different numbers of factors in the 5 random splits are shown in Figure 2. The values of the hyper parameters used in each of the four MF models can be found in Table II. All of our new MF models produce better average recommendation accuracies when $K \geq 10$ than the hybrid kNN model does. But HKV produce worse accuracies than the hybrid kNN model does, even the number of factors K in HKV is taken to 50. BWMF exhibits slight superiority over WMF for all the values of K.But its superiority seems to vanish gradually when K increases. Compared with the other models, HBWMF obtains dramatic improvement from user features. The average recommendation precision increases to 0.2497 from 0.2437 of BWMF when $K = 50$.

VII. Conclusions

In this paper we explore some methods to use implicit information ,and they performs well in our experiments. For future work, we will explore other methods to solve the new MF models, although alternate least squares method can be used to produce efficient algorithms for them. Methods which efficiently solve original MF models for explicit ratings are our first candidates, such as Variational Expectation Maximization (VEM) [15] or Markov Chain Monte Carlo (MCMC) [16]. We will also explore the probabilistic interpretations of the new MF models for implicit information.

On the other side, it is worth exploring more efficient approaches to combine implicit and feature information. One reasonable choice is to use kernel regression instead of linear regression in HBWMF.

Finally, we will also try to consider more practical factors when a practical recommender system is constructed. One reasonable consideration is the repetition of user actions. For example, should we recommend to a user an item which he(she) has already purchased? We may introduce a value for each item to represent the probability that the item will be purchased again. Actually we applied the idea to our new kNN model, and obtained slight improvement of the recommendation accuracy. Another consideration is to produce recommendation lists with different sizes for different users.

References

[1] Y. H. Cho and K. Kim, "Application of web usage mining and product taxonomy to collaborative recommendations in e-commerce," *Expert Systems with Applications*, vol. 26, pp. 233–246, 2004.

[2] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *IEEE International Conference on Data Mining*, Los Alamitos, CA, USA, 2008.

[3] A. Albadvi and M. Shahbazi, "A hybrid recommendation technique based on product category attributes," *Expert Systems with Applications*, 2009.

[4] Netflix, "Netflix prize." [Online]. Available: http://www.netflixprize.com

[5] D. Oard and J. Kim, "Implicit feedback for recommender systems," in *Proceedings of the AAAI Workshop on Recommender Systems*, 1998, pp. 81–83.

[6] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998, pp. 43–52.

[7] Wikipedia, "tf-idf."

[8] R. M. Bell, Y. Koren, and C. Volinsky, "Modeling relationships at multiple scales to improve accuracy of large recommender systems," in *KDD*, P. Berkhin, R. Caruana, and X. Wu, Eds. ACM, 2007, pp. 95–104.

[9] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *KDD-Cup and Workshop*. ACM press, 2007.

[10] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Investigation of various matrix factorization methods for large recommender systems," in *2nd Netflix-KDD Workshop*, Las Vegas, NV, USA, August 2008.

[11] Y. Koren, R. M. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[12] J. Wu, "Binomial matrix factorization for discrete collaborative filtering," in *ICDM*, W. Wang, H. Kargupta, S. Ranka, P. S. Yu, and X. Wu, Eds. IEEE Computer Society, 2009, pp. 1046–1051.

[13] R. Bell and Y. Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," in *Proceedings of IEEE International Conference on Data Mining (ICDM'07)*, 2007.

[14] S. Singer and J. Nelder, "Nelder-mead algorithm," 2009. [Online]. Available: http://www.scholarpedia.org/article/Nelder-Mead_algorithm

[15] Y. J. Lim and Y. W. Teh, "Variational Bayesian approach to movie rating prediction," in *Proceedings of KDD Cup and Workshop*, 2007.

[16] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *ICML*, 2008, pp. 880–887.