Two-Phase Layered Learning Recommendation via Category Structure

Ke Ji[1], Hong Shen[2,3], Hui Tian[4], Yanbo Wu[1], and Jun Wu[1]

[1] School of Computer and Information Tech., Beijing Jiaotong University, China
[2] School of Information Science and Technology, Sun Yat-sen University, China
[3] School of Computer Science, University of Adelaide, Australia
[4] School of Electronics and Info. Engineering, Beijing Jiaotong University, China
{12120425,htian,ybwu,wuj}@bjtu.edu.cn, hongsh01@gmail.com,

Abstract. Context and social network information have been introduced to improve recommendation systems. However, most existing work still models users' rating for every item directly. This approach has two disadvantages: high cost for handling large amount of items and unable to handle the dynamic update of items. Generally, items are classified into many categories. Items in the same category have similar/relevant content, and hence may attract users of the same interest. These characteristics determine that we can utilize the item's content similarity to overcome the difficultiess of large amount and dynamic update of items. In this paper, aiming at fusing the category structure, we propose a novel two-phase layered learning recommendation framework, which is matrix factorization approach and can be seen as a greedy layer-wise training: first learn user's average rating to every category, and then, based on this, learn more accurate estimates of user's rating for individual item with content and social relation ensembled. Based on two kinds of classifications, we design two layered gradient algorithms in our framework. Systematic experiments on real data demonstrate that our algorithms outperform other state-of-the-art methods, especially for recommending new items.

Keywords: Collaborative filtering, Matrix Factorization, Recommender Systems, Layered Learning.

1 Introduction

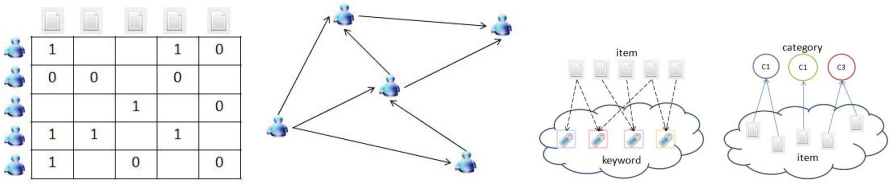
With the rapid development of the Internet, information growth has gone beyond the capacity of our social infrustucture. Recommendation systems that can suggest users with useful information become a powerful way to solve the information overload. A successful technique in recommendation systems is *collaborative filtering* (CF) [1]. It has been applied in many areas, such ase ecommerce (e.g., Amazon) and social networks (e.g., Twitter). Two primary approaches to CF are memory based [2] and model based [3,4] algorithms. The basic difference is that memory based algorithms predict the missing rating based on similar users or items which can be found from the whole user-item rating matrix

V.S. Tseng et al. (Eds.): PAKDD 2014, Part II, LNAI 8444, pp. 13–24, 2014.

(Figure 1(a)) using the similarity measurement (PCC, VSS [5]), whereas model based algorithms explore the training data to train a model, which can make fast prediction using only a few parameters of the model instead of manipulating the whole matrix.

Traditional CF algorithms have several challenges. Due to the sparsity, they cannot make reliable recommendation for lazy users who have rated few items or cold start users who have never rated any items because of insufficient data to capture their tastes accurately. Mining purely the rating matrix may give unrealistic recommendation. In order to solve these problems, lots of studies have been done. Matrix factorization can solve the sparsity problem [4]. Context-aware algorithms [6] that incorporate contextual information have improved the accuracy. With the popularity of online social networks, social recommendation models [7,8,9,10,11] that incorporate social networks information (Figure 1(b)) not only improve the recommendation quality, but also solve the cold start problem.

Even so, there are still some drawbacks. They typically model users' rating for every item. As the number of items increases, the rating matrix becomes very large so that matrix operations in all CF algorithms become exceedingly expensive which may even go beyond the physical computation/storage power. Beside that, attention to an individual item does not reveal users' tastes explicitly, and provides no ability to deal with new items to arrive in the future. There is therefore an urgent need to establish a general system that can provide scalable solutions for both the large amount and dynamic update of data. As nowadays we can easily get a greater variety of data than ever before, information extraction methods that can extract keyword from item content (Figure 1(c)) are widely adopted. There are classification methods that can accurately classify the items into many categories (Figure 1(d)). Intuitively, for items under the same category, their content is relevant, and hence the user's tastes to them may well be similar. This means that we can explore the category structure to find user's similar tastes. Since this information is comparatively static, we can use it to improve the scalability of a recommendation system. However, the current models cannot be adopted to incorporate this information. Therefore, a more flexible recommendation mechanism that can efficiently integrate this information is needed.



(a) rating matrix (b) social network (c) keyword (d) category

Fig. 1. A Toy Example

To address the above problems, we apply a new strategy of *layered learning* toconsider separately different factors in different layers. Motivated by this idea, we propose a two-phase layered learning recommendation framework integrating various information. The main process is defined as: we first learn user's average tastes to every category of items in phase one, then we regard them as baseline estimates and learn more accurate estimates of user's rating for each item with content and social relation ensembled in phase two. We employ matrix factorization to factorize different user preference matrixes: user-category preference matrix and user-keyword preference matrix. According to the two kinds of classification, we design two layered gradient algorithms in our framework, and conduct experiments on real dataset. The experimental result and analysis demonstrate that our framework not only increases the classification accuracy, but also has good performance for dynamic updates of items.

The rest of this paper is organized as follows. In Section 2, we introduce the related work. Our recommendation framework is formulated in Section 3, and experimental results are reported in Section 4. Section 5 is the conclusion.

2 Related Work

2.1 Matrix Factorization(MF)

Matrix factorization is one of the most popular approaches for low-dimensional matrix decomposition. Here, we review the basic MF method [4]. The rating matrix $R \in R^{M \times N}$ (M is the number of users and N is the number of items) can be predicted by UV^T with the user latent factor matrix $U \in R^{D \times M}$ and item latent factor matrix $V \in R^{D \times N}$, where D is the dimension of the vectors. In order to learn the two matrices, the sum-of-squared-error function $\mathcal{L}$ is defined (with Frobenius regularization $\| \cdot \|_F$).

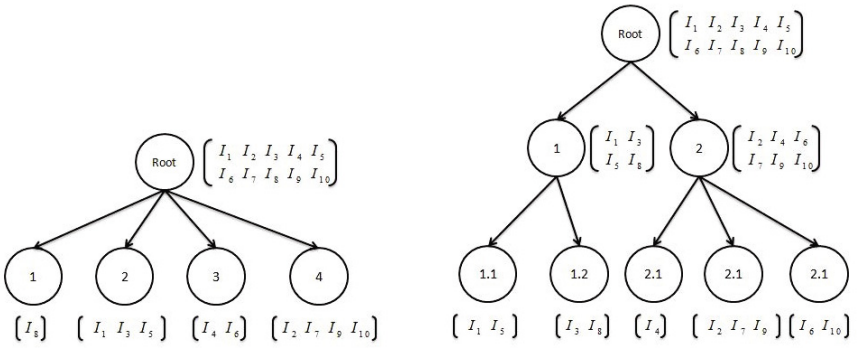
$$\mathcal{L} = \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij} \left(R_{ij} - U_i^T V_j \right)^2 + \lambda_1 \|U\|_F^2 + \lambda_2 \|V\|_F^2 \tag{1}$$

where λ_1 or λ_2 is the extent of regularization and I_{ij} is the indicator function that is equal to 1 if user i rated item j and equal to 0 otherwise. The optimization problem $\arg\min_{U,V} \mathcal{L}$ can be solved using gradient descent method.

2.2 Classification Based on Flat Approache and Top-Down Approache

Classification is an important data analysis method. It can help us better understand data. Classification can be artificial, also can be automatic based on machine learning. According to the division structure, there are two main classification methods: flat approach and top-down approach [12] (Figure 2). Flat approach divides the data into multi-category directly, not considering the hierarchical relation between categories. Top-down approach uses the divide and

conquer technique: classify the current category into some small-scale subcategories, perform the step iteratively until a reasonable classification. In this paper, we introduce the category of items to find the similarity among items.



(a) Flat Approaches (b) Top-down Approaches

Fig. 2. Two kinds of classifications

2.3 Social Recommendation

Traditional recommendation systems assume users are *i.i.d* (independent and identically distributed). In real life, people's decision is often affected by friends' action or recommendation. How to utilize social information has been extensively studied. Trust-aware models [7,9,13] fusing users' social network graph with the rating matrix move an important step forward for recommendation systems. Recently, social-based models make some further improvements. [10] proposed two better methods to leverage the social relation. [8] revealed two important factors: individual preference and interpersonal influence for better utilization of social information. CircleCon [11] used the domain-specific "Trust Circles" to extend the SocialMF [7]. However, all of them give no consideration to item content and the similarity among items. In this paper, we incorporate this information to elaborate recommendation.

3 Layered Learning Frameworks for Recommendation

We introduce the problem description, basic idea and define notations in Section 3.1, and present two layered gradient algorithms in Section 3.2 and 3.3.

3.1 Preliminaries

Because of the weakness of directly modeling every rating mentioned in Section 1, we take advantage of user's tastes to the information of items and indirectly

model the rating. Choosing the appropriate category and keyword from the information, we can present the rating matrix as the combination of the user-category preference matrix and user-keyword preference matrix. The first problem is how to fuse the two matrices into the CF model. We apply the *two-phase layered learning* strategy: **Find user's average tastes to every category** and **Ensemble it with content and social relation**. The second problem is how to deal with different classifications. For the flat approach, we directly learn user's tastes to every category , whereas for the top-down approach, we apply the same layered learning strategy: after learning user's average taste to current category, we learn their tastes to the subcategories.

Suppose that we have M users, N items and K keywords. Every item belongs to one category. For the flat approach, we assume that the values of a category are discrete variables in the range $c = \{1, 2, \ldots, n\}$. For the top-down approach, we assume that the category is expressed hierarchically as a string $c_1.c_2.c_3.c_4$, where the categories are delimited by the character '.', ordered in top-down fashion (i.e., category 'c_1' is a parent category of 'c_2', and category 'c_3' is a parent category of 'c_4', and so on). $c_i = \{1, 2, \ldots, n_i\}$ is the set of discrete values of a category in the i-th layer. The rating matrix is denoted by $R \in R^{M \times N}$. We also have a directed social follow graph $G = (\nu, \varepsilon)$ where ν represents the users and the edge set ε represents the following relationships between users.

3.2 Layered Learning Framework on Flat Approach

For the flat approach, we directly learn user's average tastes to every category.

Phase One: Find User's Average Tastes to Every Category. We associate user i with factor vector $U_i \in R^D$ and category k with factor vector $C_k \in R^D$. R_{ij} can be computed by $\hat{R_{ij}} = U_i^T C_{Ca(j)}$, where $Ca(j)$ is the category that item j belongs to. The sum-of-squared-error function $\mathcal{L}_1$ is defined:

$$\mathcal{L}_1 = \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij} \left(R_{ij} - \hat{R_{ij}} \right)^2 + \lambda_u \|U\|_F^2 + \lambda_c \|C\|_F^2 \tag{2}$$

We perform gradient descent in U_i and C_k (Eq.3 and 4) to minimize $\mathcal{L}_1$.

$$\frac{\partial \mathcal{L}_1}{\partial U_i} = \sum_{j=1}^{N} I_{ij} C_{Ca(j)} \left(\hat{R_{ij}} - R_{ij} \right) + \lambda_u U_i, \frac{\partial \mathcal{L}_1}{\partial C_k} = \sum_{j \in \phi(k)} \sum_{i \in \varphi(j)} \left(\hat{R_{ij}} - R_{ij} \right) + \lambda_c C_k \tag{3}$$

where $\phi(k)$ is the set of the items belong to category k, $\varphi(j)$ is the set of users who have rated item j and λ_u or λ_c is the extent of regularization. After the optimization, we can get the user-category preference matrix $R^c = U^T C$. The matrix *base*, where $base_{ij} = R^c_{iCa(j)} = U_i^T C_{Ca(j)}$ means user i's average taste to item j's category is taken as the initial prediction to R.

Phase Two: Ensemble User's Rating with Content and Social Relation. Although we have user's tastes to every category, user's preference for individual

Algorithm 1. Layered gradient algorithm for flat approach

Require: $0 < \alpha_u, \alpha_c, \alpha_k < 1$, $t = 0$.
Ensure: $\mathcal{L}_1^{(0)}(U_i^{(0)}, C_k^{(0)}) \geq 0$, $\mathcal{L}_2^{(0)}\left(U_i^{(0)}, k_z^{(0)}\right) \geq 0$, $\mathcal{L}_1^{(t+1)} < \mathcal{L}_1^{(t)}$, $\mathcal{L}_2^{(t+1)} < \mathcal{L}_2^{(t)}$.
Phase one:
Initialization:$U_i^{(0)}$, $C_k^{(0)}$
for $t = 1, 2, \cdots$ do
Calculate $\frac{\partial \mathcal{L}_1^{(t-1)}}{\partial U_i}$, $\frac{\partial \mathcal{L}_1^{(t-1)}}{\partial C_k}$
$U_i^{(t)} = U_i^{(t-1)} - \alpha_u \frac{\partial \mathcal{L}_1^{(t-1)}}{\partial U_i}, C_k^{(t)} = C_k^{(t-1)} - \alpha_c \frac{\partial \mathcal{L}_1^{(t-1)}}{\partial C_k}$
end for
Generate the baseline estimate matrix *base* whose elements are $base_{ij} = U_i^T C_{Ca(j)}$
Phase two:
Initialization: $K_z^{(0)}$. Take current value U_i as the initial value: $U_i^{(0)} \leftarrow U_i$
for $t = 1, 2, \cdots$ do
Calculate $\frac{\partial \mathcal{L}_2^{(t-1)}}{\partial U_i}$, $\frac{\partial \mathcal{L}_2^{(t-1)}}{\partial K_z}$
$U_i^{(t)} = U_i^{(t-1)} - \alpha_u \frac{\partial \mathcal{L}_2^{(t-1)}}{\partial U_i}, K_z^{(t)} = K_z^{(t-1)} - \alpha_k \frac{\partial \mathcal{L}_2^{(t-1)}}{\partial K_z}$
end for

item is around the average estimate. For example, a user's taste to one category is 3, but the user's rating for individual item may be some higher 3.3 or some lower 2.9. We introduce user's preference for item's keywords to help optimize the initial estimates. We associate keyword t with factor vector $K_t \in R^D$. The user-keyword preference matrix is denoted by $R^K = U^T K$. $I(j)$ is the set of the keywords extracted from item j. User i's preference for item j's keywords is denoted by $\tilde{R_{ij}} = \sum_{t \in I(j)} R_{it}^K = \sum_{t \in I(j)} U_i^T K_t$. Given the $base_{ij}$, we define the new prediction:$\hat{R_{ij}} = base_{ij} + \tilde{R_{ij}}$. The error function is redefined:

$$\mathcal{L} = \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij} \left(R_{ij} - base_{ij} - \tilde{R_{ij}} \right)^2 + \lambda_u \|U\|_F^2 + \lambda_k \|K\|_F^2 \tag{4}$$

Beside item content, we have social network information. Inspired by SoReg [10], with the same assumption that if user i has a friend f, there is a similarity between their tastes, the regularization term to impose constraints between one user and their friends is formulated as:

$$\lambda_f \sum_{i=1}^{M} \sum_{f \in \mathcal{F}^+(i)}^{N} Sim(i, f) \|U_i - U_f\|_F^2 \tag{5}$$

where $\mathcal{F}^+(i)$ is the set of outlink friends of user i and $Sim(i, f) \in [0, 1]$ is the similarity function. We use PCC to compute this value. We change Eq.4 to $\mathcal{L}_2$:

$$\mathcal{L}_2 = \mathcal{L} + \frac{\lambda_f}{2} \sum_{i=1}^{M} \sum_{f \in \mathcal{F}^+(i)}^{N} Sim(i, f) \|U_i - U_f\|_F^2 \tag{6}$$

We perform gradient descent in U_i and K_z (Eq.7 and 8) to minimize $\mathcal{L}_2$.

$$\frac{\partial \mathcal{L}_2}{\partial U_i} = \sum_{j=1}^{N} I_{ij} \left(\sum_{t \in I(j)} K_t \right) \left(base_{ij} + \tilde{R_{ij}} - R_{ij} \right) + \lambda_f \sum_{f \in \mathcal{F}^+(i)}^{|\mathcal{F}^+(i)|} Sim\left(i, f\right) \left(U_i - U_f\right) + \lambda_f \sum_{g \in \mathcal{F}^-(i)}^{|\mathcal{F}^-(i)|} Sim\left(i, g\right) \left(U_i - U_g\right) + \lambda_u U_i \tag{7}$$

$$\frac{\partial \mathcal{L}_2}{\partial K_z} = \sum_{j \in \psi(z)} \sum_{i \in \varphi(j)} U_i \left(base_{ij} + \tilde{R_{ij}} - R_{ij} \right) + \eta K_z \tag{8}$$

where $\psi\left(z\right)$ is the set of the items that contain the keyword z, $\mathcal{F}^-\left(i\right)$ is the set of inlink friends of user i and $\mid \mathcal{F}^+\left(i\right) \mid / \mid \mathcal{F}^-\left(i\right) \mid$ denote the number of friends in the set $\mathcal{F}^+\left(i\right) / \mathcal{F}^-\left(i\right)$. The whole algorithm is presented in Algorithm 1.

3.3 Layered Learning Framework on Top-Down Approach

In order to adapt our framework to the multi-layer category, we do some adjustments to Algorithm 1. The improved algorithm is shown in Algorithm 2.

Phase One: Find User's Average Tastes to Every Category. Suppose that the category has L layers. $Ca^l\left(j\right)$ is the category that item j belongs to in the l-th layer. We associate user i with latent factor $U_i^l \in R^D$ and category k with latent factor $C_k^l \in R^D$ in the l-th layer. In the $1st$ layer, the method is consistent with phase one of Algorithm 1. In the l-th layer, user's taste to item's category is denoted by $\tilde{R_{ij}^l} = \left(U_i^l\right)^T C_{Ca^l(j)}$. Given user's average taste to the parent category $base_{ij}^{l-1}$ in the $l-1$-th layer, R_{ij} can be predicted by $base_{ij}^{l-1} + \tilde{R_{ij}^l}$. The sum-of-squared-error function $\mathcal{L}_1^l$ given by:

$$\mathcal{L}_1^l = \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij} \left(R_{ij} - base_{ij}^{l-1} - \tilde{R_{ij}^l} \right)^2 + \lambda_u \|U^l\|_F^2 + \lambda_c \|C^l\|_F^2 \tag{9}$$

We perform gradient descent in U_i^l and C_k^l (Eq.10 and 11) to minimize $\mathcal{L}_1^l$ in the l-th layer given by Eq.9.

$$\frac{\partial \mathcal{L}_1^l}{\partial U_i^l} = \sum_{j=1}^{N} I_{ij} C_{Ca^l(j)} \left(base_{ij}^{l-1} + \tilde{R_{ij}^l} - R_{ij} \right) + \lambda_u U_i^l \tag{10}$$

$$\frac{\partial \mathcal{L}_1^l}{\partial C_k^l} = \sum_{j = \in \phi^l(k)} \sum_{i = \in \varphi(t)} U_i^l \left(base_{ij}^{l-1} + \tilde{R_{ij}^l} - R_{ij} \right) + \lambda_c C_k^l \tag{11}$$

where $\phi^l\left(k\right)$ is the set of the items belonging to category j in the l-th layer. Then basic estimate $base_{ij}^l$ for the category in the l-th layer is given by: $base_{ij}^l = base_{ij}^{l-1} + \left(U_i^l\right)^T C_{Ca^l(j)}$. Repeat the operation down the categories until the

Algorithm 2. Layered gradient algorithm for top-down approach

Require: $0 < \alpha_u, \alpha_c, \alpha_k < 1$, $t = 0$, $l = 1$.

Ensure: $\mathcal{L}_1^{(0)}(U_i^{(0)}, C_k^{(0)}) \geq 0$, $\mathcal{L}_2^{(0)}\left(U_i^{(0)}, k_z^{(0)}\right) \geq 0$, $\mathcal{L}_1^{(t+1)} < \mathcal{L}_1^{(t)}$, $\mathcal{L}_2^{(t+1)} < \mathcal{L}_2^{(t)}$.

Phase one:

for $l = 1, 2, \cdots, L$ do

Initialization: $U_i^{l\,(0)} \leftarrow U_i^{l-1}$, $C_k^{l\,(0)}$

for $t = 1, 2, \cdots$ do

Calculate $\frac{\partial \mathcal{L}_1^{l\,(t-1)}}{\partial U_i^l}$, $\frac{\partial \mathcal{L}_1^{l\,(t-1)}}{\partial C_k^l}$

$U_i^{l\,(t)} = U_i^{l\,(t-1)} - \alpha_u \frac{\partial \mathcal{L}_1^{l\,(t-1)}}{\partial U_i^l}$, $C_k^{l\,(t)} = C_k^{l\,(t-1)} - \alpha_k \frac{\partial \mathcal{L}_1^{l\,(t-1)}}{\partial C_k^l}$

end for

Generate the baseline estimate matrix $base^l$ in the l-th layer.

$base_{ij}^l = base_{ij}^{l-1} + \left(U_i^l\right)^T C_{Ca^l(j)}$

end for

Generate the baseline estimate matrix $base$ whose elements are $base_{ij} = base_{ij}^L$

Phase two:

Initialization: $K_z^{(0)}$. Take current value U_i as the initial value: $U_i^{(0)} \leftarrow U_i^L$

The following process is the same as phase two in Algorithm 1

lowest layer. Finally, we can get the more accurate baseline estimate $base_{ij} = base_{ij}^L$ in the L-th layer.

Phase Two: Ensemble User's Rating with Content and Social Relation. Given the baseline estimate $base$, the phase is the same as the phase two in Algorithm 1.

4 Experimental Results

4.1 Datasets and Metrics

In our experiments, we use the real Tencent Weibo[1] data published by KDD Cup 2012[2]. Beside the social network information, it contains much context information such as keyword, category and timestamp. The items have been organized using four-layer categories, such as "1.2.5.8"; each category belongs to another category, and all categories together form a hierarchy. This structure is suitable for our framework. We predict user's action to items, where "1" represents that the user accepts the item, and "0" otherwise.

We extract a small dataset over a period of time randomly. It is much bigger and richer than other datasets used by [7,11]. The statistics of the dataset are summarized in Table 1.

For the flat approach, we only use the categories in the $4th$ layer. The density of the rating matrix is $\frac{379598}{12518 \times 3610} = 0.84\%$. We divide the dataset into three parts: the training set R_{train}., test set R_{test}, and set R_{new} containing all items not in R_{train}.

[1] http://t.qq.com/

[2] http://www.kddcup2012.org/

Table 1. Statistics of dataset extracted

(a) The basic statistics

Description	Number	Description	Number
user	12518	user-item rating	375989
item	3610	item-keyword pair	85107
keyword	1102	Min.Num.of Rating per user	1
Social link	3898	Max.Num.of rating per user	325

(b) Category

category	Number
1st layer	6
2nd layer	23
3rd layer	83
4th layer	258

The evaluation metrics we use in our experiments are two popular error metrics: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). A smaller MAE or RMSE value means higher accuracy.

4.2 Implementation and Comparisons

We compare our algorithms with four state-of-art CF algorithms.

- PMF [4]: It is a Low-rank matrix factorization based on minimizing the sum-of-squared-error. It does not take into account the social information.
- SocialMF [7] is a trust-based model incorporating the mechanism of trust propagation. It can reduce recommendation error for cold start users.
- SoReg [10]: It is a matrix factorization model with social regularization, which treats dissimilar tastes of friends with different social regularization.
- CircleCon [11]: It incorporates the concept of circle-based recommendation, which only considers the trust circle specific to one category.

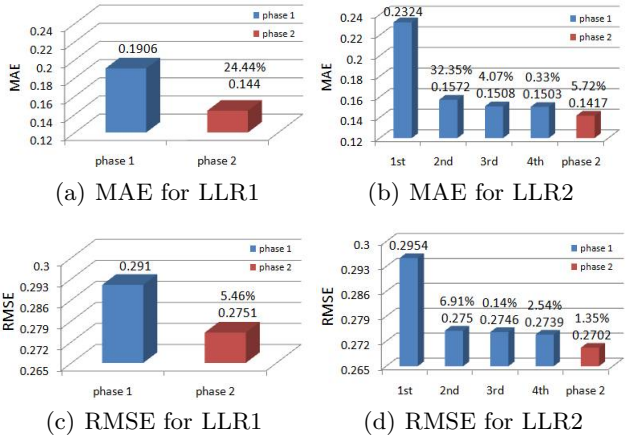
We call our Algorithm 1/Algorithm 2 proposed in section 3 LLR1/LLR2. In all the experiments, the tradeoff parameter settings are $\lambda_u = \lambda_c = \lambda_k = \lambda_f = 0.001$.

JAMA[3] is an open matrix package for Java, developed at NIST and the University of Maryland. It provides the fundamental operations of numerical linear algebra. All algorithms are implemented using this library.

4.3 Impacts of Different Factors

The Number of Layers of Category: The difference between LLR1 and LLR2 is the number of layers of category. The results of LLR1/LLR2 (Figure 3) show in phase one, LLR1 achieves 0.1906/0.2910 on MAE/RMSE, but for LLR2, the training of each layer decreases the values: the training of the $1st$ layer and $2nd$ layer reduce the values greatly, the training of the $3rd$ layer and $4th$ layer have made only minor changes to the results of the $2nd$ layer, and after the training of the $4th$ layer, the values can be reduced to 0.1503/0.2739. Contrasts looked, LLR2 has smaller prediction error than LLR1 in phase one. So our framework benefits more from the top-down approaches than the flat approach. We believe classification based on hierarchy can better model the similarity among items.

[3] `http://math.nist.gov/javanumerics/jama/`



(a) MAE for LLR1 (b) MAE for LLR2

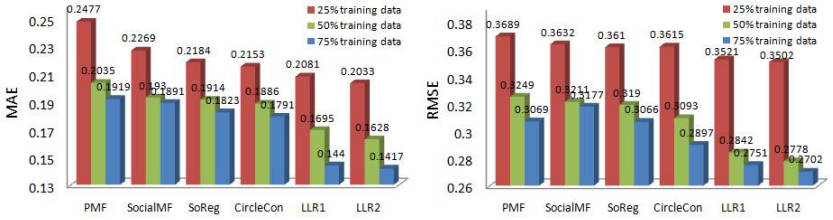
(c) RMSE for LLR1 (d) RMSE for LLR2

Fig. 3. The results of different phases of LLR1 and LLR2 (Dimensionality = 5)

The Item Content and Social Networks Information: After we get the baseline estimate, we discuss how the content and social information may contribute to improving the values. The results of phase two (Figure 3) show we get more accurate estimate of user's rating for individual item: LLR1 and LLR2 achieve 0.1440/0.2751 and 0.1417/0.2702 on MAE/RMSE respectively. For LLR1, this information improves the accuracy as high as 24.44%/5.46% in contrast to phase one. For LLR2, this information improves as high as 5.72%/1.35%. The improvement demonstrates that the content and social information are helpful to boost the performance, especially for LLR1, although classification on the flat approach improves much less than LLR2 based on the top-down approach in phase one, the information significantly enhance more accuracy than LLR2 in phase two. Overall, final results show LLR2 achieves better performance than LLR1.

4.4 Analysis of Recommendation Performance

Figure 4 shows the results of the algorithms on different amounts of training data (25%, 50%, 75%). We observe PMF has the worst MAE/RMSE. SocialMF and SoReg have almost the same accuracy, both superior to PMF, but SoReg is a bit lower because it uses better social regularization terms. CircleCon viewed as an extension of SocialMF is better than the three algorithms. This demonstrates only considering the trust circle belong to one category is useful for learning user's tastes. Our algorithms have the minimum of MAE/RMSE: when the training is 25%, LLR2 gets the decrease by 5.57%/2.99% over CircleCon/SoReg , when the training is 50%, the decrease is 13.68%/10.18% over CircleCon, when the training is 75%, the decrease is 20.88%/6.37% over CircleCon. Experiments demonstrate that our algorithms have higher accuracy than purely using the user-item rating matrix, purely utilizing social networks information or purely considering category-specific circles.



(a) MAE (b) RMSE

Fig. 4. Performance comparison with other algorithms (Dimensionality = 5)

4.5 Performance on Dynamic Update of Items

We analyze the performance of our algorithms on dynamic update of items, i.e., addition of new items. Except some very special items, usually we can know the keywords and category of new items before their addition. The predicted value of $R_{ij_{new}}$ in R_{new} can be computed by $base_{ij_{new}} + \tilde{R}_{ij_{new}}$. Figure 5 shows the results of our algorithms on training data (25%, 50%, 75%). We observe although the new items are not in the rating matrix, our algorithms still make very good prediction using the new item's category and keywords.

(a) MAE (b) RMSE

Fig. 5. The results on addition of new items (Dimensionality = 5)

5 Conclusion

In this paper, based on the similarity in the classification, we proposed a novel two-phase layered learning framework, which incorporates the category, item content and social networks information. For two kind of classifications, we designed two layered gradient algorithms in our framework. We conducted extensive experiments on real data. Comparison results of different phases of LLR1 and LLR2 show that the top-down approaches are more helpful to find user's similar tastes than the flat approach, and item content and social networks information contribute to improve the classification accuracy. The analysis results show that our algorithms outperform other state-of-the-art methods. The results also show that our algorithms has good scalability for the dynamic update of items to cope with addition of new items.

Acknowledgement. This work is supported by National Science Foundation of China under its General Projects funding # 61170232 and # 61100218, Fundamental Research Funds for the Central Universities # 2012JBZ017, Research Initiative Grant of Sun Yat-Sen University (Project 985), State Key Laboratory of Rail Traffic Control and Safety Research Grant # RCS2012ZT011. The corresponding author is Hong Shen.

References

1. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Adv. in Artif. Intell. 2009, 4:2–4:2 (2009)
2. Wang, J., de Vries, A.P., Reinders, M.J.T.: Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2006, pp. 501–508. ACM, New York (2006)
3. Hofmann, T.: Latent semantic models for collaborative filtering. ACM Trans. Inf. Syst. 22(1), 89–115 (2004)
4. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: Platt, J., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems 20, pp. 1257–1264. MIT Press, Cambridge (2008)
5. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial intelligence, UAI 1998, pp. 43–52. Morgan Kaufmann Publishers Inc., San Francisco (1998)
6. Baltrunas, L., Ludwig, B., Ricci, F.: Matrix factorization techniques for context aware recommendation. In: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys 2011, pp. 301–304. ACM, New York (2011)
7. Jamali, M., Ester, M.: A matrix factorization technique with trust propagation for recommendation in social networks. In: Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys 2010, pp. 135–142. ACM, New York (2010)
8. Jiang, M., Cui, P., Liu, R., Yang, Q., Wang, F., Zhu, W., Yang, S.: Social contextual recommendation. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM 2012, pp. 45–54. ACM, New York (2012)
9. Ma, H., King, I., Lyu, M.R.: Learning to recommend with social trust ensemble. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, pp. 203–210. ACM, New York (2009)
10. Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.: Recommender systems with social regularization. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM 2011, pp. 287–296. ACM, New York (2011)
11. Yang, X., Steck, H., Liu, Y.: Circle-based recommendation in online social networks. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2012, pp. 1267–1275. ACM, New York (2012)
12. Silla Jr., C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Min. Knowl. Discov. 22(1-2), 31–72 (2011)
13. Bedi, P., Kaur, H., Marwaha, S.: Trust based recommender system for semantic web. In: IJCAI, pp. 2677–2682 (2007)