This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Dynamic Personalized Recommendation on Sparse Data

Xiangyu Tang and Jie Zhou, *Senior Member, IEEE*

Abstract—Recommendation techniques are very important in the fields of E-commerce and other Web-based services. One of the main difficulties is dynamically providing high-quality recommendation on sparse data. In this paper, a novel dynamic personalized recommendation algorithm is proposed, in which information contained in both ratings and profile contents are utilized by exploring latent relations between ratings, a set of dynamic features are designed to describe user preferences in multiple phases, and finally a recommendation is made by adaptively weighting the features. Experimental results on public datasets show that the proposed algorithm has satisfying performance.

Index Terms—dynamic recommendation, dynamic features, multiple phases of interest.

1 INTRODUCTION

Nowadays the internet has become an indispensable part of our lives, and it provides a platform for enterprises to deliver information about products and services to the customers conveniently. As the amount of this kind of information is increasing rapidly, one great challenge is ensuring that proper content can be delivered quickly to the appropriate customers. Personalized recommendation is a desirable way to improve customer satisfaction and retention [1], [2].

There are mainly three approaches to recommendation engines based on different data analysis methods, i.e., rule-based, content-based and collaborative filtering [3], [4]. Among them, collaborative filtering (CF) requires only data about past user behavior like ratings, and its two main approaches are the neighborhood methods and latent factor models. The neighborhood methods can be user-oriented or item-oriented. They try to find like-minded users or similar items on the basis of co-ratings, and predict based on ratings of the nearest neighbors [5], [6], [7]. Latent factor models try to learn latent factors from the pattern of ratings using techniques like matrix factorization [8] and use the factors to compute the usefulness of items to users. CF has made great success and been proved to perform well in scenarios where user preferences are relatively static.

In most dynamic scenarios, there are mainly two issues that prevent accurate prediction of ratings – the sparsity [3] and the dynamic nature. Since a user could only rate a very small proportion of all items, the $U \times I$ rating matrix is quite sparse and the amount of information for estimating a candidate rating is far from enough. While latent factor models involve most ratings to capture the general taste of users, they still have difficulties in catching up with the drifting signal in dynamic recommendation because of sparsity, and it is hard to physically explain the reason of the involving. The dynamic nature decides that users' preferences may drift over time in dynamic recommendation, resulting in different taste to the items in different phases of interest, but it is not well studied in previous studies [9]. In our experiences, the interest cycle differs from user to user, and the pattern how user preferences changes cannot be precisely described by several simple decay functions. Moreover, CF approaches usually accounter the cold-start problem which is amplified in the dynamic scenario since the rate of new users and new items would be high.

Some researchers have previously attempted to solve the above problems. Hybrid approaches which combine content-based and collaborative filtering in different ways were proposed to alleviate the sparsity problem [3], [10], where more information were mined than just in each of them. Prassas et al. [11] classified items into many categories using content information and chose recent categories to perform Item-Based Collaborative Filtering (IBCF). Kim and Li [10] introduced group similarity by clustering and used it to modify original item-item similarity matrix. The principle of utilization of rating data in these algorithms is shown in Fig. 1.(a). Some approaches emphasize utilization of time information to deal with the dynamic nature. Koren [4] proposed to model temporal dynamics to separate transient factors from lasting ones. Rendle et al. [12] brought matrix factorization and Markov chains together to form a factorized personalized MC model. Xia and Jiang [13] proposed a dynamic IBCF with users' implicit feedback by using time decay functions in the calculation of the similarities.

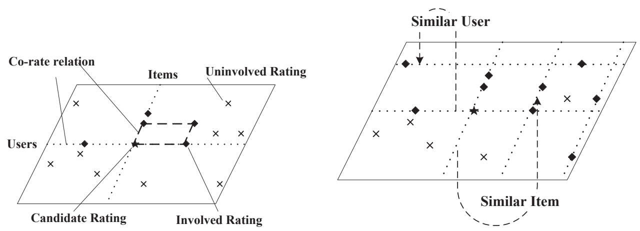
In this paper, we present a novel hybrid dynamic recommendation approach. Firstly, in order to utilize more information while keeping data consistency, we use user profile and item content to extend the co-rate relation between ratings through each attribute, as shown in Fig. 1.(b). The involved ratings can reflect similar users' preferences and provide useful information for recommendation. Correspondingly, in order to enable the algorithm to catch up with the changing of signals quickly and to be updated conveniently, a set of dynamic features are proposed based on time series analysis (TSA) technique, and relevant ratings in each phase of interest are added up

• *Xiangyu Tang and Jie Zhou are with the Department of Automation, Tsinghua University, Beijing, China, 100084.*
E-mail: tangxy03@mails.thu.edu.cn, jzhou@tsinghua.edu.cn

Digital Object Indentifier 10.1109/TKDE.2012.229

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

by applying TSA to describe users' preferences and items' reputations. Then we propose a personalized recommendation algorithm by adaptively weighting the features according to the amount of utilized rating data. The experimental results show that the proposed algorithm is effective with dynamic data and significantly outperforms previous algorithms.



(a) Common Hybrid filtering[10] (b) The proposed approach

Fig. 1. Ratings associated in different methods, where ★, ⧫ and $\times$ represent destination rating, involved rating and uninvolved rating, respectively. In the $U \times I$ plane, ratings along a horizontal line are from the same user and ratings along a vertical line are of the same item. "Similar" here means "identical or close in some attribute of the profiles".

The main contributions of this paper can be summarized as follows: (a) More information can be used for recommender systems by investigating the similar relation among related user profile and item content. Compared with the previous works such as [4], [12], [10], we utilize the similarity among content in each profile attribute so that more content information is used, especially content in those attributes which are hard to be quantified. (b) A novel set of dynamic features is proposed to describe users' preferences, which is more flexible and convenient to model the impacts of preferences in different phases of interest compared with dynamic methods used in previous works, since the features are designed according to periodic characteristics of users' interest and a linear model of the features can catch up with changes in user preferences. (c) An adaptive weighting algorithm is designed to combine the dynamic features for personalized recommendation, in which time and data density factors are considered to adapt with dynamic recommendation on sparse data.

2 The proposed method

In most cases, the drifting of users' preferences or items' reputations is not too rapid, which makes it possible to describe temporal state of them by using some features. In this section, firstly we introduce a way to make use of profiles to extend the co-rating relation, and then we propose a set of dynamic features to reflect users' preferences or items' reputations in multiple phases of interest, and after that we propose an adaptive algorithm for dynamic personalized recommendation.

2.1 Relation mining of rating data

For the sparsity of recommendation data, the main difficulty of capturing users' dynamic preferences is the lack of useful information, which may come from three sources - user profiles, item profiles and historical rating records. Traditional algorithms heavily rely on the co-rate relation (to the same item by different users or to different items by the same user), which is rare when the data is sparse. Useful ratings are discovered using the co-rate relation, which is simple, intuitional and physically significant when we go one or two steps along, but it strongly limits the amount of data used in each prediction.

Instead of searching neighboring nodes along co-rate edges in the $U \times I$ plane, we try to find a different way to find useful ratings. We notice that when considering the factors which affect a rating $r(u,i)$, we may focus more on some attributes of u and i in their profiles, instead of the user himself or the item itself. For example, if the movie "Gone with the Wind" is given high ratings by middle-aged people and lower ratings by teenagers with no doubt, we would primarily check on the age attribute in a user's profile when predicting probable rating the user would give to the movie, instead of other descriptions of the user or how the user has rated other movies. As is evident, it may not be necessary to stick only to the co-rate relation, and we introduce the *semi-co-rate* relation between ratings whose corresponding user profiles or item contents have similar or identical content in one or more attributes. Since *semi-co-rate* is much less constrained, we extend the co-rate relation to it using user profile and item content, and propose a new way of finding useful ratings for dynamic personalized recommendation.

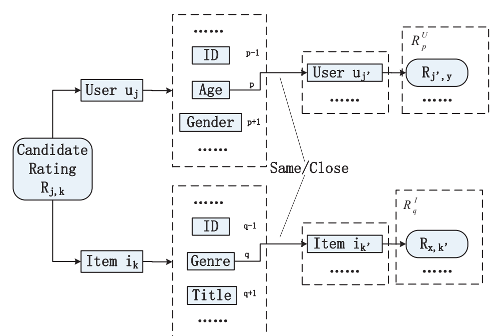


Fig. 2. Finding neighboring ratings in the new relation

Let $\mathcal{U} = \{u_j\}_{j=1}^m$ be the entire user set with $|\mathcal{U}| = m$, $\mathcal{I} = \{i_k\}_{k=1}^n$ be the entire item set with $|\mathcal{I}| = n$, R be a $m \times n$ matrix such that its element $R_{j,k}$ refers to the rating user u_j gave to item i_k, and T be the corresponding time matrix such that $T_{j,k}$ denotes the timestamp of $R_{j,k}$. We note the set whose ratings is *semi-co-rate* related with the candidate rating via the p-th attribute in user profile as R_p^U, and similarly we define R_q^I, as shown in Fig. 2. If we note the set whose rating is co-rate related with the candidate rating via user as R_0^U and similarly we define R_0^I, we have $R_0^U = (\bigcap_p R_p^U)$ and $R_0^I = (\bigcap_q R_q^I)$. Clearly the *semi-co-rate* is much looser than the co-rate relation, and now that we have found much more related ratings via the relation instead of co-rate, we take only one step neighboring nodes, $(\bigcup_p R_p^U) \bigcup (\bigcup_q R_q^I)$ in this newly defined graph to keep consistency of utilized data.

To avoid overwhelming computation in finding R_p^U $(p = 1, 2, ...)$ and R_q^I $(q = 1, 2, ...)$ for all candidate ratings and to solve the difficulty in quantification of some contents,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

classification and clustering are performed as the content of each attribute in profiles of users and items, and R is then separated into rating subsets $R_p^{U,c}$ $(c = 1, 2, ...)$ or $R_q^{I,c}$ $(c = 1, 2, ...)$ accordingly in actual implementation and c is the class number. For example, suppose "Age" is the p-th attribute in user description, then R could be divided into six disjoint subsets for this attribute. By using clustering techniques like K-Means [14] on content of "Age", we divide user age values into several disjoint ranges based on relevant users' ages of all ratings in R. Then for candidate rating $R_{j,k}$, we could conveniently find its neighboring ratings in our algorithm by reaching relevant subsets, which is done by matching each attribute content of u_j and i_k to the nearest subsets. These subsets are representative and can be directly used to extract the dynamic features. We limit the size of each separated subset in online calculation, and the earliest ratings would be removed from the rating subset when new ratings are added in. Through these techniques, we have introduced a more general relation between ratings and an extended way of information mining in personalized recommendation.

2.2 Dynamic feature extraction

Users' preferences or items' reputations are drifting, thus we have to deal with the dynamic nature of data to enhance the precision of recommendation algorithms, and recent ratings and remote ratings should have different weights in the prediction. Three kinds of methods were proposed in concept drift [15] to deal with the drifting problem as *instance selection*, *time-window* (usually time decay function) and *ensemble learning*. Koren [4] also proposed an algorithm to isolate transient noise in data using temporal dynamics to help recommendation. These methods help to make progress in precision of dynamic recommendation, but they also have their weaknesses: decay functions cannot precisely describe the evolution of user preferences and only isolating transient noise cannot catch up with the change in data.

So we propose a set of dynamic features to describe users' multi-phase preferences in consideration of computation, flexibility and accuracy. It is impossible to learn weights of all ratings for each user, but it is possible to learn the general weights of ratings in the user's different phases of interest if the phases include ranges of time that are long enough. For convenience of notation, we relabel all subsets $R_p^{U,c}$ and $R_q^{I,c}$ acquired through the extended information mining as R_s $(s = 1, 2, ...)$.

To enable the features to describe users' preferences in multiple phases of interest, we divide each rating subset R_s into several disjoint secondary subsets R_s^d $(d = 1, 2, ...)$ using the time distances between each rating in R_s and the candidate rating $R_{j,k}$, where each secondary subset is manually assigned with a range of time-distance (corresponding to multiple phases of interest), and then we calculate the features on each secondary subsets using some basic algorithms such as *time series analysis* (TSA).

Since each secondary subset is naturally an array of ratings arranged by time order, and TSA technique [16] is a most widely used and effective method dealing with such data, we choose TSA as the basic feature extraction method. In fact, methods for concept drift [15] and Koren's method [4] are also variants of TSA algorithms in the angle of prediction. More importantly, since the results of TSA are generally representative and predictive of the utilized data in relevant time ranges, we could conveniently use and update the results as features and "expectations" of certain phases of interest for further analysis.

In the theory of time series analysis, earlier ratings should impact the predictive features less, and thus they should have lower weights. So if we perform TSA algorithm on a secondary subset of R (i.e. R_s^d) to get a feature $fea_{s,d}$, there would be an uniform formulation as:

$$fea_{s,d} = \sum_{l=1}^{o} \frac{w_l}{w} R_{s,l}^d, \tag{1}$$

where $\#R_s^d = o$, $R_{s,l}^d (l = 1, 2, ..., o)$ are the rating values which are from the subset R_s^d and listed in reversed time order. And positive weight parameters w_l, $(l = 1, 2, ..., o)$ and normalization factor w should satisfy

$$\begin{cases} w = \sum_{l=1}^{o} w_l, \\ w_{l_1} \geq w_{l_2} \; if \; l_1 < l_2. \end{cases} \tag{2}$$

Since the subsets are updated frequently, index smoothing [16], which is a classic TSA algorithms, is chosen as the basic TSA algorithm:

$$\begin{cases} R_s^d = \{R_{j',k'} | R_{j',k'} \in R_s \; and \; T_{j,k} - T_{j',k'} \geq T_d\}, \\ fea_{s,d} = \sum_{l=1}^{o} \mu(1-\mu)^{l-1} R_{s,l}^d, \end{cases} \tag{3}$$

where R_s^d $(d = 1, 2, ...)$ are the secondary subsets, T_d $(d = 1, 2, ...)$ are a sequence of time differences manually set, $R_{s,l}^d$ $(l = 1, 2, ..., o)$ are the rating values listed in reversed order in the subset, μ is the forgetting factor for index smoothing. We have tested different values for mu in the experiments and set $\mu = 0.95$ empirically.

All $fea_{s,d}$ $(d = 1, 2, ...)$ and the sizes of R_s^d $(d = 1, 2, ...)$ are recorded as dynamic features. With the dynamic features, we only have to optimize their weights to get the best estimation of the candidate rating, and in this way we have transformed the training of a recommendation model into weight learning across different secondary rating subsets. Now that the features are related to phases of interest and latent relations between ratings, we would see how the preferences differ with each other in impacting the candidate rating by analyzing optimal weights of the features. We can also see in Eq(3) that the feature extraction does not need heavy computation. Finding all R_s^d needs only comparison in time one by one, and the computation of $fea_{s,d}$ is very efficient. In this way we have proposed a flexible way of feature extraction, where weights in TSA can be different for different rating subsets and the weights for different phases of interest can be variable and learned from the data. The proposed algorithm is termed as *Multiple Phase Division* (MPD).

2.3 Adaptive weighting algorithm

As features like $fea_{s,d}$ ($s = 1, 2, ..., \ d = 1, 2, ...$) gained by applying *Multiple Phase Division* are all normalized rating values, in other words, as content of user and item profiles have been quantified in the feature extraction, it is convenient for us to organize them for accurate rating estimation by adaptive weighting. Sizes of the relevant subsets are also recorded in MPD and could reflect data density.

We incorporate these features for recommendation with a linear model since they are homogeneous and it is efficient to learn their weights. $\hat{R}_{j,k}$ is used to note the estimated rating that user u_j could give to item i_k at time point $T_{j,k}$, and the adaptive linear model can be formulated as:

$$\hat{R}_{j,k} = \sum_s \sum_d (\alpha_{s,d} + \beta(\#R_s^d)) b_{u_j}(s) b_{i_k}(s) fea_{s,d}, \quad \text{with} : \alpha_{s,d} \geq 0, \ \beta \geq 0, \tag{4}$$

where sizes of relevant subsets are used as prior information in weighting the features to improve recommendation accuracy, $fea_{s,d}$ ($s = 1, 2, ..., \ d = 1, 2, ...$) are the features calculated in Eq.(3), R_s^d ($s = 1, 2, ..., d = 1, 2, ...$) denote their relevant secondary rating subsets, b_{u_j} and b_{i_k} are binary functions denoting the relating state of candidate rating and relevant subset and $\alpha_{s,d}$ and β are weighting parameters which should balance the weights of features and data density, or, balance the affection of data consistency and quantity of information. In detail, $b_{u_j}(s) = 1$ if $R_{j,k}$ is *semi-co-rate* related with all ratings in secondary subset R_s through attribute of the user u_j denoted by s, else $b_{u_j}(s) = 0$, $b_{i_k}(s) = 1$ if $R_{j,k}$ is *semi-co-rate* related with all ratings in secondary subset R_s through attribute of the item i_k also denoted by s, else $b_{i_k}(s) = 0$.

It is difficult to solve all parameters in Eq.(4) at once, hence we use sequential optimization. Let

$$\delta_{s,d} = \alpha_{s,d} + \beta(\#R_s^d), \tag{5}$$

in Eq.(4) and we first solve for the combined weights $\delta_{s,d}$ ($s = 1, 2, ..., \ d = 1, 2, ...$) by minimizing the differences between prediction results of the recommendation algorithm and the real rating values in the training set, where RLS algorithm [17] could be used for optimization, i.e.,

$$E = \sum_{R_{j,k} \in R_{Train}} (\hat{R}_{j,k} - R_{j,k}))^2, \tag{6}$$

where R_{Train} is the training set or known rating set. But we notice that a user's preferences or an item's reputations are commonly affected by only a few principle factors, indicating that using more features might also bring noise into the recommendation. So we changed the destination of the optimization and limited the quantity of the features by regularization, and the training problem can be formulated as:

$$\begin{aligned} \min_{\delta} : & \sum_{R_{j,k} \in R_{Train}} (\hat{R}_{j,k} - R_{j,k}))^2 + \lambda ||\delta||_1, \\ \text{with} : & \ 0 \leq \delta_{s,d} \leq 1 \ and \ \sum_s \sum_d \delta_{s,d} = 1. \end{aligned} \tag{7}$$

This is a typical LASSO optimization problem which can be solved via ADMM [18].

Provided the δs are solved, we turn to the second step of the sequential optimization: to solve αs and β. To deal with the uncertainty in solving αs and β from Eq.(5), we introduce the generalization error like in SVM [19]. Here the generalization error is $\max(\sum_s \sum_d \alpha_{s,d}^2, \ \sum_s \sum_d \beta^2(\#R_s^d)^2)$, and we minimize it to gain satisfying performance as:

$$\begin{aligned} & \min_{\alpha,\beta} : \max(\sum_s \sum_d \alpha_{s,d}^2, \ \sum_s \sum_d \beta^2(\#R_s^d)^2), \\ & \text{with} : \forall s, d, \ \alpha_{s,d} + \beta(\#R_s^d) = \delta_{s,d}, \ \alpha_{s,d} \geq 0, \ \beta \geq 0. \end{aligned} \tag{8}$$

This optimization problem has explicit solution as:

$$\begin{cases} \beta = \dfrac{\sum_s \sum_d \delta_{s,d}}{2 \sum_s \sum_d (\#R_s^d)}, \\ \alpha_{s,d} = \delta_{s,d} - \beta(\#R_s^d) \ for \ all \ s, d. \end{cases} \tag{9}$$

Now we have a practical way of solving all the parameters. Firstly we solve δs from Eq.(7) using Lasso algorithm, then use Eq.(8) and Eq.(9) to compute αs and β.

3 Experiments

3.1 Datasets

MovieLens 100k data[1] and Netflix Competition data[2] are two datasets in studying personalized recommendation which were collected from online movie recommender services [1], [8]. These two datasets contain abundant rating records which last in a reasonable time, and they are different in composition and dynamic nature of data. We use them for our case study. Time distances between the target rating and historical ratings are defined as time of interest, and we manually assigned 6 time intervals to classify times of interest into multiple phases, i.e., within 1 day, 1 to 7 days, 1 to 4 weeks, 1 to 3 months, 3 to 12 months and more than a year.

3.2 Evaluation

The frequently used accuracy indicator for predictive algorithms, Root-mean-square error (RMSE), is used to evaluate the proposed recommendation algorithm. In previous studies [1], [4], [12], the training and testing data are randomly chosen for the experiments. But this is unsuitable for the evaluation of dynamic recommendation. With respect to general causality, it is a critical fact in dynamic recommendation that we can use only historical data but not future data for current prediction in real applications. Unfortunately, the fact is often ignored in previous studies. In traditional RMSE evaluations (even for the Netflix competition), training and testing data are randomly sampled and the train and test split is not based on time. This would produce current prediction based on future data. Even if it is guaranteed that testing instances of each user/item come later than its training instances, the aforementioned issue still exists in algorithms like *IBCF* and *latent factor models* due to the utilization of other users' future ratings.

Replay-match evaluation has been proposed to address this issue by Li et al. [20], whose evaluation is like replaying a

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

match from the beginning to the end. Its evaluation results are more stable for dynamic recommendation compared with results of traditional evaluation. Accordingly, so as to provide a better simulation of practical recommendation systems' working in evaluation, we split training and testing data based on time in the same way and evaluate the accuracies of dynamic recommendation algorithms as follows:

1) Sort the entire dataset in normal time order, use a certain training ratio to determine its splitting.
2) Use the earlier part as the training set to adjust all parameters in the recommendation algorithm.
3) Run algorithm on testing set, generate estimated rating for each user-item pair in testing set.
4) Compare estimated ratings and real ratings in the testing set, and calculate RMSE.
5) Use different ratios and repeat last four steps.

3.3 Experiment setup

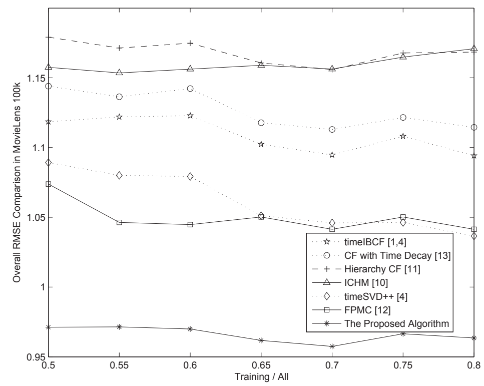
We compared the proposed algorithm with some representative and widely-used dynamic recommendation algorithms. In the comparisons, all the competing algorithms were in online-updating forms and their parameters were set to their empirically best. Here we briefly introduce them.

TimeSVD++ [4] is extended from SVD++ [21] by accounting for temporal dynamics. TimeIBCF is extended from IBCF by accounting for temporal dynamics as in [4]. Factorized Personalized Markov Chain (FPMC) [12] combines *matrix factorization* and *markov chain* together to handle both the sparsity and the dynamic nature of dynamic personalized recommendation. IBCF with time decay [13] weights the similarities of CF by time decay functions to deal with the dynamic problems. Hierarchy CF [11] is a hybrid of content-based and collaborative filtering methods using category information in items' content. ICHM [10] introduces group similarity by clustering to modify similarities in IBCF, and K-Means is adopted for the clustering of item content.

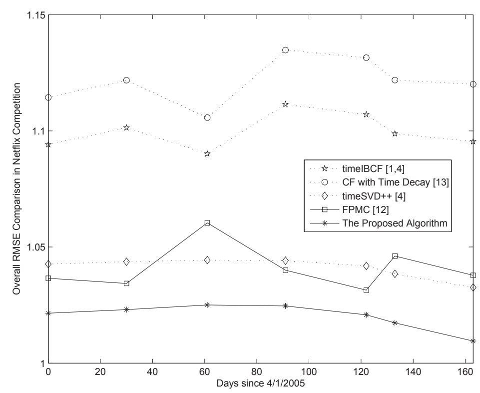
In the experiments, the MovieLens 100k dataset was split as described above with different training ratios ($\frac{\#Trainingset}{\#Dataset}$) – from 50% to 80% stepped by %5, and Netflix Competition was split with different time points – from 4/1/2005 to 10/1/2005 stepped by one month for the amount of its ratings. The former parts of the sorted-by-time datasets are used as the training sets while the latter parts are used as relevant testing sets. We ran almost all algorithms on MovieLens 100k and Netflix Competition for comparison, but we did not perform Hierarchy CF and ICHM on Netflix Competition because these two algorithms require fertile and well-quantified content in item profiles, which is not satisfied by Netflix Competition.

3.4 Comparison results

In Fig. 3 the performances of the algorithms on MovieLens 100k and Netflix Competition are reported. The proposed algorithm significantly outperforms the other algorithms in accuracy on MovieLens 100k and has comparable performance with timeSVD++ on Netflix Competition. On MovieLens 100k, the RMSE of the proposed algorithm is consistently the lowest, and the average RMSE of the proposed method is about 10%-15% lower than other algorithms. On Netflix Competition, the accuracy of the proposed algorithm is comparable with timeSVD++, due to the lack of user profiles and lack of information in item profiles. The experimental results also show that the proposed algorithm and timeIBCF are robust with time evolving, indicating that the proportion of ratings by new users or of new items does not change a lot.



(a)



(b)

Fig. 3. Accuracy comparison (a)MovieLens 100k and (b)Netflix Competition

Comparing Fig. 3(a) and Fig. 3(b), we can infer that (i) the accuracies of algorithms would be enhanced when data, especially recent data, gets dense, and (ii) the utilization of profile content in the proposed algorithm is effective and helps improve the quality of recommendation. Compared to timeSVD++ and timeIBCF, in which only rating information is utilized, hybrid approaches make use of more information and may achieve better recommendation accuracies if the information mined is sufficient and the dynamic nature of data is well handled. The experimental results also show that users' preferences could be well described and learned by the MPD-based features.

As the efficiency of common IBCF is high in all recommendation algorithms [1], [3], we listed the computational time cost of the proposed algorithm and IBCF on the dataset of MovieLens 100k in Table 1 to illustrate its efficiency. All experiments are performed on a workstation with an Intel

Core2 Q8300 CPU and 8GB RAM. We can see that the proposed approach has a comparable computational cost with IBCF. As the size of the testing set increases, the difference of time consumed between two algorithms decreases. indicating that the proposed approach has satisfying performance and can handle larger datasets.

TABLE 1
Computational cost of the proposed algorithm and IBCF [1] on MovieLens 100k

#Test Set	Common IBCF [1]	Proposed Approach
50000	1.8 s	1.8 s
40000	1.5 s	1.5 s
30000	1.2 s	1.3 s
20000	0.8 s	1.0 s

We also conducted experiments to test the robustness of the proposed algorithm on different phases of historical data. We applied the proposed algorithm on the data in each single phase defined before, and the RMSEs are calculated separately according to the definition of users' multiple phases of interest. In Fig. 4, we presented the RMSEs of the proposed recommendation algorithms using the data in different phases of interest at different training ratios. It is clear that the proposed algorithm is quite robust in the phases, and we found it is not true that the more recent ratings should have heavier weights across the whole time, which illustrates the advantages of the features – light computation, flexibility and high accuracy.

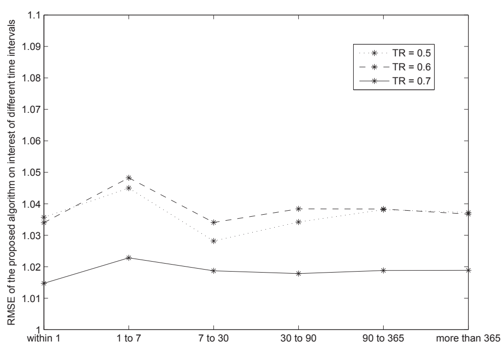


Fig. 4. Accuracy of the proposed algorithm on different phases of interest. TR means training ratio.

Comparing Fig. 4 with Fig. 3(a), we can see that the proposed algorithm has better performances than other algorithms even when it uses only the data in some single phases. We can see that the accuracies become higher when we use all the data, which illustrated that mining and making use of more related data can provide more useful information.

4 Conclusion

In this paper, we proposed a novel dynamic personalized recommendation algorithm for sparse data, in which more rating data is utilized in one prediction by involving more neighboring ratings through each attribute in user and item profiles. A set of dynamic features are designed to describe the preference information based on TSA technique, and finally a recommendation is made by adaptively weighting the features using information in multiple phases of interest. Experimental results on public MovieLens 100k and Netflix Competition data indicate that the proposed algorithm is effective, and its computational cost is also acceptable.

Acknowledgment

This work is supported by National Natural Science Foundation of China (No.61021063, No.61020106004). The authors would like to thank Dr. Brandon Norick, Dr. Jianjiang Feng and Mr. Quanquan Gu for their help in revising this paper.

References

[1] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: WWW, 2001, pp. 285–295.

[2] P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, Methods and Strategies of Web Personalization, Lecture Notes in Computer Science, Springer, 2007.

[3] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 734–749.

[4] Y. Koren, Collaborative filtering with temporal dynamics, Communications of the ACM 53 (4) (2010) 89–97.

[5] L. Candillier, F. Meyer, M. Boullé, Comparing state-of-the-art collaborative filtering systems, in: P. Perner (Ed.), MLDM, Vol. 4571 of Lecture Notes in Computer Science, Springer, 2007, pp. 548–562.

[6] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, H. Kriegel, Probabilistic memory-based collaborative filtering, IEEE Transactions on Knowledge and Data Engineering 16 (1) (2004) 56–69.

[7] F. Fouss, A. Pirotte, J. Renders, M. Saerens, Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation, IEEE TKDE 19 (3) (2007) 355–369.

[8] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.

[9] S. Boutemedjet, D. Ziou, Long-term relevance feedback and feature selection for adaptive content based image suggestion, Pattern Recognition 43 (12) (2010) 3925–3937.

[10] B. M. Kim, Q. Li, C. S. Park, S. G. Kim, J. Y. Kim, A new approach for combining content-based and collaborative filters, J. Intell. Inf. Syst. 27 (1) (2006) 79–91.

[11] G. Prassas, K. C. Pramataris, O. Papaemmanouil, Dynamic recommendations in internet retailing, in: ECIS, 2001.

[12] S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Factorizing personalized markov chains for next-basket recommendation, in: Proceedings of the 19th WWW, ACM, 2010, pp. 811–820.

[13] C. Xia, X. Jiang, S. Liu, Z. Luo, Z. Yu, Dynamic item-based recommendation algorithm with time decay, in: ICNC, IEEE, 2010, pp. 242–247.

[14] J. Lai, T. Huang, Y. Liaw, A fast k-means clustering algorithm using cluster center displacement, PR 42 (11) (2009) 2551–2556.

[15] A. Tsymbal, The problem of concept drift: definitions and related work, Computer Science Department, Trinity College Dublin.

[16] X. Tang, C. Yang, J. Zhou, Stock price forecasting by combining news mining and time series analysis, in: Web Intelligence, IEEE, 2009, pp. 279–282.

[17] J. Mohammed, Real-time implementation of an efficient rls algorithm based on iir filter for acoustic echo cancellation, in: IEEE/ACS ICCSA, IEEE, 2008, pp. 489–494.

[18] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression, The Annals of statistics 32 (2) (2004) 407–499.

[19] B. Boser, I. Guyon, V. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the fifth annual workshop on Computational learning theory, ACM, 1992, pp. 144–152.

[20] L. Li, W. Chu, J. Langford, X. Wang, Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms, in: Proceedings of the fourth ACM international conference on Web search and data mining, ACM, 2011, pp. 297–306.

[21] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2008, pp. 426–434.