Temporal Matrix Factorization for Tracking Concept Drift in Individual User Preferences

Yung-Yin Lo, Wanjiun Liao, *Fellow, IEEE,* Cheng-Shang Chang, *Fellow, IEEE,* and Ying-Chin Lee

***Abstract*—The matrix factorization (MF) technique has been widely adopted for solving the rating prediction problem in recommender systems. The MF technique utilizes the latent factor model to obtain *static* user preferences (user latent vectors) and item characteristics (item latent vectors) based on historical rating data. However, in the real world user preferences are not static but full of dynamics. Though there are several previous works that addressed this time varying issue of user preferences, it seems (to the best of our knowledge) that none of them is specifically designed for tracking concept drift in *individual user preferences*. Motivated by this, we develop a Temporal Matrix Factorization approach (TMF) for tracking concept drift in each individual user latent vector. There are two key innovative steps in our approach: (i) we develop a modified stochastic gradient descent method to learn an individual user latent vector at each time step, and (ii) by the Lasso regression we learn a linear model for the transition of the individual user latent vectors. We test our method on a synthetic dataset and several real datasets. In comparison with the original MF, our experimental results show that our temporal method is able to achieve lower root mean square errors (RMSE) for both the synthetic and real datasets. One interesting finding is that the performance gain in RMSE is mostly from those users who indeed have concept drift in their user latent vectors at the time of prediction. In particular, for the synthetic dataset and the Ciao dataset, there are quite a few users with that property and the performance gains for these two datasets are roughly 20% and 5%, respectively.**

keywords: Recommender systems, Rating prediction, Matrix factorization, Temporal dynamics, Concept drift

I. Introduction

With the accelerated growth of the Internet and a wide range of web services such as electronic commerce and online video streaming, people are easily overwhelmed by massive amounts of information and therefore recommender systems are indispensable tools to alleviate the information overload problem. At the heart of each recommender system, there is an algorithm that handles the rating prediction task and the accuracy of the rating prediction algorithm is the foundation of the system. The most successful and widely used approach to implement such an algorithm is collaborative filtering with matrix factorization (MF). Such an approach has the advantage of high accuracy, robustness and scalability, and it is thus more favorable than the other approaches, such as the neighborhood-based approach and the graph-based approach [1], [2]. The MF approach proved its success in the Netflix Prize competition [3] as the winning submission of this competition was heavily relied on it to predict unobserved ratings. The MF approach decomposes a user-item rating matrix into two low-rank matrices which directly profile *users* and *items* to the latent factor space respectively and these representative latent factors form the main basis for further prediction in the future.

Although MF is the state-of-the-art approach that can successfully process the relational rating data, its capability of capturing the temporal dynamics of users' preferences is quite limited. As we are facing the fast-moving business environment, the real world is not static but full of dynamics. There are a great variety of sources that can cause the changes of users' behavior, including shifting trends in the community, the arrival of new products, the changes in users' social networks, and so on. Recent research in [4] considered the aspect of personal development and pointed out that user's expertise may change from amateurs to connoisseurs as they become more experienced. To satisfy users' current taste and need, a key building block for recommender systems is to accurately model such user preferences as they evolve over time.

The need to model the temporal dynamics of user preferences raises some fundamental challenges. First of all, the amount of available data is significantly reduced in a specific time step and the sparsity problem of recommender systems is more severe in this situation. In addition, how can we generally incorporate the temporal dimension and further capture the evolution of preferences at the individual level for every time step? Finally, what is the principled method to model this kind of transition for every user in order to make more accurate predictions in the future? Toward this end, we propose a general and principled temporal dynamic model for tracking concept drift in each individual user latent vector. Such an approach can further effectively and efficiently achieve a lower RMSE than that of MF.

The main contributions of this paper include:

- We propose a Temporal Matrix Factorization approach (TMF) for tracking concept drift in each individual user latent vector. Such a method not only breaks the limit of using static decompositions in the original MF approach, but also provides a tool for recommender systems to better serve "valuable" customers in the future.
- We develop a modified stochastic gradient descent method to learn an individual user latent vector at each time step by using both the overall rating logs and the rating logs within the specific time step.
- By using the Lasso regression for the user latent vector at every time step, we learn a linear system model that

Y. Y. Lo and W. Liao are with Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C. email: {r02921080,wjliao}@ntu.edu.tw.

C. S. Chang and Y. C. Lee are with the Institute of Communications Engineering, National Tsing Hua University, Hsinchu 300, Taiwan, R.O.C. email: cschang@ee.nthu.edu.tw,ward8212@gmail.com

TABLE I
LIST OF NOTATIONS

M	The number of users
N	The number of items
T	$T-1$ training periods and prediction at time T
D	The number of latent factors
$R=(R_{i,j})$	The $M \times N$ rating matrix
$\hat{R}_{i,j}$	The prediction of $R_{i,j}$ via a prediction algorithm
P	The $D \times M$ user latent matrix
P_i	The i^{th} user latent vector
$P_i(t)$	The "learned" i^{th} user latent vector at time $t=1,2,\ldots,T-1$
$P_i(T)$	The predicted i^{th} user latent vector at time T
Q	The $D \times N$ item latent matrix
Q_j	The j^{th} item latent vector
A_i	The transition matrix of user i in (15)
b_i	The bias vector of user i in (15)
$b_i^{(k)}$	The k^{th} element of b_i
I	The identity matrix with an appropriate dimension
I_{ij}	The indicator function in (1) that is equal to 1 if user i rated item j and equal to 0 otherwise
$\tilde{A}_i$	The "modified" transition matrix with $\tilde{A}_i = A - I$
$Z_i(t)$	$Z_i(t) = P_i(t) - P_i(t-1)$
$Z_i^{(k)}(t)$	The k^{th} element of $Z_i(t)$
α	The learning rate of the (modified) SGD method
$e_{i,j}$	The prediction error in (3) and (9)
λ	The regulator parameter for the Lasso regression

can be used for modelling the transition pattern at the individual level.

- We conduct comprehensive experiments on a synthetic dataset and four real datasets, Ciao, Epinions, Flixster and MovieLens. In comparison with the original MF, our experimental results show that our TMF approach is able to achieve lower root mean square errors (RMSE) for both the synthetic and real datasets. In particular, there is roughly a 17-26% improvement on the synthetic dataset and a 5% improvement on the Ciao dataset. Such an improvement is quite significant.
- Our experiments also reveal one interesting finding. The performance gain in RMSE is mostly from those users who indeed have concept drift in their user latent vectors at the time of prediction. In particular, for the synthetic dataset and the Ciao dataset, there are quite a few users with that property and the performance gains for these two datasets are more significant than those for the other datasets.

The rest of paper is organized as follows. In Section 2, we provide a review of related work. We define the rating prediction problem in Section 3 and propose the method of incorporating temporal dynamics including capturing and predicting the user preferences in Section 4. In Section 5, we conduct experiments on both synthetic and several real datasets to validate our proposed temporal method. Finally, we conclude our work and point out the future research directions in Section 6.

In Table I, we provide a list for the notations that are used in the paper.

II. RELATED WORK

In this section, we first briefly review the MF approach for recommender systems and several recent approaches that intend to incorporate temporal dynamics with MF, including time-dependent collaborative filtering, tensor factorization, and collaborative Kalman filter.

A. Matrix Factorization

Matrix Factorization (MF) performs well in the rating prediction task and has attracted considerable attention. The rationale behind the MF approach is to characterize each user and item by a series of latent factors that can be used for representing or approximating the interactions between users and items from the historical rating logs. Specifically, given an $M \times N$ rating matrix $R=(R_{i,j})$ with M users and N items, the MF approach considers the following optimization problem:

$$\min_{P,Q} \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}\left(R_{ij}-Q_j^T\cdot P_i\right)^2+\frac{\lambda}{2}\left(\|P\|^2+\|Q\|^2\right), \tag{1}$$

where P and Q are the latent matrices which record the latent factors of users and items respectively. Also, P_i is the i^{th} column of P, Q_j is the j^{th} column of Q, and I_{ij} is an indicator function that is equal to 1 if user i rated item j and equal to 0 otherwise. The vector P_i, called the user latent vector of user i, is commonly used for representing (latent) user preferences, and the vector Q_j, called the item latent vector of item j, is commonly used for representing (latent) item characteristics. The regularization terms are added in the optimization problem to prevent overfitting.

We can also view MF from a probabilistic perspective. Probabilistic Matrix Factorization (PMF) [5], [6] defines the following conditional distribution over the observed ratings based on the linear model with Gaussian observation noise:

$$p\left(R|P,Q,\sigma^2\right)=\prod_{i=1}^{M}\prod_{j=1}^{N}\left[N\left(R_{ij}|Q_j^T\cdot P_i,\sigma^2\right)\right]^{I_{ij}}, \tag{2}$$

where $N\left(x|\mu,\sigma^2\right)$ denotes the probability density function of the Gaussian distribution with mean μ and variance σ^2. With placing zero-mean spherical Gaussian priors on the latent factors, the problem of maximizing the log-posterior with the fixed variance is equivalent to the optimization problem in (1).

Note that both P_i and Q_j are unknowns in (1) and the objective function is not *convex* [7]. Simon Funk [8] popularized a stochastic gradient descent (SGD) algorithm which loops through all ratings in the training set to find the latent matrices P and Q. For each training example R_{ij}, one first computes the associated prediction error

$$e_{ij}=R_{ij}-Q_j^T\cdot P_i. \tag{3}$$

One then updates P_i and Q_j in the opposite direction of the gradient as follows:

$$P_i \leftarrow P_i+\alpha\left(e_{ij}Q_j-\lambda P_i\right), \tag{4}$$

$$Q_j \leftarrow Q_j+\alpha\left(e_{ij}P_i-\lambda Q_j\right), \tag{5}$$

where α is the learning rate and λ is the regulator parameter. This incremental and iterative approach provides a practical way to scale the MF method to large datasets.

B. Time-dependent Collaborative Filtering

In order to provide recommendations that fit users' present preferences, time-dependent collaborative filtering (CF) [2] employs the availability of temporal information (time stamps) associated with user-item rating logs to put more emphasis on the recent ratings. Such an approach is based on the plausible assumption that recent logs have bigger influence on future events than old and obsolete logs. There are many prior works on time-dependent collaborative filtering, including neighborhood-based CF [9], [10], social influence analysis [11], [12], [13], temporal bipartite projection [14] and timeSVD++ [15]. Among all these prior works, timeSVD++ [15] is perhaps the most related work to MF. In [15], Koren proposed adding a time-varying rating bias for each user and each item to the estimate from the original MF. As such, the temporal dynamics of user latent vectors are only modelled by a simple sum of three factors, the stationary portion, a possible gradual change with linear equation of a deviation function, and a day-specific parameter for sudden drift. Even so, it was reported in [15] that timeSVD++ significantly outperforms SVD and SVD++ [16] (that considered implicit feedback).

Although these methods improve the accuracy of the prediction compared to the baseline MF estimator, there are some difficulties in the time-dependent CF approach. The system model in timeSVD++ for the user latent vectors is too simple to have any structural characterizations or constraints on their parameters. As such, these parameters (in various aspects and time steps) have to be learned individually and need lots of efforts on fine tuning. Thus, timeSVD++ maybe too data-specific to be used as a general model. Also, the assumption that claims recent ratings are always more important than old data may be oversimplified.

C. Tensor Factorization

Tensor factorization (TF) extends MF into a three-dimensional tensor by incorporating the temporal features into the prediction model. The underlying physical meaning of TF is that the given ratings not only depend on the user preferences and the item characteristics but also the current trend. There are two kinds of popular tensor factorization models in CF [17]: the CANDECOMP/PARAFAC (CP) model that decomposes the tensor into same rank of latent factors, and the Tucker model that considers the problem as the higher-order PCA.

There are some works that adopt the TF model for exploiting temporal information associated with user-item interactions. The Bayesian Probabilistic Tensor Factorization (BPTF) [18] extended PMF to CANDECOMP/PARAFAC tensor factorization that models each rating as the inner product of the latent factors of user, item, and time slice as well. It also imposes constraints that the adjacent time slices should share similar latent factors. The advantage of BPTF is its almost parameter-free probabilistic tensor factorization algorithm with a fully Bayesian treatment derivation while the drawback is it is not sensitive enough to capture the local changes of preferences compared with timeSVD++. Recently, Rafailidis and Nanopoulos [19] modeled continuous user-item interactions over time and defined a new measure of user preference dynamics to capture the shifting rate for each user. In a broader sense, recommendation can be regarded as a bipartite link prediction problem that aims to infer new interactions between users and items which are likely to occur in the near future. Based on this idea, Dunlavy et al. [20] considered bipartite graphs that evolve over time and demonstrated that tensor-based methods are effective for temporal data with varying periodic patterns. Apart from incorporating the temporal information, tensor factorization is a popular approach to integrate further information such as the context of implicit feedback in content-based recommender systems. For instance, Moghaddam et al. [21] added *review* as the third dimension based on the Tucker tensor model to address the problem of personalized review quality prediction and Shi et al. [22] directly trained the tensor model for creating an optimally ranked list of items for individual users in the context-aware recommender systems.

Tensor factorization provides a principled and well-structured approach to incorporate the temporal dynamics in recommender systems; however, the structure also limits the flexibility of the model so that it is hard to process and solve the decomposition especially for a large-scale and sparse tensor. Given the same amount of rating data, the higher order the tensor model is, the more severe the sparsity problem is. The sparsity problem leads to time-consuming computing, high space complexity and the convergence issues in the decomposition procedure.

D. Collaborative Kalman Filter

Inspired by the success of PMF that places Gaussian priors on the latent factors and formulates the matrix factorization problem as an optimization problem for obtaining the Maximum-a-Posteriori (MAP) estimate, there are some recent works that compute the MAP optimally by using the Kalman filter [23]. Considering the observed measurements over time with noise and uncertainties, the Kalman filter is the optimal linear estimator of unknown variables. Its recursive structure also allows new measurements to be processed as they arrive. The Kalman filter can be conceptualized in two phases: the *predict* phase is called a priori estimate which produces an estimation without the observation at the current time step, and the *update* phase is known as a posteriori estimate which refines the estimation with the current observation. The refinements of the state and covariance estimates are based on the optimal Kalman gain computed at every time step.

The paper [24] by Lu et al. might be the first paper to use the Kalman filter in recommender systems. In that paper, they exploited the Kalman filter to model the change of user preferences in its temporal component. Though they provided a new perspective on the recommender systems, their approach is still not general enough as the transition matrix used in the Kalman filter was only modeled by an *identity* matrix.

As such, one can only capture the drifts of user preferences. In another recent paper [25], Gultekin and Paisley proposed the collaborative Kalman filter (CKF) approach that used a geometric Brownian motion to model the dynamically evolving drift of each latent factor. The dynamic state space model proposed by [26], [27] is most related to our work. To solve the system identification problem for the linear system in the Kalman filter, they develop an EM algorithm that performs the Kalman filter and the RTS smoother. The EM algorithm is an iterative two-pass algorithm that yields estimates for the model parameters by using all observations in the expectation step, and then refines the estimates of the model parameters in the maximization step. Although the model is comprehensive and provides better results compared to the SVD and timeSVD++ approaches, there are some limitations in practice: (i) it makes a very strong assumption that assumes the transition matrix is homogeneous for all users. Such a homogeneous assumption is needed to simplify the model (as otherwise it is very difficult, if not impossible, to determine all their parameters from the EM algorithm [27]), and (ii) it is not suitable for large datasets due to the tractability and runtime performance.

In this paper, we will remove the assumption that the transition matrix is homogeneous for all users. By doing so, we allow our system to track concept drift in each individual user latent vector. Our experiments further verify that users do have different transition matrices. Some of them are simply governed by the identity matrix and have no concept drift in their latent vectors. On the other hand, some of them have significant changes of their latent vectors and the improvement of the rating for those users is the key factor for lowering RMSE in our temporal approach.

III. Problem Definition

In this paper, we study the rating prediction problem with time-stamped logs. Specifically, there are M users, indexed from $i = 1, 2, \ldots, M$, and N items, indexed from $j = 1, 2, \ldots, N$. For these users and items, we are given a set of time-stamped logs, where each log is represented by the four-tuple:

(user, item, rating, time).

We assume that every rating is a real-valued number and each item can be rated by a user at most once. If we neglect the time stamps of these logs, then the ratings of these logs can be represented by an $M \times N$ matrix $R = (R_{ij})$, where R_{ij} is the rating of user i on item j if item j has been rated by user i. On the other hand, if item j has not been rated by user i, then R_{ij} is said to be *missing*. In practice, the matrix R is a *sparse* matrix and there are many missing values. The rating prediction problem is then to predict the missing values in the matrix R.

To evaluate the performance of a rating prediction algorithm, the rating logs are partitioned into two sets: the training set and the testing set. The training set is given to a rating prediction algorithm to "learn" the needed parameters for rating prediction. On the other hand, the testing set is not revealed to a rating prediction algorithm and is only available for testing the accuracy of a rating prediction algorithm. Though there are many metrics for evaluating the performance of rating prediction algorithms, in this paper we adopt the root mean square error (RMSE) that can be computed as follows:

$$RMSE = \sqrt{\frac{\sum_{(i,j) \in Testing\ Set} \left(R_{ij} - \hat{R_{ij}}\right)^2}{|Testing\ Set|}}, \quad (6)$$

where $\hat{R}_{ij}$ is the prediction for R_{ij} via a rating prediction algorithm. The RMSE has been widely used in the literature, including the competition for the Netflix Prize. Although the range of RMSE might be quite small, there is evidence (see e.g., [16]) that small improvement in RMSE can have a significant impact on the quality of the top few recommendations from a rating prediction algorithm.

The original MF does not use the information of time stamps and simply decomposes the matrix R approximately into a product of two matrices: the user latent matrix and the item latent matrix. Though the item latent matrix could be quite stationary with respect to time, it is a general belief (see e.g., [15], [24], [26], [27]) there might be concept drift in the user latent matrix as users tend to change their mind over time. In view of this, our aim is to develop a temporal dynamic model for tracking concept drift in each individual user latent vector by using the time stamps of rating logs. By doing so, we can effectively and efficiently achieves a lower RMSE than that of MF.

IV. Temporal Matrix Factorization

For the rating prediction problem described in the previous section, we propose a Temporal Matrix Factorization (TMF) approach that is capable of tracking concept drift in each individual user latent vector. Our approach is based on the following assumptions that were previously used in the literature (see e.g., [15], [24], [26], [27]):

(i) Like the original MF, there are a user latent matrix $P \in \mathcal{R}^{D \times M}$ and an item latent matrix $Q \in \mathcal{R}^{D \times N}$ that can be used for approximating the rating matrix R. The i^{th} column of the user latent matrix P, denoted by P_i, is called the user latent vector of user i, $i = 1, 2, \ldots, M$, that can be viewed as the preferences of user i for the D latent factors. Similarly, the j^{th} column vector of the item latent matrix Q, denoted by Q_j, is called the item latent vector of item j, $j = 1, 2, \ldots, N$. The rating for user i on item j is then predicted by the inner product of P_i and Q_j.

(ii) There is concept drift in each individual user latent vector as people might change their preferences over time. Though it is arguable whether user preferences can be predicted or tracked, in particular sudden and random changes of user preferences, in this paper we assume that the changes of user preferences are rather smooth and can be tracked. For this, we denote by $P(t)$ the user latent matrix at time t, and $P_i(t)$ the user latent vector of user i at time t, $i = 1, 2, \ldots, M$.

(iii) As the characteristics of items are stationary, we assume that the item latent matrix Q is invariant with respect to time.

In view of these assumptions, the key ingredient of our TMF approach is to use the training data set to capture the dynamics of the concept drift in each individual user latent vector. For this, our approach consists of the following steps:

(i) Use the rating logs in the training data set to construct a time series of $M \times N$ rating matrices, $\{R(t), t = 1, 2, \ldots, T-1\}$.

(ii) Use the time series of rating matrices $\{R(t), t = 1, 2, \ldots, T-1\}$ to learn a time series of $D \times 1$ user latent vectors, $\{P_i(t), t = 1, 2, \ldots, T-1\}$, $i = 1, 2, \ldots, M$.

(iii) For each user i, use the time series of user latent vectors, $\{P_i(t), t = 1, 2, \ldots, T-1\}$ to learn the dynamics of the concept drift in the user latent vector.

(iv) Use the dynamics of the concept drift in each individual user latent vector to predict the user latent vector at time T, i.e., $P(T)$. Then use the product of $P(T)$ and the item latent matrix Q to predict the missing values in the testing data set.

A. Construction of a time series of rating matrices

The simplest way to construct a time series of rating matrices $\{R(t), t = 1, 2, \ldots, T-1\}$ is to partition rating logs into equally spaced time slices according to their time stamps. But, as the original rating matrix in a real data set might have already been very sparse, further partitioning of the rating logs might yield a time series of extremely sparse rating matrices that might not have any statistical significance at all. In view of the sparsity problem, the number of time slices T cannot be too large. To further mitigate the sparsity problem, one can consider a sliding window approach that merges the rating logs in several consecutive time slices into a single step. By doing so, there are *overlapping* rating logs in such a time series of rating matrices. Such an approach can not only mitigate the sparsity problem but also ensure smooth change of rating matrices so that prediction could be possible.

B. Learning a time series of user latent vectors

To learn a time series of user latent vectors for each user, we first perform MF for the rating matrix R to obtain the user latent matrix P and the item latent matrix Q. As we assume that the item latent matrix Q is invariant with respect to time, one might expect that

$$R_i(t) = Q^T \cdot P_i(t), \tag{7}$$

where $R_i(t)$ is the rating vector for user i on the N items (that can be extracted from the rating matrix $R(t)$). In view of this, a näive way to learn a time series of $D \times 1$ user latent vectors, $\{P_i(t), t = 1, 2, \ldots, T-1\}$, is to simply compute the Moore-Penrose pseudoinverse of Q^T from (7). It is well-known that the Moore-Penrose pseudo inverse computes a "best fit," i.e., the least squared solution to a system of linear equations and its uniqueness follows from the SVD theorem in matrix algebra. Such an approach works fine if all the entries in the vector $R_i(t)$ are known. In reality, there are many missing values in the vector $R_i(t)$ and thus make a direct computation of the Moore-Penrose pseudo inverse infeasible. One way to remedy this is to pad the missing values in $R_i(t)$ with the predicted values of user i by MF, i.e., $Q^T \cdot P_i$. In particular, one can generate another vector $\tilde{R}_i(t)$ as a linear combination of these two vectors, i.e.,

$$\tilde{R}_i(t) = \beta R_i(t) + (1-\beta) Q^T \cdot P_i. \tag{8}$$

If β is small, the padded values in $\tilde{R}_i(t)$ are all from the vector $Q^T \cdot P_i$. As such, the vectors $\tilde{R}_i(t), t = 1, 2, \ldots, T-1$, are all very similar and the corresponding Moore-Penrose pseudo inverse vectors also very similar. As a result, there is basically no change of the user latent vectors and that defeats the purpose of tracking the dynamics of user latent vectors. On the other hand, if β is large, then we basically ignore all the missing values in $R_i(t)$ and that causes great fluctuation of the user latent vectors which makes it extremely difficult to track the dynamics of user latent vectors. Also, as there are many missing values in $R_i(t)$, it is not clear whether the user latent vectors obtained this way possess any statistical significance.

The key insight to tackle this problem is that the user preferences at a specific time step are not only related to the ratings during that specific time step but also related to his/her overall behavior. In view of this, we first set $P_i(t)$ as the original user latent vector P_i. Then we use the observed ratings during that time step to "learn" $P_i(t)$. Specifically, we propose the following modified stochastic gradient descent method:

$$e_{ij}(t) = R_{ij}(t) - Q_j^T \cdot P_i(t), \tag{9}$$

$$P_i(t) \leftarrow P_i(t) + \alpha \left[e_{ij}(t) Q_j - \lambda P_i(t) \right]. \tag{10}$$

Unlike the standard stochastic gradient descent method for MF in (3)–(5), here we only update the latent vector $P_i(t)$ for every rating provided by user i at time t (as the item matrix Q is stationary). By doing so, the user latent vector $P_i(t)$ only updates his/her preferences for those items rated during time t and thus retains his/her overall behavior for those items not rated during time t. Such an approach not only overcomes the obstacle of data sparsity but also possesses meaningful user preferences in the temporal setting.

C. Learning the dynamics of the concept drift in the user latent vector

To track concept drift in the user latent vector, we need to identify a system model for the dynamics of the time series of the user latent vectors. Such a problem is known as the *system identification* problem in the literature [28]. One of the most commonly used models for system identification problems is the linear system model. As such, we consider the linear system model for the latent vector of each user. Specifically, we consider $P_i(t)$ as the state vector at time t and model the evolution of the state vector by

$$P_i(t) = A_i \cdot P_i(t-1) + b_i, \tag{11}$$

where A_i is a $D \times D$ matrix and b_i is a $D \times 1$ vector. The matrix A_i is called the *transition* matrix for user i and b_i is called the *bias* vector of user i.

We note that such a linear system model is not capable of detecting/tracking sudden and random changes of user preferences. However, as we use the sliding window approach (with overlapping rating logs) in Section IV-A to construct the time series of rating matrices and the modified stochastic gradient descent method in Section IV-B to learn the latent vectors, we believe the latent vectors learned this way should be relatively smooth and could be tracked by using the simple linear system model in (11). On the other hand, it is possible to approximate a more complicated nonlinear system by combining various linear systems that switch between the critical points of the original nonlinear system. However, as the number of time slices in our datasets is rather small, it would be difficult to use a set of linear systems and select the most appropriate linear system when a sudden change of user preference is detected.

It seems plausible to assume that the user latent vectors do not vary a lot in each time step. As such, we replace the transition matrix A_i by $\left(I + \tilde{A}_i\right)$ in (11). This then leads to

$$Z_i(t) = \tilde{A}_i \cdot P_i(t-1) + b_i, \tag{12}$$

where

$$Z_i(t) = P_i(t) - P_i(t-1). \tag{13}$$

By doing so, we expect that the matrix $\tilde{A}_i$ is sparse and it only contains a small number of nonzero entries. It is known [29] that the Lasso regression provides parameter shrinkage and variable selection that limit the number of nonzero elements in the parameters. As such, we apply the Lasso regression to estimate $\tilde{A}_i$ and b_i in (12) from the $T-2$ "observations" of the output $\{Z_i(t), t = 2, \ldots, T-1\}$ with the input $\{P_i(t), t = 2, \ldots, T-1\}$. Specifically, for each factor k, $k = 1, 2, \ldots, D$, we let $A_i^{(k)}$ be the k^{th} row of A_i, $b_i^{(k)}$ be the k^{th} element in b_i, $Z_i^{(k)}(t)$ be the k^{th} element in $Z_i(t)$ and consider the following optimization problem:

$$\min_{\tilde{A}_i^{(k)}, b_i^{(k)}} \frac{1}{2(T-2)} \sum_{t=2}^{T-1} \left(Z_i^{(k)}(t) - \tilde{A}_i^{(k)} \cdot P_i(t) - b_i^{(k)} \right)^2 + \lambda \| \tilde{A}_i{}^{(k)} \|_1, \tag{14}$$

where $\| \tilde{A}_i{}^{(k)} \|_1$ is the L_1-norm of the vector $\tilde{A}_i{}^{(k)}$ and λ is a nonnegative regulator parameter for the Lasso regression. As λ increases, the number of nonzero elements in the vector $\tilde{A}_i{}^{(k)}$ decreases. In our experiments, we will use the Matlab tool [30] to solve the above Lasso regression.

D. Rating prediction

Once we obtain the transition matrix A_i and the bias vector b_i, we can use the system dynamic in (11) to predict the latent vector of user i at time T by the following equation:

$$P_i(T) = A_i \cdot P_i(T-1) + b_i. \tag{15}$$

As in the original MF, the missing values in the testing data set are then predicted by using the product of the user latent vector and the item latent matrix, i.e.,

$$R_i(T) = Q^T \cdot P_i(T). \tag{16}$$

ALGORITHM 1: The Temporal Matrix Factorization (TMF) approach

Input: A collection of time-stamped rating logs $(user, item, rating, time)$ with M users and N items.

(0) (Matrix Factorization) Ignore the time stamps and represent the rating logs by an $M \times N$ rating matrix $R = (R_{i,j})$, where $R_{i,j}$ is the rating of user i on item j if item j has been rated by user i. Perform the matrix factorization by using LIBMF [31], [32] to obtain the $D \times M$ user latent matrix P and the $D \times N$ item latent matrix Q.

(1) (Construction of a time series of rating matrices) Construct a time series of rating matrices $\{R(t), t = 1, 2, \ldots, T-1\}$ by using the sliding window approach in Section IV-A.

(2) (Learning a time series of user latent vectors) Set $P_i(t)$ as the original user latent vector P_i (the i^{th} column of P). Use the modified stochastic gradient descent method in (9) and (10) to "learn" $P_i(t)$:

$$e_{ij}(t) = R_{ij}(t) - Q_j^T \cdot P_i(t),$$
$$P_i(t) \leftarrow P_i(t) + \alpha \left[e_{ij}(t) Q_j - \lambda P_i(t) \right].$$

The latent vector $P_i(t)$ is only updated for every rating provided by user i at time t. See the pseudo-code in Algorithm 2.

(3) (Learning the dynamics of the user latent vectors) For each user i, solve the Lasso regression in (14) to find the $D \times D$ transition matrix A_i and the $D \times 1$ bias vector b_i for the linear system model

$$P_i(t) = A_i \cdot P_i(t-1) + b_i.$$

(4) (Rating prediction) Predict the latent vector of user i at time T by

$$P_i(T) = A_i \cdot P_i(T-1) + b_i.$$

Predict the (missing) rating of user i at time T by

$$R_i(T) = Q^T \cdot P_i(T).$$

The complete learning procedure of our TMF method is outlined in Algorithm 1 and a pseudo-code of the modified stochastic gradient descent method is given in Algorithm 2.

V. Experimental Results

In this section, we perform various experiments to evaluate the performance and efficiency of our temporal method via a synthetic dataset and four real datasets. All our experiments are implemented in MATLAB and executed on a server equipped with an Intel Core i7 (4.2GHz) processor and 64G memory on the Linux system.

A. Experiments on the Synthetic Dataset

We first conduct our experiments on synthetic data. The main reason for doing this is to test our method in a *controllable* environment so that we can gain insights of the effects

ALGORITHM 2: A pseudo-code of the modified stochastic gradient descent method

function $P(t)$ = modifiedSGD(ratingLogs, P, Q, D, T, iterations, α, λ)
for $t = 1 : T - 1$
  $P(t) \leftarrow P$;
  for iter = 1:iterations
    for index = 1:size(ratingLogs, 1)
      $i \leftarrow$ ratingLogs(index,1,t);
      $j \leftarrow$ ratingLogs(index,2,t);
      $R_{ij}(t) \leftarrow$ ratingLogs(index,3,t);
      $e_{ij}(t) \leftarrow R_{ij}(t) - Q_j^T \cdot P_i(t)$;
      for $d = 1 : D$
        $P_{i,d}(t) \leftarrow P_{i,d}(t) + \alpha * \Big(e_{ij}(t) * Q_{j,d} - \lambda * P_{i,d}(t)\Big)$;
      end
    end
  end
end

of various parameters and thus better understand when our method could be effective.

To generate the synthetic data, we set $M = 10,000$ and $N = 10,000$, i.e., there 10,000 users and 10,000 items. The density of the rating matrix R is set to 1%, and that gives 1,000,000 ratings. We generate all the entries in both the initial user latent matrix $P(1)$ and the item latent matrix Q by uniformly distributed random variables over $(0, 1)$. To model the evolution of the user latent vector for each user i, the transition matrix A_i is generated by the sum of the identity matrix and a random matrix R' with all its entries generated from a uniform distribution. The entries in the bias vector b_i are also generated from a uniform distribution. Various ranges of the entries in R' and b_i are specified in our experiments (see Table II). The number of steps T is set to 10 and the rating logs are then generated according to equations (7) and (11).

In Table II, we report the RMSE for both the original MF (implemented by the LIBMF library [31], [32]) and our method for various parameter settings. The parameters that we choose for MF are the learning rate α= 0.02, the regulator parameter $\lambda = 0.02$, and the number of latent factors $D = 30$. We run 50 iterations of SGD to obtain the latent factors in the original MF. The learning rate α and the regulator parameter λ in equation (10) for computing the user latent vector at every time step in our method are set to be the same as those for MF. As can be seen from this table, our method consistently and significantly outperforms MF in all the parameter settings. The improvement depends on the transition matrix and the bias vector that are selected to control the concept drift in the user latent vector. A more careful examination reveals that the improvement for our method is relatively small if the range of the entries in the random matrix R' is small, e.g., $(-0.01, 0.01)$. This is because the transition matrix in such a scenario is very close to the identity matrix and there is almost no change of the user latent vectors. As such, MF performs well and yields a low RMSE. On the other hand, if such a range is large, the prediction accuracy of MF is low as MF relies on the assumption of stationary user preferences. As our method is capable of tracking the concept drift in the user latent vector, our method achieves roughly 17-26% improvement in terms of RMSE.

TABLE II
THE RMSE RESULTS ON THE SYNTHETIC DATASET FOR VARIOUS PARAMETER SETTINGS.

Range of R'	Range of b_i	MF	TMF	Improvement
(-0.01, 0.01)	(-0.01, 0.01)	0.4566	0.4246	7.01%
(-0.05, 0.05)		0.8468	0.6758	20.19%
(-0.1, 0.1)		1.2667	0.9790	22.71%
(-0.3, 0.3)		1.7007	1.3780	18.97%
(-0.5, 0.5)		1.8046	1.4967	17.06%
(-0.01, 0.01)	(-0.1, 0.1)	0.5763	0.5124	11.09%
(-0.05, 0.05)		0.9645	0.7660	21.20%
(-0.1, 0.1)		1.2985	0.9539	26.54%
(-0.3, 0.3)		1.7092	1.3673	19.96%
(-0.5, 0.5)		1.8091	1.4787	18.26%

Next, we study the effect of the range of b_i. In Table II, we consider two different ranges of b_i. The experimental results show that the values of RMSE are larger when the given range is larger (under the condition of using the same transition matrix). Moreover, the performance gain from our method is also larger. This shows the importance of adding the bias vectors in our linear system model.

In addition to the prediction results presented above, we demonstrate the state tracking ability of our temporal method. Denote by $\hat{P}_i(t)$ the user latent vector of user i at time t (computed by our temporal method) and $P_i(t)$ the given ground truth in the synthetic dataset. We compute the dissimilarity measure of these two latent vectors by using the RMSE metric as follows:

$$s\left(P_i(t), \hat{P}_i(t)\right) = \sqrt{\frac{\sum_{d=1}^{D}\left(P_{id}(t) - \hat{P_{id}}(t)\right)^2}{D}}, \qquad (17)$$

where D is the dimensions of these two latent vectors. To compare the tractability of the MF approach and our temporal method, we measure the average of the dissimilarities among all the users at a specific time step t and plot the results in Figure 1. As can be seen from Figure 1, the gain of using our temporal method to track the user latent vectors increases over time and at the time for prediction, i.e., the 10^{th} time step, the gain is near 13%.

To dig further, we examine the rating prediction result for each user. We observe that our temporal method consistently obtains better prediction results if the MF approach always overestimates (or underestimates) all the ratings provided by a user in the testing dataset. For instance, we show in Table III a user (user ID 26) who gives extreme high ratings and another user (user ID 28) who gives extremely low ratings. This is because the MF approach cannot track the concept drift in the user latent vector even when there is a distinct downward (or upward) trend in the evolution of user preferences. On the

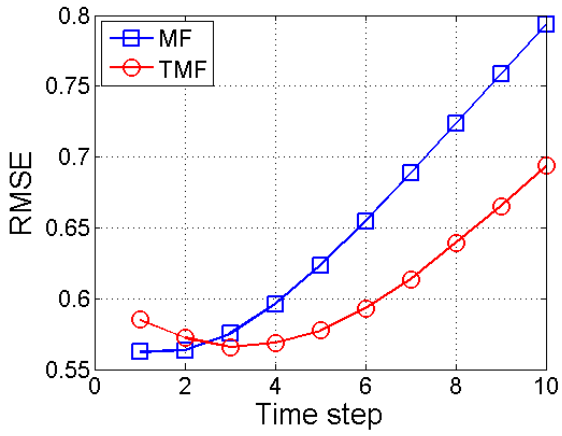

Fig. 1. The average dissimilarities among all users at each time step.

TABLE III
THE RATING PREDICTION RESULTS WITH EXTREME ACTUAL RATINGS.

user ID	item ID	actual rating	MF	TMF
26	1139	5	4.51	4.95
	1303	5	4.62	5.07
	2092	5	3.87	4.26
	2200	5	4.22	4.63
	2625	5	3.83	4.21
	2867	5	4.43	4.86
	3515	5	4.69	5.14
	6495	5	4.67	5.13
	7864	5	3.71	4.07
	8693	5	4.21	4.62
28	92	1	2.24	0.88
	1440	1	2.44	0.96
	1626	1	3.08	1.18
	1917	1	3.15	1.27
	3234	1	2.95	1.15
	3556	1	2.62	0.98
	4425	1	2.91	1.10
	8990	1	2.23	0.85
	9262	1	2.37	0.94
	9978	1	2.79	1.09

other hand, if the ratings provided by a user are distributed over the entire range as shown in Table IV, then the ratings predicted by our temporal method are not always better than those from those predicted by the MF. But, the overall accuracy of our temporal method is still better. In summary, our temporal method is good at capturing the general trend and thus yields improvement on the overall performance.

To further understand the effect of the number of latent factors D, we show in Figure 2 the comparison results on the synthetic dataset (with both R' and b_i being $(-0.1, 0.1)$ in Table II) for the three approaches: MF, TMF and CKF [25]. As in the previous experiment, we use the LIBMF library [31], [32] to implement MF. We are very grateful to Mr. San Gultekin (one of the authors of [25]) for providing us the source code for CKF. In order to have a fair comparison of MF and TMF, we use the same parameter setting for the synthetic dataset and set the learning rate $\alpha = 0.01$, the regulator

TABLE IV
THE RATING PREDICTION RESULTS WHEN ACTUAL RATINGS ARE DISTRIBUTED OVER THE ENTIRE RANGE.

user ID	item ID	actual rating	MF	TMF
32	976	3	2.74	3.09
	1224	4	2.63	2.98
	1379	5	3.17	3.64
	1691	2	2.72	3.06
	3989	4	2.86	3.25
	5079	1	2.62	2.95
	6541	2	2.60	2.95
	7455	4	2.58	2.93
	9691	3	2.46	2.79
	9944	3	2.47	2.81

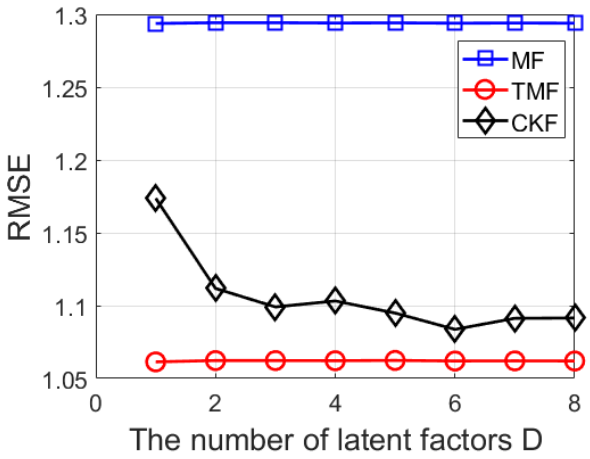

Fig. 2. Comparison results for various values of D with MF, TMF and CKF on the synthetic dataset.

parameter $\lambda = 0.02$ and 50 iterations for each run. We run CKF by using the default setting in the source code. As shown in Figure 2, TMF is rather insensitive to the number of latent factors D and has a lower RMSE than those of MF and CKF in this range of D.

B. Experiments on the Real Datasets

To validate our temporal method in practical environments, in this section we conduct our experiments on real datasets. Motivated by the increasing importance of recommender systems on electronic commerce and online video streaming services, we consider Ciao, Epinions, Flixster and MovieLens. With Web 2.0 technique to gather users' feedbacks such as explicit ratings and implicit reviews, Ciao and Epinions are two of the most popular online-shopping websites. Ciao is a European-based online-shopping portal with websites and claims and it reaches an audience of 28.4 million monthly unique visitors in Europe [33]. Epinions is established in 1999 and now is the largest consumer review site with thousands of product reports in the world. Another focus is the movie/video recommendation platform which becomes more popular in our daily life. For this, we consider Flixster and MovieLens in our work. Flixster is an American social movie site which

TABLE V
STATISTICS OF THE REAL DATASETS.

	Ciao	Epinions	Flixster	MovieLens
Users	1,947	21,752	114,747	125,041
Items	5,004	242,842	44,439	17,951
Training Ratings	22,068	830,043	7,376,472	18,000,243
Testing Ratings	826	23,621	416,293	353,575
Density	0.23%	0.02%	0.14%	0.80%
Earliest Rating	Jun. 2000	Jul. 1999	Dec. 2005	Jan. 1995
Latest Rating	Apr. 2011	May 2011	Nov. 2009	Mar. 2015

TABLE VI
THE RMSE RESULTS FOR THE FOUR REAL DATASETS.

	Ciao	Epinions	Flixster	MovieLens
MF	1.1099	1.1287	1.1189	0.8170
TMF	1.0540	1.1189	1.1102	0.8150
Improvement	5.04%	0.87%	0.78%	0.24%

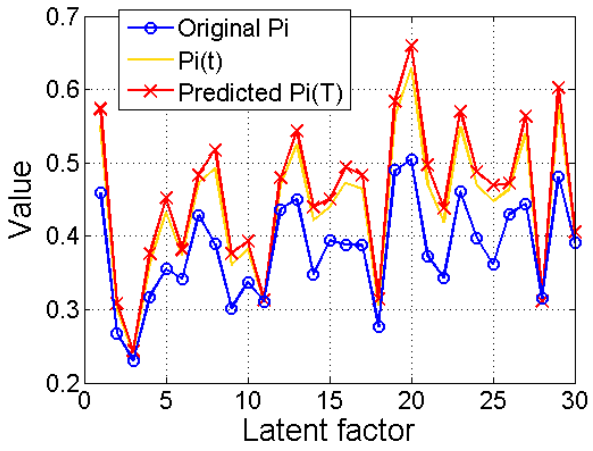

Fig. 3. The evolution of the user latent vector for user 49 in the Ciao dataset.

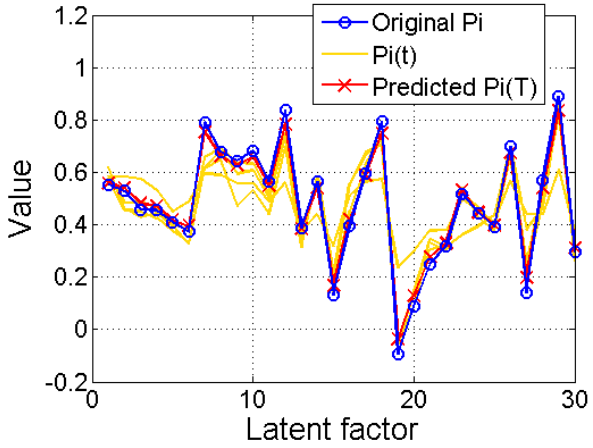

Fig. 4. The evolution of the user latent vector for user 108 in the Ciao dataset.

also provides applications in Facebook and MySpace for users to share film reviews and ratings whereas MovieLens is a recommender system for research of collaborative filtering run by GroupLens Research. These datasets with the information of ratings and the associated time stamps are publicly available in [34], [35], [36] and their statistics information is shown in Table V.

All of these platforms provide services for users to rate items using a 5-point Likert scale while Flixster adopts 10 discrete numbers in the range [0.5,5] with step size 0.5. Each log in a dataset contains the information of user ID, item ID, rating and timestamp. First of all, we sort these logs in the chronological order to form a time series. Note that this setting is more practical than the traditional approach because we are only allowed to use the past data to predict future events. We partition the whole dataset into 10 time slices equally and leave the last slice as the testing set. In order to have an enough number of representative ratings and have smooth and trackable transitions, we apply the sliding window approach to combine the logs in every 5 consecutive time slices to form a time step. By doing so, 4 of the slices in each step overlap with those in the next time step and there are totally 6 time steps (T = 6) in this setting. Next, we remove new users and new items, i.e., the users and items appear only once in the testing set, and focus on tracking the evolution for the latent vectors of existing users. We adopt the same parameter settings as those in the synthetic dataset except that we choose the learning rate α = 0.01 (in (10)) for the Flixster and MovieLens datasets. The experimental results for RMSE by the MF method and our method are shown in Table VI.

In comparison with the MF approach, we can see from Table VI that our temporal method improves the performance in RMSE in all four real datasets. However, the gain varies from one to another. In Ciao, we obtain 5% improvement. For the other three datasets, the improvements are not that significant. One possible explanation for this is that the temporal effect depends on the dataset due to its intrinsic properties. We also observe the improvements in real datasets are not as significant as those in the synthetic dataset. The main reason is that we purposely construct the synthetic dataset so that it possesses the desired concept drift in the user latent vector. It is not clear whether there are concept drifts in the user latent vectors in the Epinions, Flixster, and MovieLens datasets.

To see whether there is performance gain by tracking concept drift in the latent vector of a user, we further examine the evolution of various user latent vectors and their corresponding rating prediction results (see Figures 3–5). An interesting finding is that users basically can be classified into two types: one is *beneficial* to track concept drift in his/her user latent vector and the predicted latent vector from our method is substantially different from that from MF, and the other is *worthless* to track concept drift in his/her user latent vector as the predicted latent vector from our method is quite close to that from MF. In the Ciao dataset, we observe that the majority of improvement made by the users whose latent vectors evolve in a consistent direction. A plain example of this case is that the user latent vector changes in only one step from the original latent vector. We list the evolution of the first five latent factors of the latent vector (due to space limitation) and the corresponding ratings in Table VII. To visualize this case

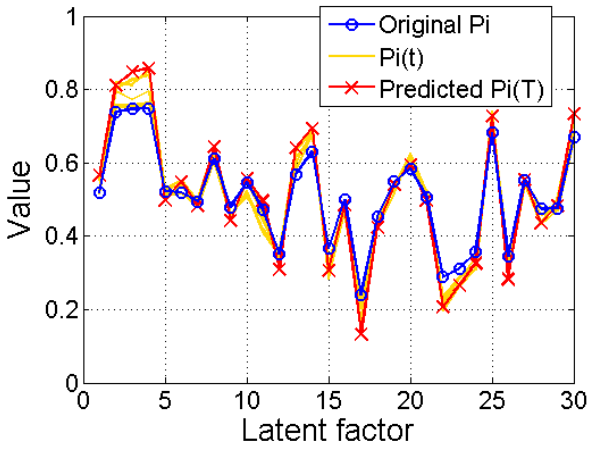

Fig. 5. The evolution of the user latent vector for user 339 in the Epinions dataset.

more clearly, we also plot the factors of the latent vector for the original P_i (computed by MF and marked in blue), the factors of the user latent vector at time step $T-1$ (marked in yellow), and the factors of the predicted user latent vector $P_i(T)$ with $T = 6$ (marked in red) for user 49 in the Ciao dataset in Figure 3. As can be seen from this figure, the predicted latent vector from our method is substantially different from that from MF. For this user, the corresponding ratings in Table VII show that such a user is beneficial to track concept drift in his/her user latent vector. On the other hand, we plot in Figure 4 the factors of the latent vector for the original P_i (computed by MF and marked in blue), the factors of the user latent vector at time step t (marked in yellow), and the factors of the predicted user latent vector $P_i(T)$ with $T = 6$ (marked in red) for user 108 in the Ciao dataset. For user 108, the predicted latent vector by our method is very close to that from MF. As such, the predicted rating by using our method and MF are also very close as shown in Table VIII. For such a user, there is little performance gain to track concept drift in his/her latent vector.

In Epinions, Flixster and MovieLens datasets, we observe that the latent vectors of most users change smoothly but not usually in a consistent direction. As such, the predicted $P_i(T)$ is similar to the original latent factors P_i. The corresponding latent vectors and the prediction results are shown in Table IX and Figure 5 for a typical user (user 339) in the Epinions dataset. In this case, considering the overall ratings without using the information of time stamps (such as MF) is capable of yielding good estimations and there is little performance gain to track concept drift in the user latent vector for most users.

Now we report the run-time of our temporal method on these datasets in Table X. The run-time includes learning the user latent vectors, learning the transition matrices, and further performing rating prediction which quantify the additional efforts after obtaining the original latent matrices P and Q from MF. As we use Matlab to implement our temporal method (except we use LIBMF [31], [32] in the step for MF), we also implement MF by using Matlab and report the run-

TABLE VII
THE FIRST FIVE FACTORS OF USER 49 IN THE CIAO DATASET AND THE CORRESPONDING PREDICTION RESULTS.

Factor	1	2	3	4	5
P_i	0.4592	0.2673	0.2304	0.3172	0.3549
$\hat{P}_i(1)$					
$\hat{P}_i(2)$	0.4592	0.2673	0.2304	0.3172	0.3549
$\hat{P}_i(3)$					
$\hat{P}_i(4)$					
$\hat{P}_i(5)$	0.5507	0.3000	0.2411	0.3640	0.4328
$\hat{P}_i(6)$	0.5737	0.3082	0.2438	0.3757	0.4522

user ID	item ID	actual rating	MF	TMF
49	36	4	3.13	3.76
	138	4	3.50	4.20
	711	5	3.11	3.74
	712	5	3.80	4.57
	713	5	3.78	4.55

TABLE VIII
THE FIRST FIVE FACTORS OF USER 108 IN THE CIAO DATASET AND THE CORRESPONDING PREDICTION RESULTS.

Factor	1	2	3	4	5
P_i	0.5814	0.5842	0.5738	0.5327	0.4485
$\hat{P}_i(1)$	0.5814	0.5842	0.5738	0.5327	0.4485
$\hat{P}_i(2)$	0.5777	0.4566	0.4518	0.4261	0.3749
$\hat{P}_i(3)$	0.6218	0.4630	0.4285	0.4449	0.3949
$\hat{P}_i(4)$	0.5849	0.4845	0.4855	0.4655	0.4043
$\hat{P}_i(5)$	0.5590	0.5398	0.4818	0.4703	0.4174
$\hat{P}_i(6)$	0.5534	0.5287	0.4588	0.4547	0.4096

user ID	item ID	actual rating	MF	TMF
108	122	5	4.16	4.12
	251	4	3.57	3.59
	447	5	4.47	4.49
	469	5	4.83	4.88
	531	5	4.49	4.57
	768	5	5.01	5.01
	823	5	3.78	3.86
	1258	5	3.65	3.70
	1319	4	4.16	4.13
	1320	5	3.74	3.78
	1321	5	5.17	5.16
	1322	5	4.78	4.82
	1323	4	3.72	3.62
	1324	5	4.74	4.79
	1325	5	4.74	4.77
	1326	5	5.03	5.14

TABLE IX
THE FIRST FIVE FACTORS OF USER 339 IN THE EPINIONS DATASET AND THE CORRESPONDING PREDICTION RESULTS.

Factor	1	2	3	4	5
P_i	0.4953	0.5479	0.4734	0.5847	0.5540
$\hat{P}_i(1)$	0.4781	0.5194	0.4089	0.6105	0.5400
$\hat{P}_i(2)$	0.4810	0.5245	0.4178	0.6249	0.5505
$\hat{P}_i(3)$	0.4968	0.5065	0.4450	0.6246	0.5427
$\hat{P}_i(4)$	0.4793	0.5483	0.4625	0.6249	0.5398
$\hat{P}_i(5)$	0.4825	0.5496	0.4789	0.5966	0.5512
$\hat{P}_i(6)$	0.4837	0.5571	0.4964	0.5932	0.5540

user ID	item ID	actual rating	MF	TMF
339	9188	4	3.57	3.59
	9189	1	0.95	0.96
	9190	4	3.38	3.39
	9191	4	3.93	3.93
	9192	4	3.54	3.56
	9193	5	3.98	4.11
	9194	3	4.02	4.03

TABLE X
THE RUN-TIME FOR OUR TEMPORAL METHOD AND MF ON VARIOUS DATASETS.

	Synthetic	Ciao	Epinions	Flixster	MovieLens
TMF	57.60m	3.79m	25.90m	78.42m	143.08m
MF (LIBMF)	2.32s	0.18s	6.85s	20.47s	44.27s
MF (Matlab)	27.61m	0.67m	24.58m	218.75m	532.93m

time for performing MF on these datasets. As shown in Table X, the run-time of MF by LIBMF is in the order of seconds and the run-time of TMF and MF by Matlab is in the order of minutes. The additional efforts of our temporal methods (in terms of run-time) are comparable to those for performing MF by using Matlab.

To further understand the effect of the number of latent factors D, we show in Figure 6 the comparison results on the Ciao dataset for the three approaches: MF, TMF, and CKF [25]. For MF and TMF, we set the learning rate $\alpha = 0.02$, the regulator parameter $\lambda = 0.02$ and 50 iterations for each run. We run CKF by using the default setting in the source code. We note that there is a version update of the LIBMF library [31], [32] and the MF version of this experiment, i.e., the 2016 version, is different from the old version used in Table VI. As such, the RMSE results for both MF and TMF are slightly different from those in Table VI.

VI. CONCLUSIONS AND DISCUSSIONS

In this paper, we proposed a Temporal Matrix Factorization approach (TMF) for tracking concept drift in each individual user latent vector. There are two key innovative steps in our approach: (i) a modified stochastic gradient descent method to learn an individual user latent vector at each time step, and (ii) a linear model for the transition of the individual user latent vectors by the Lasso regression. In comparison with the other approaches that intend to incorporate temporal dynamics with

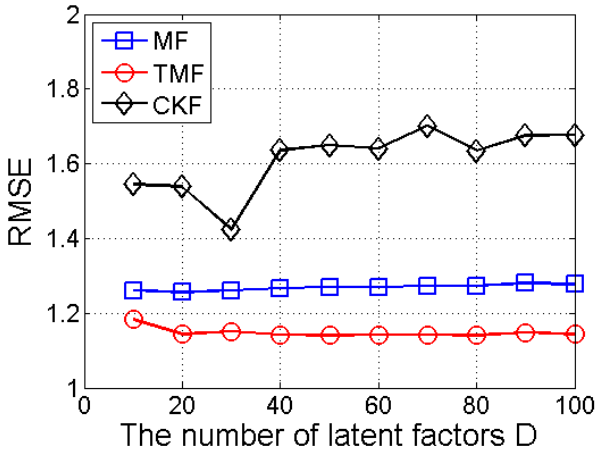

Fig. 6. Comparison results for various values of D with MF, TMF, and CKF on the Ciao dataset.

MF in the literature, there are several distinctive features of our temporal method:

(i) Our modified stochastic gradient descent method is able to alleviate the data sparsity problem for learning the user preferences at a certain time step. This overcomes the data sparsity problem in tensor factorization.

(ii) Unlike the CKF approach [27], we do not need to assume the transition matrix is *homogeneous*. Thus, we are allowed to track concept drift in each individual user latent vector.

In comparison with the original MF, our temporal method is able to achieve lower root mean square errors (RMSE) for both the synthetic and real datasets. One interesting finding is that the performance gain in RMSE is mostly from those users who indeed have concept drift in their user latent vectors at the time of prediction. As our temporal method is specifically designed for each user, one can save a lot of efforts by only tracking those users who indeed have concept drift in their user latent vectors at the time of prediction. However, identifying those users is not an easy task and might require further study. One possible approach for this is to examine the transition matrix for each user. In our experiments, we found that there are many users whose transition matrices are the identity matrix and those users are not worth tracking.

Another research direction is to study the effect of cold start users (who have very few ratings). One might think cold start users are difficult to predict and then immediately filter out their ratings in the preprocessing step. However, in our temporal method, the ratings of cold start users might be valuable as they contribute to the item latent matrix Q which in turn affects the accuracy of estimating the time series of the latent vectors of other users.

As pointed out by one of the reviewers, the effects of user-biases and item-biases could be important for real datasets. Such effects are not taken into account in our TMF approach. To see the effects of user-biases and item-biases, we compare the MF, the biased MF [7] and the TMF for the synthetic dataset and the four real datasets. In Table XI, we show the comparison results for the synthetic dataset by using various

TABLE XI
THE RMSE RESULTS OF MF, TMF AND BIASED MF FOR THE SYNTHETIC DATASET UNDER VARIOUS PARAMETER SETTINGS.

Range of R'	Range of b_i	MF	TMF	biased MF
(-0.01, 0.01)	(-0.01, 0.01)	0.4566	0.4246	0.4898
(-0.05, 0.05)		0.8468	0.6758	0.9122
(-0.1, 0.1)		1.2667	0.9790	1.3884
(-0.3, 0.3)		1.7007	1.3780	1.9489
(-0.5, 0.5)		1.8046	1.4967	2.1029
(-0.01, 0.01)	(-0.1, 0.1)	0.5763	0.5124	0.7690
(-0.05, 0.05)		0.9645	0.7660	0.9791
(-0.1, 0.1)		1.2985	0.9539	1.4003
(-0.3, 0.3)		1.7092	1.3673	1.9552
(-0.5, 0.5)		1.8091	1.4787	2.0997

TABLE XII
THE RMSE RESULTS OF MF, TMF AND BIASED MF FOR THE FOUR REAL DATASETS.

	Ciao	Epinions	Flixster	MovieLens
MF	1.1099	1.1287	1.1189	0.8170
TMF	1.0540	1.1189	1.1102	0.8150
biased MF	0.9131	1.0564	0.9855	0.7866

parameter settings as in Table II. As there are no user-biases and item-biases in the synthetic dataset, the RMSE of the biased MF is even worse than that of MF. This is because the biased MF has to "learn" additional bias parameters that are known to be 0 in the synthetic dataset. On the other hand, we show in Table XII the comparison results for the four real datasets used in Table VI. The RMSE of the biased MF is better than that of MF and that of TMF. This suggests that there are indeed user-biases and item-biases in the four real datasets and exploiting the effects of user-biases and item-biases in the real datasets can lead to good performance improvement. In view of this, one of our future research directions is to incorporate the user-biases and item-biases into our TMF approach.

REFERENCES

[1] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, p. 4, 2009.

[2] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, p. 3, 2014.

[3] X. Amatriain, "Mining large streams of user data for personalized recommendations," *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 37–48, 2013.

[4] J. J. McAuley and J. Leskovec, "From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews," in *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013, pp. 897–908.

[5] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, 2007, pp. 1257–1264.

[6] N. D. Lawrence and R. Urtasun, "Non-linear matrix factorization with gaussian processes," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 601–608.

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

[8] B. Webb, "Netflix update: Try this at home," *Blog post sifter. org/simon/journal/20061211. html*, 2006.

[9] Y. Ding and X. Li, "Time weight collaborative filtering," in *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, 2005, pp. 485–492.

[10] N. Lathia, S. Hailes, and L. Capra, "Temporal collaborative filtering with adaptive neighbourhoods," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2009, pp. 796–797.

[11] X. Yang, Y. Guo, Y. Liu, and H. Steck, "A survey of collaborative filtering based social recommender systems," *Computer Communications*, vol. 41, pp. 1–10, 2014.

[12] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 241–250.

[13] R. Pálovics, A. A. Benczúr, L. Kocsis, T. Kiss, and E. Frigó, "Exploiting temporal influence in online recommendation," in *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 2014, pp. 273–280.

[14] T. Wu, S.-H. Yu, W. Liao, and C.-S. Chang, "Temporal bipartite projection and link prediction for online social networks," in *IEEE International Conference on Big Data (Big Data), 2014*. IEEE, 2014, pp. 52–59.

[15] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010.

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

[17] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[18] L. Xiong, X. Chen, T.-K. Huang, J. G. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with Bayesian probabilistic tensor factorization." in *SDM*, vol. 10. SIAM, 2010, pp. 211–222.

[19] D. Rafailidis and A. Nanopoulos, "Modeling the dynamics of user preferences in coupled tensor factorization," in *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 2014, pp. 321–324.

[20] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 5, no. 2, p. 10, 2011.

[21] S. Moghaddam, M. Jamali, and M. Ester, "Etf: extended tensor factorization model for personalizing prediction of review helpfulness," in *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 2012, pp. 163–172.

[22] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver, "Tfmap: Optimizing map for top-n context-aware recommendation," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2012, pp. 155–164.

[23] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[24] Z. Lu, D. Agarwal, and I. S. Dhillon, "A spatio-temporal approach to collaborative filtering," in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 13–20.

[25] S. Gultekin and J. Paisley, "A collaborative kalman filter for time-evolving dyadic processes," in *IEEE International Conference onData Mining (ICDM), 2014*. IEEE, 2014, pp. 140–149.

[26] J. Z. Sun, K. R. Varshney, and K. Subbian, "Dynamic matrix factorization: A state space approach," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012*. IEEE, 2012, pp. 1897–1900.

[27] J. Z. Sun, D. Parthasarathy, and K. R. Varshney, "Collaborative kalman filtering for dynamic matrix factorization," *IEEE Transactions on Signal Processing*, vol. 62, no. 14, pp. 3499–3509, 2014.

[28] R. Johansson, "System modeling and identification," 1993.

[29] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[30] "Lasso," http://www.mathworks.com/help/stats/lasso.html.

[31] W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin, "A fast parallel stochastic gradient method for matrix factorization in shared memory systems," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 1, p. 2, 2015.

[32] W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin, "A learning-rate schedule for stochastic gradient methods to matrix factorization," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2015, pp. 442–455.

[33] "Ciao - wikipedia," https://en.wikipedia.org/wiki/Ciao.

[34] "The data source of ciao and epinions datasets," http://www.public.asu.edu/~jtang20/datasetcode/truststudy.htm.
[35] "The data source of flixster datasets," http://www.cs.sfu.ca/~sja25/personal/datasets/.
[36] "The data source of movielens datasets," http://grouplens.org/datasets/movielens/.

Yung-Yin Lo received her B.S and M.S. degree in Computer Science from National Taiwan University, Taiwan in 2013 and 2015, respectively. She is currently a software development engineer at Amazon Japan. Her research interests are focused on social network analysis and recommender systems.

Wanjiun Liao (S'96-M'97-SM'06-F'10) received her Ph.D. degree in Electrical Engineering from the University of Southern California, USA, in 1997. She is a Distinguished Professor of Electrical Engineering Department, National Taiwan University (NTU), Taipei, Taiwan, where she was the Department Chair. She is the Director General of the Engineering and Technologies Department, Ministry of Science and Technology (MOST), Taiwan, and also an Adjunct Research Fellow of the Research Center for Information Technology Innovation, Academia Sinica, Taiwan. Her research is focused on the design and analysis of wireless networking, green communications, cloud networking and network virtualization.

Dr. Liao was an Associate Editor of IEEE Transactions on Wireless Communications and IEEE Transactions on Multimedia, and is on the Steering Committee of the IEEE Transactions on Mobile Computing. She was an IEEE Communications Society (ComSoc) Distinguished Lecturer, IEEE ComSoc Fellow Evaluation Committee, and IEEE Fellow Committee. She helped organize many IEEE conferences, including serving as the symposium Co-Chair of the IEEE GLOBECOM and the IEEE ICC, and the TPC CoChair of the IEEE VTC 2010 Spring and the IEEE PIMRC 2015. She received many awards and recognitions from government and different organizations. She is a Fellow of the IEEE.

Cheng-Shang Chang (S'85-M'86-M'89-SM'93-F'04) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1983, and the M.S. and Ph.D. degrees from Columbia University, New York, NY, USA, in 1986 and 1989, respectively, all in electrical engineering.

From 1989 to 1993, he was employed as a Research Staff Member with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. Since 1993, he has been with the Department of Electrical Engineering, National Tsing Hua University, Taiwan, where he is a Tsing Hua Distinguished Chair Professor. He is the author of the book Performance Guarantees in Communication Networks (Springer, 2000) and the coauthor of the book Principles, Architectures and Mathematical Theory of High Performance Packet Switches (Ministry of Education, R.O.C., 2006). His current research interests are concerned with network science, big data analytics, mathematical modeling of the Internet, and high-speed switching.

Dr. Chang served as an Editor for Operations Research from 1992 to 1999, an Editor for the *IEEE/ACM TRANSACTIONS ON NETWORKING* from 2007 to 2009, and an Editor for the *IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING* from 2014 to 2017. He is currently serving as an Editor-at-Large for the *IEEE/ACM TRANSACTIONS ON NETWORKING*. He is a member of IFIP Working Group 7.3. He received an IBM Outstanding Innovation Award in 1992, an IBM Faculty Partnership Award in 2001, and Outstanding Research Awards from the National Science Council, Taiwan, in 1998, 2000, and 2002, respectively. He also received Outstanding Teaching Awards from both the College of EECS and the university itself in 2003. He was appointed as the first Y. Z. Hsu Scientific Chair Professor in 2002. He received the Merit NSC Research Fellow Award from the National Science Council, R.O.C. in 2011. He also received the Academic Award in 2011 and the National Chair Professorship in 2017 from the Ministry of Education, R.O.C. He is the recipient of the 2017 IEEE INFOCOM Achievement Award.

Ying-Chin Lee received his B.S. degree in electrical engineering from the National Tsing-Hua University, Hsinchu, Taiwan, in 2016. He is currently pursuing the M.S. degree in the Institute of Communications Engineering, National Tsing-Hua University. His research interest is in recommendation systems and deep learning algorithms.