Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-020-02714-4

ORIGINAL RESEARCH

Modeling user preference dynamics with coupled tensor factorization for social media recommendation

Hamidreza Tahmasbi[1] · **Mehrdad Jalali**[2,3] · **Hassan Shakeri**[3]

Received: 23 June 2019 / Accepted: 18 November 2020

Abstract
An essential problem in real-world recommender systems is that user preferences are not static and users are likely to change their preferences over time. Recent studies have shown that the modelling and capturing the dynamics of user preferences lead to significant improvements on recommendation accuracy and, consequently, user satisfaction. In this paper, we develop a framework to capture user preference dynamics in a personalized manner based on the fact that changes in user preferences can vary individually. We also consider the plausible assumption that older user activities should have less influence on a user's current preferences. We introduce an individual time decay factor for each user according to the rate of his preference dynamics to weigh the past user preferences and decrease their importance gradually. We exploit users' demographics as well as the extracted similarities among users over time, aiming to enhance the prior knowledge about user preference dynamics, in addition to the past weighted user preferences in a developed coupled tensor factorization technique to provide top-K recommendations. The experimental results on the two real social media datasets—Last.fm and Movielens—indicate that our proposed model is better and more robust than other competitive methods in terms of recommendation accuracy and is more capable of coping with problems such as cold-start and data sparsity.

Keywords User preference dynamics · Coupled tensor factorization (CTF) · Temporal recommendation · Social recommendation

1 Introduction

Recommender systems have become an important tool for addressing the information overload problem of web users (Shi et al. 2014) by providing personalized recommendations to a user that he might like based on past preferences, interest, or observed behavior about one or various items. An essential problem in real-world recommender systems is that users are likely to change their preferences over time. A user's preference dynamics (UPD) is known in the literature as temporal dynamics (Zhang et al. 2014; Rafailidis et al. 2017) that may be caused by various reasons. According to Rafailidis et al. (2017), Koren (2010) and Lo et al. (2018), the most important of these reasons are: (i) User experiences: the past interaction of users and items make users like some items and dislike some others. For example, if a user is satisfied with the purchase on an auction website then he will probably continue buying from it in future. (ii) New items: the appearance of new items may change the focus of users. For example, users usually like to explore new items over time instead of interacting multiple times with the same items (Rafailidis et al. 2017). (iii) Time frames: different time frames may affect the preferences of users. For instance, a user may tend to watch a family movie on a weekend compared to weekdays when he has to go to work. (iv) Item popularity: popular items may affect user interactions, regardless of a user's preferences or his past behavior. For example, if there is a popular action movie but the user is interested in romantic films, the user may prefer to watch

✉ Mehrdad Jalali
mehrdad.jalali@kit.edu

Hamidreza Tahmasbi
htahma@gmail.com

Hassan Shakeri
shakeri@mshdiau.ac.ir

[1] Department of Computer Engineering, Kashmar Branch, Islamic Azad University, Kashmar, Iran

[2] Institute of Functional Interfaces, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

[3] Department of Computer Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran

Published online: 23 December 2020

this action movie instead. (v) Social influence: friends' preferences may affect a user's decision and change the user preferences over time.

Many traditional recommender systems usually assume that data is static and use historical data without considering the temporal effects in natural user-generated data and time-related phenomena. Recent studies show that the modelling and capturing the dynamics of user preferences lead to significant improvements on recommendation accuracy and, consequently, user satisfaction (Rana and Jain 2015; Cheng et al. 2015). It is also shown that disregarding such changes result in a progressive deterioration in the quality of recommender systems (Matuszyk and Spiliopoulou 2014; Zafari et al. 2019). The adaptability of recommender systems to capture the evolving user preferences (which are constantly changing) is a growing research field (Rana and Jain 2014) that has recently attracted significant attention.

The accurate modeling of the temporal dynamics of user preferences is a crucial challenge in designing efficient personalized recommendation systems (Rana and Jain 2014). To address this issue, many methods have been proposed recently that take into account the effect of time on modeling the dynamics of user preferences over time. However, some of these methods assume that the preference dynamics are homogeneous for all users, whereas in reality, the changes in user preferences may be individual (Rafailidis and Nanopoulos 2016; Wu et al. 2018). Moreover, most of the proposed methods for capturing the user preference dynamics, exploit only a single type of user-item interaction without any side information. These methods suffer from some inherent limitations including cold-start (Barjasteh et al. 2016) as well as data sparsity (Hafshejani et al. 2018) problems and generally perform poorly for users who interacted with a few items. To alleviate these problems in temporal recommendation systems, several methods (Rafailidis and Nanopoulos 2016; Liu et al. 2013; Aravkin et al. 2016; Bao et al. 2013; Yin et al. 2015) have been proposed which commonly exploit the side information such as user profile or trust relations among users, in addition to the interaction data that are usually available (Barjasteh et al. 2016; Lee and Ma 2016). Most of these methods only exploit a single type of side information at a time. The cold-start problem can be tackled by exploiting several types of side information to bridge the gap between existing users (or items) and new users (or items) (Barjasteh et al. 2016). Also, exploiting these additional data can help relieve the sparsity problem (Pan 2016) and thus provide users with better personalized recommendations (Sun et al. 2019).

The recommender systems may include different types of heterogeneous side information. Blending this heterogeneous information is an open problem. Coupled tensor factorization (CTF) (Do and Liu 2016) is an effective scheme to tackle this situation and prove that it is useful to address the cold-start and sparsity problems (Acar et al. 2011b, 2015). In the basic form of CTF, historical users' preferences can be modeled into a sparse tensor whose users, items, and time-periods correspond to its modes (dimensions). Also, the side information can be considered a matrix or a tensor that shares at least a common mode with the main tensor. In this case, the main tensor and additional tensor/matrix are coupled in a common mode. Then, the user preferences dynamics are captured by factorizing these coupled matrices/tensors. One of the state-of-the-art temporal recommendation models based on CTF has been presented by Rafailidis and Nanopoulos (2016) and is called UPD-CTF. In this model, the importance of users' past preferences is weighted based on the UPD rate of each user. The model exploits the weighted users' past preferences and demographics into the CTF scheme.

The need to model the dynamics of user preferences over time in recommender systems poses several essential challenging problems. First of all, based on the intuition that the time change pattern for each user may differ, how can the temporal dimension be incorporated to capture each individual user preference dynamics? Moreover, how can different types of heterogeneous side information be blended to alleviate the cold-start user and sparsity issues? Finally, what is the efficient approach to model the dynamics of user preferences in order to generate more accurate recommendations? To this end, in this paper, we propose a social temporal recommendation model by extending the UPD-CTF method. Our model, in addition to considering which changes in user preferences can vary individually (Rafailidis and Nanopoulos 2016; Wu et al. 2018) and the time change pattern for each user differs, supposes that the importance of users' past preferences decreases according to the rate of user-preference dynamics. Hence, we introduce an appropriate time decay factor for each user according to UPD. In other words, based on the plausible assumption that more recent activities could describe the users' current preferences better (Bao et al. 2013; Su et al. 2015), we decrease the influence of past activities in our model gradually. This is done so that older preferences for users with high preference dynamics rate have less influence on the user's current preferences compared to a user with a low preference dynamics rate. Moreover, based on the fact that user preferences may be influenced by friends' opinions over time (Rafailidis et al. 2017), we extract the similarity information among users as implicit social information from users' interactions with items and exploit it to enhance the prior knowledge about user preference dynamics in each time period which can help alleviate the cold-start and data sparsity problems, in addition to using user demographics. We jointly factorize an incomplete tensor corresponding to user-item interactions over time with an incomplete tensor showing user-user similarities in various time periods as well as an incomplete

matrix corresponding to user demographics, aiming to estimate the missing entries of the first tensor. To this end, we use an extended CTF that optimizes the factorization of each coupled tensors so that the accuracy of none of these is sacrificed. We validate the performance of the proposed model compared with competitive methods with respect to top-K recommendation quality on two public benchmark datasets. The experimental results show that our model is more accurate than others and can help cope with the cold-start user and data sparsity problems.

The rest of this paper is organized as follows. Section 2 describes the preliminaries about coupled tensor factorization. Section 3 summarizes the related works. Section 4 details our proposed temporal recommendation model. Section 5 provides the experimental results. Finally, Sect. 6 presents the conclusions and future research directions.

2 Preliminaries

A tensor is a multidimensional array which is often specified by its number of dimensions, also known as order or mode. It is a generalized concept of vector (first-order tensor) and matrix (second-order tensor). We use boldface Euler script letters, e.g. $\mathcal{X}$ to denote tensors. Matrices are denoted by boldface capital letters, e.g., $\boldsymbol{A}$. Vectors are denoted by boldface lowercase letters, e.g., $\mathbf{a}$. The transpose of matrix $\boldsymbol{A}$ is denoted by $\boldsymbol{A}^{\mathrm{T}}$.

A boldface lower letter with a subscript, is used to denote the ith column of a matrix. For example, $\mathbf{a}_i$ is the ith column of matrix $\boldsymbol{A}$. Entries of a tensor or matrix are denoted by lowercase letters with subscripts. For example, $x_{i,j,k}$ is the element (i,j,k) of a third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$. The tensor $\mathcal{X}$ can be unfolded into matrices through one of the three modes denoted by $\boldsymbol{X}_{(1)} \in \mathbb{R}^{I \times JK}$, $\boldsymbol{X}_{(2)} \in \mathbb{R}^{J \times IK}$ and $\boldsymbol{X}_{(3)} \in \mathbb{R}^{K \times IJ}$ (Wang et al. 2017). We use * and $\odot$ to represent the Hadamard product and Khatri–Rao product of matrices, respectively. The symbol $\circ$ is used to denote the outer product of vectors. We also use $\|\cdot\|$ to denote the Frobenius norm of a tensor and two-norm in the case of matrices. For details about tensor notations and operations, refer to Kolda and Bader (2009).

The CTF is the common scheme which has been widely exploited for joint analysis of heterogeneous data (Acar et al. 2011b). In particular, CTF has proved useful in applications where the goal is estimation of missing data such as recommendation systems (Acar et al. 2015; Frolov and Oseledets 2017; Balasubramaniam et al. 2020). Suppose we have the incomplete third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ coupled with matrix $\boldsymbol{Y} \in \mathbb{R}^{I \times M}$, in the first mode as in Fig. 1. The goal is to find the tensor $\widehat{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$ as an approximation of the tensor $\mathcal{X}$.

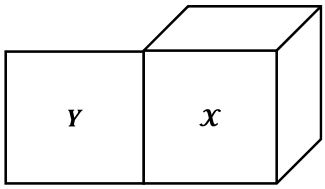

Fig. 1 Third-order tensor $\mathcal{X}$ and matrix $\mathbf{Y}$ coupled in the first mode

According to Acar et al. (2011b), we formulate CTF as an optimization problem, and thus for coupled analysis of tensor $\mathcal{X}$ and matrix $\boldsymbol{Y}$, we define the objective function as

$$f\left(\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\boldsymbol{A}^{(3)},\boldsymbol{V}\right) = \frac{1}{2} \parallel \mathcal{X} - \left[\left[\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\boldsymbol{A}^{(3)}\right]\right] \parallel^2 + \frac{1}{2} \parallel \boldsymbol{Y} - \boldsymbol{A}^{(1)}\boldsymbol{V}^T \parallel^2 \tag{1}$$

where $\boldsymbol{A}^{(1)} \in \mathbb{R}^{I \times R}$, $\boldsymbol{A}^{(2)} \in \mathbb{R}^{J \times R}$ and $\boldsymbol{A}^{(3)} \in \mathbb{R}^{K \times R}$ are the factor matrices of $\mathcal{X}$ that can be extracted through the R-component CANDECOMP/PARAFAC (CP) decomposition model (Kolda and Bader 2009). R denotes the number of factors (rank of decomposition). $\mathbf{A}^{(1)}$ and $\boldsymbol{V} \in \mathbb{R}^{M \times R}$ are factor matrices extracted from $\boldsymbol{Y}$ by performing matrix factorization. Note that $\mathbf{A}^{(1)}$ is the common factor matrix corresponding to the shared mode of $\mathcal{X}$ and $\boldsymbol{Y}$. The notation $\left[\left[\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\boldsymbol{A}^{(3)}\right]\right]$ is used to define tensor $\widehat{\mathcal{X}}$, which can be concisely expressed as follows:

$$\left[\left[\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\boldsymbol{A}^{(3)}\right]\right] = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \mathbf{a}_r^{(3)} \tag{2}$$

where $\mathbf{a}_r^{(1)} \in \mathbb{R}^I$, $\mathbf{a}_r^{(2)} \in \mathbb{R}^J$ and $\mathbf{a}_r^{(3)} \in \mathbb{R}^K$ are the rth column of matrices $\boldsymbol{A}^{(1)}$, $\boldsymbol{A}^{(2)}$, and $\boldsymbol{A}^{(3)}$, respectively, for r = 1,…, R. In case of two matrices ($\mathbf{A}^{(1)}$ and $\boldsymbol{V}$), the Eq. (2) reduces to $\left[\left[\boldsymbol{A}^{(1)},\boldsymbol{V}\right]\right] = \boldsymbol{A}^{(1)}\boldsymbol{V}^T$.

Usually, the columns of factor matrices are normalized to length one (Acar et al. 2011b, 2015) with the weights absorbed into the vector $\lambda \in \mathbb{R}^R$, so that, for example, $\widehat{\mathcal{X}}$ is defined as.

$$\hat{\mathcal{X}} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \mathbf{a}_r^{(3)}. \tag{3}$$

To minimize the objective in (1), we must find optimal matrices $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$, $\mathbf{A}^{(3)}$ and $\mathbf{V}$. Gradient-based optimization algorithms are the most popular techniques to minimize this objective function (Do and Liu 2016).

In the case where tensor $\mathcal{X}$ has missing entries, these are ignored and the model is fitted only to known data entries (Acar et al. 2011b). Accordingly, the objective function (1) is modified as

$$f_{\mathcal{W}}(\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\boldsymbol{A}^{(3)},\boldsymbol{V}) = \frac{1}{2} \parallel \mathcal{W} * \left(\mathcal{X} - [\![\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\boldsymbol{A}^{(3)}]\!]\right) \parallel^2 + \frac{1}{2} \parallel \boldsymbol{Y} - \boldsymbol{A}^{(1)}\boldsymbol{V}^T \parallel^2 \tag{4}$$

where $\mathcal{W} \in \mathbb{R}^{I \times J \times K}$ is an indicator tensor that denotes the missing entries of $\mathcal{X}$ in such a way that each cell $w_{i,j,k}$ of $\mathcal{W}$ is set as follows:

$$w_{i,j,k} = \begin{cases} 1 & \text{if } x_{i,j,k} \text{ is known} \\ 0 & \text{if } x_{i,j,k} \text{ is missing} \end{cases} \tag{5}$$

for all $i \in \{1, \ldots, I\}, j \in \{1, \ldots, J\}$ *and* $k \in \{1, \ldots, K\}$.

3 Related work

Some studies on capturing the dynamics of user preferences in recommender systems are based on the computing user or item neighborhoods (known as neighborhood-based approaches (Liu et al. 2018)). These approaches generally boost recent ratings and penalize older ratings that possibly have less relevance at recommendation time, by employing time windows or a decay function (Vinagre 2012). For instance, in the works of Su et al. (2015) and Liu et al. (2010), they have given more weight to recently rated items and reduced the importance of past rated items gradually in rating prediction using an exponential time decay function. They consider the importance of a specific time period is the same for all users. However, as mentioned before, in reality, the time change pattern for each user may be different (Rafailidis and Nanopoulos 2016). Therefore, the importance of previous time periods varies for each user.

Most of the temporal recommender systems are based on the matrix factorization (MF) scheme (Yin et al. 2014). In this technique, each users and items is characterized by a series of features showing latent factors of the users and items in the system. It decomposes the matrix of users' ratings on items into two low-dimensional matrices, which directly profile users and items to the latent feature space, respectively, and these latent features are later used to make user behavior predictions. One of the first temporal models, namely TimeSVD++ has been proposed by Koren (2010). This model adopts the singular value decomposition (SVD) that is the most basic technique to MF (Yang et al. 2014). TimeSVD++ incorporates time-varying rating biases of each item and user into the MF. It assumes that older ratings are less important in rating prediction. The parameters of this method in different aspects and time periods must be learned individually, so it needs considerable effort for parameter tuning (Lo et al. 2018). A temporal MF (TMF) approach has been proposed by Zhang et al. (2014) that captures the temporal dynamics of user preferences by designing a transition matrix for each user latent feature vectors between two consecutive time periods. Next, this approach is extended to a fully Bayesian treatment called BTMF by introducing priors for the hyperparameters to control the complexity and improve the accuracy of TMF. Another temporal MF to track the temporal dynamics in each of the individual user preferences has been proposed by Lo et al. (2018). The model introduces a modified stochastic gradient descent method to learn the individual user latent vector at each time period by using both the rating logs within the specific time period and overall rating logs. This method learns a linear model to extract the transition pattern for each user's latent feature vector using Lasso regression. The work of Wu et al. (2017) presented a temporal model based on recurrent neural networks (RNNs) (Sherstinsky 2020). This method adopts long short-term memory networks (Sherstinsky 2020) for capturing the dynamics of both users and items. It also incorporates the stationary latent features of users and items extracted by MF into the model. An approach based on multi-task non-negative MF was presented by Ju et al. (2015) that uses a transition matrix to map between latent features of users in two successive time periods in order to track the temporal dynamics of user preferences. The transition matrix used in this method needs to be fixed, while in practice, this matrix is different for each user and each time period. An approach which extends the Gaussian probabilistic matrix factorization to capture user preference dynamics by using a state transition matrix has been proposed by Sun et al. (2014). For learning model parameters from previously available observations, it exploits an expectation–maximization (EM) algorithm, which uses Kalman filter in the EM expectation step. Despite the comprehensiveness of this method, the transition matrix used in it is homogeneous for all users. Moreover, the method is impractical for large datasets due to the run-time performance.

The above-mentioned methods do not exploit any side information and most of them result in limited recommendation accuracy by not handling the cold-start and data sparsity (Rafailidis et al. 2017) problems. A series of studies based on MF exploit the side information to alleviate cold-start and sparsity problems in temporal recommendation systems and thus improve the recommendation performance. A method based on MF has been presented by Wu et al. (2018) that fuses ratings, review texts and item correlation by considering the temporal dynamics of user preferences to improve prediction results. The authors use TimeSVD++ as part of the model to capture temporal dynamics. However, the rating prediction for new users is difficult in this method. Moreover, it assumes that the number of latent factors in ratings is equal to the number of hidden topics in reviews, while, as the authors point out, the number of latent factors is more than the number of hidden topics. Another SVD-based method has been presented by Tong et al. (2019) that integrates rating, trust and time information to model user

preference dynamics. This method includes time-variant biases for each item and each user. However, in this method, the feature vectors of users are not optimized with temporal information.

A temporal collective MF method to generate the recommendations has been proposed by Rafailidis et al. (2017). This work jointly factorizes the multimodal user-item interactions to extract the user temporal pattern. The method introduces a transition matrix of users' preferences between two consecutive user latent feature matrices. Similarly, a dynamic collective MF approach to predict the behavior of users has been presented by Li and Fu (2017), which introduces a transition matrix of users' behaviors. This method models the temporal dynamics between purchase activity and click response behavior of users. It exploits the side information of users and items to alleviate the sparsity problem. The transition matrix used in these two methods is homogeneous for all users; which is a major limitation of them. A framework has been presented by Aravkin et al. (2016) that incorporates trust relations into MF-based dynamic modeling of user preferences. The method defines a transition matrix of users' preferences, assumes that trust relations among users are a graph at each time period, and considers a regularization term for dynamics that can incorporate known trust relations via the graph Laplacian. A method based on social probabilistic MF has been proposed by Bao et al. (2013) which exploits the evolution of user preferences and the social relations to predict user preferences in micro-blogging. In this method, by employing an exponential time decay function, the users' latent features and the topics associated with previous latent features are made. It gives more weight to users' current preferences and decreases the importance of past users' activities gradually in making recommendations. Although the four aforementioned methods exploit side information to make more accurate recommendations, they ignore the personal preferences dynamics and consider that users' preferences change similarly with over time.

Recently, exploitation of the tensor factorization scheme was considered to deal with temporal information because the principle and well-structured approach provided incorporation of the temporal dynamics in recommender systems (Lo et al. 2018). Xiong et al. (2010) proposed a temporal recommendation method based on the Bayesian probabilistic tensor factorization. This method introduces a set of additional time features and adds constraints in the time mode of the tensor to model the evolution of data over time. It uses a fully Bayesian treatment that leads to an almost parameter-free probabilistic tensor factorization approach. Dunlavy et al. (2011) and Spiegel et al. (2011) proposed the temporal link prediction based on tensor factorization. The work of Dunlavy et al. (2011) considers bipartite graphs that evolve over time and gives a CP tensor decomposition approach. It demonstrates that tensor based methods are effective for capturing and exploiting temporal patterns. In the work of Spiegel et al. (2011), the importance of past user preferences using a smoothing factor was reduced. Nevertheless, it gives all user preferences the same weight at a specific time period.

The aforementioned three tensor factorization-based methods do not exploit any side information. Moreover, a major problem of these studies is that they ignore the fact that the time change pattern for each user may be different. Rafailidis and Nanopoulos (2016) proposed a temporal recommendation model based on the CTF called UPD-CTF, where the importance of user past preferences are weighted based on the UPD rate of each user. In this method, the weighted user past preferences are modeled into a third-order tensor with users, items, and time dimensions. The tensor is coupled with a matrix which contains the users' demographic data. Our work is an extension of UPD-CTF. In UPD-CTF, for each user, the importance of all his preferences in previous time periods is considered the same. In contrast, we introduce a decay function and decrease the importance of user past preferences gradually, according to the rate of user-preference dynamics. The UPD-CTF exploits the weighted user past preferences and user demographics into a CTF scheme. However, in our model, in addition to these data, we join the temporal similarity information as an implicit social relation to an extended CTF scheme for mitigating the cold-start user problem and improving the recommendation performance. This is unlike most previous studies on social recommendation that ignore the role of implicit social relation in boosting accuracy of the recommendations whenever explicit social information is not available (Reafee et al. 2016). The extended CTF method that we use optimizes the factorization of each coupled tensors so that the accuracy of none of them is sacrificed.

4 Proposed approach

In this section, we describe our proposed model for predicting users' preferences by considering their preference dynamics. As mentioned, user-item interaction data at a time period is very sparse. Therefore, the dynamic user preference at a time period may be easily underestimated or overestimated. For mitigating the cold-start user and data sparsity problems and enhancing the prior knowledge about dynamic user preferences, we exploit users' demographics and the implicit similarity values between the users in previous and current times, in addition to the historical user-item interaction data. We weigh the historical user-item interactions by applying a new decay function according to user preference dynamics in order to give more importance to

recent interactions. To predict the user preferences, we use an extended CTF method for joint analysis of user-item interactions and side information aimed to improve the top-K recommendations. The proposed model consists of five steps including:

(1) Modeling the historical users' preferences in third-order tensor $\mathcal{X}$ and users' demographic data in matrix $\mathbf{Y}$.
(2) Reconstructing tensor $\mathcal{X}$ by down weighting the entries of tensor $\mathcal{X}$ based on the proposed decay function.
(3) Constructing tensor $\mathcal{Z}$, including the implicit users' similarities, over time from tensor $\mathcal{X}$.
(4) Coupling the tensors $\mathcal{X}$, and $\mathcal{Z}$, and matrix $\mathbf{Y}$ together, and using the extended CTF to generate tensor $\widehat{\mathcal{X}}$ as an approximate of $\mathcal{X}$.
(5) Generating the personalized recommendations for each user based on $\widehat{\mathcal{X}}$.

4.1 Modeling user preferences and user demographic data

We consider the temporal dynamics of user preferences, assuming U, I, and T to be the sets of users, items and time periods, respectively. We store the users' preferences in the third-order tensor including users, items, and time period modes called $\mathcal{X} \in \mathbb{R}^{|U|\times|I|\times|T|}$, where the value of each non-empty cell $x_{u,i,t}$ corresponds to the number of interactions of user u with item i at time period t ($u \in U, i \in I \, and \, t \in T$). The size of the time periods, for example days, months or years depends on the application of the recommender system (Rafailidis and Nanopoulos 2016). The goal is to predict the missing entries of the current/last time period in $\mathcal{X}$, because the time that we have to generate the personalized recommendations is within the current/last time period. Note that there are a few user-item interactions in each time period, and hence $\mathcal{X}$ is a sparse tensor.

Considering the user demographic data, let M be the set of private attributes of users. We construct the matrix $\boldsymbol{Y} \in \mathbb{R}^{|U|\times|M|}$. Similar to Rafailidis and Nanopoulos (2016), we transform the numerical private attributes such as age into bins using equal-width binning, and for every numerical value in $\mathbf{Y}$, the corresponding number of the bin is stored in this matrix. Also, the categorical attributes such as country are transformed into a binary valued vector. By concatenating all the transformed attributes, we create the final |M| different attributes in matrix $\mathbf{Y}$.

Although the user attributes can be dynamic and demographic information may change over time, most of the available datasets such as those we utilized in our experiments provide static user attributes (Rafailidis and Nanopoulos 2016). Hence, we consider that user demographic information is static for all time periods, as in the case of Rafailidis and Nanopoulos (2016).

4.2 User preference dynamics

Discarding old user activities is a simple method to adapt recommender system to changes of user preferences (Su et al. 2015). However, it may lead to sparseness problem. Moreover, normally, user's current preferences are affected by his historical activities (Bao et al. 2013). Nevertheless, these historical activities reflect users' previous preferences and should have less influence on their current preferences (Tang et al. 2015; Cai et al. 2014). In this regard, decaying the weight on old user activities is an appropriate method which is widely used in such applications (Tang et al. 2015; Cai et al. 2014; Su et al. 2015). Based on these intuitions, we suppose the importance of users' previous preferences decrease according to the user preference dynamics and we introduced appropriate personalized time decay factor for each user according to his preference dynamics rate. According to Rafailidis and Nanopoulos (2016), for each user $u \in U$, the user preference dynamics rate, UPD_u is calculated as follows:

$$UPD_u = 1 - \frac{|I^u_{prev} \cap I^u_{cur}|}{|I^u_{prev} \cup I^u_{cur}|} \tag{6}$$

where $I^u_{prev} \subset I$ is the union set of the items that a user u has interacted with at all the previous time periods and $I^u_{cur} \subset I$ indicates the set of items that user u has interacted with in the current/last time period (Rafailidis and Nanopoulos 2016). The high UPD_u values indicate that user u has a great desire to change his preferences in the current time period, whereas low ones mean that user preferences changes are negligible. We construct the exponential decay function $dec_u(t)$ for each user u individually to decrease the influences of each of his interactions in each different (past) time period t ($t = 1, \ldots, |T| - 1$) based on UPD_u, as follows:

$$dec_u(t) = e^{-UPD_u(|T|-t)} \tag{7}$$

where the value of |T|-t is the number of time periods between the tth time period and the current/last time period |T| and parameter UPD_u controls the amount of decay. The lower value of $dec_u(t)$ indicates that user preferences at the time period t have less impact on his current preferences.

By exploiting the proposed decay function in Eq. (7), the weight of each entry $x_{u,i,t}$ in tensor $\mathcal{X}$ for $|T| > 1$ is decreased as follows:

$$x_{u,i,t} = dec_u(t) \cdot x_{u,i,t} \tag{8}$$

where $t = 1, \ldots, |T| - 1$ and $i \in I^u_{prev}$.

We reconstruct the tensor $\mathcal{X}$ based on recalculating the respective entries of $\mathcal{X}$ by exploiting the Eq. (8).

4.3 Calculating user similarities

Based on the fact that friends' preferences may affect a user's decision and change the user preferences over time (Rafailidis et al. 2017), we exploit the similarity relationships between users in each time period, which can help alleviate the cold-start user and data sparsity problems and lead to improved personalized recommendations. Note that social relationships may change over time as well. Therefore, we consider users' similarities over time. The explicit users' similarity information is not available in the benchmark datasets that we used in our experimental evaluation. Hence, we exploit a reforming method to extract the implicit users' similarity from user-item interaction data.

Cosine similarity and Pearson correlation coefficient are the most widely used methods to measure the similarity between users (Wang and Ma 2016). Since we have to measure the similarities based on implicit ratings from user-item interactions, we exploit the Cosine measure. However, if the number of items the two users have both interacted with is small, they will have very high similarities. In our experiments, to alleviate this problem, we decrease the similarity between the two users, if their commonly interested items are less than a certain threshold. Therefore, we introduce a significant weight parameter β as follows:

$$\beta = \begin{cases} \frac{n}{\omega} & n < \omega \\ 1 & otherwise \end{cases} \tag{9}$$

where n denotes the number of items interacted with by both users, and ω is the threshold. Considering β, the similarity between two users u and v in time period t, could be expressed as follows:

$$Sim_{u,v,t} = \beta \frac{\sum_{i \in I_{u,t} \cap I_{v,t}} x_{u,i,t} \cdot x_{v,i,t}}{\sqrt{\sum_{i \in I_{u,t} \cap I_{v,t}} (x_{u,i,t})^2} \cdot \sqrt{\sum_{i \in I_{u,t} \cap I_{v,t}} (x_{v,i,t})^2}} \tag{10}$$

where $x_{u,i,t}$ and $x_{v,i,t}$ denote the number of interactions of user u and user v respectively, with item i in time period t. Also, $I_{u,t}$ and $I_{v,t}$ are the sets of items interacted with by u and v in time period t, respectively.

We obtain user-user similarities in different time periods from tensor $\mathcal{X}$ by applying Eq. (10) and store them in a tensor $\mathcal{Z} \in \mathbb{R}^{|U| \times |U| \times |T|}$, including users, users, and time period modes so that the value of each cell $z_{u,v,t}$ corresponds to the similarity between the two users u and v at the time period t ($u, v \in U$ and $t \in T$).

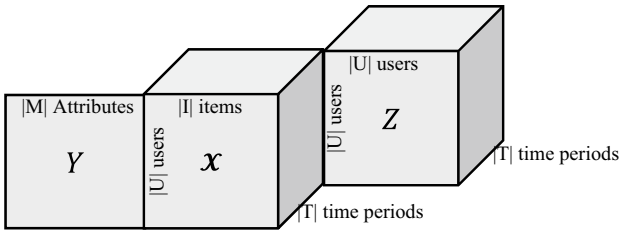

Fig. 2 Third-order tensor $\mathcal{X}$ corresponding to the dynamics of users' preferences over time, coupled with the matrix $\mathbf{Y}$ and third-order tensor $\mathcal{Z}$, corresponding to users' demographic and similarities over time, respectively, in users mode

4.4 Coupled tensor factorization

Aiming to predict the missing entries for the current/last time period in $\mathcal{X}$, we jointly factorize the incomplete tensor $\mathcal{X}$, coupled with additional side information including the sparse matrix $\mathbf{Y}$ showing user demographics as well as tensor $\mathcal{Z}$ showing user-user similarities in different time periods. Figure 2 shows this coupled model. Tensors $\mathcal{X}$ and $\mathcal{Z}$ and matrix $\mathbf{Y}$, share the users mode and are coupled in that mode.

Let $\mathcal{W}_{\mathcal{X}} \in \mathbb{R}^{|U| \times |I| \times |T|}$ and $\mathcal{W}_{\mathcal{Z}} \in \mathbb{R}^{|U| \times |U| \times |T|}$ be tensors indicating the missing values of $\mathcal{X}$ and $\mathcal{Z}$, respectively, such that:

$$w_{x_{u,i,t}} = \begin{cases} 1 & if\ x_{u,i,t}\ is\ known \\ 0 & if\ x_{u,i,t}\ is\ missing \end{cases} \tag{11}$$

$$w_{Z_{u,v,t}} = \begin{cases} 1 & if\ z_{u,v,t}\ is\ known \\ 0 & if\ z_{u,v,t}\ is\ missing \end{cases} \tag{12}$$

for all$u, v \in U, i \in Iandt \in T$. Therefore, we define the objective function of CTF as follows:

$$\begin{aligned} & f_{\mathcal{W}}\left(\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)}, \boldsymbol{V}, \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}\right) \\ & = \frac{\| \mathcal{W}_{\mathcal{X}} * \left(\mathcal{X} - \left[\!\left[\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)}\right]\!\right]\right) \|^2}{2size(\mathcal{X})} \\ & + \frac{\| \boldsymbol{Y} - \boldsymbol{A}^{(1)} \boldsymbol{V}^T \|^2}{2size(\boldsymbol{Y})} \\ & + \frac{\| \mathcal{W}_{\mathcal{Z}} * \left(\mathcal{Z} - \left[\!\left[\boldsymbol{A}^{(1)}, \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}\right]\!\right]\right) \|^2}{2size(\mathcal{Z})} \end{aligned} \tag{13}$$

where $\boldsymbol{A}^{(1)} \in \mathbb{R}^{|U| \times R}$, $\boldsymbol{A}^{(2)} \in \mathbb{R}^{|I| \times R}$ and $\boldsymbol{A}^{(3)} \in \mathbb{R}^{|T| \times R}$ are the factor matrices of $\mathcal{X}$. $\boldsymbol{A}^{(1)}$ and $\boldsymbol{V} \in \mathbb{R}^{|M| \times R}$ are the factor matrices of $\mathbf{Y}$, and $\boldsymbol{A}^{(1)} \in \mathbb{R}^{|U| \times R}$, $\boldsymbol{U}^{(1)} \in \mathbb{R}^{|U| \times R}$, $\boldsymbol{U}^{(2)} \in \mathbb{R}^{|T| \times R}$ are the factor matrices of $\mathcal{Z}$. $size(\mathcal{X})$, $size(\mathbf{Y})$, and $size(\mathcal{Z})$ indicate the number of non-empty entries of $\mathcal{X}$, $\mathbf{Y}$, and $\mathcal{Z}$, respectively. $\mathbf{A}^{(1)}$ is the common factor matrix corresponding to the shared mode of $\mathcal{X}$ and $\mathbf{Y}$, as well as $\mathcal{X}$ and $\mathcal{Z}$. In Eq. (13), inspired by Do and Liu (2016), we

divide the approximation error of $\mathcal{X}$, $\mathbf{Y}$, and $\mathcal{Z}$ by their sizes, respectively, that causes the difference in the sizes of $\mathcal{X}$, $\mathbf{Y}$ and $\mathcal{Z}$ does not have any impact on distribution of their loss to total decomposition's error (Do and Liu 2016), and all three $\mathcal{X}$, $\mathbf{Y}$ and $\mathcal{Z}$ will be optimized simultaneously, without sacrificing the accuracy of one of the three.

In order to use the gradient-based optimization methods to solve this minimization problem, we rewrite the objective function in Eq. (13) as three components, $f_{\mathcal{W}_1}$, f_2 and $f_{\mathcal{W}_3}$:

$$f_{\mathcal{W}}(\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\boldsymbol{A}^{(3)},\boldsymbol{V},\boldsymbol{U}^{(1)},\boldsymbol{U}^{(2)}) = \underbrace{\frac{\| \mathcal{W}_{\mathcal{X}} * (\mathcal{X} - [\![\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\boldsymbol{A}^{(3)}]\!]) \|^2}{2size(\mathcal{X})}}_{f_{\mathcal{W}_1}} + \underbrace{\frac{\| \boldsymbol{Y} - \boldsymbol{A}^{(1)}\boldsymbol{V}^T \|^2}{2size(\boldsymbol{Y})}}_{f_2} + \underbrace{\frac{\| \mathcal{W}_{\mathcal{Z}} * (\mathcal{Z} - [\![\boldsymbol{A}^{(1)},\boldsymbol{U}^{(1)},\boldsymbol{U}^{(2)}]\!]) \|^2}{2size(\mathcal{Z})}}_{f_{\mathcal{W}_3}} \tag{14}$$

Let $\mathcal{P} = [\![\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\boldsymbol{A}^{(3)}]\!]$ and let $\mathcal{Q} = [\![\boldsymbol{A}^{(1)},\boldsymbol{U}^{(1)},\boldsymbol{U}^{(2)}]\!]$. According to Acar et al. (2011a), for i = 1, 2, 3 and for j = 1, 2, the partial derivatives of $f_{\mathcal{W}_1}$ with respect to $\boldsymbol{A}^{(i)}$, $\mathbf{V}$ and $\boldsymbol{U}^{(j)}$ can be taken as

$$\frac{\partial f_{\mathcal{W}_1}}{\partial \boldsymbol{A}^{(i)}} = \frac{(\boldsymbol{W}_{\mathcal{X}_{(i)}} * \boldsymbol{P}_{(i)} - \boldsymbol{W}_{\mathcal{X}_{(i)}} * \boldsymbol{X}_{(i)})\boldsymbol{A}^{(-i)}}{size(\mathcal{X})} \tag{15}$$

$$\frac{\partial f_{\mathcal{W}_1}}{\partial \boldsymbol{V}} = 0 \tag{16}$$

$$\frac{\partial f_{\mathcal{W}_1}}{\partial \boldsymbol{U}^{(j)}} = 0 \tag{17}$$

In Eq. (15), $\boldsymbol{A}^{(-i)} = \begin{cases} \boldsymbol{A}^3 \odot \boldsymbol{A}^2, \text{ if } i = 1 \\ \boldsymbol{A}^3 \odot \boldsymbol{A}^1, \text{ if } i = 2 \\ \boldsymbol{A}^2 \odot \boldsymbol{A}^1, \text{ if } i = 3. \end{cases}$

Similarly, the partial derivatives of f_2 with respect to $\boldsymbol{A}^{(i)}$, $\mathbf{V}$, and $\boldsymbol{U}^{(j)}$ can be taken as

$$\frac{\partial f_2}{\partial \boldsymbol{A}^{(i)}} = \begin{cases} \frac{-\boldsymbol{Y}\boldsymbol{V} + \boldsymbol{A}^{(-i)}\boldsymbol{V}^T\boldsymbol{V}}{size(\boldsymbol{Y})}, \text{ for } i = 1 \\ 0 \text{ for } i \neq 1 \end{cases} \tag{18}$$

$$\frac{\partial f_2}{\partial \boldsymbol{V}} = \frac{-\boldsymbol{Y}^T\boldsymbol{A}^{(1)} + \boldsymbol{V}\boldsymbol{A}^{(1)^T}\boldsymbol{A}^{(1)}}{size(\boldsymbol{Y})} \tag{19}$$

$$\frac{\partial f_2}{\partial \boldsymbol{U}^{(j)}} = 0 \tag{20}$$

Also, the partial derivatives of $f_{\mathcal{W}_3}$ with respect to $\boldsymbol{A}^{(i)}$, $\mathbf{V}$, and $\boldsymbol{U}^{(j)}$ can be taken as

$$\frac{\partial f_{\mathcal{W}_3}}{\partial \boldsymbol{A}^{(i)}} = \begin{cases} \frac{(\boldsymbol{W}_{\mathcal{Z}_{(i)}} * \boldsymbol{Q}_{(i)} - \boldsymbol{W}_{\mathcal{Z}_{(i)}} * \boldsymbol{Z}_{(i)})(\boldsymbol{U}^{(2)} \odot \boldsymbol{U}^{(1)})}{size(\mathcal{Z})}, \text{ for } i = 1 \\ 0 \text{ for } i \neq 1 \end{cases} \tag{21}$$

$$\frac{\partial f_{\mathcal{W}_3}}{\partial \boldsymbol{V}} = 0 \tag{22}$$

$$\frac{\partial f_{\mathcal{W}_3}}{\partial \boldsymbol{U}^{(j)}} = \frac{(\mathcal{W}_{\mathcal{Z}_{(j)}} * \mathcal{Q}_{(j)} - \mathcal{W}_{\mathcal{Z}_{(j)}} * \boldsymbol{Z}_{(j)})U^{(-j)}}{size(\mathcal{Z})} \tag{23}$$

where $\boldsymbol{U}^{(-j)} = \begin{cases} \boldsymbol{U}^{(2)} \odot \boldsymbol{A}^{(1)}, \text{ if } j = 1 \\ \boldsymbol{U}^{(1)} \odot \boldsymbol{A}^{(1)}, \text{ if } j = 2 \end{cases}$

According to Eqs. (15)–(23), we write the partial derivatives of $f_{\mathcal{W}}$ with respect to $\boldsymbol{A}^{(i)}$, $\mathbf{V}$ and $\boldsymbol{U}^{(j)}$ as

$$\frac{\partial f_{\mathcal{W}}}{\partial \boldsymbol{A}^{(i)}} = \begin{cases} \frac{\partial f_{\mathcal{W}_1}}{\partial \boldsymbol{A}^{(i)}} + \frac{\partial f_2}{\partial \boldsymbol{A}^{(i)}} + \frac{\partial f_{\mathcal{W}_3}}{\partial \boldsymbol{A}^{(i)}}, \text{ for } i = 1 \\ \frac{\partial f_{\mathcal{W}_1}}{\partial \boldsymbol{A}^{(i)}} \text{ for } i \neq 1 \end{cases} \tag{24}$$

$$\frac{\partial f_{\mathcal{W}}}{\partial \boldsymbol{V}} = \frac{\partial f_2}{\partial \boldsymbol{V}} \tag{25}$$

$$\frac{\partial f_{\mathcal{W}}}{\partial \boldsymbol{U}^{(j)}} = \frac{\partial f_{\mathcal{W}_3}}{\partial \boldsymbol{U}^{(j)}} \tag{26}$$

Finally, we form the gradient of $f_{\mathcal{W}}$, $\nabla f_{\mathcal{W}}$ by vectorizing the partial derivatives of Eqs. (24)–(26) as a vector of size $R(2|U| + 2|T| + |I| + |M|)$ i.e.,

$$\nabla f_{\mathcal{W}} = \left[vec\left(\frac{\partial f_{\mathcal{W}}}{\partial \boldsymbol{A}^{(1)}}\right) vec\left(\frac{\partial f_{\mathcal{W}}}{\partial \boldsymbol{A}^{(2)}}\right) vec\left(\frac{\partial f_{\mathcal{W}}}{\partial \boldsymbol{A}^{(3)}}\right) vec\left(\frac{\partial f_{\mathcal{W}}}{\partial \boldsymbol{V}}\right) vec\left(\frac{\partial f_{\mathcal{W}}}{\partial \boldsymbol{U}^{(1)}}\right) vec\left(\frac{\partial f_{\mathcal{W}}}{\partial \boldsymbol{U}^{(2)}}\right) \right] \tag{27}$$

where vec (.) is the vectorization operator (Kolda and Bader 2009).

Once we have the objective function of $f_{\mathcal{W}}$ in Eq. (14), and the gradient values, $\nabla f_{\mathcal{W}}$ in Eq. (27), we use the nonlinear conjugate gradient (NCG) (Nocedal et al. 2006) (with Hestenes–Stiefel updates and the More–Thuente line search (Moré and Thuente 1994)) from the Poblano Toolbox (Dunlavy et al. 2010) as a gradient-based optimization method to compute the factor matrices $\boldsymbol{A}^{(1)}$, $\boldsymbol{A}^{(2)}$

and $\boldsymbol{A}^{(3)}$. Then, according to Eq. (3), we calculate tensor $\widehat{\mathcal{X}} = [\![\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)}]\!]$ ($\widehat{\mathcal{X}} \in \mathbb{R}^{|U|\times|I|\times|T|}$) as an approximation of the tensor $\mathcal{X}$.

4.5 Generating personalized recommendations

To generate the final top-K personalized recommendations at the current/last time that is within the last time period |T|, for each user u, we sort the entries within the current/last time period of $\widehat{\mathcal{X}}$ i.e., $\widehat{x}_{u,:,|T|}$ in descending order. Then, the items that correspond to the top-K sorted entries are recommended for user u.

4.6 Complexity analysis

The main computation cost of learning our model is to evaluate the objective function $f_{\mathcal{W}}$ and its partial derivatives with respect to factor matrices. The evaluation of $f_{\mathcal{W}}$ in Eq. (14) involves calculating three components $f_{\mathcal{W}_1}$, f_2 and $f_{\mathcal{W}_3}$ which have time complexities of $O(|U| \cdot |I| \cdot |T| \cdot R)$, $O(|U| \cdot |M| \cdot R)$, and $O(|U|^2 \cdot |T| \cdot R)$, respectively. The time complexities for calculating partial derivatives $\frac{\partial f_{\mathcal{W}_1}}{\partial \boldsymbol{A}^{(i)}}$, $\frac{\partial f_2}{\partial \boldsymbol{A}^{(i)}}$, $\frac{\partial f_2}{\partial \boldsymbol{V}}$, $\frac{\partial f_{\mathcal{W}_3}}{\partial \boldsymbol{A}^{(i)}}$, and $\frac{\partial f_{\mathcal{W}_3}}{\partial \boldsymbol{U}^{(j)}}$ are $O(|U| \cdot |I| \cdot |T| \cdot R)$, $O(|U| \cdot |M| \cdot R + |I| \cdot |T| \cdot R)$, $O(|U| \cdot |M| \cdot R + MR^2)$, $O(|U|^2 \cdot |T| \cdot R)$, and $O(|U|^2 \cdot |T| \cdot R)$, respectively. Therefore, the overall time complexity for each iteration of our model is $O\big(|U| \cdot |I| \cdot |T| \cdot R + |U| \cdot |M| \cdot R + |U|^2 \cdot |T| \cdot R + |I| \cdot |T| \cdot R + MR^2\big)$ which is reduced to $O\big(|U| \cdot |I| \cdot |T| \cdot R + |U| \cdot |M| \cdot R + |U|^2 \cdot |T| \cdot R + MR^2\big)$.

5 Experiments

In this section, we performed our experiments on two datasets with time stamp information including Last.fm and Movielens obtained from social networking websites to validate the performance of our proposed model against competitive methods.

5.1 Datasets

5.1.1 Last.fm dataset

The Last.fm-1 K[1] dataset contains the music listening behavior of |U|= 992 users. This dataset contains 19,150,868 listening events with |I|= 1,766,948 music artists over 54 months. In our experiments, we considered every 6 months as a time period and grouped data into nine time periods (|T|= 9), similar to Rafailidis and Nanopoulos (2016). We extracted the number of times that each user has listened to songs of an artist within a time period as the entry of tensor $\mathcal{X}$.

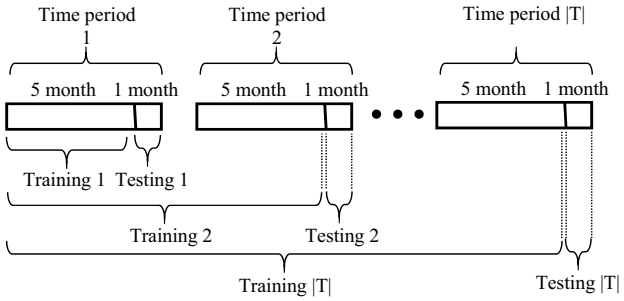

Fig. 3 The dataset splitting for evaluation of methods

This dataset also contains the private attributes for a few users including gender, age, and country. The country and gender have 66 and 2 as different categorical values, respectively. We utilized the transformation method mentioned in Sect. 4.1, and generated |M|= 69 attributes in total, which is 1, 2, and 66 for age, gender, and country attributes, respectively. The proposed model recommends the top-K artists to the user.

5.1.2 Movielens dataset

The Movielens-1 M[2] dataset contains 1,000,209 anonymous ratings, with a scale of 1 to 5, from |U|= 6040 users on 3952 movies over 36 months. We considered every 6 months as a time period and grouped data into six time periods (|T|= 6). Since a user rarely watches the same movie several times, instead of movie recommendations, we performed movie-genre recommendations. Thus, the proposed model recommends the top-K movie-genres to the user. There are |I|= 18 movie-genres and each movie belongs to one or more genres. It is assumed that each user has rated a movie after watching it (Rafailidis and Nanopoulos 2016) and we extracted the number of times that each user has watched a certain movie-genre within a time period as entry of tensor $\mathcal{X}$.

This dataset also contains user attributes including gender, age and 21 categorical occupation types. Similar to Last.fm, we used the transformation method and generated |M|= 24 attributes in total, which are 1, 2, and 21 for age, gender and occupation attributes, respectively.

[1] http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html.

[2] https://grouplens.org/datasets/movielens/.

5.2 Experimental settings

As shown in Fig. 3, we considered the last (sixth) month of each time period as the test set and used the first five months of this time period and all the months of previous time periods as the training set. Therefore, we had nine and six different training/test cases for Last.fm and Movielens, respectively. This validation method is also defined and used in (Rafailidis and Nanopoulos 2016).

The proposed model was validated on Last.fm by predicting the top-K artists a user is likely to listen to during the test month, and in Movielens by predicting the top-K movie-genres that a user is likely to watch at the test month. In our experiments, we set the maximum number of K to 100 and three for Last.fm and Movielens, respectively. Because in a test month, each user does not listen to more than 100 different artists in Last.fm dataset and each user does not watch more than three different movie genres in the Movielens dataset (Rafailidis and Nanopoulos 2016).

The metrics we adopted to measure the recommendations quality are recall, precision and F1-measure that are widely used for the top-K recommendation strategy evaluation. For a test user u, recall is the fraction of relevant items that are in the top-K list of recommended items for u, among the total number of his relevant items that should have been recommended. A high recall indicates the user's adoption. In addition, for test user u, precision is the fraction of the relevant items in top-K list of recommended items for u. We report the recall and precision that are obtained by averaging the recall values and precision values over all users with at least one relevant item in test month. Recall and precision measures assess diverging properties, and if more items are recommended, the recall will increase, but precision will decrease. The F1-measure finds a suitable trade-off between recall and precision, and it is calculated as follows:

$$F1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{28}$$

A higher F1-measure corresponds to better top-K recommendation performance. We compared the recommendation results of the following methods in our work:

(1) DeepRec (Zhang et al. 2019): This is a state-of-the-art recommender model that is based on deep neural network learning (Mu 2018) with item embedding and weighted loss function. It does not consider the temporal dynamics.
(2) TimeSVD++ (Koren 2010): This method is a baseline for modeling the user preference dynamics. It incorporates the time-variant biases of each user and item into the MF and generates the recommendations. TimeSVD++ exploits only the user-item interactions without any side information.
(3) BTMF (Zhang et al. 2014): This is a Bayesian temporal MF approach that captures the temporal dynamics of user preferences by learning a transition matrix for each user latent feature vectors between two consecutive time periods.
(4) Recurrent recommender networks (RRN) (Wu et al. 2017): This is a deep learning method based on RNNs that fuses a long short-term memory model with MF for capturing the dynamics of both users and items.
(5) TF (Dunlavy et al. 2011): This method only considers the user-item interactions in different time periods as a third-order tensor and factorizes this tensor, aiming to recommend items by taking into account user preference dynamics.
(6) UPD-CTF (Rafailidis and Nanopoulos 2016): This method, models user preference dynamics based on UPD by incorporating the weighted user-item interactions as well as user demographics into the CTF scheme.
(7) EUPD-CTF1 (Extended UPD-CTF1): It is our model proposed in Sect. 4 (as an extension of UPD-CTF) except that only exploits both user-item interactions in different time periods and user demographics based on the CTF scheme.
(8) EUPD-CTF2 (Extended UPD-CTF2): It is our model proposed in Sect. 4 (as an extension of UPD-CTF) that in addition to user-item interactions in different time periods and user demographics, also exploits the temporal similarity information between users into CTF scheme.

The EUPD-CTF1, which does not consider the similarity among over time in comparison with EUPD-CTF2, is used to examine the effect of using similarity in the proposed model. Since the TimeSVD++ was originally designed for rating prediction task, not top-K recommendations, we modified this algorithm to use the same top-K strategy as the other comparison algorithms.

The optimal parameters for each method are determined either by our experiments or suggested by their corresponding references. We set parameters of UPD-CTF according to the respective reference. For making a fair comparison, we fix the number of factors R to be 20 in all comparison methods. Also, for simplicity, we used $\lambda_r = 1$, for r = 1,2,…,R in CP decomposition. The number of features in item embedding for Last.fm and Movielens is set to 200 and 150, respectively in DeepRec. We set learning rate $\eta = 0.001$, regularization terms for the user vectors $\lambda_U = 0.01$ and regularization terms for the item vectors $\lambda_V = 0.01$, with default setting for other internal parameters in TimeSVD++, and set $\upsilon_0 = R$, $\mu_0 = 0$, $\beta_0 = 2$, $W_0 = Z_0 = \mathrm{I}$ (I is the identity

Table 1 Recall of comparative methods on testing all users for Last.fm

Method	Top5	Top10	Top15	Top20	Top50	Top100
DeepRec	0.07309	0.0773	0.0832	0.0921	0.1271	0.1971
p	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
TimeSVD++	0.0818	0.09914	0.1123	0.1296	0.1561	0.2108
p	0.0000	0.0000	0.0000	0.0000	0.0000	0.0098
BTMF	0.1096	0.1181	0.147	0.1903	0.2317	0.2614
p	0.0000	0.0000	0.0100	0.0000	0.0109	0.0032
RRN	0.0931	0.1108	0.1259	0.151	0.1935	0.2293
p	0.0000	0.0000	0.0131	0.0000	0.0011	0.0000
TF	0.0424	0.0521	0.0602	0.0734	0.0935	0.1437
p	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
UPD-CTF	0.1514	0.1839	0.1907	0.2114	0.2975	0.3403
p	0.0000	0.0107	0.0183	0.0290	0.0144	0.0215
EUPD-CTF1	0.1703	0.1961	0.2098	0.2204	0.3085	0.3564
p	0.0000	0.0212	0.0307	0.0116	0.0217	0.0291
EUPD-CTF2	**0.1814**	**0.2101**	**0.2135**	**0.2583**	**0.3218**	**0.3989**

Table 2 Precision of comparative methods on testing all users for Last.fm

Method	Top5	Top10	Top15	Top20	Top50	Top100
DeepRec	0.1827	0.1711	0.1636	0.1509	0.1188	0.0729
p	0.0000	0.0000	0.0000	0.0000	0.0000	0.0019
TimeSVD++	0.2128	0.1907	0.1798	0.1639	0.1276	0.0884
p	0.0000	0.0000	0.0059	0.0000	0.0194	0.0314
BTMF	0.2436	0.2111	0.2104	0.1986	0.1707	0.0912
p	0.0000	0.0000	0.0136	0.0076	0.0244	0.0160
RRN	0.2203	0.2086	0.2006	0.1842	0.1398	0.092
p	0.0000	0.0000	0.0118	0.0000	0.0199	0.0171
TF	0.1396	0.1308	0.1259	0.1132	0.094	0.0591
p	0.0000	0.0000	0.0000	0.0000	0.0000	0.0015
UPD-CTF	0.2735	0.2598	0.2471	0.2311	0.1631	0.0951
p	0.0121	0.0430	0.0351	0.0317	0.0392	0.0205
EUPD-CTF1	0.2859	0.2708	0.2592	0.2467	0.1854	0.0979
p	0.0000	0.0004	0.0000	0.0001	0.0043	0.0000
EUPD-CTF2	**0.3225**	**0.3126**	**0.2954**	**0.2799**	**0.2037**	**0.1017**

matrix) in BTMF. We set $\omega = 50$ for Last.fm and three for Movielens in our proposed methods as they provided good results in comparison with other tested values. As stopping conditions for NCG, we set the maximum number of iterations and function evaluations to 10^4 and 10^5, respectively. For other parameters that control the termination of NCG, we used the default values in the Poblano Toolbox.

All the experiments were conducted by using MATLAB 2016a on Windows 10 PC with Intel Core i5 2.53 GHz with 8 GB memory.

5.3 Experimental results

We performed the experiments on Last.fm and MovieLens datasets to evaluate the performance of our two proposed approaches compared against competitive methods via different values of K. We conducted the statistical significance tests (paired t-tests with the significant level of 0.05) between the results of the proposed EUPD-CTF2 and the other methods. The results demonstrate the significance of the difference between the proposed EUPD-CTF2 and the other methods in terms of recall, precision, and F1.

5.3.1 Validation on all users

Performance of the methods compared in terms of recall, precision and F1 for different values of K on the Last.fm dataset is shown in Tables 1, 2, 3. The boldface numbers in tables highlight the best results in each metric. The results also are presented in Fig. 4.

Table 3 F1 value of comparative methods on testing all users for Last.fm

Method	Top5	Top10	Top15	Top20	Top50	Top100
DeepRec	0.1044	0.1065	0.1103	0.1144	0.1228	0.1064
p	0.0000	0.0000	0.0000	0.0000	0.0000	0.0023
TimeSVD++	0.1182	0.1305	0.1383	0.1447	0.1404	0.1246
p	0.0000	0.0000	0.0230	0.0014	0.0307	0.0073
BTMF	0.1512	0.1515	0.1731	0.1944	0.1966	0.1352
p	0.0000	0.0011	0.0000	0.0193	0.0037	0.0042
RRN	0.1309	0.1447	0.1547	0.166	0.1623	0.1313
p	0.0000	0.0000	0.0000	0.0180	0.0000	0.0000
TF	0.065	0.0745	0.0815	0.0891	0.0937	0.0838
p	0.0000	0.0000	0.0000	0.0000	0.0000	0.0237
UPD-CTF	0.1949	0.2154	0.2153	0.2208	0.2107	0.1487
p	0.0023	0.0000	0.0174	0.0470	0.0251	0.0091
EUPD-CTF1	0.2135	0.2275	0.2319	0.2328	0.2316	0.1536
p	0.0000	0.0000	0.0210	0.0315	0.0009	0.0401
EUPD-CTF2	**0.2322**	**0.2513**	**0.2479**	**0.2687**	**0.2495**	**0.1621**

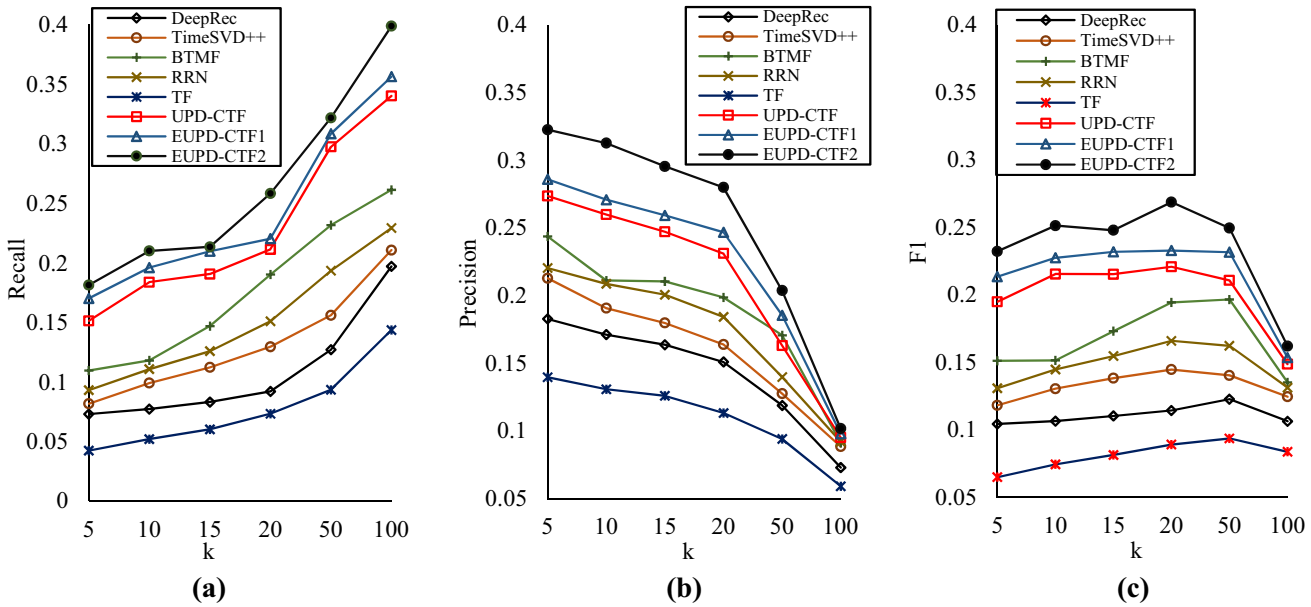

Fig. 4 The performance of comparative methods in terms of **a** recall, **b** precision and **c** F1, on testing all users for Last.fm

As it is shown in Table 1 and Fig. 4a, EUPD-CTF2 performs better in terms of recall than the other compared methods at different Ks for the Last.fm dataset. Tables 2 and 3 and Fig. 4b, c show that EUPD-CTF2 also outperforms the competitive methods in terms of precision and F1 with different Ks for Last.fm. In addition, based on the results in Tables 1, 2, 3 and Fig. 4, it can be observed that the proposed EUPD-CTF1 method has the second-best performance in terms of all metrics for the Last.fm. The p-values of the t-test in Tables 1, 2, 3 demonstrate that EUPD-CTF2 obtains statistically significantly better performance in terms of recall, precision and F1 than the other methods on Last.fm.

Tables 4, 5, 6 respectively show the recall, precision and F1 obtained by compared methods with different values of K for the Movielens dataset. Also Fig. 5 shows the performance of the methods compared in terms of these metrics for Movielens. As it can be observed, the proposed EUPD-CTF2 has the best performance in terms of recall, precision and F1 among the compared methods at different Ks for Movielens. In addition, EUPD-CTF1 has the second-best performance except for precision in Top3 (K = 3) on this dataset. The p values in Tables 4, 5, 6 show that the proposed EUPD-CTF2 has significantly better performance on all metrics than the other methods on Movielens.

Table 4 Recall of comparative methods on testing all users for Movielens

Method	Top1	Top2	Top3
DeepRec	0.1604	0.2537	0.2928
p	0.0000	0.0000	0.0000
TimeSVD++	0.1736	0.2549	0.3037
p	0.0021	0.0039	0.0114
BTMF	0.2211	0.3209	0.3514
p	0.0000	0.0000	0.0238
RRN	0.1794	0.2714	0.3354
p	0.0000	0.0000	0.0000
TF	0.1474	0.2496	0.3117
p	0.0000	0.0000	0.0000
UPD-CTF	0.2472	0.3194	0.3699
p	0.0254	0.0240	0.0107
EUPD-CTF1	0.2592	0.3305	0.3762
p	0.0000	0.0060	0.0001
EUPD-CTF2	**0.2713**	**0.3565**	**0.41**

Table 5 Precision of comparative methods on testing all users for Movielens

Method	Top1	Top2	Top3
DeepRec	0.8025	0.7357	0.6954
p	0.0000	0.0000	0.0000
TimeSVD++	0.8113	0.7642	0.7211
p	0.0000	0.0012	0.0000
BTMF	0.8413	0.8025	0.7504
p	0.0000	0.0249	0.0374
RRN	0.8235	0.7803	0.7315
p	0.0000	0.0000	0.0093
TF	0.0000	0.0000	0.0000
p	0.0091	0.0004	0.0136
UPD-CTF	0.8594	0.7905	0.731
p	0.0091	0.0004	0.0136
EUPD-CTF1	0.8837	0.8219	0.7374
p	0.0176	0.0112	0.0039
EUPD-CTF2	**0.91**	**0.8512**	**0.7695**

Table 6 F1 value of comparative methods on testing all users for Movielens

Method	Top1	Top2	Top3
DeepRec	0.2674	0.3773	0.4121
p	0.0000	0.0085	0.0000
TimeSVD++	0.286	0.3823	0.4274
p	0.0000	0.0106	0.0058
BTMF	0.3502	0.4585	0.4787
p	0.0115	0.0370	0.0219
RRN	0.2946	0.4027	0.4599
p	0.0000	0.0000	0.0000
TF	0.2474	0.3678	0.4264
p	0.0000	0.0000	0.0000
UPD-CTF	0.384	0.455	0.4912
p	0.0314	0.0101	0.1920
EUPD-CTF1	0.4008	0.4714	0.4982
p	0.0023	0.0174	0.0098
EUPD-CTF2	**0.418**	**0.5025**	**0.535**

The superiority of EUPD-CTF1 compared to UPD-CTF in both datasets means that considering the proposed personalized time decay factor based on UPD for each user to capture user preference dynamics can improve the quality of recommendations. Although the accuracy of EUPD-CTF1 is very close to UPD-CTF and the relative improvements are small, even small improvements may lead to significant improvements in the quality of recommendations in practice (Guo et al. 2016).

The proposed EUPD-CTF2 method achieves better results than EUPD-CTF1 in both datasets, which indicates that incorporating the temporal similarity information between users in our proposed model leads to better improvement in the recommendation accuracy. The experimental results demonstrate that the proposed EUPD-CTF2 method consistently and significantly outperforms other competitive methods on all metrics in Last.fm and MovieLens. Especially, in comparison with UPD-CTF, which is the basis of the proposed EUPD-CTF2, the results of t-test on these two methods indicate that the EUPD-CTF2 performs better significantly.

The main different between UPD-CTF and EUPD-CTF2 is that in UPD-CTF, the user past preferences are weighted based on UPD. While EUPD-CTF2 exploits a decay function and decreases the importance of user past preferences gradually, according to the UPD. EUPD-CTF2 in addition to the past user preferences and user demographics, also utilizes the temporal users' similarity into the developed CTF scheme. The experimental results show that these designs allow EUPD-CTF2 to capture the temporal dynamics of user preferences better, thus, boosting the recommendation performance.

From Figs. 4 and 5, it can be observed that the deep learning-based method RRN outperforms the temporal methods TimeSVD++ and TF, while it performs worse than EUPD-CTF2, EUPD-CTF1, EUPD-CTF, and BTMF. This finding confirms that although using recurrent neural networks in recommender systems can help to capture the temporal dynamics of user preferences, this type of models still requires significant improvements in recommendation accuracy (Batmaz et al. 2018). The results also show that in both datasets, the deep learning-based method DeepRec performs worse than other methods except TF. This is because

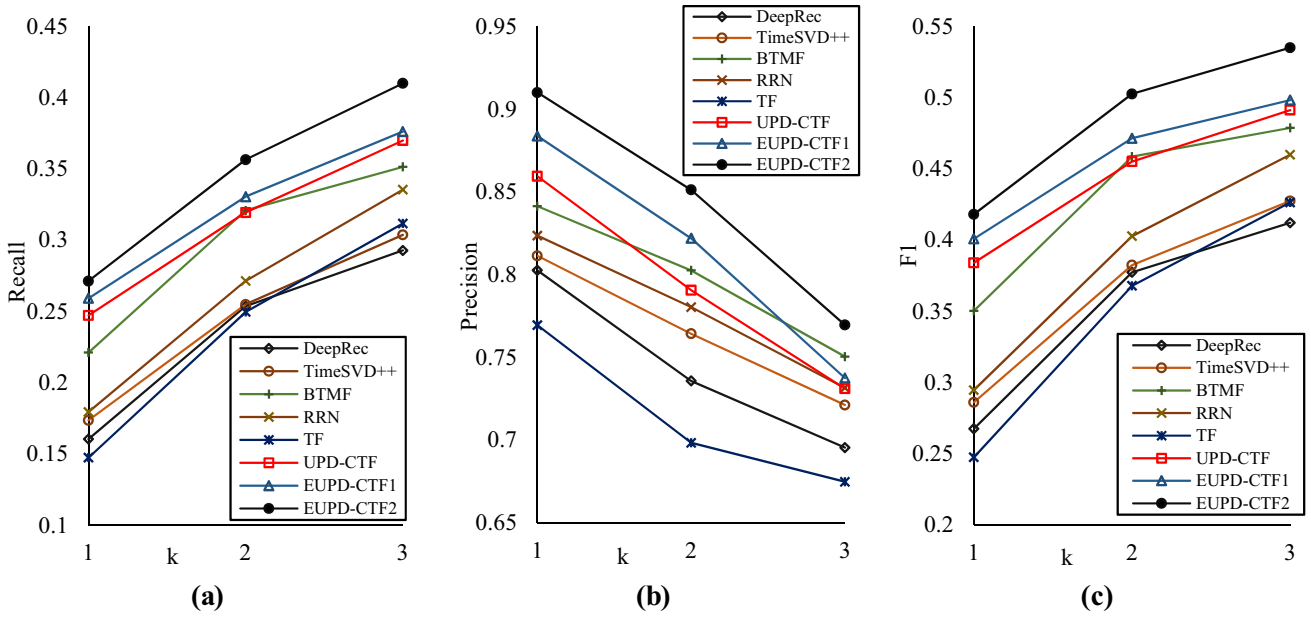

Fig. 5 The performance of comparative methods in terms of **a** recall, **b** precision and **c** F1, on testing all users for Movielens

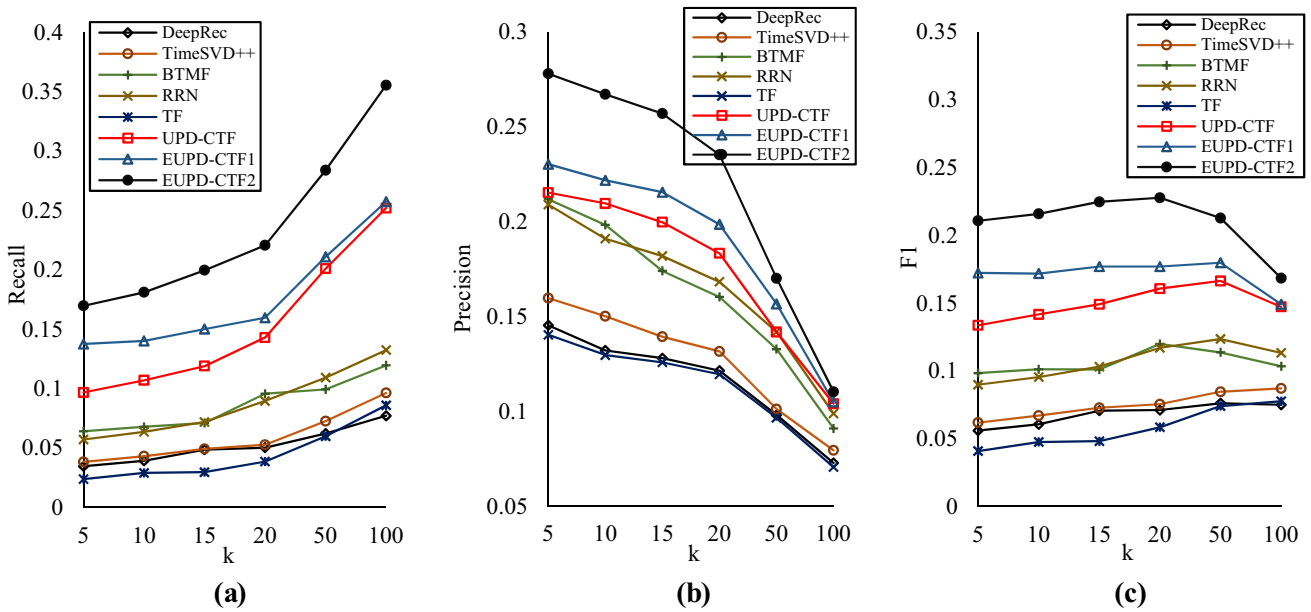

Fig. 6 The performance of comparative methods in terms of **a** recall, **b** precision and **c** F1, on testing cold-start users for Last.fm

DeepRec does not consider the temporal dynamics of user preference into the model.

5.3.2 Validation on cold-start users

We evaluated the capability of our proposed model to cope with the cold-start user problem. In this scenario, we only cared about the recommendation accuracy for users who interact with up to five items in the training set. The performance of the compared methods in terms of recall, precision and F1 for different values of K on the Last.fm dataset is shown in Tables 7, 8, 9. The results also are presented in Fig. 6. The results show that the EUPD-CTF2 has the best performance once again in terms of all metrics in all cases. Specially, as shown in Table 7 and Fig. 6, the recall obtained by EUPD-CTF2 is significantly higher than the other methods. The EUPD-CTF1 has the second-best performance on all metrics.

Table 10 Recall of comparative methods on testing cold-start users for Movielens

Method	Top1	Top2	Top3
DeepRec	0.1476	0.1659	0.1792
p	0.0000	0.0000	0.0000
TimeSVD++	0.1569	0.1736	0.1841
p	0.0000	0.0023	0.0000
BTMF	0.1601	0.1878	0.2103
p	0.0000	0.0119	0.0094
RRN	0.1656	0.1922	0.2194
p	0.0000	0.0000	0.0211
TF	0.1398	0.1504	0.1591
p	0.0000	0.0000	0.0000
UPD-CTF	0.2103	0.2307	0.2462
p	0.0000	0.0208	0.0115
EUPD-CTF1	0.2259	0.2411	0.2496
p	0.0128	0.0075	0.0381
EUPD-CTF2	**0.2321**	**0.2704**	**0.2896**

Table 11 Precision of comparative methods on testing cold-start users for Movielens

Method	Top1	Top2	Top3
DeepRec	0.5121	0.4983	0.4762
	0.0109	0.0058	0.0000
TimeSVD++	0.5196	0.5113	0.5046
p	0.0000	0.0000	0.0000
BTMF	0.5811	0.5688	0.5494
p	0.0012	0.0109	0.0253
RRN	0.5914	0.5720	0.5691
p	0.0000	0.0137	0.0291
TF	0.4955	0.4711	0.4697
p	0.0000	0.0114	0.0000
UPD-CTF	0.6914	0.6727	0.6541
p	0.0248	0.0198	0.0209
EUPD-CTF1	0.7007	0.6814	0.6555
p	0.0095	0.0171	0.0244
EUPD-CTF2	**0.7197**	**0.7091**	**0.6998**

In addition, Tables 10, 11, 12 and Fig. 7 show that EUPD-CTF2 has the best performance in terms of recall, precision and F1 for different values of K on the Movielens. The method of EUPD-CTF1 has the second best performance, but is very close to UPD-CTF. The p values in Tables 7, 8, 9, 10, 11, 12 indicate that EUPD-CTF2 obtains significantly better performance on all metrics than the other methods in both datasets. These observations demonstrate that our EUPD-CTF2 model can mitigate the cold start problem better than other competitive methods.

Table 12 F1 value of comparative methods on testing cold-start users for Movielens

Method	Top1	Top2	Top3
DeepRec	0.2292	0.2489	0.2604
	0.0000	0.0000	0.0135
TimeSVD++	0.241	0.2592	0.2698
p	0.0000	0.0000	0.0256
BTMF	0.251	0.2824	0.3042
p	0.0115	0.0341	0.0319
RRN	0.2587	0.2877	0.3167
p	0.0093	0.0117	0.0208
TF	0.2181	0.228	0.2377
p	0.0000	0.0000	0.0000
UPD-CTF	0.3225	0.3436	0.3577
p	0.0000	0.0223	0.0291
EUPD-CTF1	0.3417	0.3562	0.3615
p	0.0252	0.0109	0.0214
EUPD-CTF2	**0.351**	**0.3915**	**0.4097**

Except for our proposed EUPD-CTF2, the obtained results demonstrate that our proposed EUPD-CTF1, which does not exploit user-user similarities in comparison with EUPD-CTF2, performs better than other competitive methods to mitigate the cold-start user problem in both datasets. On the other hand, from the superiority of the EUPD-CTF2 compared to EUPD-CTF1, we can find that considering the temporal users' similarity in EUPD-CTF2 lead to better cope with the cold-start user problem and provides that social information is very effective in improving recommendation accuracy once again. The results also show that unlike the first scenario (validation on all users), in this scenario, RRN performs almost better than BTMF. In other words, RRN works better than BTMF in dealing with cold-start users.

5.4 Validation on data sparsity

Inspired by the works of Huang et al. (2004), Hafshejani et al. (2018), Forsati and Mahdavi (2014), and Reafee et al. (2016), to evaluate the performance of our proposed model against different levels of data sparsity, we used different amounts of training data (100%, 90%, 80%, 70%, 60%). Training data 60%, for example, means we randomly eliminated 40% of user-item interactions from each original training set. The performance of the compared methods in terms of recall, precision and F1 for $K = 100$ on the Last.fm and $K = 3$ on the Movielens is shown in Figs. 8 and 9. Similar results were obtained for other values of K. As it can be observed, decreasing the number of training data leads to a decrease in recommendation performance in all methods. However, when the number of training data decreases, the performance of CTF-based methods (i.e., UPD-CTF,

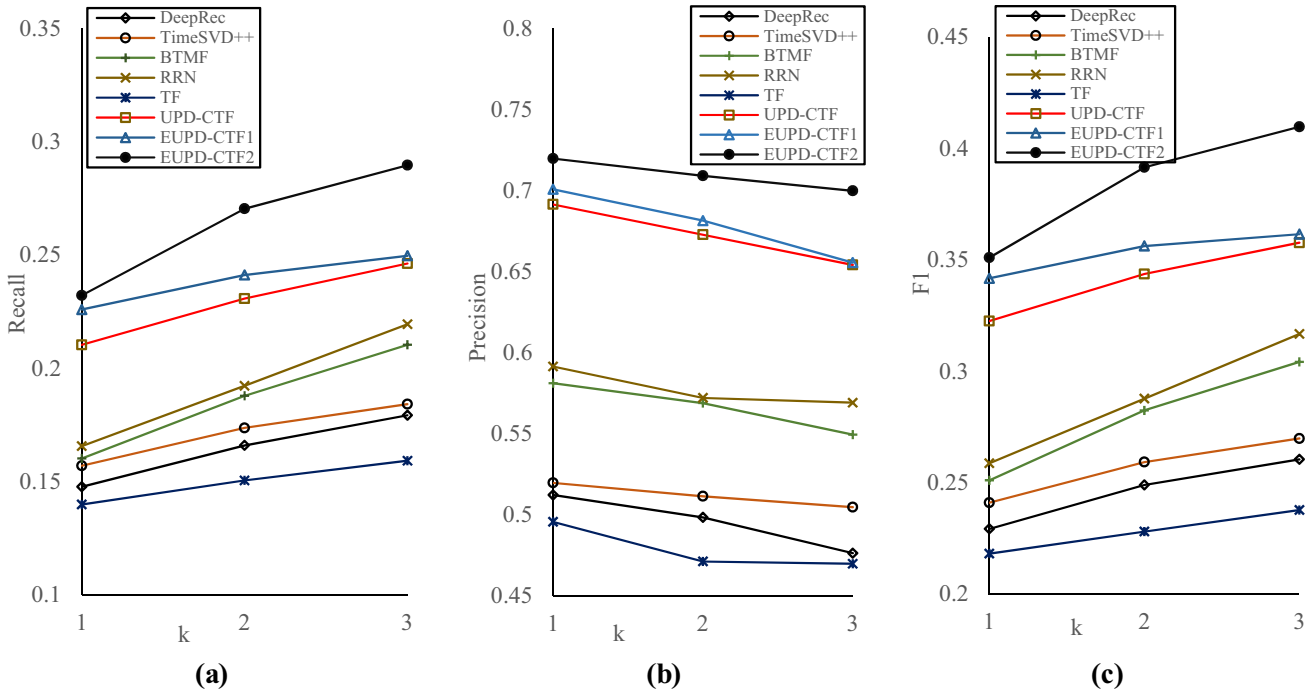

Fig. 7 The performance of comparative methods in terms of **a** recall, **b** precision and **c** F1, on testing cold-start users for Movielens

Table 7 Recall of comparative methods on testing cold-start users for Last.fm

Method	Top5	Top10	Top15	Top20	Top50	Top100
DeepRec	0.0347	0.0393	0.0487	0.0503	0.0622	0.0772
p	0.0000	0.0000	0.0000	0.0093	0.0029	0.0016
TimeSVD++	0.0383	0.0431	0.0493	0.0528	0.0726	0.0964
p	0.0000	0.0000	0.0000	0.0000	0.0014	0.0107
BTMF	0.0641	0.0679	0.0711	0.0957	0.0994	0.1196
p	0.0000	0.0000	0.0000	0.0145	0.0320	0.0318
RRN	0.0572	0.0636	0.0719	0.0896	0.1093	0.1325
p	0.0000	0.0000	0.0000	0.0122	0.0096	0.0029
TF	0.0239	0.0291	0.0297	0.0386	0.0599	0.086
p	0.0000	0.0000	0.0101	0.0074	0.0000	0.0000
UPD-CTF	0.0968	0.107	0.119	0.143	0.201	0.252
p	0.0000	0.0311	0.0258	0.0185	0.0058	0.0120
EUPD-CTF1	0.1376	0.1402	0.1501	0.1596	0.2109	0.2574
p	0.0012	0.0214	0.0025	0.0107	0.0210	0.0192
EUPD-CTF2	**0.1697**	**0.1811**	**0.1997**	**0.2206**	**0.2838**	**0.3554**

EUPD-CTF1, and EUPD-CTF2) slightly decreases, whereas the performance of other compared methods dramatically decreases. This indicates that CTF-Based methods, which jointly analyze heterogeneous information, can help relieve the data sparsity problem. The results show that the proposed EUPD-CTF2 consistently outperforms the other methods in all cases. In addition, with a decreasing number of training data, the performance of EUPD-CTF2 decreases less compared to UPD-CTF and EUPD-CTF1. These observations demonstrate that the proposed EUPD-CTF2 can better alleviate the data sparsity problem. The superiority of EUPD-CTF2 compared to EUPD-CTF1 for all the different training set sizes means that incorporating the temporal users' similarity into the model is effective in alleviating the data sparsity problem in recommender systems.

Table 8 Precision of comparative methods on testing cold-start users for Last.fm

Method	Top5	Top10	Top15	Top20	Top50	Top100
DeepRec	0.1454	0.1322	0.1281	0.1215	0.0981	0.0729
p	0.0000	0.0000	0.0000	0.0000	0.0029	0.0104
TimeSVD++	0.1598	0.1502	0.1394	0.1317	0.1013	0.0795
p	0.0000	0.0000	0.0000	0.0000	0.0230	0.0000
BTMF	0.2119	0.1984	0.1741	0.1604	0.1329	0.0911
p	0.0000	0.0000	0.0000	0.0000	0.0149	0.0068
RRN	0.2091	0.1911	0.1820	0.1684	0.1419	0.0989
p	0.0000	0.0000	0.0000	0.0154	0.0109	0.0000
TF	0.1404	0.1297	0.1259	0.1196	0.0967	0.0707
p	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
UPD-CTF	0.2154	0.2097	0.1998	0.1834	0.1419	0.1041
p	0.0109	0.0411	0.0186	0.0095	0.0174	0.0403
EUPD-CTF1	0.2304	0.2219	0.2156	0.1987	0.1567	0.105
p	0.0236	0.0102	0.0201	0.0371	0.0163	0.0473
EUPD-CTF2	**0.2781**	**0.2672**	**0.2571**	**0.2354**	**0.1702**	**0.1104**

Table 9 F1 value of comparative methods on testing cold-start users for Last.fm

Method	Top5	Top10	Top15	Top20	Top50	Top100
DeepRec	0.056	0.0606	0.0706	0.0711	0.0761	0.075
p	0.0000	0.0000	0.0000	0.0119	0.0000	0.0093
TimeSVD++	0.0618	0.067	0.0728	0.0754	0.0846	0.0871
p	0.0000	0.0000	0.0000	0.0162	0.0023	0.0299
BTMF	0.0984	0.1012	0.101	0.1199	0.1137	0.1034
p	0.0000	0.0000	0.0000	0.0000	0.0117	0.0120
RRN	0.0898	0.0954	0.1031	0.117	0.1235	0.1133
p	0.0000	0.0000	0.0000	0.0000	0.0091	0.0104
TF	0.0408	0.0475	0.0481	0.0584	0.074	0.0776
p	0.0000	0.0000	0.0000	0.0312	0.0000	0.0000
UPD-CTF	0.1336	0.1417	0.1492	0.1607	0.1664	0.1473
p	0.0000	0.0036	0.0104	0.0029	0.0209	0.0197
EUPD-CTF1	0.1723	0.1718	0.177	0.177	0.1798	0.1492
p	0.0139	0.0212	0.0410	0.0379	0.0431	0.0079
EUPD-CTF2	**0.2108**	**0.2159**	**0.2248**	**0.2278**	**0.2128**	**0.1685**

6 Conclusion

User preferences in real-world recommender systems change over time. Modeling the user preferences dynamics lead to significant improvements on accuracy of personalized recommendation systems. Accurate modeling of the user preferences dynamics is a crucial challenge in designing efficient recommendation systems. In this paper, we developed a state-of-the-art method to capture the temporal dynamics of user preferences in a personalized manner based on a proposed weighting scheme. We introduced a personalized time decay factor for each user according to the rate of his preference dynamics and exploited the extracted similarities among users over time in addition to the historical user-item interaction data and user demographics in a developed coupled tensor factorization framework to generate personalized top-K recommendations. Evaluation of the results on two real-world social media datasets indicates the superiority of our proposed model over competitive methods for the social temporal recommendation. We can also conclude that our

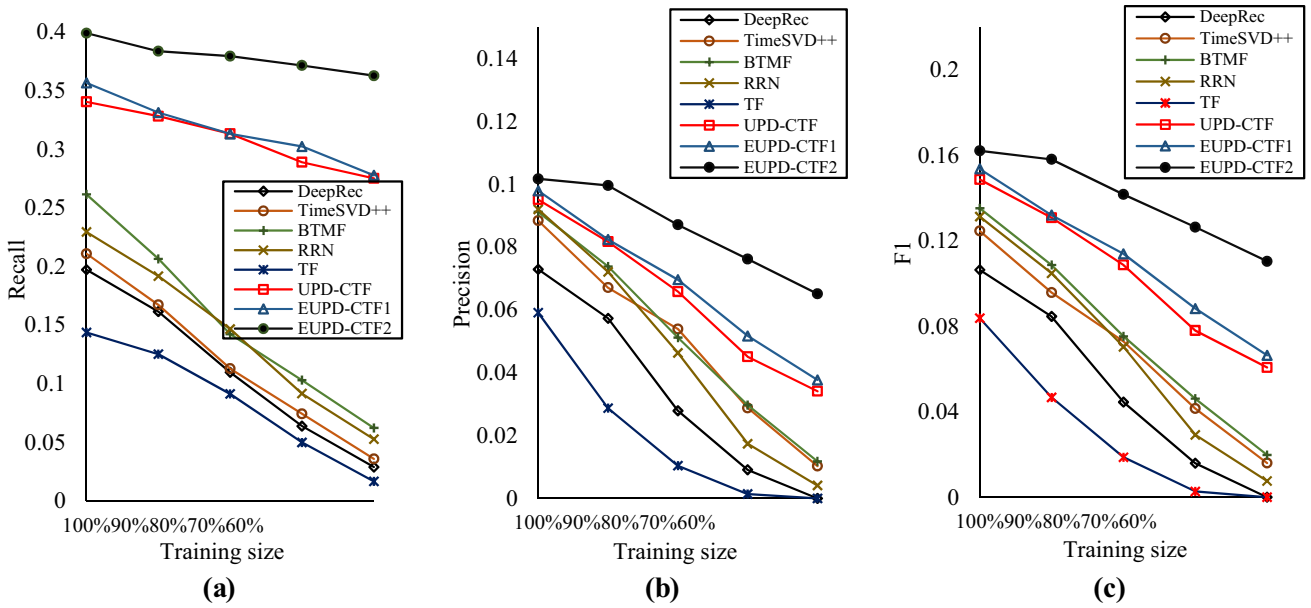

Fig. 8 The performance of comparative methods in terms of **a** recall, **b** precision and **c** F1, on Last.fm for different training sizes (K = 100)

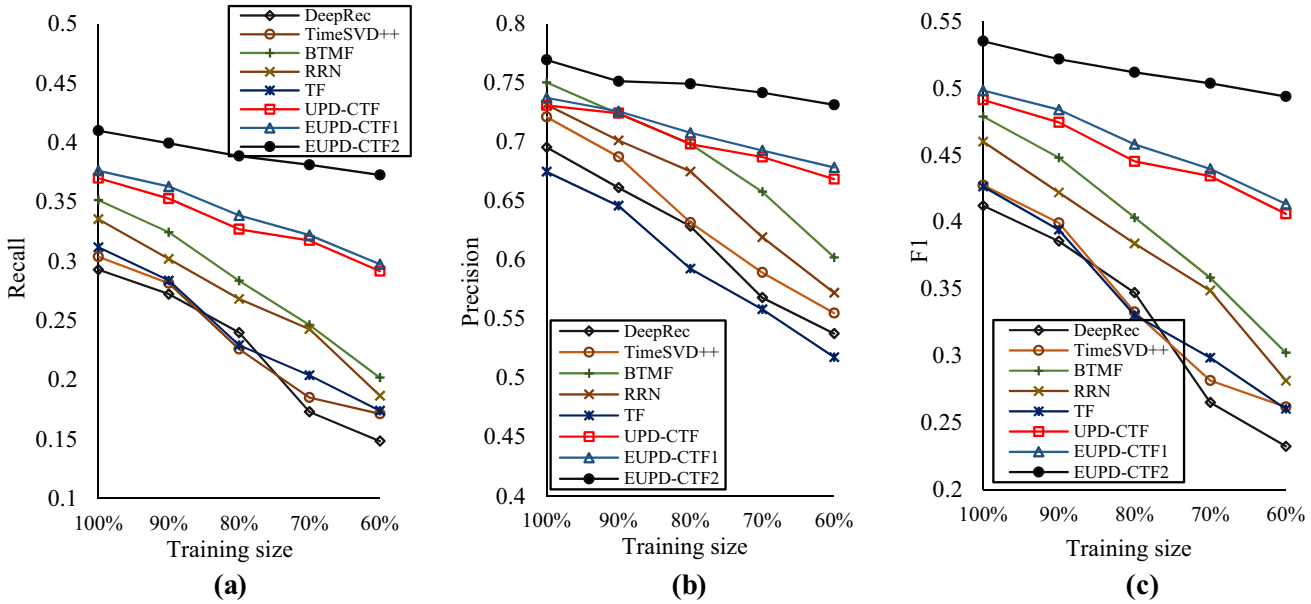

Fig. 9 The performance of comparative methods in terms of **a** recall, **b** precision and **c** F1, on Movielens for different training sizes (K = 3)

approach can handle the cold-start user and data sparsity problems effectively.

We plan to study the dynamics of user preferences by considering the trust evolution that may be evolved with different speeds under different situations. We also want to design a parallel implementation of our model, in order to make it scalable to large-scale datasets.

Funding Open Access funding enabled and organized by Projekt DEAL..

References

Acar E, Dunlavy DM, Kolda TG, Morup M (2011) Scalable tensor factorizations for incomplete data. Chemometr Intell Lab Syst 106:41–56. https://doi.org/10.1016/j.chemolab.2010.08.004

Acar E, Kolda TG, Dunlavy DM (2011b) All-at-once optimization for coupled matrix and tensor factorizations. arXiv:1105.3422

Acar E, Bro R, Smilde AK (2015) Data fusion in metabolomics using coupled matrix and tensor factorizations. Proc IEEE 103:1602–1620. https://doi.org/10.1109/JPROC.2015.2438719

Aravkin AY, Varshney KR, Yang L (2016) Dynamic matrix factorization with social influence. In: 2016 IEEE international workshop on machine learning for signal processing. IEEE, pp 1–6

Balasubramaniam T, Nayak R, Yuen C, Tian Y (2020) Column-wise element selection for computationally efficient nonnegative coupled matrix tensor factorization. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2020.2967045

Bao H, Li Q, Liao SS et al (2013) A new temporal and social PMF-based method to predict users' interests in micro-blogging. Decis Support Syst 55:698–709. https://doi.org/10.1016/j.dss.2013.02.007

Barjasteh I, Forsati R, Ross D et al (2016) Cold-start recommendation with provable guarantees: a decoupled approach. IEEE Trans Knowl Data Eng 28:1462–1474. https://doi.org/10.1109/TKDE.2016.2522422

Batmaz Z, Yurekli A, Bilge A, Kaleli C (2018) A review on deep learning for recommender systems: challenges and remedies. Artif Intell Rev 52:1–37. https://doi.org/10.1007/s10462-018-9654-y

Cai G, Lv R, Tang J, Liu H (2014) Temporal dynamics in social trust prediction. Wuhan Univ J Nat Sci 19:369–378. https://doi.org/10.1007/s11859-014-1027-z

Cheng J, Liu Y, Zhang H et al (2015) A new recommendation algorithm based on user's dynamic information in complex social network. Math Probl Eng. https://doi.org/10.1155/2015/281629

Do Q, Liu W (2016) ASTEN: an accurate and scalable approach to coupled tensor factorization. In: Proceedings of the international joint conference on neural networks (IJCNN), Vancouver, pp 99–106

Dunlavy DM, Kolda TG, Acar E (2010) Poblano v1. 0: a matlab toolbox for gradient-based optimization. Sandia National Laboratories, Albuquerque, NM and Livermore, CA, Tech Rep SAND2010-1422

Dunlavy DM, Kolda TG, Acar E (2011) Temporal link prediction using matrix and tensor factorizations. ACM Trans Knowl Discov Data 5:1–27. https://doi.org/10.1145/1921632.1921636

Forsati R, Mahdavi M (2014) Matrix factorization with explicit trust and distrust relationships. arXiv:1408.0325. https://doi.org/10.1145/2641564

Frolov E, Oseledets I (2017) Tensor methods and recommender systems. Wiley Interdiscip Rev Data Min Knowl Discov 7:1–41. https://doi.org/10.1002/widm.1201

Guo G, Zhang J, Yorke-Smith N (2016) A novel recommendation model regularized with user trust and item ratings. IEEE Trans Knowl Data Eng 28:1607–1620. https://doi.org/10.1109/TKDE.2016.2528249

Hafshejani ZY, Kaedi M, Fatemi A (2018) Improving sparsity and new user problems in collaborative filtering by clustering the personality factors. Electron Commerce Res 18:813–836. https://doi.org/10.1007/s10660-018-9287-x

Huang Z, Zeng D, Chen H (2004) A link analysis approach to recommendation under sparse data. In: Amcis, pp 1–9

Ju B, Qian Y, Ye M et al (2015) Using dynamic multi-task non-negative matrix factorization to detect the evolution of user preferences in collaborative filtering. PLoS ONE 10:1–20. https://doi.org/10.1371/journal.pone.0135090

Kolda TG, Bader BW (2009) Tensor decompositions and applications. SIAM Rev 51:455–500. https://doi.org/10.1137/07070111X

Koren Y (2010) Collaborative filtering with temporal dynamics. Commun ACM 53:89–97. https://doi.org/10.1145/1721654.1721677

Lee WP, Ma CY (2016) Enhancing collaborative recommendation performance by combining user preference and trust-distrust propagation in social networks. Knowl Based Syst 106:125–134. https://doi.org/10.1016/j.knosys.2016.05.037

Li S, Fu Y (2017) Robust representations for response prediction. In: Robust representation for data analytics. Springer, pp 147–174

Liu NN, Zhao M, Xiang E, Yang Q (2010) Online evolutionary collaborative filtering. In: Proceedings of the fourth ACM conference on recommender systems, pp 95–102

Liu NN, He L, Zhao M (2013) Social temporal collaborative ranking for context aware movie recommendation. ACM Trans Intell Syst Technol. doi 10(1145/2414425):2414440

Liu Z, Tan K, Wang X-Q, Tang S-H (2018) A learning framework for temporal recommendation without explicit iterative optimization. Appl Soft Comput 67:529–539

Lo Y-Y, Liao W, Chang C-S, Lee Y-C (2018) Temporal matrix factorization for tracking concept drift in individual user preferences. IEEE Trans Comput Soc Syst 5:156–168. https://doi.org/10.1109/TCSS.2017.2772295

Matuszyk P, Spiliopoulou M (2014) Selective forgetting for incremental matrix factorization in recommender systems. International conference on discovery science. Springer, Cham, pp 204–215

Moré JJ, Thuente DJ (1994) Line search algorithms with guaranteed sufficient decrease. ACM Trans Math Softw 20:286–307. https://doi.org/10.1145/192115.192132

Mu R (2018) A survey of recommender systems based on deep learning. IEEE Access 6:69009–69022. https://doi.org/10.1109/ACCESS.2018.2880197

Nocedal J, Wright SJ (2006) Sequential quadratic programming. Numer Optim. https://doi.org/10.1007/978-0-387-40065-5_18

Pan W (2016) A survey of transfer learning for collaborative recommendation with auxiliary data. Neurocomputing 177:447–453. https://doi.org/10.1016/j.neucom.2015.11.059

Rafailidis D, Nanopoulos A (2016) Modeling users preference dynamics and side information in recommender systems. IEEE Trans Syst Man Cybern Syst 46:782–792. https://doi.org/10.1109/TSMC.2015.2460691

Rafailidis D, Kefalas P, Manolopoulos Y (2017) Preference dynamics with multimodal user-item interactions in social media recommendation. Expert Syst Appl 74:11–18

Rana C, Jain SK (2014) An evolutionary clustering algorithm based on temporal features for dynamic recommender systems. Swarm Evolut Comput 14:21–30. https://doi.org/10.1016/j.swevo.2013.08.003

Rana C, Jain SK (2015) A study of the dynamic features of recommender systems. Artif Intell Rev 43:141–153. https://doi.org/10.1007/s10462-012-9359-6

Reafee W, Salim N, Khan A (2016) The power of implicit social relation in rating prediction of social recommender systems of social recommender. PLoS ONE. https://doi.org/10.1371/journal.pone.0154848

Sherstinsky A (2020) Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Phys D 404:132306. https://doi.org/10.1016/j.physd.2019.132306

Shi Y, Larson M, Hanjalic A (2014) Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges. ACM Comput Surv (CSUR) 47:3. https://doi.org/10.1145/2556270

Spiegel S, Clausen J, Albayrak S, Kunegis J (2011) Link prediction on evolving data using tensor factorization. New frontiers in applied data mining. Springer, Berlin, pp 100–110

Su H, Lin X, Yan B, Zheng H (2015) The collaborative filtering algorithm with time weight based on mapreduce. International conference on big data computing and communications. Springer, Cham, pp 386–395

Sun JZ, Parthasarathy D, Varshney KR (2014) Collaborative Kalman filtering for dynamic matrix factorization. IEEE Trans Signal Process 62:3499–3509

Sun Z, Guo Q, Yang J et al (2019) Research commentary on recommendations with side information: a survey and research directions. Electron Commerce Res Appl 37:100879. https://doi.org/10.1016/j.elerap.2019.100879

Tang J, Gao H, Das SA et al (2015) Trust evolution: modeling and its applications. IEEE Trans Knowl Data Eng 27:1724–1738. https://doi.org/10.1109/TKDE.2014.2382576

Tong C, Qi J, Lian Y et al (2019) TimeTrustSVD: a collaborative filtering model integrating time, trust and rating information. Future Gener Comput Syst 93:933–941. https://doi.org/10.1016/j.future.2017.07.037

Vinagre J (2012) Time-aware collaborative fitering: a review. In: Doctoral symposium in informatics engineering, p 43

Wang M, Ma J (2016) A novel recommendation approach based on users' weighted trust relations and the rating similarities. Soft Comput 20:3981–3990. https://doi.org/10.1007/s00500-015-1734-1

Wang H, Zhang Q, Yuan J (2017) Semantically enhanced medical information retrieval system: a tensor factorization based approach. IEEE Access 5:7584–7593. https://doi.org/10.1109/ACCESS.2017.2698142

Wu C, Amr A, Alex B et al (2017) Recurrent recommender networks. In: Proceedings of the tenth ACM international conference on web search and data mining, pp 495–503

Wu T, Feng Y, Sang J et al (2018) A novel recommendation algorithm incorporating temporal dynamics, reviews and item correlation. IEICE Trans Inf Syst 101:2027–2034. https://doi.org/10.1587/transinf.2017EDP7387

Xiong L, Chen X, Huang T-K et al (2010) Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In: Proceedings of the 2010 SIAM international conference on data mining, pp 211–222

Yang X, Guo Y, Liu Y, Steck H (2014) A survey of collaborative filtering based social recommender systems. Comput Commun 41:1–10. https://doi.org/10.1016/j.comcom.2013.06.009

Yin H, Cui B, Chen L et al (2014) A temporal context-aware model for user behavior modeling in social media systems. In: Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pp 1543–1554

Yin H, Cui B, Chen L et al (2015) Dynamic user modeling in social media systems. ACM Trans Inf Syst (TOIS) 33:1–44

Zafari F, Moser I, Baarslag T (2019) Modelling and analysis of temporal preference drifts using a component-based factorised latent approach. Expert Syst Appl 116:186–208. https://doi.org/10.1016/j.eswa.2018.09.010

Zhang C, Wang K, Yu H et al (2014) Latent factor transition for dynamic collaborative filtering. In: Proceedings of the 2014 SIAM international conference on data mining, pp 452–460

Zhang W, Du Y, Yoshida T, Yang Y (2019) DeepRec: a deep neural network approach to recommendation with item embedding and weighted loss function. Inf Sci 470:121–140