Modeling Users Preference Dynamics and Side Information in Recommender Systems

Dimitrios Rafailidis and Alexandros Nanopoulos

***Abstract*—In recommender systems user preferences can be fairly dynamic, as users tend to exploit a wide range of items and modify their tastes accordingly over time. In this paper, we model user-item interactions over time using a tensor that has time as a dimension (mode). To account for the fact that user preferences change individually, we propose a new measure of user-preference dynamics (UPD) that captures the rate with which the current preferences of each user have been shifted. UPD shows the variability in how users interact with items in recommender systems. We generate recommendations based on a tensor factorization technique, where the importance of past user preferences are weighted according to their UPD values, that is, higher UPD values downweigh more past user preferences. Additionally, we exploit users' side data, such as demographics, which improve the accuracy of recommendations based on a coupled tensor-matrix factorization scheme. Our empirical evaluation uses two real benchmark datasets from the social media platforms Last.fm and MovieLens, containing users' history records pertaining to listening to songs and viewing movies, respectively. We demonstrate that in both datasets, there are users with a varying level of dynamics, expressed by the UPD metric. Our experimental results show that the proposed method outperforms several baselines, by taking into account both dynamics and side data of users.**

***Index Terms*—Coupled tensor factorization (CTF), recommender systems, users' dynamics.**

I. Introduction

COLLABORATIVE filtering plays a vital role in recommender systems [1], [2], where users with the same activity, for example, rating or tagging behavior tend to get similar recommendations. Following the collaborative filtering paradigm, several matrix and tensor factorization methods have been proposed, such as [3]–[7]. In addition to these studies, in this paper, we focus on the user's preference dynamics (UPD) in temporal collaborative filtering methods. We consider user-item interactions over time and capture how users preferences shift within certain time-periods. There are many factors that affect UPDs in temporal collaborative filtering. According to [8], [12], [13], and [15] users in recommender systems shift their preferences over time mainly for the following reasons.

1) *New Items Exploration:* Users in recommender systems tend to explore new items over time, instead of interacting with the same items multiple times.
2) *Users' Past Experience:* Users tend to interact with items that they have previously liked. For instance, if the first song of a certain artist that a user listens is particularly good, then the user will probably keep listening songs from the same artist. Similarly, if the user did not enjoy the first song, then the user will probably dislike the rest of this artist's songs.
3) *Popular Items Bias:* Users may interact with popular items to a great extent, irrespective of their history record. For instance, if a user likes "romantic" movies but there is a popular "sci-fi" movie, then despite his past preference the user may prefer to watch it.
4) *Neighbors' Influence:* Users preferences may be updated based on the preferences of their neighbors/friends over time. According to [17], users can be categorized to: (a) "stubborn," that is, users who do not change preferences; (b) "compromising," that is, users who update their preferences by combining their initial preferences and those of their neighbors; or (c) "conforming," that is, users who inherit their neighbors' preferences, ignoring their own. However, user behavior may be dynamic, because a user may interact differently when faced with different amounts and types of items.

A. Motivation

For the aforementioned reasons users may change their interests over time, especially in recommender systems where they interact customarily with a wide range of items, for example, when listening to music or watching movies. Recent research has started to incorporate temporal effects into matrix factorization models [8], [9], by observing "drifts" in the rating behavior and modeling the way user and item characteristics change over time [10]–[12]. For instance, users like cartoons when they are young, but dislike them when growing up. However, such approaches do not take into account the fact that in several applications: users interact with items over time [13], or unlike [13] that focuses on explicit feedback in the form of ratings, implicit quantitative feedback may be provided by users, such as the number of times they listened to a song or downloaded a video within a time period.

User-item interactions over time can be captured with a sparse tensor whose dimensions (modes) correspond to

Manuscript received May 16, 2014; revised September 17, 2014 and February 21, 2015; accepted June 13, 2015. This paper was recommended by Associate Editor S. Tong.

D. Rafailidis is with the Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki GR-54124, Greece, and also with the Department of Computer Science, University of Cyprus, Nicosia 1678, Cyprus (e-mail: draf@csd.auth.gr).

A. Nanopoulos is with the School of Management, Catholic University of Eichstät-Ingolstadt, Ingolstadt 85049, Germany (e-mail: alexandros.nanopoulos@ku.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2015.2460691

users, items and time-periods [14]. Each nonempty tensor cell records the count information within the given period. For instance, Xiong *et al.* [15] used a Bayesian probabilistic tensor factorization model for movie recommendations. Spiegel *et al.* [16] used tensor factorization on evolving data with the different time-periods being modeled as time slices in the tensor. In this paper, the importance of users' past preferences was decreased according to a smoothing factor (SF). Nevertheless, these approaches ignore the fact that changes in user preferences can vary individually, that is, some users' tastes may be very volatile, whereas others may keep their tastes relatively stable over time.

In addition to UPDs, several studies [18]–[20] exploited auxiliary information (also known as side information) of users or items to improve the recommendation accuracy. For instance, Beel *et al.* [21] discussed the importance of users' demographics and private attributes for evaluating recommender systems. In their case study, they showed that it is crucial to consider users' age and other private data, when evaluating recommender systems. For the evaluation, they used the click-through rate, which expresses how many times the displayed recommendations were clicked. In their case study, the authors revealed that younger users (20–24 years old) tend to have a lower click-through rate than older ones. Nevertheless, in many cases the private attributes may be missing, thus prohibiting their direct use in recommender systems. In addition, the aforementioned methods that exploit side information do not consider the UPDs.

B. Contribution and Outline

In [22], we introduced a new measure of UPD that captures the rate with which the current preferences of each user have been shifted, when listening to music. We generated artist recommendations, by weighting the importance of past users' preferences according to their UPD values, and we exploited users' demographics in a coupled tensor factorization (CTF) scheme to improve the recommendation accuracy. In this paper, we extend our preliminary study as follows: we introduce the generic optimization problem of CTF, then, we present the optimization problem of CTF for missing data, and we describe our CTF model calculation. We extend our experiments, using two benchmark datasets from the social media platforms Last.fm[1] and MovieLens.[2] We show that the UPD values for the majority of users significantly vary, indicating that users significantly shift their preferences over time, when listening to music or watching movies. In addition, we experimentally show the effectiveness of our approach against competitive methods, and we discuss the main findings of our experimental evaluation. Finally, according to the aforementioned findings, we provide the managerial insights which could result in more accurate recommendations.

The rest of this paper is organized as follows: after summarizing related studies in Section II, we present some preliminaries of tensors in Section III. Next, we formulate our problem in Section IV and we present the proposed model in Section V. In our experimental results on the two benchmark datasets, we show the effectiveness of the proposed approach against competitive methods in Section VI. Finally, we draw the basic conclusion of this paper in Section VII.

[1]http://www.last.fm: a music discovery service, providing personal recommendations based on the music that users listen to.

[2]http://www.movielens.org/: a personalized movie recommendation platform.

II. Related Study

The related study can be divided into: 1) collaborative filtering methods that handle temporal information; 2) tensor factorization techniques that consider data dynamics; and 3) tensor factorization with side information, mainly including coupled-based tensor factorization techniques. The first two groups differ, as tensor factorization techniques that consider data dynamics can also handle temporal information, but not vice versa.

A. Temporal Collaborative Filtering Methods

With respect to the temporal dimension, several studies exploited the time information. Stefanidis *et al.* [24] presented a framework for generating recommendations based on users' recent preferences and providing different suggestions under different temporal specifications. Gao *et al.* [25] presented a model for location-based recommendations to a user based on his personal preferences, to facilitate his exploration of new areas of a city. This was achieved by exploring temporal patterns, modeling temporal effects on location-based social networks. Wang *et al.* [26] modeled tuples {tourist, time, location} into tensors, to perform time-aware recommendations for travel destinations, by maximizing the temporal-spatial correlation for tourists. The proposed context-aware recommendation model tried to predict the items and their best corresponding temporal context. However, the aforementioned methods focused on the spatial and temporal information of the data, even as ignoring the UPDs in recommender systems.

B. Tensor Factorization for Data Dynamics

To predict a future link, Dunlavy *et al.* [14] considered bipartite graphs with tuples in the form {author, conference, relationship} that evolve over time. In this paper, the CANDECOMP/PARAFAC (CP) decomposition [23] (see also Section III-C) of a tensor $\mathcal{X}(a, c, t) = 1$ was used, denoting if author a links to conference c at time t. The goal was to predict a new link at time $t + 1$, by exploring the 3-D structure of temporal data. Liu *et al.* [27] proposed an iterative tensor factorization technique with linear time complexity for mining time-evolving graph data. Xiong *et al.* [15] added constraints in the time dimension of a Bayesian probabilistic tensor factorization model, to process time-evolving data and furthermore to generate sales prediction and movie recommendations. Spiegel *et al.* [16] proposed a prediction algorithm on evolving data using tensor factorization. They used the CP decomposition to a tensor, constructed by tuples in the form {user, item, time, rating}. Different time-periods were modeled as time slices in the tensor. Using a SF, exponentially decreasing weights were assigned over time. This was achieved by multiplying different weights to the time slices of the tensor, with the weights being calculated based on the time difference of the recently examined time-period and the past ones. Nevertheless, the fundamental problem of this paper was that the algorithm assigned the same weight for all users' ratings at the same time-period, by ignoring the personal preference dynamics. Finally, in these studies the side information of users' private attributes were not exploited. In addition, there are several studies that consider data streams in tensor factorization techniques such as [28]. However, following [14]–[16], [26], and [27], user-item interactions are considered within certain time periods, for example, days, weeks, and so on, and thus data streams are out of this paper's scope.

C. Tensor Factorization With Side Information

Several studies have addressed the recommendation problem with tensor-based approaches using auxiliary information. For instance, Ermis *et al.* [19] presented a generalized CTF method for generating personalized activity recommendations based on geo-locations. According to a probabilistic approach of tensor factorization models, they generated recommendations using a coupled analysis of relational datasets, which were represented as heterogeneous data in the form of matrices with side information: 1) the location features from the points of interest and 2) user-location preferences from the GPS trajectory data. In this paper, the main data were tuples in the form {user, location, activity}, which were modeled to high-order tensors. Narita *et al.* [20] introduced two regularization approaches for exploiting the auxiliary information in tensor factorization. However, both regularization approaches used an alternative least squares-based technique [23] for fitting the reconstructed tensor to the initial tensor over the factorization, thus making the method have poor convergence in the presence of missing data; as for example usually happens in recommender systems. Acar *et al.* [18] proposed an all-at-once optimization approach, which is a gradient-based approach for fitting outer-product models to high-order tensors and matrices in a coupled manner. Additionally, they extended their approach to handle incomplete datasets, that is, missing entries either in the tensor or in the coupled matrices. Although the aforementioned studies incorporated auxiliary information to generate more accurate recommendations, the UPDs were ignored.

III. Preliminaries

A. Notations

N-order tensors are denoted by the Euler script letter, for example, $\mathcal{X}$. In this paper, we consider three-order tensors (i.e., $N = 3$). Matrices and sets are denoted by italic capital letters, for example, A, vectors are denoted by boldface lowercase letters, for example $\mathbf{a}$, and scalars are denoted by lowercase letters for example, a. The ith column of a matrix is denoted by a boldface lower letter with a subscript, for example, $\mathbf{a}_i$ of matrix A. The entries (nonempty cells) of a matrix or a tensor are denoted by lowercase letters with subscripts, that is, the $i_1, i_2, \ldots,$ and i_N entry of a N-way tensor $\mathcal{X}$ is denoted by $x_{i_1 i_2 \cdots i_N}$.

B. Tensors Operators and Products

In this section, we present some basic operators and products of matrices and tensors. For further details, interested readers can refer to [23]. Given a matrix $A \in \mathbb{R}^{m\times p}$, the operator vec($A$) concatenates the columns of the matrix and forms the following vector $\in \mathbb{R}^{mp\times 1}$: $\text{vec}(A) = \begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_p\end{bmatrix}$.

Let $A \in R^{m\times n}$ and $B \in R^{p\times q}$ be two matrices, the Kronecker product, denoted by $\otimes$, results in a matrix $\in R^{mp\times qn}$ as follows:

$$\begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}. \tag{1}$$

Given two matrices $A \in \mathbb{R}^{m\times p}$ and $B \in \mathbb{R}^{q\times p}$, their Khatri–Rao product is denoted by $A \odot B$ and it is defined as the columnwise Kronecker product, resulting in the following matrix $\in R^{(mq)\times p}$:

$$A \odot B = \begin{bmatrix}\mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_K \otimes \mathbf{b}_p\end{bmatrix}. \tag{2}$$

A three-order tensor can be rearranged as a matrix, which is called the unfolding of the tensor. The mode-n unfolding of a tensor $\mathcal{X} \in R^{I_1\times I_2\times I_3}$, with $n = 1, 2, 3$, is denoted by $X_{(n)}$ and arranges the mode-n 1-D "fibers" [23], as the columns of the resulting matrix. Tensor $\mathcal{X}$ can be unfolded to all its three modes-dimensions. Thus, after unfolding the tensor $\mathcal{X}$, we create three new matrices $X_{(1)}, X_{(2)}$, and $X_{(3)}$, as follows:

$$X_{(1)} \in \mathbb{R}^{I_1\times I_2 I_3}, \quad X_{(2)} \in \mathbb{R}^{I_2\times I_1 I_3}, \quad X_{(3)} \in \mathbb{R}^{I_3\times I_1 I_2}. \tag{3}$$

Given two tensors $\mathcal{X} \in R^{I_1\times I_2\times I_3}$ and $\mathcal{Y} \in R^{I_1\times I_2\times I_3}$, their Hadamard (elementwise) product is denoted by

$$(\mathcal{X} * \mathcal{Y})_{i_1 i_2 i_3} = x_{i_1 i_2 i_3} y_{i_1 i_2 i_3}. \tag{4}$$

Let $\langle \mathcal{X}, \mathcal{Y}\rangle$ be the inner product of tensors $\mathcal{X}$ and $\mathcal{Y}$, which is the sum of products of their entries

$$\langle \mathcal{X}, \mathcal{Y}\rangle = \sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\sum_{i_1=3}^{I3} x_{i_1 i_2 i_3} y_{i_1 i_2 i_3}. \tag{5}$$

Based on (5), the norm of tensor $\mathcal{X}$ is defined as

$$||\mathcal{X}|| = \sqrt{\langle \mathcal{X}, \mathcal{Y}\rangle} \tag{6}$$

where $||\cdot||$ denotes the Frobenius norm and the two-norm in the case of matrices.

The three way outer product of vectors $\mathbf{a}, \mathbf{b}$, and $\mathbf{c}$ is defined as

$$(\mathbf{a} \circ \mathbf{b} \circ \mathbf{c})_{i,j,k} = \mathbf{a}_i \mathbf{b}_j \mathbf{c}_k. \tag{7}$$

Given a sequence of matrices $A^{(n)} \in \mathbb{R}^{I_n\times R}$, for $n = 1, 2, 3$ the notation $[\![A^{(1)}, A^{(2)}, A^{(3)}]\!]$ defines a three-order tensor $\in \mathbb{R}^{I_1\times I_2\times I_3}$ whose elements are calculated as follows:

$$\left([\![A^{(1)}, A^{(2)}, A^{(3)}]\!]\right)_{i_1 i_2 i_3} = \sum_{r=1}^{R}\prod_{n=1}^{3} a^{(n)}_{i_n r}. \tag{8}$$

In the case of two matrices, it holds that $[A, B] = AB^T$.

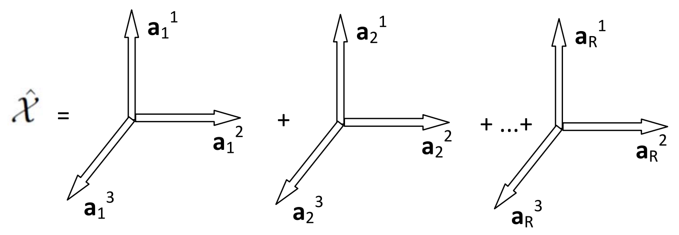


Fig. 1. R-component CP model for a third-order tensor $\mathcal{X}$ results in the low-rank approximation tensor $\hat{\mathcal{X}}$.

C. Low-Rank R Approximation of Tensors

The rank of a three-order tensor $\mathcal{X}$, denoted as $\text{rank}(\mathcal{X}) = R$, is the smallest number of rank-one tensors that generate the tensor as their sum, that is, the smallest R is defined as follows:

$$\mathcal{X} = \sum_{i=1}^{R} \mathbf{a}_r^1 \circ \mathbf{a}_r^2 \circ \mathbf{a}_r^3$$

where $\circ$ is the outer product operation of (7) and $\mathbf{a}_r^1 \in \mathbb{R}^{I_1 \times 1}$, $\mathbf{a}_r^2 \in \mathbb{R}^{I_2 \times 1}$, $\mathbf{a}_r^3 \in \mathbb{R}^{I_3 \times 1}$ are called factor vectors. As determining the rank R of the tensor is a nondeterministic polynomial-time (NP)-hard problem [29], several methods consider the low-rank R approximation of a tensor $\mathcal{X}$. In this paper, we consider the CP decomposition [23], which in its original form calculates the $\hat{\mathcal{X}}$ low-rank R approximation of a tensor $\mathcal{X}$ as follows:

$$\hat{\mathcal{X}} = \sum_{r=R}^{R} \lambda_r \mathbf{a}_r^1 \circ \mathbf{a}_r^2 \circ \mathbf{a}_r^3 \tag{9}$$

where the factor vectors are normalized by a scalar term λ_r, one for each rank-one factor of the decomposition (comprising a vector $\in \mathbb{R}^{R \times 1}$), thus forcing the factor vectors to be of unit norm. Also, factor vectors are considered as the rth columns of matrices, which are called factor matrices of CP, denoted by $A^{(1)} \in \mathbb{R}^{I_1 \times R}$, $A^{(2)} \in \mathbb{R}^{I_2 \times R}$, and $A^{(3)} \in \mathbb{R}^{I_3 \times R}$, with $A^{(n)} = [\mathbf{a}_1^n \ \mathbf{a}_2^n \ \cdots \ \mathbf{a}_R^n]$. According to (8) and (9), it holds that $\hat{\mathcal{X}} = [\![A^{(1)}, A^{(2)}, A^{(3)}]\!]$. The goal of the R-component CP decomposition is to minimize the low rank approximation error $||\mathcal{X} - \hat{\mathcal{X}}||$, where $|| \cdot ||$ indicates the Frobenius norm, as described in (6). An overview of the R-component CP decomposition is depicted in Fig. 1.

IV. Problem Formulation

Given the sets U, I, and T of users, items, and time-periods, respectively, the training set consists of tuples in the form {user, item, time-period, # of interactions}. The training tuples are stored into a tensor $\mathcal{X} \in \mathbb{R}^{|U| \times |I| \times |T|}$, with users, items, time dimensions, and each nonempty cell $x_{u,i,t}$= # of interactions. Tensor $\mathcal{X}$ is coupled with the auxiliary matrix $Y \in \mathbb{R}^{|U| \times |D|}$, which contains the side information of the $|D|$ users' attributes.

The setting of the examined problem is presented in Fig. 2. Tensor $\mathcal{X}$ can be viewed as an aggregation of many slices, which for the rest of this paper we will call time slices. Each time slice corresponds to a time-period $t = 1 \ldots |T|$, denoted by $(|U| \times |I|)_t$. The entries in the current/last time

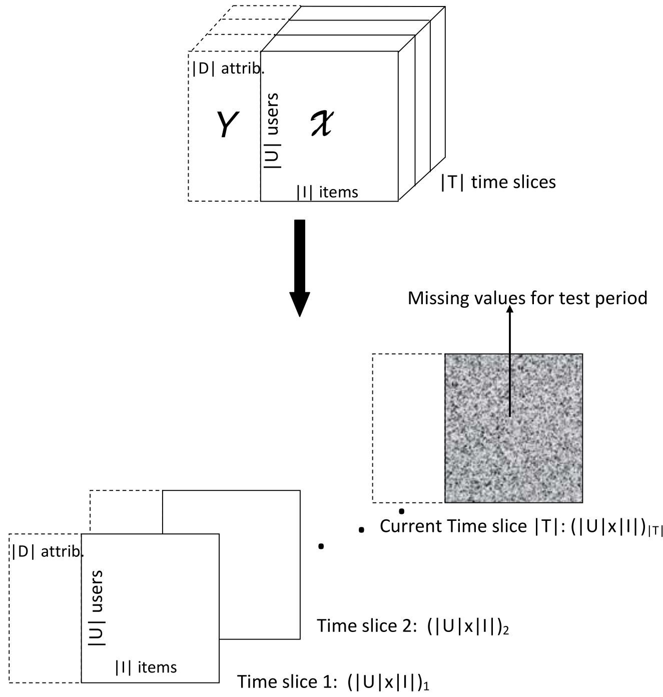


Fig. 2. Tensor $\mathcal{X}$ with the auxiliary matrix Y can be viewed as an aggregation of $|T|$ different time slices. In the current/last time slice $(|U| \times |I|)_{|T|}$, there are the missing values for a future (test) period. For example, time slices could be semiannuals and the future (test) period could be the last sixth months of the ongoing semiannual, corresponding to the current/last time $(|U| \times |I|)_{|T|}$.

slice $(|U| \times |I|)_{|T|}$ are the number of users' interactions within the ongoing time-period. Note that in the current/last time slice, a few entries have already been filled by the users' interactions at the ongoing time-period. The goal is to predict the missing values for a future (test) period within the ongoing time-period, that is, the missing values of the current/last time slice $(|U| \times |I|)_{|T|}$.

Please note that the auxiliary matrix $Y \in \mathbb{R}^{|U| \times |D|}$ may also change over time, as users' attributes can be dynamic too. As depicted in Fig. 2, each of the $|T|$ time slices of tensor $\mathcal{X}$ may potentially have a different instance of the auxiliary matrix Y. However, motivated by the fact that most publicly available benchmark datasets (such as those used in our experimental evaluation) provide static users' attributes, we henceforth focus on the case of a single auxiliary matrix Y for all $|T|$ time slices of tensor $\mathcal{X}$.

V. Proposed Model

The inputs of the proposed model are: 1) a third-order tensor $\mathcal{X} \in \mathbb{R}^{|U| \times |I| \times |T|}$, where U, I and T denote the sets of users, items and time-periods, respectively; 2) an auxiliary matrix $Y \in \mathbb{R}^{|U| \times |D|}$ of users' side information with a set of D private attributes, coupled in the user dimension (mode) U of tensor $\mathcal{X}$; and 3) the rank R of $\mathcal{X}$. The proposed model consists of the following three steps.

1) First, we model the temporal and users' side information in tensor $\mathcal{X}$, which is also coupled with the auxiliary matrix Y, as presented in Fig. 2 (Section V-A).
2) Then, the proposed model downweighs the past entries of tensor $\mathcal{X}$ based on the users' recent interaction behavior (Section V-B).

3) According to a CP decomposition for CTF, the output is a low-rank R approximation of $\mathcal{X}$ coupled with Y, which is denoted by a tensor $\hat{\mathcal{X}}$. Finally, based on $\hat{\mathcal{X}}$, we generate personalized recommendations for a user at the future (test) time-period in the current/last time slice $(|U| \times |I|)_{|T|}$ of $\hat{\mathcal{X}}$ (Section V-C).

A. Modeling Changing User Preferences and Side Information

Each nonempty tensor cell $x_{u,i,t}$ contains the number of interactions of user u with item i at the time period t. The time period can be days, months, semiannuals or years, corresponding to the $|T|$ different time slices $(|U| \times |I|)_t$ of tensor $\mathcal{X}$, with $t \in 1 \ldots |T|$. The choice of the time period mainly depends on the application of the recommender system. This means that the test time-period, for example, the test month where the personalized recommendations have to be generated, is included in the current/last time slice $(|U| \times |I|)_{|T|}$ of the ongoing time-period.

With respect to the users' side information, the uth row of the auxiliary matrix $Y \in \mathbb{R}^{|U|\times|D|}$, with $u \in U$, corresponds to the set D of the users' private attributes. In case of numerical private attributes, for example, age, we perform an equal-width binning method, where for each numerical entry in the matrix Y, we store the respective number of the bin. In the case of categorical attributes, such as county and gender, we calculate the c distinct categorical values and then we create a binary vector $\mathbf{c}_i \in \mathbb{R}^{c\times 1}$, where 1 denotes the categorical attribute of each user. Finally, we concatenate the transformed numerical and categorical attributes to generate the final $|D|$ different private attributes in matrix Y. Because a matrix can be considered as a two-way tensor, we model the auxiliary matrix Y to a sparse two-order tensor with the help of the tensor MATLAB toolbox [38]. The reason for using sparse tensor structures is that in many cases users' private attributes are missing in recommender systems.

B. Users Preference Dynamics

Given a test period t within the current/last time slice $(|U| \times |I|)_{|T|}$ of the tensor $\mathcal{X}$, for example, the test month in the last semiannual, for each user u we calculate the UPD_u value as follows:

$$\mathrm{UPD}_u = 1 - \frac{\left|I^u_{\mathrm{cur}} \cap I^u_{\mathrm{prev}}\right|}{\left|I^u_{\mathrm{cur}} \cup I^u_{\mathrm{prev}}\right|} \tag{10}$$

where $I^u_{\mathrm{cur}} \subseteq I$ denotes the set of items that user u has interacted at the current/last time slice $(|U| \times |I|)_{|T|}$ of $\mathcal{X}$ and $I^u_{\mathrm{prev}} \subseteq I$ is the union set of the items that user u has interacted at all the previous time slices $|U| \times |I|_t$, with $t = 1 \ldots |T|-1$. The nominator of the fraction in (10) is the number of common items that user u has interacted in the current and the previous time-periods, whereas the dominator of the fraction is the number of the distinct items that user u has interacted overall. According to (10), low UPD_u values indicate that user u preserved his preferences, whereas high ones correspond to user's u high tendency to change his preferences at the current/last time-period $(|U| \times |I|)_{|T|}$. After calculating the $|U|$ different UPD values, we decrease the weights of each number of interactions of user u at the $|T|-1$ different past time slices $(|U| \times |I|)_t$, with $t = 1 \ldots |T|-1$. In particular, the weights are decreased by multiplying them with the SF $\mathrm{sf}_u = 1 - \mathrm{UPD}_u$ $\forall u \in U$ and $|T| > 1$, as follows:

$$x_{u,i,t} := \mathrm{sf}_u \cdot x_{u,i,t} \tag{11}$$
$$\text{with } t = 1, \ldots, |T|-1,\ i \in I^u_{\mathrm{prev}},\ \text{and } \mathrm{sf}_u = 1 - \mathrm{UPD}_u$$

where $:=$ is the assignment operator. In the special case of $|T| = 1$, $\forall u \in U$ then $\mathrm{sf}_u = 1$. Note that the SF sf_u is different for each user u, depending on the users' interaction behavior within the current/last time-period. This comes in contrast to several studies such as [16], where the SF is applied in a nonpersonalized manner. The outcome of this process is the recalculation of the respective entries of $\mathcal{X}$ based on (11).

C. CP Decomposition for Coupled Tensor Factorization

In our model, we use the CP decomposition of $\mathcal{X}$ with the recalculated entries based on (11). As $\mathcal{X}$ is coupled with the auxiliary matrix Y in the users' dimension, we have to follow a CP decomposition for CTF. The rest of this section is organized as follows: 1) the generic optimization problem of CTF; 2) the special case of CTF with the missing entries for the test period at the current/last time slice $(|U| \times |I|)_{|T|}$, defining thus the objective function for our CTF problem; 3) the computation of the partial derivatives of the objective function for CTF with the missing entries; 4) the calculation of the gradient values of the objective function based on the previously computed partial derivatives; 5) the construction of the final R-component CP model, that is, the reconstructed tensor $\hat{\mathcal{X}}$; and 6) the generation of the recommendations for a user u at the test time-period $t = |T|$ within the current/last time slice $(|U| \times |I|)_{|T|}$ of $\hat{\mathcal{X}}$.

1) Generic Optimization Problem of CTF: According to [18], the CP decomposition of a tensor $\mathcal{X}$ with the coupling of the auxiliary matrix Y is defined as

$$f\left(A^{(1)}, A^{(2)}, A^{(3)}, V\right) = \frac{1}{2}\left|\left|\mathcal{X} - \left[\!\left[A^{(1)}, A^{(2)}, A^{(3)}\right]\!\right]\right|\right|^2 + \cdots + \frac{1}{2}\left|\left|Y - A^{(1)}V^T\right|\right|^2. \tag{12}$$

Following the notation of Section III-C, let matrices $A^{(1)} \in \mathbb{R}^{|U|\times R}$, $A^{(2)} \in \mathbb{R}^{|I|\times R}$, and $A^{(3)} \in \mathbb{R}^{|T|\times R}$ be the factor matrices of $\mathcal{X}$ and let $A^{(1)}$ and $V \in \mathbb{R}^{|D|\times R}$ be the factor matrices extracted from Y (with users' private attributes), by performing matrix factorization. As aforementioned in Section III-C, according to (8), we use the notation $\hat{\mathcal{X}} = [\![A^{(1)}, A^{(2)}, A^{(3)}]\!]$ in (12) to denote the low-rank approximation based on the CP model. Our goal is to calculate the factor matrices $A^{(1)}, A^{(2)}, A^{(3)}$, and V that minimize the objective function of (12). To solve the minimization problem we use a gradient-based algorithm [18], which also handles missing data.

2) Optimization Problem of CTF for Missing Entries: Let $\mathcal{W} \in \mathbb{R}^{|U|\times|I|\times|T|}$ be a tensor with missing entries in the current/last time slice $(|U| \times |I|)_{|T|}$, that is, those of the test time-period

$$w_{i,j,t} = \begin{cases} 1, & \text{if } x_{i,j,t} \text{ is known} \\ 0, & \text{if } x_{i,j,t} \text{ is missing.} \end{cases} \tag{13}$$

By modifying (12) let $f_{\mathcal{W}}$ be the respective objective function for tensor $\mathcal{W}$ and auxiliary matrix Y. In particular, objective function $f_{\mathcal{W}}$ is defined as

$$f_{\mathcal{W}}\Big(A^{(1)}, A^{(2)}, A^{(3)}, V\Big) = \frac{1}{2}\Big|\Big|\mathcal{W} * \Big(\mathcal{X} - \Big[\!\!\Big[A^{(1)}, A^{(2)}, A^{(3)}\Big]\!\!\Big]\Big)\Big|\Big|^2 \\ \dots + \frac{1}{2}\Big|\Big|Y - A^{(1)V^T}\Big|\Big|^{(2)}. \quad (14)$$

The objective function of (14) is divided into the following two terms: 1) $f_{\mathcal{W}1}(A^{(1)}, A^{(2)}, A^{(3)})$ and 2) $f_2(A^{(1)}, V)$, which are the terms of sum of the squared norms in (14). The first term $f_{\mathcal{W}1}$ corresponds to the least squares problem for fitting the CP model for tensor X, whereas the second term f_2 is the respective objective function for factorizing the auxiliary matrix Y.

3) Partial Derivatives of the Objective Function: Let $\mathcal{Z} = [\![A^{(1)}, A^{(2)}, A^{(3)}]\!]$ based on (8) and let $A^{(-n)} = A^{(N)} \cdots \odot A^{(n+1)} \odot A^{(n-1)} \odot A^{(1)}$, where $\odot$ is the Khatri–Rao product of (2). According to [30], $\forall n \in 1, \dots, 3$, the partial derivatives of $f_{\mathcal{W}1}$ with respect to $A^{(n)}$ and V are

$$\frac{\partial f_{\mathcal{W}1}}{\partial A^{(n)}} = \big(\mathcal{W}_{(n)} * \mathcal{Z}_{(n)} - \mathcal{W}_{(n)} * \mathcal{X}_{(n)}\big)A^{(-n)} \quad (15)$$

$$\frac{\partial f_{\mathcal{W}1}}{\partial V} = 0 \quad (16)$$

where the partial derivative of $f_{\mathcal{W}1}$ with respect to V is always equal to 0, because, according to (14), $f_{\mathcal{W}1}$ does not depend on V. Also, as tensor $\mathcal{X}$ and the corresponding tensor $\mathcal{W}$ with the missing values of the test time-period are both coupled in the users' dimension ($n = 1$, that is, the first dimension/mode of tensor $\mathcal{X}$), the partial derivatives of the second term f_2 with respect to $A^{(n)}$ and V are

$$\frac{\partial f_2}{\partial A^{(n)}} = \begin{cases} -YV + A^{(-n)}V^TV, & \text{if } n = 1 \\ 0, & \text{if } n \neq 1 \end{cases} \quad (17)$$

$$\frac{\partial f_2}{\partial V} = -Y^TA^{(n)} + VA^{(i)^T}A^{(n)}. \quad (18)$$

Therefore, based on (15)–(18) the partial derivatives of (14) with respect to $A^{(n)}$ and V are

$$\frac{\partial f_{\mathcal{W}}}{\partial A^{(n)}} = \begin{cases} \dfrac{\partial f_{\mathcal{W}1}}{\partial A^{(i)}} + \dfrac{\partial f_2}{\partial A^{(i)}}, & \text{if } n = 1 \\ \dfrac{\partial f_{\mathcal{W}1}}{\partial A^{(n)}}, & \text{if } n \neq 1 \end{cases} \quad (19)$$

$$\frac{\partial f_{\mathcal{W}}}{\partial V} = \frac{\partial f_2}{\partial V}. \quad (20)$$

4) Gradient of the Objective Function: The respective gradient $\nabla f_{\mathcal{W}}$ of (14) is formed by vectorizing the partial derivatives of (19) and (20). This is achieved by using the vec$(\cdot)$ operator of Section III-B, thus computing a vector $\in \mathbb{R}^{(|U|+|I|+|T|+3|D|)\times 1}$ as follows:

$$\nabla f_{\mathcal{W}} = \Bigg[\text{vec}\bigg(\frac{\partial f_{\mathcal{W}}}{\partial A^{(1)}}\bigg)\ \text{vec}\bigg(\frac{\partial f_{\mathcal{W}}}{\partial A^{(2)}}\bigg) \\ \dots\ \text{vec}\bigg(\frac{\partial f_{\mathcal{W}}}{\partial A^{(3)}}\bigg)\ \text{vec}\bigg(\frac{\partial f_{\mathcal{W}}}{\partial V}\bigg)\Bigg]. \quad (21)$$

5) CP Model Calculation: Finally, given the objective function of $f_{\mathcal{W}}$ in (14) and the respective gradient values $\nabla f_{\mathcal{W}}$ in (21), we use the first-order optimization algorithm of the nonlinear conjugate gradient [31], as implemented in the coupled matrix and tensor factorization toolbox [39], to calculate the factor matrices $A^{(1)} \in \mathbb{R}^{|U|\times R}$, $A^{(2)} \in \mathbb{R}^{|I|\times R}$, $A^{(3)} \in \mathbb{R}^{|T|\times R}$. In doing so, we compute tensor $\hat{\mathcal{X}} = [\![A^{(1)}, A^{(2)}, A^{(3)}]\!]$ based on (8), where $\hat{\mathcal{X}}$ is the low-rank approximation of the initial tensor $\mathcal{X}$ based on the CP model.

6) Generation of Recommendations: The final top-k recommendations for each user u at the test time-period t are generated by ordering in descending order the entries of $\hat{x}_{u,:,t}$, that is, the respective column of items with indices $u \in U$ and $t = |T|$ within the current/last time slice of $\hat{\mathcal{X}}$. Thus, the outcome of the proposed model is the respective top-k (recommended) items.

VI. Experiments

In Sections VI-A and VI-B, we present the two benchmark evaluation datasets of Last.fm and MovieLens. Then, in Section VI-C, we show how the preference dynamics of users evolve over time in both datasets. In Section VI-D, the settings of the experiments are provided. In Section VI-E, we present the experimental results, where we evaluate the performance of the proposed method against competitive strategies, in terms of recommendation accuracy. Finally, the results are discussed in Section VI-F.

A. Last.fm Dataset

In our experiments, we used the Last.fm—1K dataset [40], which contains the listening habits of $|U| = 992$ users. The dataset consists of tuples in the form of {user, artist, song, timestamp} over 54 months (till May 5, 2009). In total, there are $|I| = 176\,948$ artists[3] and 19 150 868 listening events, corresponding to track-listenings. The distribution of the listening events is presented in Fig. 3. In our experiments, we split the dataset into nine time periods (time slots $t = S_1 \dots S_9$), corresponding to nine semiannuals. Thus, we have $|T| = 9$ different time slices in the tensor, where each slice corresponds to a period of six months. In the Last.fm dataset, the private attributes of users are also available, including age, gender and country, where in many cases the attributes are missing. Users in this dataset come from 68 different countries. As gender and country are categorical values, we used the transformation technique of Section V-A to generate $|D| = 71$ attributes in total, that is, 68, 2, and 1 for the county, genre, and age attributes, respectively. In the Last.fm dataset tuples were transformed in the form of {user, artist, time slot, # of listening events}, corresponding to the number of times a user has listened to tracks of an artist within the time slot. Given a set of training months, the proposed model performs top-k artist recommendations for a user during a test month. In our experiments, we used a time window equal to a semiannual, where as training set we considered all the past months of the previous semiannuals and the first five months of the current ongoing semiannual. Therefore, we have nine different

[3]To remove extreme sparsity for artists, we applied the p-core filtering technique, with $p = 0.2\%|I|$.

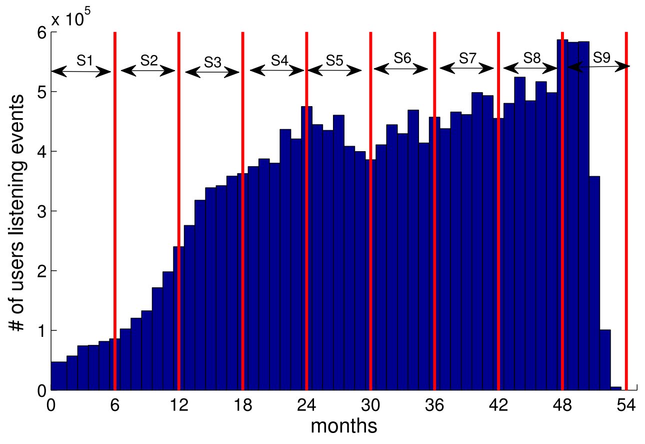


Fig. 3. Users' listening events (track-listenings) in the Last.fm dataset.

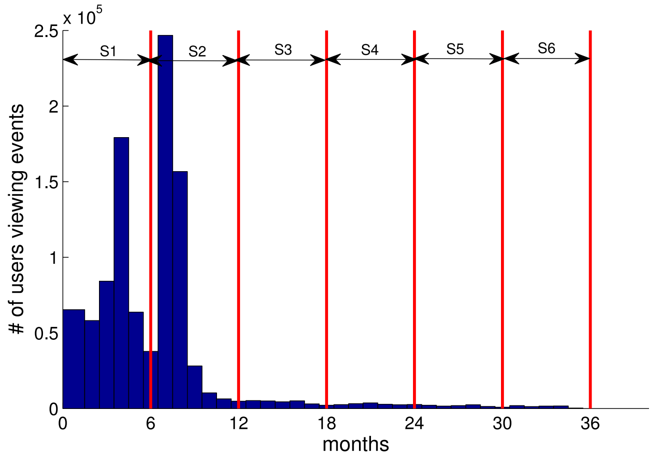


Fig. 4. Users' viewing events (movies-watched) in the MovieLens dataset.

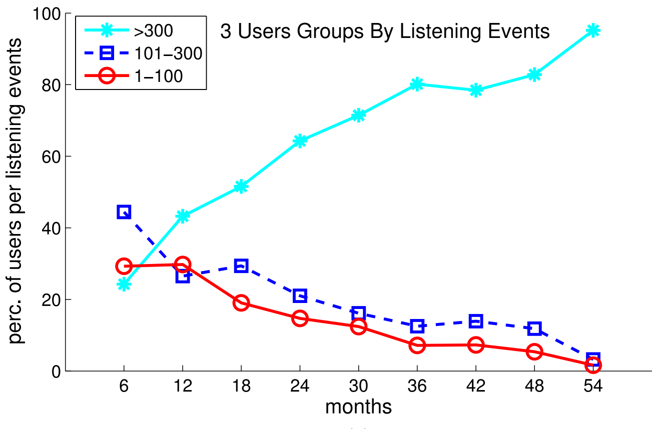
test sets of tuples at test months 6, 12, 18, 24, 30, 36, 42, 48, and 54, denoted by red lines in Fig. 3, and nine different training sets of tuples at the respective past months. The goal is to predict the artists that each user is going to listen at the last (sixth) test month of the current semiannual. Following the evaluation protocol of [7], [32], and [33], the quality of recommendations is measured in terms of recall. Thus, for a test user u that receives a list of k recommended items (top-k list) at a test period t, recall is defined as the ratio of the number of relevant artists in the top-k list over the total number of relevant artists (all artists in the hidden triplets containing the test user u and the test period t). Other commonly used measures are precision and $F1$. However, the following two factors should be clearly mentioned: 1) for each user/time-period combination in the test data, a constant number of artists has to be predicted (artists within the test period t listened by the user u) and 2) only a prespecified number k of recommendations is taken into account. Therefore, for this kind of evaluation protocol, it is redundant to evaluate precision (thus $F1$ too) because it is the same as recall up to multiplicative constants. Because in the Last.fm dataset, each user does not listen to more than 100 different artists in a test month, we report average recall (AR), with $k = 100$ artists.

B. MovieLens Dataset

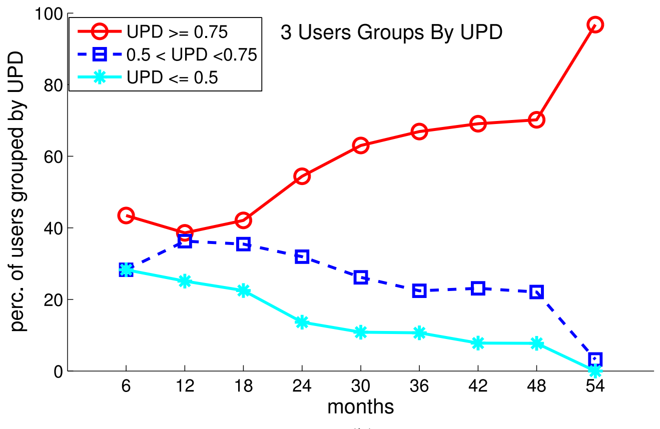
In addition, in our experiments, we used the MovieLens—1M dataset [41], with 1 000 209 anonymous ratings of 3952 movies of $|U| = 6040$ users who joined MovieLens in 2000. The dataset consists of tuples in the form {user, movie, rating, timestamp} over 36 months. Ratings are made on a five-star scale, which correspond to users' viewing events, assuming that users have rated movies after viewing them. The distribution of the viewing events is presented in Fig. 4. Accordingly, we split the dataset into six time periods (time slots $t = S_1 \dots S_6$), corresponding to six semiannuals, where we have $|T| = 6$ different time slices in the tensor, with each slice corresponding to a six-month period. In the MovieLens dataset, the private attributes of users are also available, including age, gender, and 21 occupation types, such as "academic/educator," "lawyer," "doctor/health care," "programmer," and so on. As gender and occupation type are categorical attributes, we used the same transformation technique as in the Last.fm dataset, generating $|D| = 24$ attributes in total, that is, 21, 2, and 1 for occupation, gender, and age attributes, respectively. Moreover, in the MovieLens dataset, we have $|I| = 18$ movie-genres, such as "action," "adventure," "animation," "comedy," "documentary," "thriller," "fantasy," and so on. In the MovieLens dataset, tuples were transformed in the form of {user, movie-genre, time slot, # of viewing events}, corresponding to the number of times a user watched a movie of a certain movie-genre within the time slot. The goal is to perform movie-genre recommendations, instead of movie recommendations, as users rarely watch (interact) the same movie multiple times. Given a set of training months, the goal of the proposed model is to perform top-k movie-genre recommendations for a user at a test month. Similar to the Last.fm dataset, in our experiments, we used a time window equal to a semiannual, where, as training set, we considered all the past months of the previous semiannuals and the first five months of the current ongoing semiannual. The goal is to predict the movie-genre of movies that each user is going to watch at the last (sixth) test month of the current ongoing semiannual. In our experiments, we report AR, with $k = 3$ movie-genres, as in the MovieLens dataset, users watch movies from no more than three different movie genres in a test month.

C. Preference Dynamics

For Last.fm in Fig. 5(a) and (b), we group users from the test months into three different groups based on their: 1) listening events and 2) UPD values according to (10). From Fig. 5(a), we observe that users increase their listening events over time, that is, the percentage of users in group > 300 increases over time. Fig. 5(b) shows the high variability in how users interact with items in the Last.fm dataset. The evolution of the three different groups based on the UPD metric in Fig. 5(b) shows that users tend to significantly shift their preferences over time, as the percentage of users in group UPD ≥ 0.75 is significantly increased over time. According to (10), group UPD ≥ 0.75 contains the users at the test months who have listened to more than a 75% percentage of new artists than they have listened during the past months. An interesting observation is the critical point at 18 months, where users begin to significantly shift their preferences. At this point, the percentage of

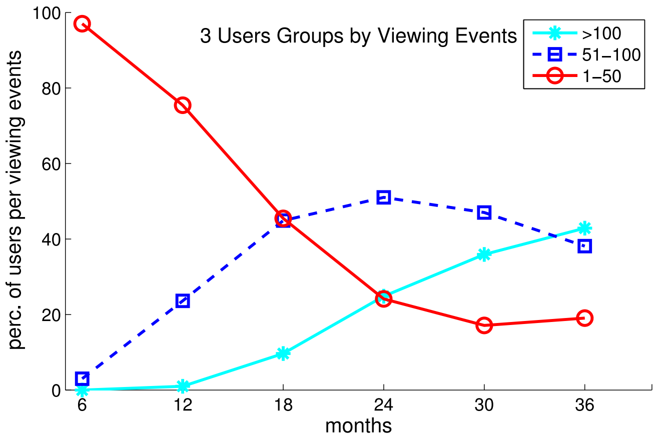


(a)

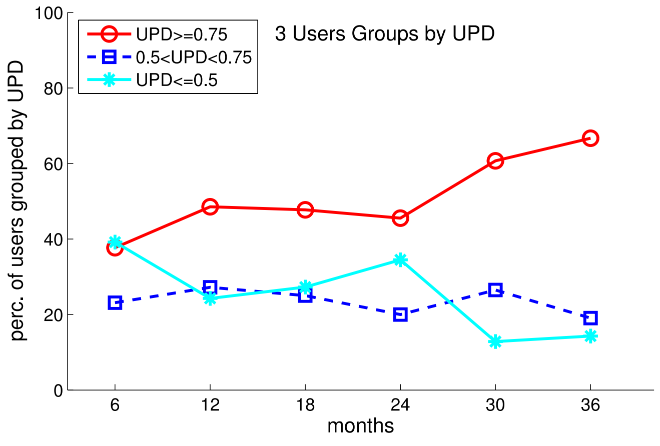


(b)

Fig. 5. Evolution of the three different users groups based on (a) listening events and (b) UPD values in Last.fm.



(a)



(b)

Fig. 6. Evolution of the three different users groups based on (a) viewing events and (b) UPD values in MovieLens.

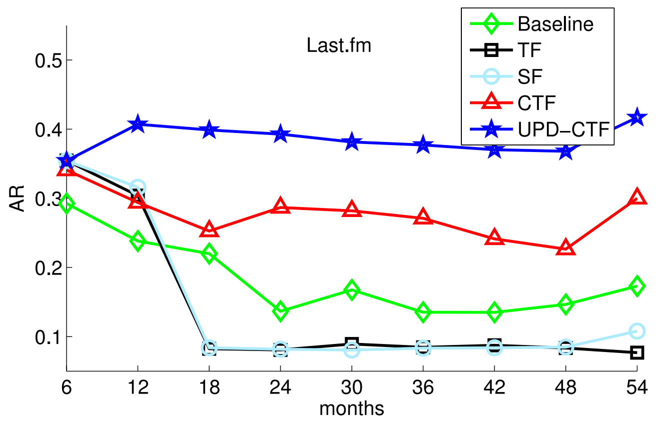
users in group UPD $\geq$ 0.75 starts to significantly increase, whereas the percentages of users in groups UPD $\leq$ 0.5 and $0.5 <$ UPD < 0.75 begin to decrease.

For MovieLens in Fig. 6(a) and (b), we group users from the test months into three different groups based on their viewing events and their UPD values, respectively. From Fig. 6(a), we observe that users increase their viewing events over time (the increase of users in group $>$100). Comparing Fig. 5(b) of Last.fm with Fig. 6(b) of MovieLens, users in the MovieLens dataset have less dynamic taste, as the percentage of users in group UPD $\geq$ 0.75 is not as significantly increased over time as users in group UPD $\geq$ 0.75 of Last.fm. This means that the percentages of users in MovieLens are more fairly distributed to the UPD groups than the respective percentages of users in Last.fm. Nevertheless, for test month 24 in MovieLens, we observe a starting point for a slight increase of the percentage of users in group UPD $\geq$ 0.75 and a decrease of the percentages of users in groups $0.5 <$ UPD < 0.75 and UPD $\leq$ 0.5.

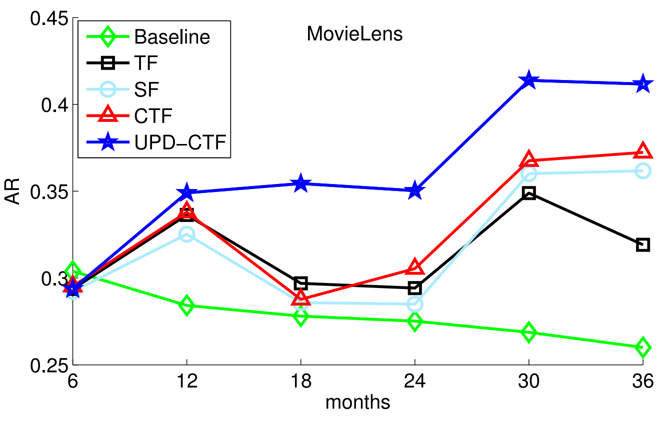
D. Settings

In our experiments, we evaluated the proposed model, by considering: 1) only the modeling of side information of the private attributes with the CP decomposition, thus performing CTF and 2) the combination of the private attributes with the weighting scheme based on UPD in (10) and (11) (UPD-CTF). As competitive methods, we considered TF [14] and SF [16], where in the latter method, we varied the SF from 0.1 to 0.9, concluding to 0.3 for both datasets. Lower values of the SF mean that the SF method downweighs more the past preferences, in our case listening and viewing events for the Last.fm and MovieLens, respectively. As baseline method, we used the most-popular artists/movie-genres method, which recommends the top-k most popular artists/movie-genres for each user in the training months.

Because determining the rank R of the tensor is a NP-hard problem [29], we varied R by 5, 10, 15, and 20, following the related studies of [14], [16], and [18] that use CP decomposition (Section III-C). We concluded to $R = 15$ for both datasets, as a further increase of rank R results in a higher computational cost without paying off in terms of recommendation accuracy. The TF and SF methods have similar computational complexities, whereas CTF and UPD-CTF have higher computational complexities for the required coupling of matrix Y with the auxiliary information of the private attributes. In the Last.fm dataset for the methods: 1) with coupling and 2) without the coupling, the computational times are: 1) 192.79, 400.65, 432.16, and 472.72 s and 2) 48.92, 108.96, 126.28,



(a)



(b)

Fig. 7. Methods comparison in terms of AR for the test months in (a) Last.fm and (b) MovieLens.

and 208.85 s for $R = 5$, 10, 15, and 20, respectively. In the MovieLens dataset, the respective computational times are: 1) 23.6, 38.05, 67.28, and 71.12 s and 2) 11.09, 18.71, 35.63, and 57.11 s. All experiments were performed on a Windows 7 PC with Intel core i7 2700K at 3.50 GHz, 8 GB RAM using MATLAB 2011a.

E. Results

In Fig. 7, we compare the performance of the examined methods in terms of recommendation accuracy (AR). In both datasets, the proposed UPD-CTF method outperforms the competitive methods of CTF, TF, SF, and baseline, by incorporating the auxiliary information of users' private attributes and capturing UPDs in a personalized manner based on the weighting scheme of (10) and (11). Using the paired t-test, we found that the differences between the reported results for the proposed UPD-CTF method against the competitive approaches were statistically significant at the 0.05 level. SF is slightly inferior to TF, as SF uses the SF. Despite the fact that the SF is low (0.3), the SF method works in a nonpersonalized manner, by ignoring users' personalized preferences over time. A significant observation for the Last.fm dataset in Fig. 7(a) is that after the critical point of 18 months, where users begin to significantly shift their preferences (Section VI-C), the recommendation accuracies of TF and SF start to decrease fast, even lower than the baseline method. This happens because users in the Last.fm dataset have very dynamic preferences and both TF and SF neither handle the users' personalized preference dynamics nor consider the users' private attributes. In the MovieLens dataset [Fig. 7(b)], both TF and SF outperform the baseline method, as the users of MovieLens have less dynamic preferences than those of Last.fm (Section VI-C). Additionally, an interesting observation is that despite the fact that CTF does not handle UPDs, the recommendation accuracy is preserved relatively well in both datasets, which means that users' private attributes play a crucial role in recommender systems, also complying with the observations of [21]. However, the proposed UPD-CTF method outperforms the competitive methods in all cases by handling both users' dynamics and side information in a personalized manner.

In Table I, we report the performance of the examined methods for the three users groups based on UPD for the

TABLE I
METHODS PERFORMANCE (AR) FOR THE THREE DIFFERENT USERS GROUPS BASED ON UPD FOR TEST MONTHS 12 AND 54 IN LAST.FM

12 months	UPD ≥ 0.75	$0.5 <$ UPD < 0.75	UPD ≤ 0.5
Perc. users	38.6%	36.28%	25.12%
Baseline	0.21 ± 0.02	0.24 ± 0.09	0.26 ± 0.06
TF	0.30 ± 0.13	0.32 ± 0.08	0.27 ± 0.11
SF	0.31 ± 0.13	0.34 ± 0.09	0.28 ± 0.13
CTF	0.27 ± 0.12	0.31 ± 0.08	0.26 ± 0.12
UPD-CTF	$\mathbf{0.41 \pm 0.11}$	$\mathbf{0.42 \pm 0.06}$	$\mathbf{0.4 \pm 0.08}$
54 months			
Perc. users	96.77%	3.23%	0%
Baseline	0.16 ± 0.09	0.30 ± 0.12	N/A
TF	0.09 ± 0.08	0.06 ± 0.05	N/A
SF	0.11 ± 0.06	0.07 ± 0.07	N/A
CTF	0.32 ± 0.12	0.34 ± 0.08	N/A
UPD-CTF	$\mathbf{0.43 \pm 0.09}$	$\mathbf{0.42 \pm 0.11}$	N/A

test months 12 and 54 in Last.fm. The reason for selecting these two months is that they are before and after the critical point of 18 months, where the UPD begin to change significantly [Fig. 5(b)]. As expected, the baseline method performs badly in the case of users in groups UPD ≥ 0.75 and $0.5 <$ UPD < 0.75 than for users in group UPD ≤ 0.5, as in the latter group users' preferences remain stable over time. Additionally, the proposed UPD-CTF method preserves the recommendation accuracy very well in all three different groups of users based on UPD, by handling the UPD and exploiting their personal attributes. Summarizing, the proposed UPD-CTF method achieves a significant improvement of the recommendation accuracy by up to 10% on average, either in cases of balanced percentages of users in the UPD groups, that is, (12 months) 38.6%, 36.28%, and 25.12% in groups UPD ≥ 0.75, $0.5 <$ UPD < 0.75, and UPD ≤ 0.5, respectively, or in cases of imbalanced percentages, where the majority of users are in group UPD ≥ 0.75, that is, (54 months) 96.77%.

In Table II, we present the respective experimental results based on the three different UPD groups for the test months 18 and 36 of the MovieLens dataset. The reason for selecting these two months in MovieLens is that they are before and after the critical point of 24 months, where the UPD begin to change on average, as presented in Fig. 6(b). The proposed UPD-CTF method achieves a recommendation accuracy improvement between 4% and 9%, compared to the

TABLE II
METHODS PERFORMANCE (AR) FOR THE THREE DIFFERENT USERS GROUPS BASED ON UPD FOR TEST MONTHS 18 AND 36 IN MOVIELENS

18 months	UPD ≥ 0.75	$0.5 <$ UPD < 0.75	UPD ≤ 0.5
Perc. users	47.73%	25%	27.27%
Baseline	0.28 ± 0.01	0.24 ± 0.02	$\mathbf{0.33 \pm 0.03}$
TF	0.34 ± 0.03	0.33 ± 0.02	0.23 ± 0.03
SF	0.31 ± 0.03	0.31 ± 0.03	0.23 ± 0.03
CTF	0.32 ± 0.03	0.32 ± 0.03	0.22 ± 0.03
UPD-CTF	$\mathbf{0.41 \pm 0.03}$	$\mathbf{0.38 \pm 0.03}$	0.26 ± 0.03
36 months			
Perc. users	66.67%	19.05%	14.29%
Baseline	0.23 ± 0.03	0.28 ± 0.02	$\mathbf{0.32 \pm 0.02}$
TF	0.36 ± 0.04	0.28 ± 0.03	0.14 ± 0.02
SF	0.42 ± 0.04	0.28 ± 0.03	0.22 ± 0.03
CTF	0.44 ± 0.04	0.29 ± 0.03	0.22 ± 0.04
UPD-CTF	$\mathbf{0.48 \pm 0.04}$	$\mathbf{0.37 \pm 0.03}$	0.27 ± 0.03

competitive methods for users with high rate of preferences change in groups UPD ≥ 0.75 and $0.5 <$ UPD < 0.75. In contrast to Last.fm, the baseline method outperforms the competitive methods, including UPD-CTF, for users with stable preferences in group UPD ≤ 0.5. However, the percentage of users with stable preferences in group UPD ≤ 0.5 is decreased over time [Fig. 6(b)], for example from 27.27% to 14.29% between 18 and 36 months. Therefore, the proposed UPD-CTF method outperforms the competitive methods for all users in the three different UPD groups on average.

F. Discussion

The main findings of our experimental evaluation are summarized as follows.

1) Our study has demonstrated the existence of preference dynamics when users interact with items over time, that is, the fact that users tend to significantly change preferences in the course of time. Nevertheless, the strength of such a shift may depend on the type of items, for example, songs, movies, and so on, which users interact with in each application. This clearly relates to the amount of exploration that users are willing to perform.
2) Compared to the baseline methods, the proposed approach takes into account UPD in a personal manner, also considering the information about the users' profiles (CTF). It is this combination that allows the proposed UPD-CTF method to outperform the baselines. It is plausible to expect that UPD are related to the users' characteristics in their profiles, for example, gender, age, or education level. Thus, the joint consideration of dynamics and profiles makes sense in this kind of application of recommender systems.

According to the aforementioned findings, the following managerial insights could result in more accurate recommendations.

1) Managers should be aware of the existence and the amount of UPD according to the type of the recommended items. As demonstrated in our experimental study, users tend to change their preferences more dynamically when interacting with songs than with movies (Section VI-C). For managers, it is worth to first investigate the users' behavior and determine the dynamics in their preferences, to assess whether a dynamic approach is required for a specific application type.
2) We have grouped users in three categories according to their preference dynamics. Thus, managers should investigate this kind of "customer segmentation," to identify how many meaningful groups can be considered with respect to UPD and the users' behavior within each group. In different applications, managers may need to explore more than three groups, also requiring examining the relative differences in the dynamics of each group.
3) Additionally, managers should consider in more detail the relation of user characteristics with the aforementioned user groups, that is, according to preference dynamics. It is insightful to investigate specific hypotheses and patterns of user characteristics within each group; for instance, age groups that show more dynamic preferences than others, as also suggested in [21].

Finally, in our experimental study, we used publicly available data to perform an offline evaluation. Nevertheless, it can be expected that the dynamics of user preferences are affected not only by the inherent user characteristics (e.g., age, gender, etc.), but also by the outcomes of the recommender system itself. Clearly, users may modify their preferences accordingly even when following the provided recommendations. The investigation of the role of a recommender system in the formation of user preferences and their dynamics is an interesting topic for future research.

VII. CONCLUSION

The proposed UPD-CTF model captures UPDs and exploits side information with private attributes in a CTF scheme for generating personalized recommendations. In our experiments, we showed that the proposed model achieves a significant improvement in recommendation accuracy when compared against competitive methods, especially in the case of dynamic users, the percentage of whom have increased over time. This happens because our model considers user-item interactions over time, by capturing in a personalized manner, the rate (UPD) with which the current preferences of each user have shifted and exploiting the user's side information in the CTF technique.

In our future study, we plan to extend the proposed model to incremental tensor factorization techniques [34], [35], as for each test time-period, we have to retrain the tensor offline. Additionally, to scale up our CTF technique for millions of users, we plan to implement a parallel version of our UPD-CTF model in a distributed framework, similar to [36] and [37]. However, these studies propose a parallel version of the baseline CP decomposition of Section III-C. Finally, an interesting research direction is to automatically determine the critical points, the points of 18 and 24 months in Last.fm and MovieLens respectively, in our experiments, where after these points users tend to significantly shift their preferences over time.

REFERENCES

[1] K. Tso-Sutter, B. Marinho, and L. Schmidt-Thieme, "Tag-aware recommender systems by fusion of collaborative filtering algorithms," in *Proc. ACM Annu. Symp. Appl. Comput. (SAC)*, Ceará, Brazil, 2008, pp. 1995–1999.

[2] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. Int. Conf. WWW*, Hong Kong, 2001, pp. 285–295.

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

[4] Y. Xu, L. Zhang, and W. Liu, "Cubic analysis of social bookmarking for personalized recommendation," in *Proc. APWeb*, Harbin, China, 2006, pp. 733–738.

[5] S. Rendle, L. B. Marinho, A. Nanopoulos, and L. Schmidt-Thieme, "Learning optimal ranking with tensor factorization for tag recommendation," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, Paris, France, 2009, pp. 727–736.

[6] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "A unified framework for providing recommendations in social tagging systems based on ternary semantic analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 2, pp. 179–192, Feb. 2010.

[7] A. Nanopoulos, "Item recommendation in collaborative tagging systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 41, no. 4, pp. 760–771, Jul. 2011.

[8] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Min. (KDD)*, Paris, France, 2009, pp. 447–456.

[9] L. Xiang and Q. Yang, "Time-dependent models in collaborative filtering based recommender system," in *Proc. IEEE/WIC/ACM Int. Joint Conf. Web Intell. Intell. Agent Technol.*, vol. 1. Milan, Italy, 2009, pp. 450–457.

[10] M. Kendall and K. D. Gibbons, *Rank Correlation Methods*. New York, NY, USA: Oxford Univ. Press, 1990.

[11] N. Koenigstein, G. Dror, and Y. Koren, "Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy," in *Proc. ACM RecSys*, Chicago, IL, USA, 2011, pp. 165–172.

[12] D. G. F. M. Bollen, M. P. Graus, and M. C. Willemsen, "Remembering the stars? Effect of time on preference retrieval from memory," in *Proc. ACM RecSys*, Dublin, Ireland, 2012, pp. 217–220.

[13] N. Lathia, S. Hailes, L. Capra, and X. Amatriain, "Temporal diversity in recommender systems," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retriev.*, Geneva, Switzerland, 2010, pp. 210–217.

[14] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM Trans. Knowl. Discov. Data*, vol. 5, no. 2, p. 10, Feb. 2011.

[15] L. Xiong, X. Chen, T.-K. Huang, J. G. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with Bayesian probabilistic tensor factorization," in *Proc. Int. Conf. Data Mining (SIAM)*, Columbus, OH, USA, 2010, pp. 211–222.

[16] S. Spiegel, J. H. Clausen, S. Albayrak, and J. Kunegis, "Link prediction on evolving data using tensor factorization," in *Proc. IEEE Pac.-Asia Conf. Knowl. Discov. Data Mining (PAKDD)*, Shenzhen, China, 2012, pp. 100–110.

[17] A. Das, S. Gollapudi, and K. Munagala, "Modeling opinion dynamics in social networks," in *Proc. ACM Int. Conf. Web Search Data Mining*, New York, NY, USA, pp. 403–412, 2014.

[18] E. Acar, T. G. Kolda, and D. M. Dunlavy, "All-at-once optimization for coupled matrix and tensor factorizations," unpublished paper. [Online]. Available: http://arxiv.org/abs/1105.3422

[19] B. Ermis, E. Acar, and A. T. Cemgil, "Generalized coupled symmetric tensor factorization for link prediction," in *Proc. IEEE Conf. Signal Process. Commun. Applicat.*, Haspolat, Turkey, 2013, pp. 1–4.

[20] A. Narita, K. Hayashi, R. Tomioka, and H. Kashima, "Tensor factorization using auxiliary information," in *Proc. Eur. Conf. Mach. Learn. Principles / Practice Knowl. Discov. Databases (ECML/PKDD)*, Athens, Greece, 2011, pp. 501–516.

[21] J. Beel, S. Langer, A. Nürnberger, and M. Genzmehr, "The impact of demographics (age and gender) and other user-characteristics on evaluating recommender systems," in *Research and Advanced Technology for Digital Libraries*. Lecture Notes in Computer Science, vol. 8092. Berlin, Germany: Springer, 2013, pp. 396–400.

[22] D. Rafailidis and A. Nanopoulos, "Modeling the dynamics of user preferences in coupled tensor factorization," in *Proc. ACM RecSys*, Foster, CA, USA, 2014, pp. 321–324.

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

[24] K. Stefanidis, I. Ntoutsi, K. Nørvåg, and H.-P. Kriegel, "A framework for time-aware recommendations," in *Proc. Database Expert Syst. Applicat. (DEXA)*, Vienna, Austria, 2012, pp. 329–344.

[25] H. Gao, J. Tang, X. Hu, and H. Liu, "Exploring temporal effects for location recommendation on location-based social networks," in *Proc. ACM RecSys*, Hong Kong, 2013, pp. 93–100.

[26] K. Wang *et al.*, "Time-aware travel attraction recommendation," in *Web Information Systems Engineering (WISE) 2013*. Lecture Notes in Computer Science, vol. 8180. Berlin, Germany: Springer, 2013, pp. 175–188.

[27] W. Liu, J. Chan, J. Bailey, C. Leckie, and K. Ramamohanarao, "Utilizing common substructures to speedup tensor factorization for mining dynamic graphs," in *Proc. ACM Conf. Inf. Knowl. Manage. (CIKM)*, Maui, HI, USA, 2012, pp. 435–444.

[28] J. Sun, D. Tao, and C. Faloutsos, "Beyond streams and graphs: Dynamic tensor analysis," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Philadelphia, PA, USA, 2006, pp. 374–383.

[29] J. Håstad, "Tensor rank is NP-complete," *J. Algorithms*, vol. 11, no. 4, pp. 644–654, Dec. 1990.

[30] E. Acar, T. G. Kolda, D. M. Dunlavy, and M. Mrup, "Scalable tensor factorizations for incomplete data," *Chemometr. Intell. Lab. Syst.*, vol. 106, no. 1, pp. 41–56, Mar. 2011.

[31] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 2006.

[32] K. Tso-Sutter, L. Marinho, and L. Schmidt-Thieme, "Tag-aware recommender systems by fusion of collaborative filtering algorithms," in *Proc. ACM Annu. Symp. Appl. Comput. (SAC)*, Fortaleza, Brazil, 2008, pp. 1995–1999.

[33] D. Rafailidis and P. Daras, "The TFC model: Tensor factorization and tag clustering for item recommendation in social tagging systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 3, pp. 673–688, May 2013.

[34] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos, "Incremental tensor analysis: Theory and applications," *ACM Trans. Knowl. Disc. Data*, vol. 2, no. 3, pp. 11:1–11:37, 2008.

[35] W. Hu, X. Li, X. Zhang, X. Shi, S. J. Maybank, and Z. Zhang, "Incremental tensor subspace learning and its applications to foreground segmentation and tracking," *Int. J. Comput. Vis.*, vol. 91, no. 3, pp. 303–327, 2011.

[36] U. Kang, E. E. Papalexakis, A. Harpale, and C. Faloutsos, "Gigatensor: Scaling tensor analysis up by 100 times—Algorithms and discoveries," in *Proc. ACM SIGKDD Conf. Knowl. Disc. Data Min. (KDD)*, Beijing, China, 2012, pp. 316–324.

[37] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Parcube: Sparse parallelizable tensor decompositions," in *Machine Learning and Knowledge Discovery in Databases*. Lecture Notes in Computer Science, vol. 7523. Berlin, Germany: Springer, 2012, pp. 521–536.

[38] Tensor Toolbox, [Online]. Available: http://www.sandia.gov/∼tgkolda/TensorToolbox/index-2.5.html, accessed Jul. 30, 2015.

[39] CMTF Toolbox, [Online]. Available: http://www.models.life.ku.dk/joda/CMTF_Toolbox/, accessed Jul. 30, 2015.

[40] The Last.fm-1K Dataset, [Online]. Available: http://www.dtic.upf.edu/∼ocelma/MusicRecommendationDataset/, accessed Jul. 30, 2015.

[41] The MovieLens-1M Dataset, [Online]. Available: http://grouplens.org/datasets/movielens/, accessed Jul. 30, 2015.

Dimitrios Rafailidis was born in Larisa, Greece, in 1982. He received the B.Sc., M.Sc., and Ph.D. degrees from the Department of Informatics of the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2005, 2007, and 2011, respectively.

He is a Post-Doctoral Research Fellow with the Department of Informatics, Aristotle University of Thessaloniki and the Department of Computer Science, University of Cyprus, Nicosia, Cyprus. His current research interests include databases, multimedia information retrieval, social media mining, and parallel and cloud computing.

Alexandros Nanopoulos received the B.Sc. and Ph.D. degrees from the Department of Informatics of the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1997 and 2003, respectively.

He was a Lecturer with the Department of Informatics, Aristotle University of Thessaloniki, from 2004 to 2008. He was an Assistant Professor with the University of Hildesheim, Hildesheim, Germany, from 2008 to 2013. He is currently an Assistant Professor with the Catholic University of Eichstaett-Ingolstadt, Ingolstadt, Germany. His current research interests include machine learning, data mining, and Web information retrieval. He has coauthored over 120 articles in international journals and conference proceedings.