Expert Systems With Applications 104 (2018) 97–106

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A Multi-Latent Transition model for evolving preferences in recommender systems

D. Rafailidis

Department of Computer Science, University of Mons, Mons, 7000, Belgium

ARTICLE INFO

Article history:
Received 24 October 2017
Revised 12 February 2018
Accepted 19 March 2018
Available online 20 March 2018

Keywords:
Recommender systems
Preference dynamics
Multi-latent analysis

ABSTRACT

In recommender systems users interact with items, while their preferences evolve over time. The challenge is how to identify the correlation between users' recent and past preferences to generate accurate recommendations. In this study, we propose a Multi-Latent Transition (MLT) model. We formulate a joint objective function to calculate the multiple transitions between an ongoing period with users' latest preferences and all the past ones, considering the multiple transitions at the user latent space of the different periods. The joint problem is solved via an efficient gradient-based alternating optimization algorithm, with convergence guarantees. Furthermore, to better capture the correlation between the ongoing period and a past one we also exploit items' metadata, accounting for the fact that users may have stable preferences over time as they may like certain attributes of items e.g., an actor or a movie director, or radically shift their preferences because they dislike them. Our experiments show that MLT significantly outperforms state-of-the art methods and boosts the recommendation accuracy for users with stable preferences and for users that tend to shift their preferences often.

1. Introduction

The collaborative filtering strategy has been widely adopted in recommender systems, where users with similar preferences tend to get similar recommendations (Sarwar, Karypis, Konstan, & Riedl, 2001). Latent models, such as matrix and tensor factorization techniques, are representative collaborative filtering strategies which factorize the data with user preferences, to reveal the latent associations between users and items (Kolda & Bader, 2009; Koren, 2008; Lee & Seung, 2000; Salakhutdinov & Mnih, 2007). In real-world recommender systems, users tend to change their behavior over time, as their preferences evolve. Users' evolving preferences have a major impact on recommender systems because they limit the recommendation accuracy of matrix and tensor factorization strategies as the recent users' preferences are ignored (Zhang, Wang, Yu, Sun, & Lim, 2014).

In an attempt to capture users' preference dynamics various latent models have been proposed, such as matrix factorization strategies. Koren (2009) tries to identify both the short-term preferences based on users' recent selections, and the long term preferences based on users' cumulative history record, that is persistent preferences that users have throughout the data time span. This method aims to capture the temporal effects by incorporating a time-variant bias for each user and item at every individual time period. Zhang et al. (2014) consider evolving preferences and model user dynamics by learning a transition matrix for each user's latent vector between consecutive time periods. In particular, this model assumes that user preferences evolve *gradually*, that is the preferences of a user j at a time period t depend on her preferences at an immediately previous period $t-1$. This method introduces a time-invariant transition matrix to capture the preferences' pattern evolution over time. However, users' preferences do not necessarily evolve gradually but can also evolve radically between two consecutive time periods, meaning that at an ongoing time period users can have either similar or completely different preferences, compared to the ones they had at an immediately previous period. Instead of considering users' short- and long-term preferences, it is important to identify the correlation between users' recent and past preferences. Meanwhile, tensor factorization strategies model tuples in the form {user, item, time period}, by modeling the time periods as slices into the tensor (Dunlavy, Kolda, & Acar, 2011; Xiong, Chen, Huang, Schneider, & Carbonell, 2010). Although these strategies try to capture the three-way correlations of user-item-time period, they compute the pairwise correlations between all slices/time periods, when factorizing the tensor. Instead of focusing on the pairwise correlations of past time periods with the ongoing one, they also compute the pairwise correlations between the past periods. As we will experimentally show in Section 5, this may introduce noise and limit the recommenda-

E-mail address: dimitrios.rafailidis@umons.ac.be

https://doi.org/10.1016/j.eswa.2018.03.033
0957-4174/© 2018 Elsevier Ltd. All rights reserved.

tion accuracy throughout the data time span. In addition, all the above studies following matrix or tensor factorization strategies do not account for the fact that user preferences may evolve or remain stable based on the items' metadata. More concretely, users may have constant preferences over time as they may like an actor or a movie director, or change their preferences significantly because they dislike them. While several factorization schemes incorporate auxiliary information such as items' metadata (Saveski & Mantrach, 2014) or social relationships (Jamali & Ester, 2010), these schemes aim at solving the data sparsity that occurs in recommender systems and do no concentrate on preferences' evolution. Meanwhile, context-aware recommender systems (Rendle, 2012; Shi, Karatzoglou, Baltrunas, Larson, & Hanjalic, 2014) utilize contextual factors to generate recommendations and do not consider evolving preferences as well.

1.1. Contribution

To overcome the limitations of existing latent models, we propose a Multi-latent Transition (MLT) model including the following contributions:

- Given in total k time periods, MLT performs a multi-latent transition analysis, computing the $k-1$-way correlations between users' recent preferences at an ongoing period and the past preferences of the previous ones. We formulate a joint objective function to simultaneously calculate the $k-1$ transitions at the users' latent space and the factor matrices of the ongoing period to generate recommendations. Also, given m users and n items we define an ℓ_1-norm regularization in our objective function, with $||X||_1 = \max_{1\leq j\leq n} \sum_{i=1}^{m} |X_{ij}|$, to force the factor matrices to be sparse, as users' feedback on items is limited at a certain time period. The joint problem is solved via an efficient gradient-based alternating optimization algorithm.
- To better capture users' evolving preferences we consider items' metadata when calculating the correlations of users' preferences in different time periods.
- We extend two benchmark datasets from MovieLens[1] and Last.fm[2] to include items' metadata and we make the extended versions publicly available. Our experimental results demonstrate the superiority of MLT over other state-of-the-art methods throughout the data time span. Also, we show that MLT preserves high recommendation accuracy for users that have either stable or evolving preferences.

1.2. Outline

The remainder of the paper is organized as follows, in Section 2 we formally define our problem and Section 3 reviews the related work. Section 4 details the proposed MLT model, Section 5 presents the experimental results and Section 6 concludes the study.

2. Problem statement

Table 1presents the main notation used in the sequel of the paper. We split the data time span into k non-overlapping time periods. The choice of time periods, for example days, weeks or months depends on how often the recommendations are generated. Let t be the present ongoing time period, with users' more recent preferences. For example, we assume that the current semi-annual is the ongoing period. In the first 5 months we have users' most recent preferences and the goal is to generate the recommendations for the last (6th) month of the ongoing semi-annual. Also, let $t-1$ be the last time period (the previous semi-annual), and $t-p$ denote a $t-p$ past period. Given a total period of $k=10$ semi-annuals, the ongoing period t is the 10th semi-annual and the past time periods $t-p$ are the past semi-annuals, with $p = 1,\ldots,9$. Given m items and u users, we assume that users express their preferences on items with different types of feedback, such as explicit feedback e.g., ratings, tag assignments or implicit one e.g., number of views, listening events, clicks, comments and so on. Users' feedback at the ongoing time period t is stored in a matrix $X^{(t)} \in \mathbb{R}^{m\times n}$. Also, for the past periods $t-p$, with $p=1,\ldots,k-1$, users' feedback is stored in the respective $k-1$ matrices $X^{(t-p)}$. Notice that a matrix $X^{(t-p)}$ only contains users' feedback in the period $t-p$ and not the cumulative history from the beginning of the data time span until period $t-p$.

Following the Non-Negative Matrix Factorization (NMF) technique, introduced in Lee and Seung (2000), recommendations based on users' recent preferences are generated in the ongoing time period t by factorizing matrix $X^{(t)}$. This is achieved by decomposing $X^{(t)}$ as follows:

$$X^{(t)} \approx Z^{(t)} V^{(t)} \tag{1}$$

$$\text{subject to} Z^{(t)}, V^{(t)} \geq 0$$

$Z^{(t)} \in \mathbb{R}^{m\times d}$ and $V^{(t)} \in \mathbb{R}^{d\times n}$ are the factor matrices of items and users, respectively, and d is the number of latent factors. The ith row of $Z^{(t)}$ and the jth column of $V^{(t)}$ express the d-dimensional latent vectors of item i and user j, accordingly. The product $Z^{(t)}V^{(t)}$ results in a factorized matrix $\hat{X}^{(t)}$, the low rank d approximation of $X^{(t)}$, with $X^{(t)} \approx \hat{X}^{(t)}$. In our setting, we have to calculate the factorized matrix $\hat{X}^{(t)} = Z^{(t)}V^{(t)}$, by also considering the evolution of user preferences, that is the transition of user preferences between a past $t-p$ and the ongoing time period t. Provided that in total there are $k-1$ past time periods, the goal is to calculate the $k-1$ users' preference transitions between the respective past periods $t-p$ and the ongoing time period t. Our problem is formally defined as follows:

"Given users' feedback on items (i) in matrix $X^{(t)}$ of the ongoing period with users' recent preferences and (ii) the $k-1$ matrices $X^{(t-p)}$ of the respective past periods $t-p$, with $p=1,\ldots,k-1$, the goal of the proposed approach is to compute the factorized matrix $\hat{X}^{(t)}$, by capturing the evolution of users' preferences in the respective $k-1$ transitions between the ongoing t and the $k-1$ past periods."

3. Related work

3.1. Dynamic collaborative filtering

Koren (2009) introduces the timeSVD++ algorithm to capture the lasting and transient factors by modeling the users' preference dynamics throughout the entire time period. Zhang et al. (2014) present two models which capture users' preference dynamics, namely the Temporal Matrix Factorization (TMF) and the Bayesian Temporal Matrix Factorization (BTMF) methods. Using the Probabilistic Matrix Factorization technique (Salakhutdinov & Mnih, 2007), TMF maps users' preferences on items into a joint latent factor with a transition matrix, which captures the users' preference dynamics between two consecutive time periods. They sample the distribution from users' feedback on items, to update the transition matrix for both past and future time periods. BTMF extends TMF by introducing priors for the hyper parameters to significantly increase the accuracy and deals with the complexity of TMF. Dunlavy et al. (2011) consider bipartite graphs with tuples in the form {author, conference, relationships} that

[1] https://movielens.org/.
[2] https://www.last.fm/.

Table 1
Notation.

Symbol	Description
m, n, d, k	Number of items, users, latent factors and time periods
$X^{(t)}, X^{(t-p)} \in \mathbb{R}^{m\times n}$	Item-user matrices in the ongoing time period t and a past time period $t-p$, with $p = 1, \dots, k-1$
$\hat{X}^{(t)}, \hat{X}^{(t-p)} \in \mathbb{R}^{m\times n}$	Factorized matrices of $X^{(t)}$ and $X^{(t-p)}$
$Z^{(t)} \in \mathbb{R}^{m\times d}, V^{(t)} \in \mathbb{R}^{d\times n}$	Item and user factor matrices of $X^{(t)}$
$\mathbf{w}^{(t-p)} \in \mathbb{R}^{n}$	Smoothing user vector for a past time period $t-p$
$T^{(t-p)} \in \mathbb{R}^{d\times d}$	Latent transition matrix between the ongoing and past user factor matrices $V^{(t)}$ and $V^{(t-p)}$
λ	l_1-norm regularization parameter

evolve over time, to recommend a future conference to an author. In this work, the CANDECOMP/PARAFAC (CP) decomposition (Kolda & Bader, 2009) of a tensor is used, denoting if author a is linked to a conference c at time t. The goal is to predict a new link at a future time, by exploring the three-dimensional structure of temporal data. Liu, Chan, Bailey, Leckie, and Kotagiri (2012) propose an iterative tensor factorization technique with linear time complexity for mining time-evolving graph data. Xiong et al. (2010) add constraints on the time dimension to a Bayesian Probabilistic Tensor Factorization (BPTF) model, in order to process time-evolving data and generate sales prediction and movie recommendation. Gao, Tang, Hu, and Liu (2013) present a model for location-based recommendation according to users' personal preferences and facilitating exploration of new areas of a city. This is achieved by exploring temporal patterns, modeling thus temporal effects on location-based social networks for location recommendation. Wang et al. (2013) model tuples {tourist, time, location} into tensors, in order to perform time-aware recommendations for travel destinations, by maximizing the temporal-spatial correlation for tourists. The proposed recommendation model tries to predict the items and their best corresponding temporal conditions. However, both methods at Gao et al. (2013) and Wang et al. (2013) focus on spatial and temporal information in the data, while ignoring the preferences' evolution in recommender systems.

3.2. Factorization with auxiliary information and context-aware recommendations

Several factorization schemes utilize auxiliary information, such as Coupled Matrix-Tensor Factorization (Acar, Kolda, & Dunlavy, 2011) and Collective Matrix Factorization (Singh & Gordon, 2008). Other studies exploit users' feedback from different domains or platforms to generate cross-domain recommendations (Cremonesi, Tripodi, & Turrin, 2011). In addition, models exploit items' metadata, such as the Local Collective Embeddings model (Saveski & Mantrach, 2014). Several studies exploit the selections of social friends, assuming that users trust the selections of their friends, such as the study reported in Jamali and Ester (2010). Nonetheless, all the above studies aim at solving the data sparsity that occurs in recommender systems and do not account for temporal dynamics. Context-aware recommender systems exploit contextual information such as time, location or social relationships (Liu, He, & Zhao, 2013; Yuan et al., 2016). For example, Yin, Cui, Chen, Hu, and Zhou (2015) point out that users' behavior is influenced by two factors: users' preferences and social preferences. Also, they show that both factors have different degree of influence on user behavior. To identify the users' behavior affected by both factors they present the Temporal Context-Aware Recommender System model, which jointly exploits both the users' preferences and their social preferences at different time periods. To leverage the context which users or items are associated to and generate context-aware recommendations, Factorization Machines have gained much popularity, integrating sparse context and exploration of context interaction (Rendle, 2010; 2012; Rendle, Gantner, Freudenthaler, & Schmidt-Thieme, 2011). Focusing on the top-N context-aware recommendation problem, various ranking-based Factorization Machines algorithms have been proposed such as the studies reported in Qiang, Liang, and Yang (2013), Shi et al. (2014, 2012) and Yuan et al. (2016). However, here we focus on temporal dynamics of users' preferences, while the contextual factors are out of this paper's scope and are left for future work.

4. Proposed model

The proposed model is divided into three steps: (i) in Section 4.1 we present the smoothing strategy to compute the preferences' shift rate for each individual; (ii) then, in Section 4.2 we formulate our joint objective function; (iii) finally, in Section 4.3 we detail our optimization strategy and illustrate our algorithm.

4.1. Smoothing strategy

Let $\mathcal{I}_j^{(t)}$ denote the set of items that a user j has expressed her preferences in the ongoing period t, with $j = 1, \dots, n$. Accordingly let $\mathcal{I}_j^{(t-p)}$ be the set of items of the same user j in a past period $t-p$. To calculate the preferences' shift rate between the ongoing period t and a past one $t-p$, we define a smoothing vector $\mathbf{w}^{(t-p)} \in \mathbb{R}^n$, where its jth element is computed as follows:

$$w_j^{(t-p)} = \frac{\sum_{a=1}^{|\mathcal{I}_j^{(t)}|}\sum_{b=1}^{|\mathcal{I}_j^{(t-p)}|} s(a,b)}{\min\left(|\mathcal{I}_j^{(t)}|, |\mathcal{I}_j^{(t-p)}|\right)} \tag{2}$$

where $s(a, b)$ is the text similarity of two movies'/artists' metadata. In our implementation we used Apache Lucene[3] to calculate the text-based similarities. A high similarity in $w_j^{(t-p)}$ reflects on users' stable preferences between the past period $t-p$ and the ongoing one t. On the other hand, a low value of $w_j^{(t-p)}$ corresponds to a dynamic user j at the ongoing period.

4.2. Objective function

To ease the presentation, we consider $k = 3$ time periods, the ongoing t, and $p = 2$ past ones, i.e., $t-1$ and $t-2$. Given the matrices $X^{(t)}$ of the ongoing period, and $X^{(t-1)}$ and $X^{(t-2)}$ of the two past periods, the goal is to generate recommendations in the low rank d approximation of $X^{(t)}$ with users' recent preferences, by also accounting the $p = 2$ transitions of the past periods. First, we downweigh the past preferences in a personalized manner, by using the respective smoothing vectors $\mathbf{w}^{(t-1)}$ and $\mathbf{w}^{(t-2)}$ (Eq. (2)) as follows:

$\forall j = 1, \dots, n$

$$(X^{(t-1)})_{*,j} \leftarrow w_j^{(t-1)} \cdot (X^{(t-1)})_{*,j} \tag{3}$$

$$(X^{(t-2)})_{*,j} \leftarrow w_j^{(t-2)} \cdot (X^{(t-2)})_{*,j} \tag{4}$$

[3] https://lucene.apache.org/.

where $(X)_{*,j}$ expresses the jth column of matrix X. Based on the minimization problem of NFM in Eq. (1), the NMFs of matrix $X^{(t)}$ and the two downweighted matrices $X^{(t-1)}$ and $X^{(t-2)}$ correspond to the following three minimization problems:

$$\min_{Z^{(t)},V^{(t)}} ||X^{(t)} - Z^{(t)}V^{(t)}||_F^2 \tag{5}$$

$$\min_{Z^{(t-1)},V^{(t-1)}} ||X^{(t-1)} - Z^{(t-1)}V^{(t-1)}||_F^2 \tag{6}$$

$$\min_{Z^{(t-2)},V^{(t-2)}} ||X^{(t-2)} - Z^{(t-2)}V^{(t-2)}||_F^2 \tag{7}$$

subject to $Z^{(t)}, V^{(t)}, Z^{(t-1)}, V^{(t-1)}, Z^{(t-2)}, V^{(t-2)} \geq 0$

where $||\cdot||_F$ denotes the Frobenius norm. To capture the evolution of user preferences between the time periods t and $t-1$, as well as the evolution of t and $t-2$, we introduce $p=2$ latent transition matrices, assuming that the transitions are in the users latent spaces. The first latent transition matrix is denoted by $T^{(t-1)} \in \mathbb{R}^{d\times d}$, expressing the evolution of preferences between the user latent matrices $V^{(t)}$ and $V^{(t-1)}$ in the time periods t and $t-1$. The latent transition matrix $T^{(t-1)}$ captures to which extent the current/ongoing user latent matrix $V^{(t)}$ can be expressed by the past user latent matrix $V^{(t-1)}$. Similarly, we define a second latent transition matrix $T^{(t-2)} \in \mathbb{R}^{d\times d}$ between the user latent matrices $V^{(t)}$ and $V^{(t-2)}$ in the time periods t and $t-2$. Using the latent transitions matrices $T^{(t-1)}$ and $T^{(t-2)}$ Eq. (5) can be reformulated as the following minimization problems:

$$\min_{Z^{(t)},V^{(t-1)}} ||X^{(t)} - Z^{(t)}T^{(t-1)}V^{(t-1)}||_F^2 \tag{8}$$

$$\min_{Z^{(t)},V^{(t-2)}} ||X^{(t)} - Z^{(t)}T^{(t-2)}V^{(t-2)}||_F^2 \tag{9}$$

By combining Eqs. (5), (8) and (9) we define the following *joint objective function* with respect to matrices/variables $Z^{(t)}$, $V^{(t)}$, $T^{(t-1)}$ and $T^{(t-2)}$:

$$\begin{aligned}\min_{Z^{(t)},V^{(t)},T^{(t-1)},T^{(t-2)}} \mathfrak{L} = {} & ||X^{(t)} - Z^{(t)}V^{(t)}||_F^2 \\ & + ||X^{(t)} - Z^{(t)}T^{(t-1)}V^{(t-1)}||_F^2 \\ & + ||X^{(t)} - Z^{(t)}T^{(t-2)}V^{(t-2)}||_F^2 \\ & + ||T^{(t-1)} - I||_F^2 + ||T^{(t-2)} - I||_F^2 \\ & + \lambda(||Z^{(t)}||_1 + ||V^{(t)}||_1 + ||T^{(t-1)}||_1 + ||T^{(t-2)}||_1)\end{aligned} \tag{10}$$

subject to $Z^{(t)}, V^{(t)}, V^{(t-1)}, V^{(t-2)} \geq 0$

where $I \in \mathbb{R}^{d\times d}$ is the identity matrix, and $||\cdot||_1$ denotes the l_1-norm. The fourth line in Eq. (10) denotes the approximation errors between the transitions and identity matrices, assuming that the $p=2$ transitions at the user latent space should be close to the identity matrix, as the respective item-user matrices have been downweighted based on Eqs. (3)-(4). In the fifth line, the terms express the ℓ_1-norm reguralizations of the variables/matrices, forcing them to be sparse (Schmidt, Fung, & Rosales, 2007), as the latent transitions are expected to be close to the identity matrices and thus sparse. The item and user factor matrices are also expected to be sparse, mainly because users' feedback on items is limited at a period t. Parameter λ controls the influence of the ℓ_1-norm regularizers, with higher values forcing the variables/matrices to be more sparse.

4.3. Model learning

As the joint objective function $\mathcal{L}$ in Eq. (10) is a non-convex function with respect to the four variables/matrices $Z^{(t)}$, $V^{(t)}$, $T^{(t-1)}$ and $T^{(t-2)}$, we propose an alternating optimization algorithm based on the strategy of multiplicative update rules (Jialu, Chi, Gao, & Jiawei, 2013; Lee & Seung, 1999), where we update one variable, while keeping the remaining three fixed. Using the Karush-Kuhn Tucker (KKT) conditions (Kuhn & Tucker, 1951) we have the following inequality constraints for the four variables/matrices and their respective gradients:

$$Z^{(t)} \geq 0, \quad V^{(t)} \geq 0, \quad T^{(t-1)} \geq 0, \quad T^{(t-2)} \geq 0 \tag{11}$$

$$\nabla_{Z^{(t)}}\mathcal{L} \geq 0, \quad \nabla_{V^{(t)}}\mathcal{L} \geq 0, \quad \nabla_{T^{(t-1)}}\mathcal{L} \geq 0, \quad \nabla_{T^{(t-2)}}\mathcal{L} \geq 0 \tag{12}$$

$$\begin{aligned}& Z^{(t)} \odot \nabla_{Z^{(t)}}\mathcal{L} = 0, \quad V^{(t)} \odot \nabla_{V^{(t)}}\mathcal{L} = 0, \\ & T^{(t-1)} \odot \nabla_{T^{(t-1)}}\mathcal{L} = 0, \quad T^{(t-2)} \odot \nabla_{T^{(t-2)}}\mathcal{L} = 0\end{aligned} \tag{13}$$

where $\odot$ denotes the element-wise product. The gradients of the joint objective function $\mathcal{L}$ in Eq. (10) with respect to each variable are equivalent to:

$$\begin{aligned}\nabla_{Z(t)}\mathcal{L} = {} & Z^{(t)}\left(V^{(t)}V^{(t)\top} + \sum_{p=1}^{2}\left[T^{(t-p)}V^{(t-p)}V^{(t-p)\top}T^{(t-p)\top}\right]\right) \\ & - \left(X^{(t)}V^{(t)\top} + \sum_{p=1}^{2}\left[X^{(t)}V^{(t-p)\top}T^{(t-p)\top}\right] - \lambda\right)\end{aligned} \tag{14}$$

$$\nabla_{V^{(t)}}\mathcal{L} = Z^{(t)\top}Z^{(t)}V^{(t)} - (Z^{(t)\top}X^{(t)} - \lambda) \tag{15}$$

$$\begin{aligned}\nabla_{T^{(t-1)}}\mathcal{L} = {} & V^{(t-1)}V^{(t-1)\top}T^{(t-1)\top}Z^{(t)\top}Z^{(t)}T^{(t-1)\top} \\ & -(V^{(t-1)}X^{(t)\top}Z^{(t)} + I - \lambda)\end{aligned} \tag{16}$$

$$\begin{aligned}\nabla_{T^{(t-2)}}\mathcal{L} = {} & V^{(t-2)}V^{(t-2)\top}T^{(t-2)\top}Z^{(t)\top}Z^{(t)}T^{(t-2)\top} \\ & -(V^{(t-2)}X^{(t)\top}Z^{(t)} + I - \lambda)\end{aligned} \tag{17}$$

Based on the gradients in Eqs. (14)–(17), by solving the conditions of Eq. (13) with respect to each variable, the following updating rules are derived:

$$Z^{(t)} \leftarrow Z^{(t)} \odot \frac{X^{(t)}V^{(t)\top} + \sum_{p=1}^{2}\left[X^{(t)}V^{(t-p)\top}T^{(t-p)\top}\right] - \lambda}{Z^{(t)}\left(V^{(t)}V^{(t)\top} + \sum_{p=1}^{2}\left[T^{(t-p)}V^{(t-p)}V^{(t-p)\top}T^{(t-p)\top}\right]\right)} \tag{18}$$

$$V^{(t)} \leftarrow V^{(t)} \odot \frac{Z^{(t)\top}X^{(t)} - \lambda}{Z^{(t)\top}Z^{(t)}V^{(t)}} \tag{19}$$

$$T^{(t-1)} \leftarrow T^{(t-1)} \odot \frac{V^{(t-1)}X^{(t)\top}Z^{(t)} + I - \lambda}{V^{(t-1)}V^{(t-1)\top}T^{(t-1)\top}Z^{(t)\top}Z^{(t)}T^{(t-1)\top}} \tag{20}$$

$$T^{(t-2)} \leftarrow T^{(t-2)} \odot \frac{V^{(t-2)}X^{(t)\top}Z^{(t)} + I - \lambda}{V^{(t-2)}V^{(t-2)\top}T^{(t-2)\top}Z^{(t)\top}Z^{(t)}T^{(t-2)\top}} \tag{21}$$

So far, we have analyzed the case of having a current period and two past ones. Accordingly, we can extend the joint objective function in Eq. (10) for $p = k-1$ past periods as follows:

$$\min_{Z^{(t)},V^{(t)},T^{(t-1)},\ldots,T^{(1)}} \mathfrak{L} = ||X^{(t)} - Z^{(t)}V^{(t)}||_F^2 + \sum_{p=1}^{k-1}\left[||X^{(t)} - Z^{(t)}T^{(t-p)}V^{(t-p)}||_F^2\right] + \sum_{p=1}^{k-1}||T^{(t-p)} - I||_F^2 + \lambda\left(||Z^{(t)}||_1 + ||V^{(t)}||_1 + \sum_{p=1}^{k-1}||T^{(t-p)}||_1\right) \tag{22}$$

It is easy to verify that the respective updating rule for a latent transition matrix $T^{(t-p)}$ is equal to:

$$T^{(t-p)} \leftarrow T^{(t-p)} \odot \frac{V^{(t-p)}X^{(t)^\top}Z^{(t)} + I - \lambda}{V^{(t-p)}V^{(t-p)^\top}T^{(t-p)^\top}Z^{(t)^\top}Z^{(t)}T^{(t-p)^\top}} \tag{23}$$

Algorithm 1: MLT algorithm.

```
   Input: $X^{(t)}$, ..., $X^{(t-p)}$, ..., $X^{(1)}$, $\{\lambda, d, \delta, maxIter\}$
   Output: $\hat{X}^{(t)}$
1  $\theta' \leftarrow$ maxInit, $\theta \leftarrow \frac{\theta'}{2}$
2  Initialize $Z^{(t)}$ and $V^{(t)}$
3  $T^{(t-p)} \leftarrow I, \forall p = 1, \ldots, k-1$
4  Compute $V^{(t-p)}$ by applying NMF on $X^{(t-p)}$while
     $(|\theta' - \theta)| > \delta) \vee (iter < maxIter)$ do
5      Update $Z^{(t)}$ (Eq. (18))
6      Update $V^{(t)}$ (Eq. (19))
7      Update $T^{(t-p)}$ $\forall p = 1, \ldots, k-1$ (Eq. (23))
8      Compute $\mathcal{L}$ based on the updated $Z^{(t)}$, $V^{(t)}$, $T^{(t-p)}$ (Eq.
     (22))
9      $\theta' \leftarrow \theta$
10     $\theta \leftarrow \mathcal{L}$
11     $iter \leftarrow iter + 1$
12 end
13 return $\hat{X}^{(t)} = Z^{(t)}V^{(t)}$
```

An overview of our algorithm is presented in Algorithm 1. In our iterative optimization algorithm the factor matrices $Z^{(t)}$ and $V^{(t)}$ are initialized by random (sparse) matrices, with $Z^{(t)}$, $V^{(t)} \geq 0$, and the $k-1$ latent transition matrices are initialized by setting $T^{(t-p)} \leftarrow I$, with $p = 1, \ldots, k-1$. Our optimization algorithm first updates variables $Z^{(t)}$ and $V^{(t)}$ based on the updating rules in Eqs. (18) and (19), respectively. Then, we update the $k-1$ latent transition matrices $T^{(t-p)}$ based on Eq. (23). The order of the variables'/matrices' updating rules does not influence the solutions, as the matrices take similar values after a few iterations (Jialu et al., 2013; Lee & Seung, 1999). At the end of each iteration we calculate the joint objective function $\mathcal{L}$ in Eq. (22) based on the updated matrices. Notice that in each iteration, the optimization algorithm minimizes the joint objective function. As we considered the KKT conditions in Eqs. (11)–(13) to generate the updating rules, the proof that the optimization algorithm minimizes the joint objective function in each iteration, and thus converges, is similar to the study reported in (Lee & Seung, 1999). After the optimization algorithm converges, we compute the factorized matrix $\hat{X}^{(t)} = Z^{(t)}V^{(t)}$, which contains the final recommendations.

5. Experiments

5.1. Data

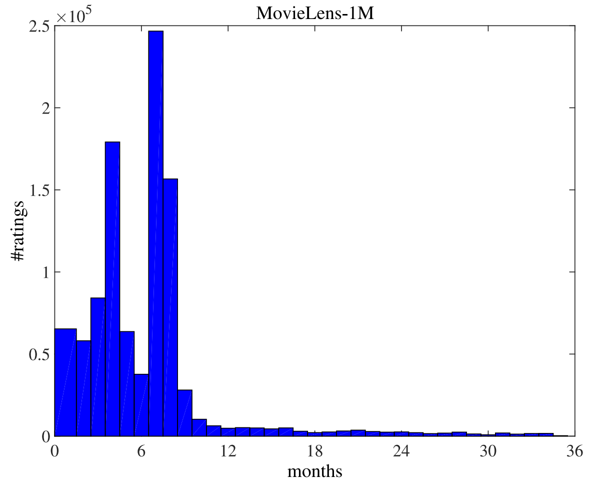
In this study we analyzed MovieLens-1M[4] and Last.fm-1K[5]. Also, we made the datasets' extended versions publicly available[6] with the crawled metadata of movies and artists.

MovieLens-1M. MovieLens-1M contains 1,000,209 anonymous ratings of $m = 3,952$ movies of $n = 6,040$ users who joined MovieLens in 2000. The dataset consists of tuples in the form {user, movie, rating, timestamp} over 36 months. User preferences are extremely sparse, as MovieLens-1M approximately includes 4.19% of all entries in the user-item matrix at the last month. Ratings are made on a 5-star scale. The distribution of users' monthly ratings throughout the 3-year time span are presented in Fig. 1(a). We split the dataset into $k = 6$ time periods corresponding to 6 semi-annuals. MovieLens-1M also contains 18 movie genres, such as "Action", "Adventure", "Animation", "Comedy", and so on. In MovieLens-1M a movie can belong to more than one genre. As genres are not accurate descriptions of the movies, we used the IMDB titles of the original version of the dataset to extend MovieLens-1M by crawling the plot, director, writer and the actors that played in each movie, with the IMDB API services[7]. Given a set of training months, we aim at generating top-N movie recommendations for a user at a test month. With a time window equal to semi-annual, we consider all the past months of the previous semi-annuals and the first five months of the current ongoing semi-annual as training set. Therefore, we have six different test sets of tuples at test months 6, 12, 18, 24, 30 and 36.

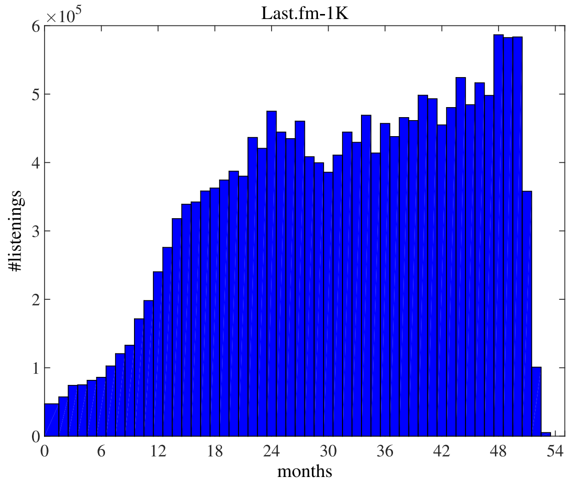
Last.fm-1K. Last.fm-1K includes the listening habits for $n = 992$ users. Last.fm consists of tuples in the form of {user, artist, track, timestamp} over 54 months (till May, 5th 2009). In total, there are $m = 176,948$ artists and 19,150,868 listening events, including 10.91% of all entries in the user-item matrix at the last month. The distribution of the monthly listening events throughout the 4.5 year time span are presented in Fig. 1(b). We split the data set into $k = 9$ time periods, corresponding to 9 semi-annuals. As there is no additional information of artists in the original version of the dataset, we fetched artists' metadata, by crawling the artists' most popular tags using the Last.fm API[8]. Given a set of training months we aim at generating top-N artist recommendations for a user at a test month. Accordingly, using a time window equal to semi-annual, we consider all the past months of the previous semi-annuals and the first five months of the current ongoing semi-annual as training set. In total, we have nine different test sets of tuples at test months 6, 12, 18, 24, 30, 36, 42, 48 and 54.

Preferences' Shift Rates. For presentation purposes, to compute the number of dynamic users, i.e., users that tend to shift their preferences at the test months of MovieLens-1M and Last.fm-1K, based on Eq. (2) we averaged the $w_j^{(t-p)}$ values over all users at the test months throughout the data time span. The average values are denoted by ϵ, based on which we generated the following groups of preferences' shift rates: $(\epsilon \leq 0.25)$, $(0.25 < \epsilon < 0.5)$, $(0.5 \leq \epsilon < 0.75)$ and $(\epsilon \geq 0.75)$. Users that belong to group $(\epsilon \leq 0.25)$ are the most dynamic users in the data, while users in group $(\epsilon \geq 0.75)$ have the most stable preferences. In Fig. 1(c), we plot the users' distribution based on the groups of preferences' shift rates. Clearly, in Last.fm-1K there are more dynamic users than

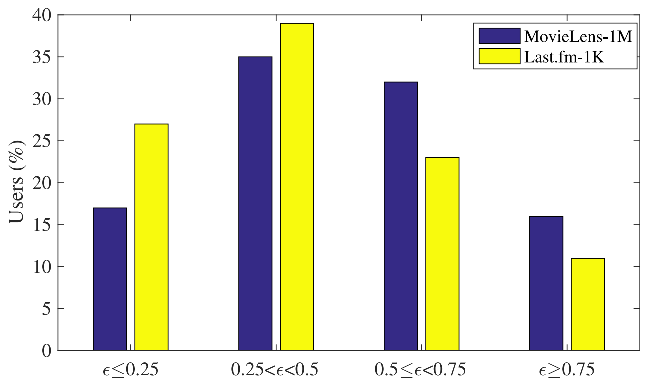
[4] https://grouplens.org/datasets/movielens/1m/.
[5] http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html.
[6] https://github.com/drafail/datasets_eswa.
[7] https://www.npmjs.com/package/imdb-api.
[8] http://www.last.fm/api.



(a)



(b)



(c)

Fig. 1. (a) Number of monthly ratings throughout the 3-year time span of MovieLens-1M. (b) Number of monthly listening events throughout the 4.5-year time span of Last.fm-1K. (c) User preferences' shift rates in both datasets.

MovieLens-1M, expressed by the high percentages of users that belong to groups ($\epsilon \leq 0.25$) and ($0.25 < \epsilon < 0.5$).

5.2. Evaluation metrics

In our experiments we use the ranking-based metrics Average Precision (AP@N) and Normalized Discounted Cumulative Gain (NDCG@N). We assume that the items are enumerated in decreasing order according to their scores. Let r be the index ranging over positions in the ranked list ($r = \{1, \dots, N\}$), and $rel(r)$ denote the relevance of the r-th item for the user. For the definition of AP@N, $rel(r) = 0$ if this is a non-relevant item for the user, $rel(r) = 1$ otherwise. For the computation of NDCG@N, $rel(r)$ is equal to the users' observed rating from the multi-level scale, that is 1–5 for MovieLens-1M. In Last.fm-1K we binarize the listening events per user, by thresholding them by user's average listening events in order to consider user's bias. AP@N is defined as the average of Precisions computed at each relevant position in the top-N recommended items of the user's ranked list, where Precision (P@N) equals the fraction of relevant items out of the top-N ranked items. AP@N is computed as follows (Weimer, Karatzoglou, Le, & Smola, 2007):

$$AP@N = \sum_{r=1}^{N} \frac{P@r \cdot rel(r)}{\min(\mathrm{N}, \#\text{of relevant items})} \tag{24}$$

DCG@N captures the importance of finding the correct ordering among higher ranked items compared to that among lower ranked items, by discounting the importance weight with the item's position in the ranked list. DCG@N is formally given by (Järvelin & Kekäläinen, 2002):

$$DCG@N = \sum_{r=1}^{N} \frac{2^{rel(r)} - 1}{l \log_2(r+1)} \tag{25}$$

NDCG@N is the ratio of DCG@N over the ideal iDCG@N per user, that is the DCG@N value given the user's recommendation list sorted according to the ground truth data. For each test month we averaged the results over all users and report the mean AP@N and NDCG@N values.

5.3. Compared methods

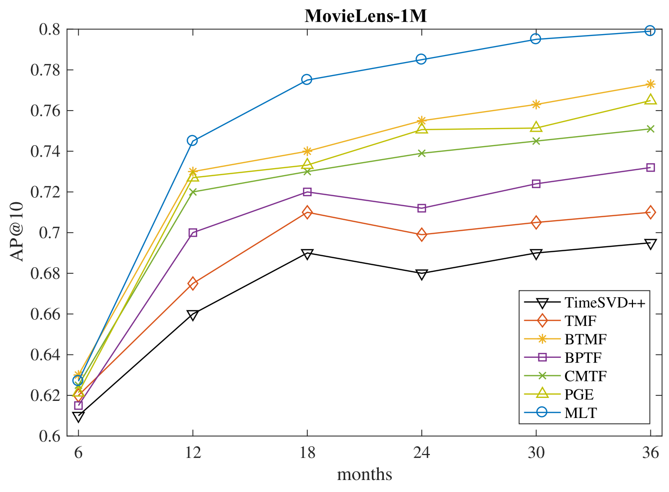
We compare the proposed Multi-Latent Transition *(MLT)* model with the following methods: *timeSVD++* (Koren, 2009): a baseline method which factorizes the item-user matrix, by taking into account users' preference dynamics. timeSVD++ extends the matrix factorization of SVD++(Koren, 2008) by introducing time dependent user/item biases and user latent factors to cope with the dynamics of user preferences. *TMF* (Zhang et al., 2014): a Temporal Matrix Factorization method that models users' preference dynamics by learning a transition matrix for each user latent vectors between two consecutive time periods. In TMF, the transition matrix considers the time-invariant pattern of the evolving preferences. *BTMF* (Zhang et al., 2014): a Bayesian Temporal Matrix Factorization technique that extends the TMF method by introducing priors for the hyperparameters which capture the conditional distributions of users, items and user's feedback on items. *BPTF* (Xiong et al., 2010): a Bayesian Probabilistic Tensor Factorization method that captures the three-way correlations among user-item-time. In BPTF, time periods (semi-annuals) are modeled as slices in the tensor. *CMTF* (Acar et al., 2011): a Coupled-Matrix Tensor Factorization strategy exploiting the items' metadata similarities in a matrix. The similarity matrix is coupled with the tensor at the items' dimension. As the BPTF method, CMTF also models time periods as slices in the tensor. *PGE* (Li, Chen, & Yan, 2017): a Product Graph Embedding model for time-aware recommendations. In PGE the item representation is learned based on a network embedding method and then, the representation of user dynamic preferences is computed by leveraging the items that the user interacted before in the form of a time decay function. In PGE, we tried different decay functions as in (Li et al., 2017), and we concluded in the triangle decay function, achieving the best performance in our experiments. The parameter tuning of the examined methods has been determined via 5-fold cross-validation and in our experiments we report the best results. The parameter analysis of MLT is studied in Section 5.5.

5.4. Performance evaluation

In Fig. 2 we show the effect on AP@10 for the test months of each evaluation dataset, throughout the data time span. Based on the results we observe the following:

Matrix Factorization strategies (TimeSVD++ / TMF / BTMF). Clearly, all the examined models have more unstable performance in Last.fm-1K than in MovieLens-1M. This occurs because there are more dynamic users in Last.fm-1K than in MovieLens-1M (Section 5.1), thus limiting the models' recommendation accuracy. Even though in more recent time periods the training sets are enlarged, the augmented training sets do not necessarily improve the AP@10 metric throughout the data time span. This happens because users shift their preferences in both datasets and consequently limit the models' performances even in the last test months. For instance, the baseline methods of TimeSVD++ and TMF have poor performance throughout the time span, by not capturing well users' preferences dynamics. Instead, BTMF achieves the best performance among the matrix factorization schemes, by introducing priors for the hyper parameters of TMF to capture the conditional distributions of users, items and user's feedback on items.

Tensor Factorization strategies (BPTF / CMTF). Both the tensor factorization strategies of BPTF and CMTF achieve higher values of AP@10 than TimeSVD++ and TMF by calculating the all-to-all pairwise correlations of preferences at different time periods. We also observe that CMTF improves the accuracy of BPTF, by exploiting the similarities of items' metadata in the coupled matrix. Compared to the second best method, that is the matrix factorization of BTMF, both BPTF and CMTF have lower performances. This happens as both BPTF and CMTF not only calculate the $k-1$ pairwise correlations between preferences at the ongoing time period of the test month and those of its past time periods, but also compute the $\frac{(k-1)!}{2!}$ all-to-all pairwise correlations of the preferences at the $k-1$ past periods over the tensor factorization. This negatively



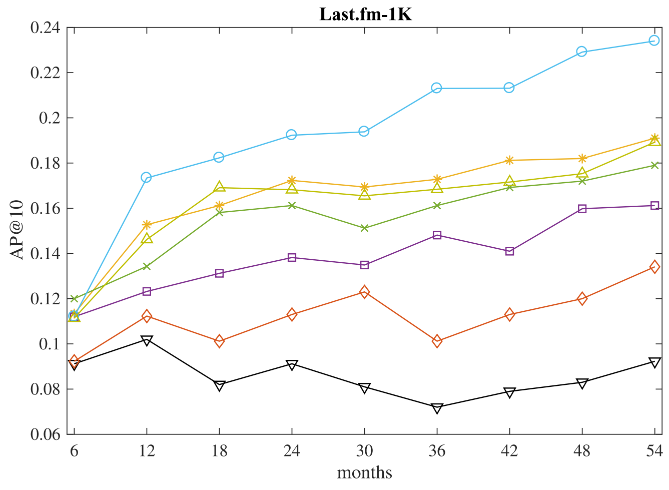


Fig. 2. Performance throughout the data time span.

affects the performance of each tensor factorization, by introducing noise when computing the pairwise correlations of the ongoing time period with the previous ones.

Graph-based strategy (PGE). The PGE model has comparable performance with BTMF, as PGE also considers the dynamics of user preferences. PGE captures the co-occurrence information of items to construct an item-item co-occurrence graph and then learns the low-dimensional representation of item vertices by the network embedding. With the item embeddings, a time decay function is used to capture the dynamics of user preferences, and the final recommendation are generated based on the embeddings of items and user preferences in the same latent space. However, PGE does not consider the items' metadata when computing the evolution of user preference in the time decay function, which explains the limited performance of PGE.

Proposed MLT model. The MLT model significantly outperforms all baselines in both datasets. Using the multi-latent transition analysis with the proposed joint objective function, MLT can accurately capture the multiple transitions at the users' latent space, and consequently captures well the preference dynamics. Compared to BTMF, the second best method, MLT can better compute the latent transitions between the ongoing time period of the test month and the past ones, as BTMF not only assumes that users' preferences evolve gradually at consecutive time periods but also BTMF does not utilize the auxiliary information of items' metadata. Also, evaluated against CMTF that also exploits items' metadata, MLT boosts the recommendation accuracy by incorporating

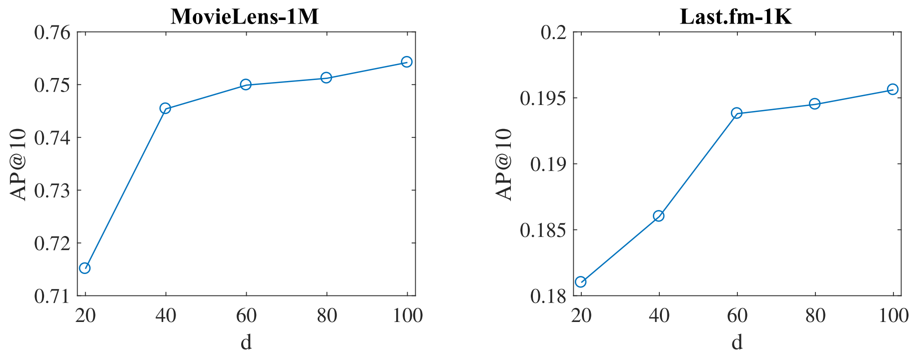


Fig. 3. Impact of the # of latent dimensions d.

Table 2
Performance evaluation in terms of AP@5, AP@10, NDCG@5 and NDCG@10.

MovieLens-1M	AP@5	AP@10	NDCG@5	NDCG@10
TimeSVD++	0.7057	0.6632	0.7012	0.6823
TMF	0.7301	0.6818	0.7096	0.6962
BTMF	0.7634	0.7236	0.7463	0.7278
BPTF	0.7488	0.6942	0.7231	0.7097
CMTF	0.7592	0.7118	0.7404	0.7195
PGE	0.7612	0.7202	0.7451	0.7209
MLT	**0.8127**	**0.7454**	**0.7719**	**0.7512**
Last.fm-1K	AP@5	AP@10	NDCG@5	NDCG@10
TimeSVD++	0.1012	0.0895	0.3572	0.3352
TMF	0.1139	0.1048	0.3692	0.3448
BTMF	0.1826	0.1664	0.4125	0.3912
BPTF	0.1683	0.1381	0.3914	0.3701
CMTF	0.1797	0.1463	0.4037	0.3875
PGE	0.1821	0.1601	0.4038	0.3895
MLT	**0.2312**	**0.1938**	**0.4412**	**0.4281**

Table 3
Effect on AP@10 based on preferences' shift rates.

MovieLens-1M	$\epsilon \leq 0.25$	$0.25 < \epsilon < 0.5$	$0.5 \leq \epsilon < 0.75$	$\epsilon \geq 0.75$
TimeSVD++	0.6124	0.6423	0.6812	0.6912
TMF	0.6723	0.6821	0.6912	0.6991
BTMF	0.7012	0.7123	0.7311	0.7423
BPTF	0.6829	0.6893	0.7012	0.7239
CMTF	0.6912	0.7012	0.7283	0.7382
PGE	0.7025	0.7104	0.7298	0.7401
MLT	**0.7199**	**0.7345**	**0 0.7554**	**0.7623**
Last.fm-1K	$\epsilon \leq 0.25$	$0.25 < \epsilon < 0.5$	$0.5 \leq \epsilon < 0.75$	$\epsilon \geq 0.75$
TimeSVD++	0.0639	0.0832	0.0971	0.1092
TMF	0.0812	0.1016	0.1182	0.1293
BTMF	0.1453	0.1625	0.1723	0.1793
BPTF	0.1123	0.1331	0.1489	0.1592
CMTF	0.1204	0.1398	0.1497	0.1769
PGE	0.1385	0.1601	0.1693	0.1782
MLT	**0.1634**	**0.1992**	**0.2013**	**0.2164**

the items' similarities when computing the multiple latent transitions of users' preferences between the ongoing and past periods.

In Table 2, we report average values of AP@5, AP@10, NDCG@5 and NDCG@10 over all the users at the 6 test months of MovieLens-1M and 9 test months of Last.fm-1K. The results show that MLT is significantly superior over all the competitive models in all evaluation metrics, due to our efficient multi-latent transitions analysis and exploitation of items' metadata when computing the correlations of users' preferences at different time periods.

5.5. Influence of Preferences' shift rate

In Table 3, we present the effect on AP@10, by evaluating the performance of the examined models on the four groups of preferences' shift rates (Section 5.1), where the group denoted by ($\epsilon \leq 0.25$) contains dynamic users that tend to shift their preferences very often, while the group denoted by ($\epsilon \geq 0.75$) consists of users with stable preferences throughout the data time span. Obviously, the performance of all models is negatively affected in the case of dynamic users. This occurs because the examined models try to capture the preferences' evolution while dynamic users radically change their tastes. On the other hand, all models perform well in the case of users with stable preferences. Due to our multi-latent analysis, we observe that MLT achieves the best performance in all cases, for users with stable preferences and for users that tend to shift their preferences often, which explains the superiority of MLT over the baselines throughout the data time span.

5.6. Parameter analysis

MLT has two important parameters, the number of latent dimensions d and the ℓ_1-norm regularization parameter λ in Eq. (22). In our experiments we observed that MLT converges in more iterations when a late test month is used, mainly because the model is trained on more months. For the largest training set, which are the cases of the 36th and 54th test months in MovieLens-1M and Last.fm-1K accordingly, the gradient-based alternating optimization algorithm converges in 54 and 86 iterations/epochs, respectively. In this set of experiments, we first evaluate the influence of the latent dimensions on the performance of the MLT model. While keeping λ fixed we vary the number of latent dimensions d from 20 to 100 by a step of 20. In Fig. 3 we present the effect on AP@10 with the changes of d. The results are reported by averaging AP@10 over users at all test months. In MovieLens-1M, we observe that after $d \geq 40$ there is a slight improvement of AP@10, thus we set $d = 40$, while in Last.fm-1K we fix $d = 60$.

In Fig. 4, we present the effect on AP@10 with the changes of the ℓ_1-norm regularization parameter λ. In this set of experiments we vary λ in {1e-5, 1e-4, 1e-3, 1e-2, 1e-1}, with high values of the λ parameter forcing the factor matrices and the $k-1$ latent transitions matrices to be sparse. In MovieLens-1M there is a significant drop after $\lambda \geq 1e-4$, thus setting $\lambda = 1e-4$. Instead, in Last.fm-1K we fix $\lambda = 1e-2$. An interesting observation is that in Last.fm-1K we fix a higher value of λ than in MovieLens-1M, meaning that we force the factor matrices and the $k-1$ latent transitions to be more sparse. This occurs because the 4.5-year time span of Last.fm-1K is longer than the 3-year time span of MovieLens-1M, thus the data in Last.fm-1K are distributed to more months, thus increasing the data sparsity at a limited time period of a semi-annual. As a consequence, in Last.fm-1K the factor matrices at each test month have to be more sparse, reflected by the larger value of λ, compared to the one in MovieLens-1M. Also, the high value of λ makes the latent transitions to be more sparse in Last.fm-1K, as in

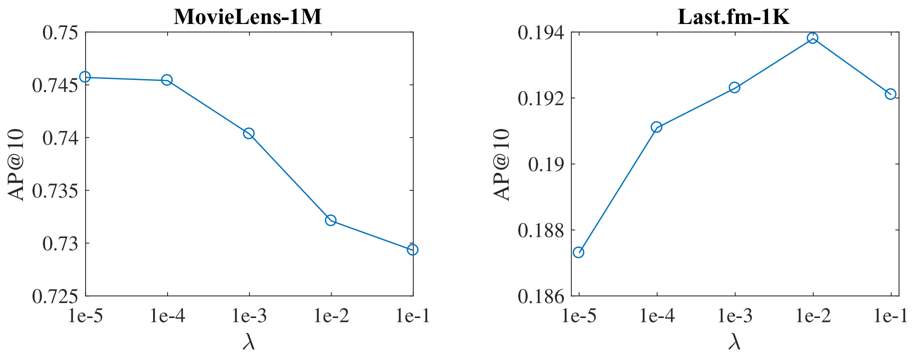


Fig. 4. Impact of the ℓ_1-norm regularization parameter λ.

Last.fm-1K there are more dynamic users than in the MovieLens-1M dataset.

6. Conclusions

In this paper we presented the MLT model, an efficient multi-latent transition model to capture users' evolving preferences in recommender systems. In our model, we formulate a joint objective function, to capture preferences' evolution by performing a multi-latent transition analysis, that is computing the pairwise correlations between the recent preferences at the ongoing period and the past preferences at the previous ones. Our joint problem is solved via an efficient gradient-based alternating optimization algorithm with convergence guarantees. Our experiments demonstrated the superiority of the proposed MLT model over several baselines throughout a 3-year and a 4.5-year data time span of two bechmark datasets. Summarizing our results, we showed that both the latent models of matrix and tensor factorization strategies, as well as a graph-based method have limited accuracy compared to our model. The latent models assume that user preferences evolve gradually over time, which is not always true in recommender systems, as users might shift their preferences radically between two consecutive periods. Also, the graph-based method uses a decay function to downweigh user past preferences, having limited accuracy as well. What really matters is to capture the correlations of user preferences between the ongoing and the past periods. We compute the dynamics of user preferences based on our multi-latent analysis. In addition, to better capture the preferences' evolution we exploit items' metadata, since users may have stable preferences over time as they may prefer certain attributes of items e.g., an actor or a movie director, or radically shift their preferences because they dislike them. Furthermore, we showed that MLT achieves high recommendation accuracy for users with stable preferences and for dynamic users that tend to shift their preferences often.

In our experiments we generate recommendations at a monthly basis - the last month of each semi-annual, by partitioning the dataset into semi-annuals. The choice of the dataset partition highly depends on the application, that is how often the recommendation are produced in practice, a challenging task which we plan to further examine in our future studies. Although MLT boosts the recommendation accuracy throughout the data time span, in real-life applications it is desirable to provide real-time recommendations. This necessitates the use of real-time streaming algorithms (Subbian, Aggarwal, & Hegde, 2016). As future work, we also plan to extend MLT to handle the problem of recommendation in a streaming setting, capturing the changes of users' preferences on existing items or when new users and items appear in real-time.

References

Acar, E., Kolda, T. G., & Dunlavy, D. M. (2011). All-at-once optimization for coupled matrix and tensor factorizations. CoRR, abs/1105.3422.

Cremonesi, P., Tripodi, A., & Turrin, R. (2011). Cross-domain recommender systems. In *Proceedings of the 11th international conference on data mining workshops (icdmw)* (pp. 496–503).

Dunlavy, D. M., Kolda, T. G., & Acar, E. (2011). Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data, 5*(2), 10:1–10:27.

Gao, H., Tang, J., Hu, X., & Liu, H. (2013). Exploring temporal effects for location recommendation on location-based social networks. In *Proceedings of the 7th acm conference on recommender systems (recsys)* (pp. 93–100).

Jamali, M., & Ester, M. (2010). A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the 4th acm conference on recommender systems (recsys)* (pp. 135–142).

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems, 20*(4), 422–446.

Jialu, L., Chi, W., Gao, J., & Jiawei, H. (2013). Multi-view clustering via joint nonnegative matrix factorization. In *Proceedings of the siam international conference on data mining (sdm)* (pp. 252–260).

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Review, 51*(3), 455–500.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th acm sigkdd international conference on knowledge discovery and data mining (kdd)* (pp. 426–434).

Koren, Y. (2009). Collaborative filtering with temporal dynamics. In *Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining (kdd)* (pp. 447–456).

Kuhn, H. W., & Tucker, A. W. (1951). Nonlinear programming. In *Berkeley symposium on mathematical statistics & probability* (pp. 481–492).

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature, 401*, 788–791.

Lee, D. D., & Seung, H. S. (2000). Algorithms for non-negative matrix factorization. In *Proceedings of the 13th international conference on neural information processing systems (nips)* (pp. 556–562).

Li, Y., Chen, W., & Yan, H. (2017). Learning graph-based embedding for time-aware product recommendation. In *Proceedings of the 2017 ACM on conference on information and knowledge management, (cikm)* (pp. 2163–2166).

Liu, N. N., He, L., & Zhao, M. (2013). Social temporal collaborative ranking for context aware movie recommendation. *ACM Transactions on Intelligent Systems and Technology, 4*(1), 15:1–15:26.

Liu, W., Chan, J., Bailey, J., Leckie, C., & Kotagiri, R. (2012). Utilizing common substructures to speedup tensor factorization for mining dynamic graphs. In *Proceedings of the 21st acm international conference on information and knowledge management (cikm)* (pp. 435–444).

Qiang, R., Liang, F., & Yang, J. (2013). Exploiting ranking factorization machines for microblog retrieval. In *Proceedings of the 22nd acm international on conference on information and knowledge management (cikm)* (pp. 1783–1788).

Rendle, S. (2010). Factorization machines. In *Proceedings of the 2010 ieee international conference on data mining (icdm)* (pp. 995–1000).

Rendle, S. (2012). Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology, 3*(3), 57:1–57:22.

Rendle, S., Gantner, Z., Freudenthaler, C., & Schmidt-Thieme, L. (2011). Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international acm sigir conference on research and development in information retrieval (sigir)* (pp. 635–644).

Salakhutdinov, R., & Mnih, A. (2007). Probabilistic matrix factorization. In *Proceedings of the 20th international conference on neural information processing systems (nips)* (pp. 1257–1264).

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web (www)* (pp. 285–295).

Saveski, M., & Mantrach, A. (2014). Item cold-start recommendations: Learning local collective embeddings. In *Proceedings of the 8th acm conference on recommender systems (recsys)* (pp. 89–96).

Schmidt, M. W., Fung, G., & Rosales, R. (2007). Fast optimization methods for L1 regularization: A comparative study and two new approaches. In *Proceedings of the 18th european conference on machine learning (ecml)* (pp. 286–297).

Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., & Hanjalic, A. (2014). Cars2: Learning context-aware representations for context-aware recommendations. In *Proceedings of the 23rd acm international conference on conference on information and knowledge management (cikm)* (pp. 291–300).

Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A., & Oliver, N. (2012). Tfmap: Optimizing map for top-n context-aware recommendation. In *Proceedings of the 35th international acm sigir conference on research and development in information retrieval (sigir)* (pp. 155–164).

Singh, A. P., & Gordon, G. J. (2008). Relational learning via collective matrix factorization. In *Proceedings of the 14th acm sigkdd international conference on knowledge discovery and data mining (kdd)* (pp. 650–658).

Subbian, K., Aggarwal, C. C., & Hegde, K. (2016). Recommendations for streaming data. In *Proceedings of the 25th acm international on conference on information and knowledge management (cikm)* (pp. 2185–2190).

Wang, K., Zhang, R., Liu, X., Guo, X., Sun, H., & Huai, J. (2013). Time-aware travel attraction recommendation. In *Proceedings of the 14th international conference on web information systems engineering (wise)* (pp. 175–188).

Weimer, M., Karatzoglou, A., Le, Q. V., & Smola, A. J. (2007). COFI RANK - maximum margin matrix factorization for collaborative ranking. In *Proceedings of the 20th international conference on neural information processing systems (nips)* (pp. 1593–1600).

Xiong, L., Chen, X., Huang, T.-K., Schneider, J., & Carbonell, J. G. (2010). Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the siam international conference on data mining (sdm)* (pp. 211–222).

Yin, H., Cui, B., Chen, L., Hu, Z., & Zhou, X. (2015). Dynamic user modeling in social media systems. *ACM Transactions on Information Systems, 33*(3), 10:1–10:44.

Yuan, F., Guo, G., Jose, J. M., Chen, L., Yu, H., & Zhang, W. (2016). Lambdafm: Learning optimal ranking with factorization machines using lambda surrogates. In *Proceedings of the 25th acm international on conference on information and knowledge management (cikm)* (pp. 227–236).

Zhang, C., Wang, K., Yu, H., Sun, J., & Lim, E. (2014). Latent factor transition for dynamic collaborative filtering. In *Proceedings of the siam international conference on data mining (sdm)* (pp. 452–460).