Information Sciences 409–410 (2017) 121–138

Contents lists available at ScienceDirect

Information Sciences

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

Learning to context-aware recommend with hierarchical factorization machines

Shaoqing Wang[a,b], Cuiping Li[a,*], Kankan Zhao[a], Hong Chen[a]

[a] Key Lab of Data Engineering and Knowledge Engineering of MOE Renmin University of China, Beijing, PR China
[b] Shandong University of Technology, Zibo, Shandong, PR China

ARTICLE INFO

Article history:
Received 20 September 2016
Revised 27 March 2017
Accepted 9 May 2017
Available online 11 May 2017

Keywords:
Collaborative filtering
Tree-structured Markov model
Context-aware recommendation
Hierarchical structure
Factorization machines
Kalman filtering

ABSTRACT

In recent years, a considerable number of context-aware recommendation methods have been proposed. One such technique, the Factorization Machines (FM) model, estimates interactions among features by working with any real valued feature vector. Although the FM model can efficiently handle arbitrary relationships (i.e., dyad, triad, etc.), it still has its limitations. In real world scenarios, contextual features can be considered as being organized in an understandable and intuitive hierarchy. However, existing the FM model performs poorly with regard to exploiting the hierarchical properties of contextual features during prediction. In this study, we consider the problem of exploiting hierarchical structures to improve recommendation quality and propose a novel two-stage recommendation model called Hierarchical Factorization Machines (HFM). In the first stage of HFM, the proposed model estimates the FM model parameters locally for each tree node and returns the initial predictions at all resolutions. Then, it finely tunes these predictions globally through a tree-structured Markov model. In the second stage, model fitting is achieved through an Expectation-Maximization (EM) algorithm, wherein the generalized Kalman filtering algorithm is used in the inner loop. Extensive experiments on real datasets verify that the proposed model is efficient and effective.

1. Introduction

In recent years, the rapid development in Internet technologies, has led to the phenomenon information explosion. Along these lines, there has emerged a need for two information processing approaches: information retrieval [30] and recommender systems [19]. It is appropriate to use information retrieval systems when users can express their requirements clearly. However, at times, users may be unable to elaborate their needs accurately. For example, users may be uncertain about their preferences for problems such as which movie to watch this weekend or which restaurant to go to for lunch? Under such circumstances, recommender systems serve a critical role in many practical applications in helping users deal with information overload by providing personalized recommendations. Recommendations can include items such as movies [17,19,24,39], music [8,20], news [9,22,25,51], books [27,35,53], spatial items [11,41,45–48], and others [5,21,31]. Collaborative filtering (CF) [3,4,7,37,42] is one of the most popular recommendation methods. CF identifies customers whose preferences

* Corresponding author.
E-mail addresses: wsq@ruc.edu.cn, wsq0533@163.com (S. Wang), licuiping@ruc.edu.cn (C. Li), zhaokankan@ruc.edu.cn (K. Zhao), chong@ruc.edu.cn (H. Chen).

http://dx.doi.org/10.1016/j.ins.2017.05.015
0020-0255/

are similar to those of the target customer and based on the choices of the identified customers recommends items to the target customer [52].

Current CF algorithms are primarily classified into two groups: context-unaware recommendation and context-aware recommendation. The former focuses on recommending the most relevant items to individual users; moreover, the users' preferences for items are quantified into a user-item rating matrix. The latter, context-aware recommendation, considers not only user and item, but also the situation in which the recommendation event happens. Such situational information usually includes the time, location, current social network of the user, etc., which can be collected easily through real world applications.

Studies on decision-making psychology have shown that environments tend to influence people's behaviors. Therefore, context-aware recommendation, which relies on contextual information, tends to be effective in recommender systems. A considerable number of context-aware recommendation methods [14,16,23,34,40,43,44] have been proposed in the literature. Among them, Factorization Machines (FM) is a popular method [34]. FM combines feature engineering with factorization models to estimate interactions between features and it can be applied successfully in many cases. Although the FM model can efficiently handle arbitrary relationships (i.e., dyad, triad, etc.), it still has its limitations. In real world scenarios, contextual features can be organized into a well-understood and intuitive hierarchy. However, the existing FM model tends to perform poorly with regard to exploiting the hierarchical properties of contextual features during prediction.

In this study, we focus on the problem of exploiting hierarchical information [1,2,39,49] to improve recommendation quality. Accordingly, we design a novel Hierarchical Factorization Machines (HFM) that can generate high-quality predictions in scenarios where each context can be organized in a natural hierarchy. Examples for hierarchical classfications in the online consumption domain and beyond include users organized in an occupation hierarchy, movies organized by genres, places organized by geographic regions, web pages organized by content, and so on.

HFM is a two-stage model. First, it estimates the FM model parameters locally for each tree node and returns the initial predictions, which can be considered as the fixed effect at all resolutions. Then, it finely tunes these predictions globally through a tree-structured Markov model which is used to ascertain the random effect. A generalized Kalman filtering algorithm [13] is used in the second stage, which consists of two steps. The first step, up-tree filtering, gathers information from the leaves to the root, and the second step, down-tree smoothing, disseminates information from the root to the leaves. Model fitting is achieved through an Expectation-Maximization (EM) algorithm [10], wherein the generalized Kalman filtering algorithm is used in the inner loop. The advantage of incorporating hierarchy into recommendations is that it provides a natural framework to generate recommendations at multiple resolutions, from coarser scales to finer scales. Such a multi-resolution model is extremely useful in handling data sparsity at finer resolutions and smoothing high variances at coarser resolutions. In particular, this work makes the following contributions.

- We propose HFM, a novel Hierarchical Factorization Machines, to incorporate hierarchy into context-aware recommendation. Hierarchy aids in realizing multi-resolution and high-quality recommendations.
- We present a two-stage modeling approach to determine the predictions. After the primary predictions are obtained in the first stage, a tree-structured Markov model is used in the second stage to perform global smoothing.
- We design a recommendation framework that provides multiple resolutions. In general, there is a relatively high degree of novelty at coarse-grained resolutions. While, fine-grained resolutions provide higher accuracy. In real scenarios, appropriate resolutions can be selected so as to make a trade-off between novelty and accuracy.
- We conduct extensive experiments on real datasets to verify the efficiency and effectiveness of the proposed model.

The remainder of this paper is organized as follows. In Section 2, we present preliminaries of context-unaware recommendation, context-aware recommendation and Factorization Machines. We then present the details of our Hierarchical Factorization Machines in Section 3. The model fitting algorithm is presented in Section 4. Detailed experimental results are demonstrated in Section 5. Section 6 discusses multiple resolutions of the proposed model. Further, appropriate resolutions are selected in order to evaluate a trade-off between novelty and accuracy. Section 7 introduces related work. Finally, Section 8 concludes this study and outlines future research directions.

2. Preliminary

In this section, we first introduce context-unaware recommendations and context-aware recommendations. Then, we present an overview of Factorization Machines, which is a popular model for context-aware recommendations.

2.1. Context-unaware recommendations

Given a set of users $U = \{u_1, u_2, \ldots, u_{|U|}\}$ and a set of items $I = \{i_1, i_2, \ldots, i_{|I|}\}$, the traditional recommendation techniques based on matrix factorization in which the ratings are provided by users on items can be represented in a sparse matrix $M \in R^{|U| \times |I|}$. Table 1 shows an example of a sparse matrix $M \in R^{6 \times 8}$. The recommendation problem then is translated into a Matrix Completion problem which infers missing entries from a partially specified matrix of observations by fitting a matrix factorization model to the data [18–20,37]. Such a system is called a traditional or context-unaware recommender since it only considers user and item dimensions in the recommendation process.

Table 1
Context-unaware recommendation.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1		1		2		1	3	
u_2	1		4		1		1	
u_3	5			1				4
u_4					2			3
u_5	4	2	5	3		2		
u_6		1	3					

Table 2
Context-aware recommendation.

	U			I				L				Ratings
x_1	1	0	0	1	0	0	0	1	0	0	0	4
x_2	1	0	0	0	1	0	0	0	1	0	0	3
x_3	0	1	0	0	0	1	0	0	1	0	0	3
x_4	0	1	0	0	0	0	1	0	0	1	0	5
x_5	0	0	1	1	0	0	0	0	0	1	0	2
x_6	0	0	1	0	1	0	0	0	0	0	1	3

2.2. Context-aware recommendation

Although the traditional 2-dimensional recommendation models can be used successfully in many cases, it is not uncommon to find a real setting in which additional contextual information (e.g., time, place, mood, the companion of other people) are involved. This information provides new dimensions for recommendations. We define contextual information with a set of contextual dimensions $\{C_3, C_4, \ldots, C_m\}$, each contextual dimension C_i being defined as a set of attributes capturing a particular type of context. In fact, users and items can be regarded as the first and second contextual feature, respectively, from a technical point of view. For simplicity, we denote the users set as C_1 and items set as C_2.

Table 2 shows an example of users U buying items I at locations L, in which $U = \{u_1, u_2, u_3\}$, $I = \{i_1, i_2, i_3, i_4\}$, and $L = \{l_1, l_2, l_3, l_4\}$. Then the first tuple x_1 means that user u_1 bought i_1 at l_1 and rated it as 4 stars. As can be observed, the categorical features are transformed into indicator variables. Let n_i denote the cardinality of contextual feature C_i, then the number of elements in feature vectors p is $n_1 + n_2 + \ldots + n_m$. Given a training set with N pair instances (x_i, y_i), where N represents the number of total instances, $x_i \in X^p$ represents the feature vector, and $y_i \in \mathbb{R}$ corresponds to the target rating for the ith instance. The goal of context-aware recommendation is to learn a function $\hat{y} : X^p \to \mathbb{R}$ that minimizes the loss function: $E = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$.

2.3. Factorization machines

Context-aware recommendation is a popular topic in the field of recommender systems, and a number of relevant pioneering works have been proposed in the literature [14,16,23,34,40]. Among them, Factorization Machines (FM) proposed by Rendle [32] is a popular model. The model equation for 2-order FM is defined as follows:

$$\hat{y}(\boldsymbol{x}) = \alpha_0 + \sum_{j=1}^{p} \alpha_j x_j + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} \theta_{j,j'} x_j x_{j'}, \tag{1}$$

in which α_0 is the global bias, α_j is the weight for feature x_j and $\theta_{j,j'}$ is the weight for the pairwise interaction of features x_j and $x_{j'}$. Pairwise interaction of features indicates that the instance has both feature value x_i and x_j. For example, the first instance of Table 2 has an interaction feature $u_1 i_1$ since it includes both u_1 and i_1.

The weight $\theta_{j,j'}$ for the interaction feature is defined as:

$$\theta_{j,j'} := \langle \boldsymbol{\theta}_j, \boldsymbol{\theta}_{j'} \rangle = \sum_{k=1}^{f} \theta_{j,k} \cdot \theta_{j',k}, \tag{2}$$

in which $\langle \cdot, \cdot \rangle$ is the dot product of two factors of size f.

The model parameters that need to be estimated are:

$$\alpha_0 \in \mathbb{R}, \quad \boldsymbol{\alpha} \in \mathbb{R}^p, \quad \boldsymbol{\theta} \in \mathbb{R}^{f \times p}. \tag{3}$$

In practice, it is not surprising that many contextual features such as users and items are often organized in a well-understood and intuitive hierarchy. On each hierarchical level, features can be classified into many groups. Note that FM converts each user and item into a set of features and learns a model to generate predictions without considering the between-level and between-group parameter heterogeneity. Since the number of features for each context dimension is

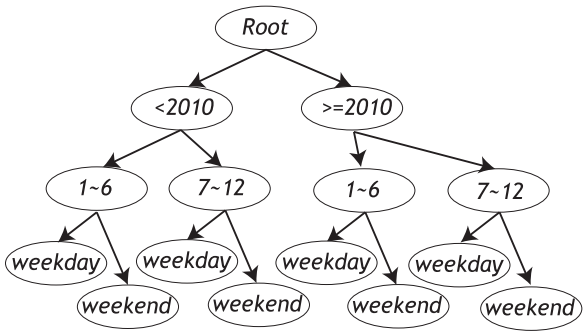

Fig. 1. An example of time hierarchy.

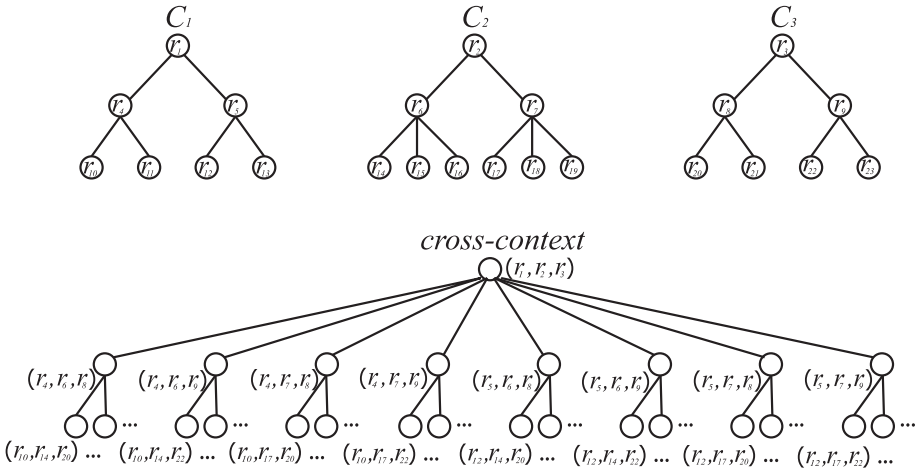

Fig. 2. An example of the construction of a cross-context hierarchy.

relatively large (for example, there are millions of users and items in many real world scenarios), aggregating results from such models to generate predictions at coarser resolutions generally involves huge biases. In this study, we propose our HFM, which incorporates hierarchy into context-aware recommendation and addresses the possibility of parameter variation across levels and regions. Our experiments on real datasets show that this parameter variation considerably improves the recommendation quality.

3. Hierarchical factorization machines

HFM predicts ratings using a two-stage model. First, it estimates the FM model parameters locally for each tree node and returns the initial predictions at all resolutions. Then, it finely tunes these predictions globally through a tree-structured Markov model. A generalized Kalman filtering algorithm is used in the second stage, which involves two steps: up-tree filtering and down-tree smoothing. The up-tree filtering step gathers information from the leaves to the root and the down-tree smoothing step disseminates information from the root to the leaves. Model fitting is achieved through an Expectation-Maximization (EM) algorithm, wherein the generalized Kalman filtering algorithm is used in the inner loop. In this section, we first discuss the construction of a cross-context hierarchy. Then we describe the first and second stage models in details.

3.1. Constructing cross-context hierarchy

Given a dataset with multiple contexts $\{C_1, C_2, \ldots, C_m\}$, we assume each context can be classified into a hierarchy where the nodes correspond to broad contextual themes (consider time as an example, Root→ Year→ Month→ day, as shown in Fig. 1). For ease of exposition, we further simplify our problem and assume that the number of levels of all hierarchies are the same. Assume that there are 3 three-level hierarchies, i.e., C_1, C_2, and C_3 hierarchy. In order to utilize the respective hierarchy, we integrate them into a new cross-context hierarchy. We denote nodes in the C_1, C_2, and C_3 hierarchy as r_i, r_j, and r_k, respectively. Let r represent an arbitrary node of a new hierarchy. We only consider that node r is constructed by r_i, r_j, and r_k where r_i, r_j, and r_k belong to the same level in the respective hierarchy. Fig. 2 shows a simple example of how to integrate 3 hierarchies into a new cross-context hierarchy, which is the foundation of our HFM model.

Table 3
The key notations used in this model.

Symbol	Definition and description
T_j	All regions at the jth level of the cross-context hierarchy T
$pa(r)$	The parent node of node r
$c_i(r)$	The ith child of node r
n_i	The number of tuples observed at node r
x_{ir}	$x_{ir} \in R^p$
y_{ir}	The ith rating at the node r
y_{ir}^{FM}	The ith rating which is estimated by FM model at node r
$\mathbf{y}_r$	$\{y_{ir}; i = 1, \ldots, n_r\}$
$\mathbf{X}_r$	$\{x_{ir}; i = 1, \ldots, n_r\}$
ϕ_r^j	The random effect of node r at level j
V_r	The variance of $y_{ir}\|\phi_r^j$
η_r	The linear transformation coefficient from the state of node $pa(r)$ to the state of node r
W_r	The variance of $\phi_r^j\|\phi_{pa(r)}^{j-1}$
Σ_d	The variance of latent state of leaf node at level d
$\hat{\phi}_{r\|r}^d$	The mean of $\phi_r^d\|\mathbf{y}_r$ at leaf node r
$\Gamma_{r\|r}^d$	The variance of $\phi_r^d\|\mathbf{y}_r$ at leaf node r
$\hat{\phi}_{r\|c_i(r)}^j$	The mean of $\phi_{r\|c_i(r)}^j\|\mathbf{y}_{c_i(r)}$ at non-leaf node r
$\Gamma_{r\|c_i(r)}^j$	The variance of $\phi_{r\|c_i(r)}^j\|\mathbf{y}_{c_i(r)}$ at non-leaf node r
$(\hat{\phi}_{r\|r}^j)^*$	The mean of $(\phi_{r\|r}^j)^*\|\mathbf{y}_r^*$ at non-leaf node r
$(\Gamma_{r\|r}^j)^*$	The variance of $(\phi_{r\|r}^j)^*\|\mathbf{y}_r^*$ at non-leaf node r
$\hat{\phi}_{r\|r}^j$	The final estimated mean of random effect at non-leaf node r at up-tree filtering step
$\Gamma_{r\|r}^j$	The final estimated variance of random effect at non-leaf node r at up-tree filtering step
$\hat{\phi}_r^j$	The final estimated mean of random effect at non-leaf node r at down-tree smoothing step
Γ_r^j	The final estimated variance of random effect at non-leaf node r at down-tree smoothing step
$\Gamma_{r,pa(r)}^{j,j-1}$	The covariance between node r and its parent node $pa(r)$

Table 4
Projected training data.

	u^{g1}		i^{g2}		l^{g1}		Ratings
x_3	0	1	1	0	0	1	3

Table 5
Aggregated training data.

	U		I		L		Ratings
$x_{1,2}$	1	0	1	0	1	0	3.5
x_3	1	0	0	1	1	0	3
x_4	1	0	0	1	0	1	5
$x_{5,6}$	0	1	1	0	0	1	2.5

Let T denote the cross-context hierarchy, which is a tree structure. We assume there are L+1 levels in T (root at level 0 and leaves at level L). We then use T_j to denote all regions at the jth level of T. Let $pa(r)$ and $c_i(r)$ represent the parent and ith child of node r, respectively.

The key notations used in this paper are summarized in Table 3.

3.2. First stage: Local estimation

In cross-context hierarchy the original data tuples with the finest resolution are partitioned into different nodes and aggregated on different levels. For example, assume that each context of Table 2 can be classified into two groups, which means that:

$U^g = \{u^{g1}, u^{g2}\}$, $u^{g1} = \{u_1, u_2\}$ and $u^{g2} = \{u_3\}$;

$I^g = \{i^{g1}, i^{g2}\}$, $i^{g1} = \{i_1, i_2\}$ and $i^{g2} = \{i_3, i_4\}$;

$L^g = \{l^{g1}, l^{g2}\}$, $l^{g1} = \{l_1, l_2\}$ and $l^{g2} = \{l_3, l_4\}$.

For the sake of simplicity, we introduced some new symbols, i.e., U^g, I^g, and L^g. It is obvious that $U^g = U$, $I^g = I$ and $L^g = L$. Correspondingly, the projected training data for region $\{u^{g1}, i^{g2}, l^{g1}\}$ is shown in Table 4. We can see that not only the number of tuples but also the number of features in the region decrease. In fact, the partition ends if there are few data in the projected training data for the region. The aggregated training data for the root is shown in Table 5. Please note that the ratings of tuples in the aggregated region are obtained by performing an average on all ratings of the same subregion.

For example, the rating 3.5 according to tuple $x_{1,2}$ in Table 2 is an average of the rating 4 and 3, which corresponds to tuple x_1 and x_2, respectively in Table 5.

With data assigned on each node of the cross-context hierarchy, the FM model parameters can be estimated locally and then primary rating predictions at multiple resolutions can be obtained. There are three methods to learn the parameters of the FM model [33]. (1) The Stochastic Gradient Descent (SGD) algorithm, which is very popular for optimizing factorization models and works well with different loss functions. (2) The Alternating Least Squares (ALS) algorithm, which iteratively solves a least-squares problem per model parameter and updates each model parameter with the optimal solution. (3) The Markov Chain Monte Carlo (MCMC) inference, which adopts Gibbs sampling in the FM model. The MCMC inference avoids a time-consuming search of the regularization parameters. We adopt the SGD algorithm to infer the parameters of FM model because of its linear computational and constant storage complexity. Conditional on these primary predictions, in the second stage, final predictions are globally adjusted through a tree-structured Markov model.

3.3. Second stage: global adjustment

We consider a Gaussian random process of the tree-structured Markov model, where each node of the tree represents random state variables and ratings denote the observations. Conditional on the states, the observed ratings are independent.

For a node $r \in T_j$, let $\mathbf{y}_r = \{y_{ir}; i = 1, \ldots, n_r\}$ denote the observed vector at node r and let n_r represent the number of tuples observed at node r.

$$y_{ir}|\phi_r^j \sim N(y_{ir}^{FM} + \phi_r^j, V_r); i = 1, \ldots, n_r; \tag{4}$$

in which:

- The level of node r is j and $j \in \{0, \ldots, L\}$.
- y_{ir}^{FM} is a fixed effect computed using the FM model in the first stage. The parameters, $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$, of the FM model can be learnt from the tuples at the node r.
- ϕ_r^j is a latent state and random effect. We consider that ϕ_r^j can smooth the effect of siblings and ancestors so that the model has a strong generalization ability.
- V_r is an unknown variance parameter. We assume that all nodes share a common variance parameter V and $V_r = V/n_r$.

To transfer knowledge across different nodes, we assume that each children state is drawn from a distribution centered around the state of their parent. A recursive one-step Markovian distribution is then defined in a joint distribution.

$$\phi_r^j|\phi_{pa(r)}^{j-1} \sim N(\eta_r \phi_{pa(r)}^{j-1}, W_r), \tag{5}$$

in which:

- Since the level of r is j, the level of its parent $pa(r)$ is $j-1$ and $\phi_{Root}^0 \sim N(0, W_0)$.
- η_r is a linear transformation coefficient from the state of node $pa(r)$ to the state of node r.
- W_r is an unknown variance parameter. For ease of exposition and to avoid over-fitting, we assume that all nodes at the jth level share a common variance parameter W_j and $j \in \{0, \ldots, L\}$.

Given Eq. (5), it is necessary to invert the state equation to represent a parent state as its children states:

$$\phi_{pa(r)}^{j-1} = E(\phi_{pa(r)}^{j-1}|\phi_r^j) + (\phi_{pa(r)}^{j-1} - E(\phi_{pa(r)}^{j-1}|\phi_r^j)) \tag{6}$$

$$= B_j \phi_r^j + \xi_r, \tag{7}$$

in which:

- $B_j = \Sigma_{pa(r)} \eta_r \Sigma_r^{-1}$, where $var(\phi_r^j) = \Sigma_j = \eta_r^2 W_{j-1} + W_j$.
- $\xi_r = \phi_{pa(r)}^{j-1} - E(\phi_{pa(r)}^{j-1}|\phi_r^j)$, where $E(\xi_r) = 0$ and $R_r = var(\xi_r) = \Sigma_{pa(r)} - \eta_r^2 \Sigma_{pa(r)} \Sigma_r^{-1} \Sigma_{pa(r)}$.

The details of the graph model are shown in Fig. 3.

4. Model fitting

Our goal is to obtain $E[\hat{\mathbf{Y}}|\mathbf{Y}]$ where $\hat{\mathbf{Y}}$ represents all final predictions and $\mathbf{Y} = \{\mathbf{y}_r | r \in T\}$. Recall that $\mathbf{y}_r = \{y_{ir}; i = 1, \ldots, n_r\}$; it denotes the observed vector at node r and n_r represents the number of tuples observed at node r. Let $\Theta = (\boldsymbol{\alpha}, \boldsymbol{\theta}, \eta, V, W)$ denote all the model parameters that can be learnt from the training data. In fact, Θ can be divided into two parts: $\Theta_1 = (\boldsymbol{\alpha}, \boldsymbol{\theta})$ and $\Theta_2 = (\eta, V, W)$. Θ_1 is used to produce fixed effect, i.e., y_{ir}^{FM}, which is learnt in the first stage of HFM using the FM model at each node. Random effect is then determined by Θ_2. We consider ϕ as the latent variable and $\mathbf{Y}$ as the observed variable. Thus, the joint distribution $p(\mathbf{Y}, \phi|\Theta_2)$ is governed by Θ_2, which will be learnt in the second stage of HFM.

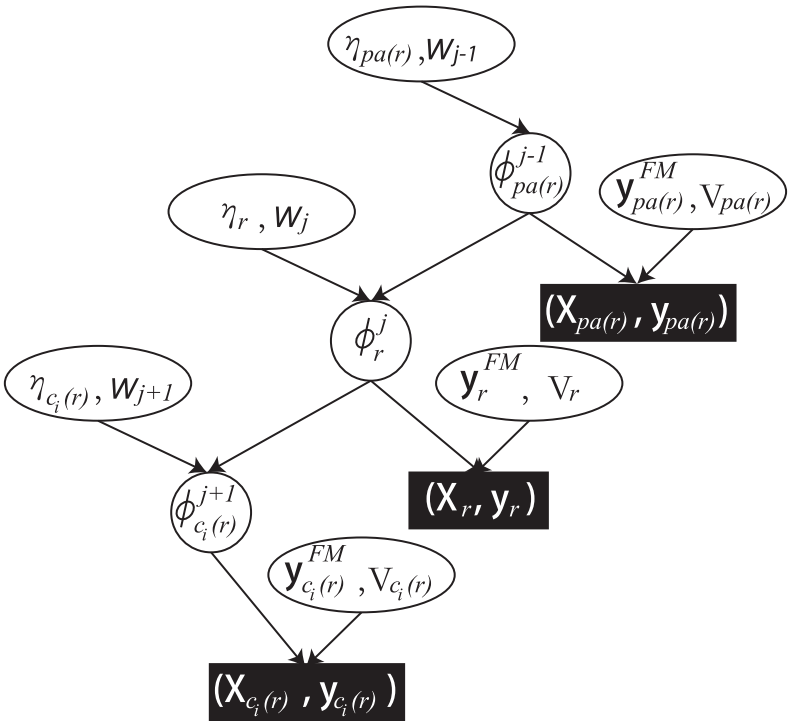

Fig. 3. An example of model details. $(\mathbf{X}_r, \mathbf{y}_r)$, $(\mathbf{X}_{pa(r)}, \mathbf{y}_{pa(r)})$ and $(\mathbf{X}_{c_i(r)}, \mathbf{y}_{c_i(r)})$ denote the observed context interaction data at node r, $pa(r)$ and $c_i(r)$ respectively. y^{FM}_r, $y^{FM}_{pa(r)}$ and $y^{FM}_{c_i(r)}$ are fixed effect computed through FM model at the first stage. The other variables are the parameters which need to be estimated at the second stage.

Θ_1 can be easily learnt by using the FM model fitting algorithm in [32]; we skip it for the want of space. Next, we focus only on how to obtain Θ_2. Since latent variables exist at this stage, the EM algorithm is a conventional method used to perform model fitting. The EM algorithm is a two-step iterative optimization technique for finding the maximum likelihood solutions [10]. In the E-step, we compute the posterior distribution of latent variable ϕ while holding Θ_2 as fixed (denoted by $\hat{\Theta}_2$). In the subsequent M-step, we update Θ_2 to obtain a new estimate.

4.1. E-step

In the E-step, the current estimates of Θ_2 can be denoted as $\hat{\Theta}_2 = (\hat{\eta}, \hat{V}, \hat{W})$. The generalized Kalman filtering algorithm [13] is adopted to compute the conditional posterior of ϕ, which is obtained by using all the observed data. That is, the conditional posterior of ϕ is determined based on three observations located at the current node, ancestors, and descendant, respectively.

The generalized Kalman filtering algorithm consists of two steps: *Up-tree filtering* and *Down-tree smoothing*.

4.1.1. Up-tree filtering

In the up-tree filtering step, the generalized Kalman filtering gathers information from level L to level 0, i.e., root node. The posterior distribution of each latent variable is obtained through its observations and descendants. That is, at each node r of level j, $j \in \{0, \ldots, L\}$, $\hat{\phi}^j_{r|r}$, and $\Gamma^j_{r|r}$ can be obtained recursively. Once the root node is reached, the conditional posterior of ϕ^0_{Root} based on all the data can be obtained because all the other nodes are descendants of the root node.

For a leaf node r at level d (Note that d is not always equal to L owing to data sparsity, which means that $d \in \{1, \ldots, L\}$). Let $e_{ir} = y_{ir} - y^{FM}_{ir}$, $E(\phi^d_r) = 0$ and $var(\phi^d_r) = \Sigma_d = \eta_r^2 W_{d-1} + W_d$. Assume that the posterior distribution of the latent variable is $\phi^d_r | \mathbf{y}_r \sim N(\hat{\phi}^d_{r|r}, \Gamma^d_{r|r})$, then

$$\hat{\phi}^d_{r|r} = \Sigma_d (n_r \Sigma_d + \hat{V})^{-1} \sum_{i=1}^{n_r} e_{ir}, \tag{8}$$

$$\Gamma^d_{r|r} = \Sigma_d \hat{V} (n_r \Sigma_d + \hat{V})^{-1}. \tag{9}$$

For a non-leaf node r, the conditional posterior distribution of the latent variable is obtained by gathering information from descendants: $\phi^j_{r|c_i(r)} | \mathbf{y}_{c_i(r)} \sim N(\hat{\phi}^j_{r|c_i(r)}, \Gamma^j_{r|c_i(r)})$. Suppose that both $\hat{\phi}^{j+1}_{c_i(r)|c_i(r)}$ and $\Gamma^{j+1}_{c_i(r)|c_i(r)}$ have been obtained, according

to Eq. (7), we can then obtain

$$\hat{\phi}^{j}_{r|c_i(r)} = B_{c_i(r)} \hat{\phi}^{j+1}_{c_i(r)|c_i(r)}, \tag{10}$$

$$\Gamma^{j}_{r|c_i(r)} = B_{c_i(r)} \Gamma^{j+1}_{c_i(r)|c_i(r)} B_{c_i(r)} + R_{c_i(r)}. \tag{11}$$

Thus, the conditional posterior distribution of the latent parent variable is obtained by combining the information of all its children: $(\phi^{j}_{r|r})^* | \mathbf{y}^*_r \sim N((\hat{\phi}^{j}_{r|r})^*, (\Gamma^{j}_{r|r})^*)$. Assume that the number of children of node r is k_r, then

$$(\hat{\phi}^{j}_{r|r})^* = (\Gamma^{j}_{r|r})^* \left\{ \sum_{i=1}^{k_r} (\Gamma^{j}_{r|c_i(r)})^{-1} \hat{\phi}^{j}_{r|c_i(r)} \right\}, \tag{12}$$

$$(\Gamma^{j}_{r|r})^* = \left\{ \Sigma_r^{-1} + \sum_{i=1}^{k_r} [(\Gamma^{j}_{r|c_i(r)})^{-1} - \Sigma_r^{-1}] \right\}^{-1}. \tag{13}$$

In addition, the observations at non-leaf node r (i.e., $\mathbf{y}_r = \{y_{ir}; i = 1, \ldots, n_r\}$) should be taken into account as well. Therefore, $\{e_{ir} = y_{ir} - y^{FM}_{ir}; i = 1, \ldots, n_r\}$ is further combined to obtain the final estimate of $\hat{\phi}^{j}_{r|r}$ and $\Gamma^{j}_{r|r}$.

$$\hat{\phi}^{j}_{r|r} = \Gamma^{j}_{r|r} \left\{ \frac{\sum_{i=1}^{n_r} e_{ir}}{\hat{V}} + \frac{(\hat{\phi}^{j}_{r|r})^*}{(\Gamma^{j}_{r|r})^*} \right\}, \tag{14}$$

$$\Gamma^{j}_{r|r} = (\Gamma^{j}_{r|r})^* - (\Gamma^{j}_{r|r})^* \left\{ (\Gamma^{j}_{r|r})^* + \frac{\hat{V}}{n_r} \right\}^{-1} (\Gamma^{j}_{r|r})^*. \tag{15}$$

As can be observed, the up-tree filtering step computes the posterior distribution of all latent variables from level L to level 0 and stops at level 0. That implies that, at node *Root*, we have

$$\hat{\phi}^{0}_{Root} = \hat{\phi}^{0}_{Root|Root},$$

$$\Gamma^{0}_{Root} = \Gamma^{0}_{Root|Root}.$$

4.1.2. Down-tree smoothing

After the up-tree filtering step, only the posterior distribution of latent variables in terms of the root node is computed by all observations. Next, we disseminate information from level 0 to level L so that all of the posterior distributions of latent variables are obtained by all observations. We obtain

$$\hat{\phi}^{j}_{r} = \hat{\phi}^{j}_{r|r} + \frac{\Gamma^{j}_{r|r} B_r (\hat{\phi}^{j-1}_{pa(r)} - \hat{\phi}^{j-1}_{pa(r)|r})}{\Gamma^{j-1}_{pa(r)|r}}, \tag{16}$$

$$\Gamma^{j}_{r} = \Gamma^{j}_{r|r} + \frac{(\Gamma^{j}_{r|r} B_r)^2 (\Gamma^{j}_{pa(r)} - \Gamma^{j-1}_{pa(r)|r})}{(\Gamma^{j-1}_{pa(r)|r})^2}, \tag{17}$$

$$\Gamma^{j,j-1}_{r,pa(r)} = \frac{\Gamma^{j}_{r|r} B_r \Gamma^{j-1}_{pa(r)}}{\Gamma^{j-1}_{pa(r)|r}}, \tag{18}$$

where $\Gamma^{j,j-1}_{r,pa(r)}$ is the covariance between node r and its parent $pa(r)$, which will be used in M-step. Please note that both the computational complexity and memory requirements of the generalized Kalman filtering algorithm are linear with the number of latent variables in the hierarchy.

4.2. M-step

The main task of M-step is to update $\Theta_2 = (\eta, V, W)$ by maximizing the expected log likelihood (shown in Table 6). Therefore, η, V and W can be updated as follows:

$$\hat{\eta}_r = \frac{\hat{\phi}^{j}_{r} \hat{\phi}^{j-1}_{pa(r)} + \Gamma^{j,j-1}_{r,pa(r)}}{\hat{\phi}^{j-1}_{pa(r)} \hat{\phi}^{j-1}_{pa(r)} + \Gamma^{j-1}_{pa(r)}}, \tag{19}$$

Table 6
Log likelihood formulas.

Complete data log-likelihood: $L(\Theta_2, \phi; Y)$
$\propto -\frac{1}{2}\sum_{j=0}^{L}\sum_{r\in T_j}\sum_{i=1}^{n_r}(\log V + \frac{n_r(e_{ir}-\phi_r^j)^2}{V}) - \frac{1}{2}\sum_{j=1}^{L}\sum_{r\in T_j}(\log W_j + \frac{(\phi_r^j-\eta_r\phi_{pa(r)}^{j-1})^2}{W_j})$,
Expected log-likelihood: $E_{(\phi\|Y,\hat{\Theta}_2)}[L(\Theta_2,\phi;Y)]$
$\propto -\frac{1}{2}\sum_{j=0}^{L}\sum_{r\in T_j}\sum_{i=1}^{n_r}(\log V + \frac{n_r[(e_{ir}-\hat{\phi}_r^j)^2+\Gamma_r^j]}{V})$
$-\frac{1}{2}\sum_{j=1}^{L}\sum_{r\in T_j}(\log W_j + \frac{(\hat{\phi}_r^j-\eta_r\hat{\phi}_{pa(r)}^{j-1})^2+\Gamma_r^j+\eta_r^2\Gamma_{pa(r)}^{j-1}-2\eta_r\Gamma_{r,pa(r)}^{j,j-1}}{W_j})$.

$$\hat{V} = \frac{\sum_{j=0}^{L}\sum_{r\in T_j}\sum_{i=1}^{n_r}[(e_{ir}-\hat{\phi}_r^j)^2+\Gamma_r^j]}{\sum_{j=0}^{L}|T_j|}, \tag{20}$$

$$\hat{W}_j = \frac{\sum_{r\in T_j}[(\hat{\phi}_r^j-\eta_r\hat{\phi}_{pa(r)}^{j-1})^2+\Gamma_r^j+\eta_r^2\Gamma_{pa(r)}^{j-1}-2\eta_r\Gamma_{r,pa(r)}^{j,j-1}]}{|T_j|}. \tag{21}$$

in which $j = 1, \ldots, L$ for Eqs. (19) and (21).

5. Experiments

In this section, we first introduce three datasets and the method to construct their hierarchical structures. Then, the metrics and experimental method are presented. Next, the other models that are used for comparison with our proposed model are presented. Finally, detailed experimental results are demonstrated.

5.1. Datasets and hierarchical structure

The Food dataset [29] contains 6360 ratings (1 to 5 stars) of 20 menus by 212 users under different kinds of contexts. Because the dataset is small, it is reasonable to construct a two-level hierarchical structure. We only select gender and genre as the attribute of users and menus, respectively. There are 3 genres: Japanese, Western and Chinese. We select 2 contextual variables: situation and virtual-flag. Situation captures how hungry the user is: normal, hungry, and full. Virtual-flag describes if the situation in which the user ratings are virtual or real. Hence, we obtain 4 two-level hierarchies which are "Root → Gender", "Root → Genre", "Root → How hungry" and "Root → Flag" for users, menus, situation and virtual-flag respectively. Finally, a cross-context hierarchy is constructed.

The MovieLens-1M dataset[1] contains 1,000,209 anonymous ratings (1 to 5 stars) of approximately 3,900 movies made by 6040 MovieLens users who joined MovieLens[2] in 2000. We consider timestamp as context and construct a three-level hierarchical structure. We select gender and age as the attributes of users. The ages can be divided into: < 25, $25-34$ and > 34. Both release-year and genre are selected as the attributes of movies. Release-years include < 1995 and ≥ 1995. There are 18 genres and most movies have multiple genres. Therefore, a tuple about movies with multiple genres may be assigned to several nodes in a cross-context hierarchical structure. We extract the day of the week (weekday and weekend) and year (2000, 2001, 2002 and 2003) as the attributes of the timestamp. Then, we obtain 3 three-level hierarchies, which are "Root → Gender → Age", "Root → Release-year → Genre" and "Root → Day-of-week → Year" for users, items, and time, respectively. Note that, the principle of hierarchy construction is discussed in Section 5.4.3. Finally, we construct a cross-context hierarchy.

The MovieLens-10M dataset[3] contains 10,000,054 anonymous ratings (1 to 5 stars) that were made by 71,567 users for 10,681 movies. Unlike the MovieLens-1M dataset, each user is represented by an id and no other information is provided. Therefore, a hierarchy of users cannot be created. We only create 2 three-level hierarchies, which are "Root → Release-year → Genre" and "Root → Day-of-week → Year" for movies and time respectively. Finally, we construct a cross-context hierarchy for the MovieLens-10M dataset.

5.2. Metrics

We assess the performance of the models using the most popular metrics: the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) defined as follows:

$$\text{MAE} = \frac{\sum_{(x_i,y_i)\in\Omega_{test}}|y_i-\hat{y}(x_i)|}{|\Omega_{test}|} \tag{22}$$

[1] http://grouplens.org/datasets/.
[2] http://movielens.org/.
[3] http://grouplens.org/datasets/.

Table 7
Performance comparison over the Food dataset.

Method	All Users		Cold Start	
	RMSE	MAE	RMSE	MAE
SVD++	1.151	0.902	1.237	1.014
FM	1.064	0.860	1.128	0.886
COT	1.031	0.829	1.065	0.841
HFM-NKF	1.057	0.847	1.083	0.869
HFM	**1.028**	**0.824**	**1.049**	**0.835**

The bold values are the best performance.

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{(x_i,y_i)\in\Omega_{test}} (y_i - \hat{y}(x_i))^2}{|\Omega_{test}|}} \tag{23}$$

where Ω_{test} denotes the test set, and $|\Omega_{test}|$ denotes the number of tuples in the test set. The smaller the value of MAE or RMSE the better the performance.

5.3. Experimental method

We adopt two different ways to split the datasets so that the performance can be examined on all users and cold start users. We repeatedly split datasets 10 times and report the average result.

- **All Users**: The original dataset is randomly partitioned into two independent sets: a training set and a test set. 20% of the tuples are randomly sampled to create the test set, and the remaining 80% tuples are treated as the training set.
- **Cold Start**: According to the above splitting result, we randomly sample about 10% of users from the training set. Then we hold less than three of their tuples and exchange the remaining tuples with randomly selected tuples in test set. Lastly, we obtain another training set and another test set.

5.4. Performance comparison

In our experiments, we compare our model with the following models.

- **SVD++** [18] is a classical context-unaware model, where only the user-item matrix is used to train the parameters so that the prediction for the missing values are generated.
- **FM** [34] is easily applicable to a wide variety of contexts by specifying only the input data and achieves fast runtime both in training and prediction.
- **COT** [23] represents the common semantic effects of contexts as a contextual operating tensor and represents a context as a latent vector. Contextual operating matrix from contextual operating tensor and latent vectors of contexts were generated to model the semantic operation of a context combination. Thus latent vectors of users and items can be operated by contextual operating matrices. COT is a state-of-the-art context-aware model and has been shown to outperform other context-aware recommendation models in terms of accuracy on datasets where the Food dataset and MovieLens-1M dataset are included.
- **HFM** is our newly proposed model that is described in Section 3.
- **HFM-NKF** is a variant of our HFM model and can be considered as a special case of our model. The difference in the HFM=NKF model is that the model only considers the fixed effect and ignores the random effect. That is, there is no Kalman filtering process in the HFM-NKF model. However, the HFM-NKF model makes use of cross-context hierarchy which is similar to [38].

5.4.1. Experimental result

For HFM, the latent dimension f is set to 8, 16 and 32 for the Food , MovieLens-1M and MovieLens-10M datasets, respectively. We select 100, 1, and 1 as the initial values of V, W_j, and η_r respectively. The detailed comparison results are shown in Tables 7–9.

From these tables, we can observe that:

- The performance of context-aware models is better than that of context-unaware one, i.e., SVD++. This shows that the performance can be improved by utilizing contextual information.
- The performance of the HFM-NKF model is better than that of the FM model. This demonstrates the importance of utilizing hierarchical information on context-aware recommendation. Higher accuracy is obtained on the MovieLens-1M and MovieLens-10M datasets owing to deeper hierarchies.
- Our proposed HFM model outperforms the HFM-NKF model. This validates that the Kalman filtering process is useful in improving the recommendation quality. This indicates that not only the first stage but also the second stage of HFM are critical to the prediction accuracy.

Table 8
Performance comparison over the MovieLens-1M dataset.

Method	All Users		Cold Start	
	RMSE	MAE	RMSE	MAE
SVD++	1.006	0.837	1.142	1.003
FM	0.896	0.713	1.027	0.825
COT	0.852	0.656	1.004	0.793
HFM-NKF	0.871	0.682	1.020	0.809
HFM	**0.842**	**0.649**	**0.992**	**0.780**

The bold values are the best performance.

Table 9
Performance comparison over the MovieLens-10M dataset.

Method	All Users		Cold Start	
	RMSE	MAE	RMSE	MAE
SVD++	0.952	0.742	1.064	0.849
FM	0.854	0.661	0.946	0.724
COT	0.817	0.618	0.903	0.705
HFM-NKF	0.843	0.636	0.921	0.711
HFM	**0.810**	**0.603**	**0.894**	**0.693**

The bold values are the best performance.

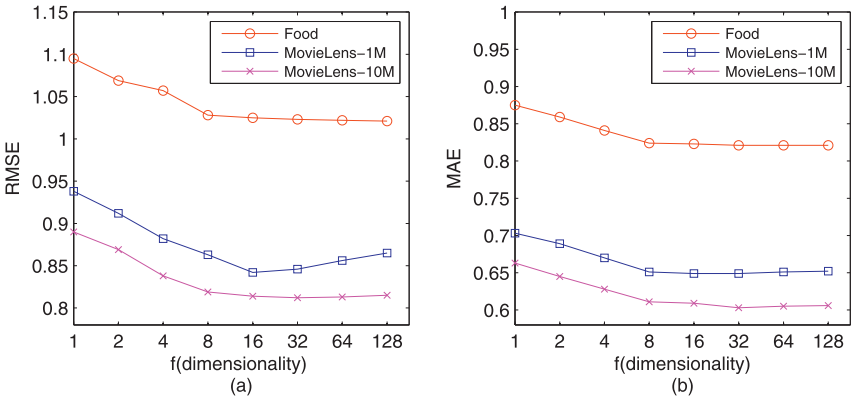

Fig. 4. The performance of HFM for an increasing number of latent factors on *All Users* data splitting in terms of RMSE and MAE.

- Using *t-test*, the performance of the HFM model is significantly higher than both that of the FM model and that of the HFM-NKF model at the confidence level of 95%.
- Our proposed HFM achieves best performance on both, all three datasets and two kinds of splitting. On *All Users* data splitting, COT obtains 0.033, 0.044, and 0.037 reduction over FM in terms of RMSE respectively. HFM further obtains 0.003, 0.01, and 0.007 reduction over COT. On *Cold Start* data splitting, the reduction becomes 0.016, 0.012, and 0.009 from COT to HFM.

5.4.2. The fixed effect and random effect

The fixed effect is generated by the first stage of HFM and affected by the number of dimensions, i.e., *f*. For simplicity, we select the same *f* for each node of the cross-context hierarchy. However, it is obvious that the convergence curves of all FM models at each node are different. Therefore, we analyze the performance of HFM on *All Users* data splitting by studying the effectiveness of the number of dimensionality. The experimental results are shown in Fig. 4(a) and (b). As can be observed, the performances for the three datasets are different. For the Food dataset, the performance improves as the dimensionality of latent factors increases. However, when the dimensionality of latent factors is larger than 8, the relative performance gain becomes small. The performance on the MovieLens-1M dataset initially improves then declines as the number of dimensions increase. This is maybe because the Food dataset has rich contextual information. There are similar performance gains between the Food and MovieLens-10M datasets.

The random effect is determined by the second stage of HFM and we verify the convergence of HFM at this stage on *All Users* data splitting. The convergence curves illustrated in Fig. 5(a) and (b) show that the RMSE/MAE of HFM becomes stable

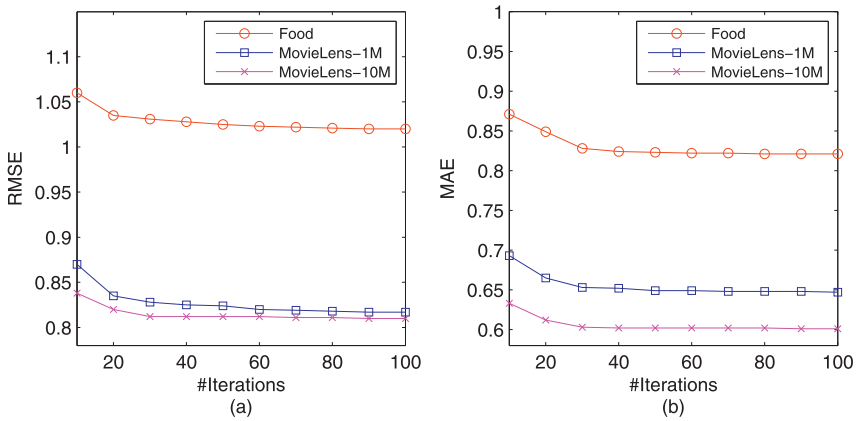

Fig. 5. The convergence of HFM for an increasing number of iterations on *All Users* data splitting in terms of RMSE and MAE.

Table 10
Four-level cross-context hierarchy over the MovieLens-1M dataset.

Attribute	The third level	The fourth level
Ages of users	< 25	< 18
		18–24
	$25 \sim 34$	25 –29
		30–34
	> 34	35–45
		> 45
Release-years of items	< 1995	< 1990
		1990 –1994
	≥ 1995	1995 –1997
		≥ 1998
Day-of-weeks of time	Weekday	Mon–Wed
		Thur–Fri
	Weekend	Sat
		Sun

after about 30 iterations. This suggests that the proposed model can fit the data well and can be trained rapidly in practical applications.

5.4.3. Constructing different hierarchies

First, the effect of the number of levels in the cross-context hierarchy is demonstrated. Then, we discuss the principle of hierarchy construction in case the number of levels of the hierarchy are fixed. We conduct experiments over the MovieLens-1M dataset.

The effect of the number of levels. We denote the project of constructing hierarchy in Section 5.1 as *Three-level*. In order to evaluate the effect of the number of levels, *Two-level* and *Four-level* cross-context hierarchies are constructed over the MovieLens-1M and MovieLens-10M datasets. Because there is a large number of ratings in the MovieLens-10M dataset, we continue to construct a *Five-level* cross-context hierarchy.

For the MovieLens-1M dataset, first, 3 two-level hierarchies are constructed. That is, "Root → Gender", "Root → Release-year" and "Root → Day-of-week" for users, items, and time, respectively. Finally, we construct a cross-context hierarchy which is denoted as *Two-level*. Next, we continue to construct a *Four-level* cross-context hierarchy based on the project of *Three-level*. It is necessary to extend the number of levels of hierarchies to four for each context. We add a level by further partitioning the data at ages, release-years, and day-of-weeks for users, items, and time respectively. For example, the node of "age < 25" of user hierarchy would have 2 children: "< 18" and "$18 \sim 25$". More details are shown in Table 10.

For the MovieLens-10M dataset, we can construct *Two-level, Four-level*, and *Five-level* cross-context hierarchies, which are similar to the MovieLens-1M dataset. Table 11 presents details.

The comparison results are shown in Figs. 6 and 7. We conclude that the performance improves with the an increase in the number of levels. That is, a higher accuracy is obtained owing to deeper hierarchies. This shows the importance of utilizing hierarchical information for context-aware recommendations. However, there is one slight difference between Figs. 6 and 7. To be specific, when the number of levels of cross-context hierarchy increase to four, the performance tends to be stable in Fig. 6. This is because the original data tuples are not partitioned if there are few data at nodes for the MovieLens-1M dataset. However, the performance notably improves with an increase in the number of levels over the MovieLens-10M

Table 11
Four-level and *Five-level* cross-context hierarchy over the MovieLens-10M dataset.

Attribute	The third level	The fourth level	The fifth level
Release-years of items	$<$ 1990	$<$ 1970	$<$ 1956
			1957–1969
		1970 –1989	1970–1978
			1979–1989
	$\geq$ 1990	1990 –1998	1990–1993
			1994–1998
		$\geq$ 1999	1999–2003
			$\geq$ 2004
Year of time	$<$ 2003	$<$ 1999	$<$ 1997
			1997–1998
		1999–2002	1999–2000
			2001–2002
	$\geq$ 2003	2003 –2006	2003–2004
			2005–2006
		$\geq$ 2007	2007$\sim$ 2008
			$\geq$ 2009

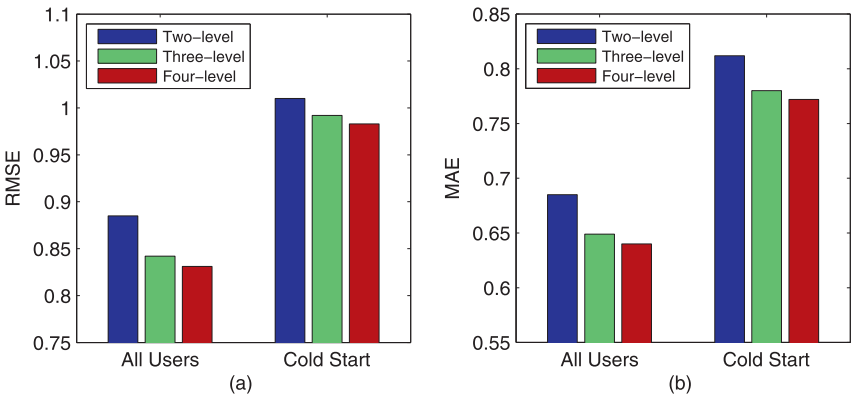

Fig. 6. The effect of the number of levels in terms of RMSE and MAE over the MovieLens-1M dataset.

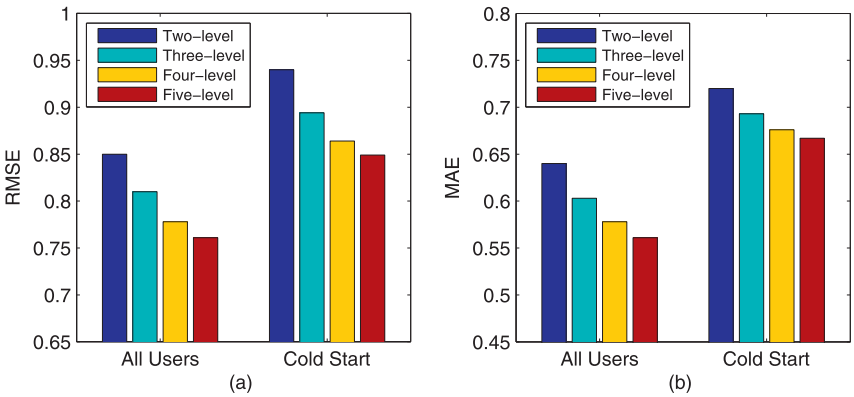

Fig. 7. The effect of the number of levels in terms of RMSE and MAE over the MovieLens-10M dataset.

dataset because the number of ratings increase. To this end, the performance of HFM is significantly better than COT for a *Five-level* cross-context hierarchy over the MovieLens-10M dataset.

The principle of hierarchy construction. There are many methods to construct a hierarchy for each context. Once the hierarchies of each context are built, the structure of cross-context hierarchy is fixed. We define a project as the method of constructing a structure. As is shown in Table 12, there are 8 projects over the MovieLens-1M dataset, since there are 3 contexts and hierarchy of each context has 3 levels.

The performance comparison of different projects for *All Users* data splitting over the MovieLens-1M dataset is shown in Fig. 8. From Fig. 8, we can see that the different structure of cross-context hierarchy leads to different performance. To be more specific, Project A achieves the best performance, and Project H performs the worst. This is because that there are

Table 12
Projects of constructing different hierarchy about *All Users* data splitting over the MovieLens-1M dataset.

Project	The hierarchy of each context
Project A	Users: Root → Gender → Age
	Items: Root → Release-year → Genre
	Time: Root → Day-of-week → Year
Project B	Users: Root → Age → Gender
	Items: Root → Release-year → Genre
	Time: Root → Day-of-week → Year
Project C	Users: Root → Gender → Age
	Items: Root → Genre → Release-year
	Time: Root → Day-of-week → Year
Project D	Users: Root → Age → Gender
	Items: Root → Genre → Release-year
	Time: Root → Day-of-week → Year
Project E	Users: Root → Gender → Age
	Items: Root → Release-year → Genre
	Time: Root → Year → Day-of-week
Project F	Users: Root → Age → Gender
	Items: Root → Release-year → Genre
	Time: Root → Year → Day-of-week
Project G	Users: Root → Gender → Age
	Items: Root → Genre → Release-year
	Time: Root → Year → Day-of-week
Project H	Users: Root → Age → Gender
	Items: Root → Genre → Release-year
	Time: Root → Year → Day-of-week

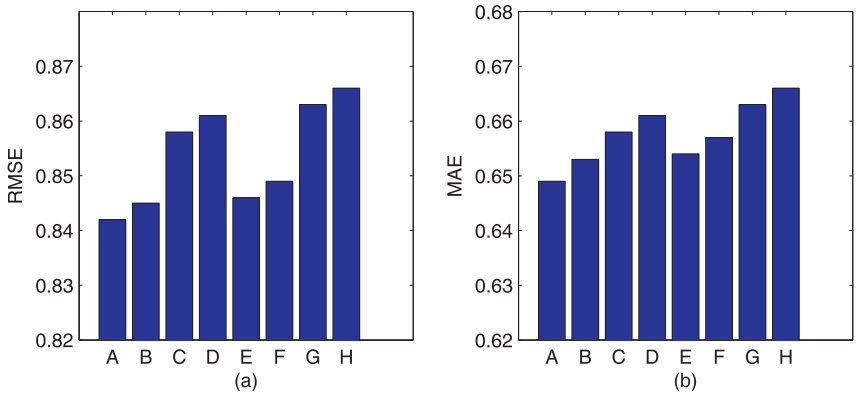

Fig. 8. Performance comparison of different projects about *All Users* data splitting over the MovieLens-1M dataset.

too many branches at the second level of cross-context hierarchy in Project H. Even though the cross-context hierarchy has 3 levels, the partition of tuples at many nodes ends at the second level owing to lack of sufficient data. In the end, the principle of hierarchy construction is suggested in that the deeper the hierarchy of the context the more the branches.

5.4.4. Computational complexity

The update to the parameters of each node in the first stage simply depends on their own data. In the second stage, the corresponding updates rely on the Markov blanket (its parent and children). As a consequence, the update to the parameters of each node is independent of all other nodes. Hence, we can easily parallelize the posterior inference computations. We implement HFM in C++ and use openMP for parallelization. Empirically, we obtain a near linear speedup in the number of threads.

The computational complexity of FM is linear in both the number of factors and the number of contexts [34]. Furthermore, the computational complexity of the generalized Kalman filtering algorithm is linear to the number of nodes in the hierarchy. Thus, the computational complexity of our proposed HFM is approximately linear with respect to the number of factors, contexts and nodes in the hierarchy. We compare the runtime of HFM with that of COT over the MovieLens-1M and MovieLens-10M datasets since the COT is a state-of-the-art context-aware model. Since HFM is a two-stage model, we record the runtime of one iteration in single-threaded scenarios of the two stages, respectively, and add them up so that the runtime of one full iteration over all the training data is learnt. To show the polynomial growth of the runtime with an increasing number of dimensions of factor, we run both models with $f \in \{1, 2, 4, 8, 16, 32, 64, 128\}$. Further, for simplicity,

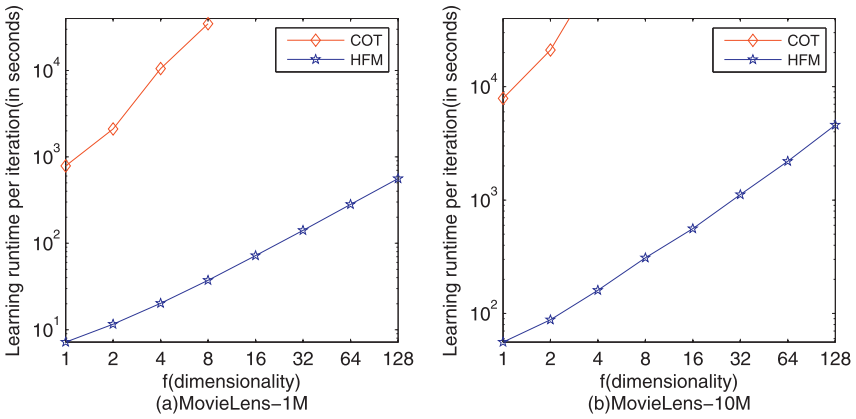

Fig. 9. Learning runtime in seconds for one iteration in single-threaded scenarios over the whole training dataset.

we simply take into account the same dimensionality of latent factor vectors of various contexts for COT and the maximal dimensionality we use is 8. The experimental results are shown in Fig. 9. We can see that the runtime of the HFM is much faster than that of COT whose computational complexity is very high.

6. Discussion: multiple resolutions

In this study, we provide a framework for rating predictions at multiple scales, from coarser to finer resolutions. Further, finer resolutions are nested within coarser resolutions in cross-context hierarchies.

After the training phase, there are estimated parameters at each node on the cross-context hierarchy. That is, for each test tuple, there are nodes at different levels on the hierarchy that can make rating predictions. Recall that the model parameters of non-leaf nodes are learnt by using aggregated training data. Thus, the rating predictions at the non-leaf nodes are coarse-grained resolutions. With an increase in the depth of the level, we can obtain finer resolutions. Finally, the finest-grained resolutions are at the leaf nodes.

In general, there is a relatively high degree of novelty at coarse-grained resolutions. However, fine-grained resolutions provide higher accuracy. It is important to consider a balance both in accuracy and novelty, as irrelevant recommendations may be new to a user, but still useless [36]. In real scenarios, the proper resolutions can be selected so as to achieve a trade-off between novelty and accuracy.

7. Related work

7.1. Collaborative filtering with hierarchies

Collaborative filtering (CF) is a very successful approach for recommender systems. The core idea of CF is that similar users will have similar preference. The low dimensional factorization model is one of the most popular approaches in collaborative filtering. Traditional factorization models do not make use of pre-existing or implied hierarchies. Recent works have incorporated the hierarchies into factorization models. Menon et al. [26] took advantage of pre-defined hierarchies of pages and ads and proposed hierarchical regularization to generate response predictions. The basic idea was to maintain a latent vector for each node in the page and ad hierarchies, and enforce priors based on this to induce correlations. However, in their study, the parent nodes failed to borrow information from its children nodes. Zhong et al. [50] proposed a random partition matrix factorization model that splits the rating matrix hierarchically by grouping similar users and items together, and employs classic matrix factorization to each sub-matrix. However, the model was a model-averaging gradient boosting model that needed to build many random decision trees and it was not appropriate for big data. Liu et al. [24] proposed to split the rating matrix by randomly selecting contextual information and only employing classic matrix factorization at leaf nodes. The final predictions were made by combining many decision trees. Besides it was not appropriate for big data, the model was unable to smooth the predictions of different nodes of each tree. Oentaryo et al. [28] developed a Hierarchical Importance-aware Factorization Machine (HIFM), which was tailored for ad response datasets, in which impressions are used as important weights. Therefore, the method could not be applied on other general data sets without impression values. In addition, HIFM only borrowed information from parents, while our method can obtain optimal estimates of parameters based on all the data observed on the cross-context hierarchy. Wang et al. [39] proposed a HSR framework to capture the implicit hierarchical structures of users and items simultaneously. However, it was a 2-dimensional factorization model essentially. Therefore, it could not take full advantage of contextual information on real world applications. Wang et al. [41] exploited the geographical correlation over a well-designed spatial index structure called spatial pyramid. However, the index structure could not be applied to the other contexts.

7.2. Context-aware recommendation

Users' decisions are usually affected by contextual information, which includes time, location, companion, mood, weather etc. Karatzoglou et al. [16] proposed a Multiverse Recommendation by modeling the data as a user-item-context N-dimensional tensor. High Order Singular Value Decomposition (HOSVD) was applied to factorize the tensor. However, this model was only applicable for categorical contextual information and the computational complexity was very high. Rendle et al. [34] employed Factorization Machines (FM) to achieve context-aware recommendations. In FM, a wide variety of context-aware data can be transformed into real valued feature vectors and FM models all interactions between pairs of variables with the target. Compared to Multiverse Recommendation, FM achieved much faster runtime in terms of training, prediction, and prediction quality. There were some extensions to the FM model. Hong et al. [12] proposed co-FM to model user interests and predict individual decisions on twitter. Cheng et al. [6] proposed a Gradient Boosting Factorization Machine (GBFM) model that incorporates a feature selection algorithm with FM in a unified framework. Wang et al. [38] proposed random partition factorization machines (RPFM) for context-aware recommendations. However, the structure of random decision trees needs to be determined manually before training in the RPFM model. In addition, the computational complexity was relatively high. Yin et al. [45–48] adopted topic models for Point-Of-Interest recommendation. An intermediate layer that included latent variables and linked users and spatial items was employed. However, their model could not exploit the natural hierarchies of many contexts. The main contribution of this study is to incorporate hierarchy into FM and to address the possibility of parameter variation across levels and groups. Our experimental results on real datasets show that this parameter variation considerably improves the recommendation quality.

7.3. Kalman filtering algorithm

The standard Kalman filtering algorithm [15] describes a recursive solution to the discrete data linear filtering problem and provides an efficient computational means to estimate the state of a process. Huang et al. [13] developed Bayes' theorem to derive the generalized Kalman filtering algorithm for tree-structured models. The generalized Kalman filtering algorithm consists of two steps: up-tree filtering and down-tree smoothing. The algorithm can explore and exploit the data of between-level and between-group nodes so that locally optimal parameters can be learnt. Agarwal et al. [1] assigned a state variable to each node in the hierarchical structure and employed the generalized Kalman filtering algorithm to compute the posterior of the states. Finally, click rates for (webpage, advertisement) pairs at all resolutions were estimated. Zhang et al. [49] made use of a multi-scale Kalman filter to realize fast computation of the posterior mode in multi-level hierarchical models. However, they simply assumed all observations were available at leaf nodes. Agarwal et al. [2] proposed localized factor models for multi-context recommendation, where the generalized Kalman filtering algorithm was used for model fitting. However, there were only two levels of hierarchical structures in their model. As for context-aware recommendation, two levels are usually not enough. Existing works prove that the Kalman filtering algorithm can obtain optimal estimates of latent states based on all the data observed on a tree, and can make predictions at multiple resolutions. Therefore, the latter can be further explored to realize context-aware recommendation through the use of a cross-context hierarchy.

8. Conclusions and future work

In this study, we consider the problem of exploiting pre-existing or implied hierarchical information to improve the recommendation quality and propose a novel two-stage recommendation model called Hierarchical Factorization Machines (HFM). In the first stage, primary predictions that can be considered as a fixed effect at all resolutions are obtained after FM model parameters of each node are estimated. Conditional on the primary predictions, in the second stage, final predictions are determined through a tree-structured Markov model which is used to ascertain the random effect. Model fitting is achieved through an EM algorithm, wherein the generalized Kalman filtering algorithm is used in the inner loop. Extensive experiments on real datasets verify that the proposed model is efficient and effective.

In conclusion, we believe that our proposal represents a new paradigm in high-quality context-aware recommendation. This work is simply the first step, and there remain numerous challenging issues that are yet to be thoroughly researched, such as how to automatically extract and construct cross-context hierarchy and, how to provide a Spark framework based implementation. We are currently investigating latter issues in detail as further study.

Acknowledgments

This work is supported by National Key Research&Develop Plan (no. 2016YFB1000702), National Basic Research Program of China (973) (no. 2014CB340402), National High Technology Research and Development Program of China (863) (no. 2014AA015204) and NSFC under the grant no. 61532021, 61502421, and the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (15XNLQ06).

References

[1] D. Agarwal, A.Z. Broder, D. Chakrabarti, D. Diklic, V. Josifovski, M. Sayyadian, Estimating rates of rare events at multiple resolutions, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2007, pp. 16–25.

[2] D. Agarwal, B.-C. Chen, B. Long, Localized factor models for multi-context recommendation, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2011, pp. 609–617.
[3] H.J. Ahn, A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem, Inf. Sci. 178 (1) (2008) 37–51.
[4] J. Bobadilla, F. Ortega, A. Hernando, A. Gutirrez, Recommender systems survey, Knowl. Based Syst. 46 (0) (2013) 109–132.
[5] S. Chang, J. Zhou, P. Chubak, J. Hu, T.S. Huang, A space alignment method for cold-start TV show recommendations, in: Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI), AAAI Press, 2015, pp. 3373–3379.
[6] C. Cheng, F. Xia, T. Zhang, I. King, M.R. Lyu, Gradient boosting factorization machines, in: Proceedings of the 8th ACM Conference on Recommender systems, ACM, 2014, pp. 265–272.
[7] C. Cornelis, J. Lu, X. Guo, G. Zhang, One-and-only item recommendation with fuzzy logic techniques, Inf. Sci. 177 (22) (2007) 4906–4921.
[8] S. Craw, B. Horsburgh, S. Massie, Music recommenders: user evaluation without real users? in: Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI), AAAI Press, 2015.
[9] A.S. Das, M. Datar, A. Garg, S. Rajaram, Google news personalization: scalable online collaborative filtering, in: Proceedings of the 16th International Conference on World Wide Web, ACM, 2007, pp. 271–280.
[10] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. Ser. B (Methodol.) 39 (1977) 1–38.
[11] H. Gao, J. Tang, X. Hu, H. Liu, Content-aware point of interest recommendation on location-based social networks., in: Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI), AAAI Press, 2015, pp. 1721–1727.
[12] L. Hong, A.S. Doumith, B.D. Davison, Co-factorization machines: modeling user interests and predicting individual decisions in twitter, in: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, ACM, 2013, pp. 557–566.
[13] H.-C. Huang, N. Cressie, Multiscale graphical modeling in space: applications to command and control, in: Spatial Statistics: Methodological Aspects and Applications, Springer, 2001, pp. 83–113.
[14] X. Jiang, W. Liu, L. Cao, G. Long, Coupled collaborative filtering for context-aware recommendation, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
[15] R.E. Kalman, A new approach to linear filtering and prediction problems, J. Fluids Eng. 82 (1) (1960) 35–45.
[16] A. Karatzoglou, X. Amatriain, L. Baltrunas, N. Oliver, Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering, in: Proceedings of the Fourth ACM Conference on Recommender Systems, ACM, 2010, pp. 79–86.
[17] D. Kim, C. Park, J. Oh, S. Lee, H. Yu, Convolutional matrix factorization for document context-aware recommendation, in: Proceedings of the 10th ACM Conference on Recommender Systems, ACM, 2016, pp. 233–240.
[18] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2008, pp. 426–434.
[19] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.
[20] S.K. Lee, Y.H. Cho, S.H. Kim, Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations, Inf. Sci. 180 (11) (2010) 2142–2155.
[21] K.H. Lim, J. Chan, C. Leckie, S. Karunasekera, Personalized tour recommendation based on user interests and points of interest visit durations, in: Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI), AAAI Press, 2015.
[22] C. Lin, R. Xie, X. Guan, L. Li, T. Li, Personalized news recommendation via implicit social experts, Inf. Sci. 254 (0) (2014) 1–18.
[23] Q. Liu, S. Wu, L. Wang, COT: contextual operating tensor for context-aware recommender systems, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
[24] X. Liu, K. Aberer, SoCo: a social network aided context-aware recommender system, in: Proceedings of the 22nd International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2013, pp. 781–802.
[25] Z. Lu, Z. Dou, J. Lian, X. Xie, Q. Yang, Content-based collaborative filtering for news topic recommendation., in: AAAI, 2015, pp. 217–223.
[26] A.K. Menon, K.-P. Chitrapura, S. Garg, D. Agarwal, N. Kota, Response prediction using collaborative filtering with hierarchies and side-information, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2011, pp. 141–149.
[27] F. Narducci, P. Basile, C. Musto, P. Lops, A. Caputo, M. de Gemmis, L. Iaquinta, G. Semeraro, Concept-based item representations for a cross-lingual content-based recommendation process, Inf. Sci. 374 (2016).
[28] R.J. Oentaryo, E.-P. Lim, J.-W. Low, D. Lo, M. Finegold, Predicting response in mobile advertising with hierarchical importance-aware factorization machine, in: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, ACM, 2014, pp. 123–132.
[29] C. Ono, Y. Takishima, Y. Motomura, H. Asoh, in: Context-Aware Preference Model Based on a Study of Difference Between Real and Supposed Situation Data, 9, Springer, 2009, pp. 102–113.
[30] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: bringing order to the web. (1999).
[31] C. Park, D. Kim, J. Oh, H. Yu, Improving top-k recommendation with truster and trustee relationship in user trust network, Inf. Sci. 374 (2016) 100–114.
[32] S. Rendle, Factorization machines, in: Proceedings of the 10th IEEE International Conference on Data Mining (ICDM), IEEE, 2010, pp. 995–1000.
[33] S. Rendle, Factorization machines with libfm, ACM Trans. Intell. Syst. Technol. (TIST) 3 (3) (2012) 57.
[34] S. Rendle, Z. Gantner, C. Freudenthaler, L. Schmidt-Thieme, Fast context-aware recommendations with factorization machines, in: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2011, pp. 635–644.
[35] J. Serrano-Guerrero, E. Herrera-Viedma, J.A. Olivas, A. Cerezo, F.P. Romero, A Google wave-based fuzzy recommender system to disseminate information in university digital libraries 2.0, Inf. Sci. 181 (9) (2011) 1503–1516.
[36] B. Shapira, F. Ricci, P.B. Kantor, L. Rokach, Recommender systems handbook.(2011).
[37] M.G. Vozalis, K.G. Margaritis, Using SVD and demographic data for the enhancement of generalized collaborative filtering, Inf. Sci. 177 (15) (2007) 3017–3037.
[38] S. Wang, C. Du, K. Zhao, C. Li, Y. Li, Y. Zheng, Z. Wang, H. Chen, Random partition factorization machines for context-aware recommendations, in: the 17th International Conference on Web-Age Information Management, Springer, 2016, pp. 219–230.
[39] S. Wang, J. Tang, Y. Wang, H. Liu, Exploring implicit hierarchical structures for recommender systems, in: Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI), 2015.
[40] S. Wang, B. Zou, C. Li, K. Zhao, Q. Liu, H. Chen, Crown: a context-aware recommender for web news, in: Proceedings of the 31st IEEE International Conference on Data Engineering (ICDE), IEEE, 2015, pp. 1420–1423.
[41] W. Wang, H. Yin, L. Chen, Y. Sun, S. Sadiq, X. Zhou, Geo-sage: a geographical sparse additive generative model for spatial item recommendation, in: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 1255–1264.
[42] B. Xie, P. Han, F. Yang, R.-M. Shen, H.-J. Zeng, Z. Chen, Dcfla: a distributed collaborative-filtering neighbor-locating algorithm, Inf. Sci. 177 (6) (2007) 1349–1363.
[43] H. Yin, B. Cui, L. Chen, Z. Hu, C. Zhang, Modeling location-based user rating profiles for personalized recommendation, ACM Trans. Knowl. Discovery Data (TKDD) 9 (3) (2015) 19.
[44] H. Yin, B. Cui, L. Chen, Z. Hu, X. Zhou, Dynamic user modeling in social media systems, ACM Trans. Inf. Syst. (TOIS) 33 (3) (2015) 10.
[45] H. Yin, B. Cui, Z. Huang, W. Wang, X. Wu, X. Zhou, Joint modeling of users' interests and mobility patterns for point-of-interest recommendation, in: Proceedings of the 23rd ACM international conference on Multimedia, ACM, 2015, pp. 819–822.
[46] H. Yin, B. Cui, X. Zhou, W. Wang, Z. Huang, S. Sadiq, Joint modeling of user check-in behaviors for real-time point-of-interest recommendation, ACM Trans. Inf. Syst. (TOIS) 35 (2) (2016) 11.
[47] H. Yin, X. Zhou, B. Cui, H. Wang, K. Zheng, Q.V.H. Nguyen, Adapting to user interest drift for poi recommendation, IEEE Trans. Knowl. Data Eng. 28 (10) (2016) 2566–2581.

[48] H. Yin, X. Zhou, Y. Shao, H. Wang, S. Sadiq, Joint modeling of user check-in behaviors for point-of-interest recommendation, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, ACM, 2015, pp. 1631–1640.
[49] L. Zhang, D. Agarwal, Fast computation of posterior mode in multi-level hierarchical models, in: Advances in Neural Information Processing Systems, 2009, pp. 1913–1920.
[50] E. Zhong, W. Fan, Q. Yang, Contextual collaborative filtering via hierarchical matrix factorization., in: SDM, SIAM, 2012, pp. 744–755.
[51] E. Zhong, N. Liu, Y. Shi, S. Rajan, Building discriminative user profiles for large-scale content recommendation, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 2277–2286.
[52] B. Zou, C. Li, L. Tan, H. Chen, Gputensor: efficient tensor factorization for context-aware recommendations, Inf. Sci. 299 (2015) 159–177.
[53] Á. Tejeda-Lorente, C. Porcel, E. Peis, R. Sanz, E. Herrera-Viedma, A quality based recommender system to disseminate information in a university digital library, Inf. Sci. 261 (0) (2014) 52–69.