Data Sci. Eng. (2017) 2:125–135
DOI 10.1007/s41019-017-0035-3

Context-Aware Recommendations with Random Partition Factorization Machines

Shaoqing Wang[1,2,3] · Cuiping Li[1,2] · Kankan Zhao[1,2] · Hong Chen[1,2]

Received: 16 November 2016 / Revised: 6 February 2017 / Accepted: 9 February 2017 / Published online: 24 February 2017

Abstract Context plays an important role in helping users to make decisions. There are hierarchical structure between contexts and aggregation characteristics within the context in real scenarios. Exist works mainly focus on exploring the explicit hierarchy between contexts, while ignoring the aggregation characteristics within the context. In this work, we explore both of them so as to improve accuracy of prediction in recommender systems. We propose a Random Partition Factorization Machines (RPFM) by adopting random decision trees to split the contexts hierarchically to better capture the local complex interplay. The intuition here is that local homogeneous contexts tend to generate similar ratings. During prediction, our method goes through from the root to the leaves and borrows from predictions at higher level when there is sparseness at lower level. Other than estimation accuracy of ratings, RPFM also reduces the over-fitting by building an ensemble model on multiple decision trees. We test RPFM over three different benchmark contextual datasets. Experimental results demonstrate that RPFM outperforms state-of-the-art context-aware recommendation methods.

Keywords Context-aware recommendations · Hierarchical information · Factorization Machines · Random decision trees

A short version of this paper appeared in the proceedings of the 17th International Conference on Web-Age Information Management (WAIM2016) [26]. Different from the conference paper, the new contents of this paper include the following. (1) We add a discussion about the relationship between the proposed RPFM and other state-of-the-art random partition based methods in Sect. 4.2. (2) We add some experiments to assess the performance of proposed RPFM with different similarity measure function in *k*-means method. (3) Besides, introduction, related works, preliminaries, and future work are all been extended in Sects. 1, 2, 3, and 6, respectively.

✉ Cuiping Li
licuiping@ruc.edu.cn

Shaoqing Wang
wsq@ruc.edu.cn

Kankan Zhao
zhaokankan@ruc.edu.cn

Hong Chen
chong@ruc.edu.cn

[1] Key Lab of Data Engineering and Knowledge Engineering of MOE, Beijing, China

[2] Renmin University of China, Beijing, China

[3] School of Computer Science and Technology, Shandong University of Technology, Zibo, China

1 Introduction

With the rapid development of web 2.0 and wireless communication technologies, we are going through a new era of information overload. That is, it is difficult to quickly find the available information for users. To cope with the challenge, there are two solutions: information retrieval [19] and recommender systems [12]. If users can express their requirement clearly, information retrieval is a good method to help them. For example, when will start the next game of Real Madrid football club? However, it is difficult to generate the specific demand in many cases. Such as, what the Internet is talking about right now, which movie is the most interesting recently, which book should I buy? Recommender systems can give the answers.

Recommender systems have become an important tool to help users to easily find the favorite items. In general, recommender systems can be divided into three categories: content-based recommendation, collaborative filtering and hybrid recommendation [9]. Content-based recommendation can play an important role to help users making decisions when the content can be abstracted from the items, e.g., news [4], jokes, books, reviews. However, the recommended items are very familiar to the users. Lack of novelty is the one of the weak points of content-based recommendation. The main idea of collaborative filtering approaches is to exploit information about the past behaviors of all users of the system for predicting which items the current user will most probably be interested in. Pure collaborative filtering approaches take a matrix of given user–item ratings as the only input. Though collaborative filtering approaches achieves great success, there are some shortcomings. For example, cold-start items cannot be recommended, and popular items often be recommended. Due to known limitations of either pure content-based recommender systems or collaborative filtering , it rather soon led to the development of hybrid recommendation that combine the advantages of different recommendation techniques. In this work, we focus on collaborative filtering by exploiting the hierarchal information implied to improve the performance of recommendations.

Collaborative filtering [11, 12, 23, 25] methods that behind the recommender systems have been developed for many years and is still a hot research topic up to now. User's decisions (e.g., clicked, purchased, re-tweeted, commented) to the relevant items are made under the certain environments which is often referred to as *context*. The contexts which include time, location, mood, companion and so on can be collected easily in real-world applications. Comparing to conventional recommendation solely based on user–item interactions, context-aware recommendation (CAR) can significantly improve the recommendation quality.

For this purpose, a great number of context-aware recommendation methods [10, 13, 21] have been proposed. Among them, Factorization Machines (FM) [21] is currently an influential and popular one. It represents the user–item–context interactions as a linear combination of the latent factors to be inferred from the data and treats the latent factors of user, item and context equality. Despite its successful application, existing FM model is weak to utilize hierarchical information. In practice, hierarchies can capture broad contextual information at different levels and hence ought to be exploited to improve the recommendation quality. The intuition here is that local homogeneous contexts tend to generate similar ratings. For example, many men who are engaged in IT department like to browse on technology Web sites in office during the day. However, they enjoy visiting sport Web sites at home in the evening. Here, users may be arranged in a hierarchy based on gender or occupation, Web sites may be characterized by contents, and there are natural hierarchies for time and location.

In this paper, we focus on solving the problem of exploiting the hierarchical information to improve the recommendation quality. We propose a Random Partition Factorization Machines (RPFM) by adopting random decision trees to split the contexts hierarchically to better capture the local complex interplay. More specifically, the user–item–context interactions are first partitioned to different nodes of a decision tree according to their local contexts. Then, FM model is applied to the interactions of each node to capture the tight impact of each other. During prediction, our method goes through from the root to the leaves and borrows from predictions at higher level when there is sparseness at lower level. Other than estimation accuracy of rating, RPFM also reduces the over-fitting by building an ensemble model on multiple decision trees. The main contribution of the paper is summarized as follows:

1. FM model is one of the most successful approaches for context-aware recommendation. However, there is only one set of the model parameters which can be learned from the whole training set. We propose the novel RPFM model which makes use of the intuition that similar ratings can be generated from homogeneous environments.
2. We adopt the *k*-means cluster method to partition the user–item–context interactions at each node of decision trees. The similarity between the latent factor vectors of FM model can be used to partition the user–item–context interactions. The subset at each node is expected to be more impacted each other.
3. We conduct experiments on three datasets and compare it with five state-of-the-art context-aware recommendations to demonstrate RPFM's performance.

The rest of the paper is organized as follows: In Sect. 2, we provide related works about context-aware and random partition-based models. In Sect. 3, we introduce the FM model. In Sect. 4, we propose the Random Partition Factorization Machines (RPFM) model which includes algorithm description and discussion with two state-of-the-art random partition-based models. In Sect. 5, we present the experimental result on three real datasets. The paper is concluded in Sect. 6, and the future research direction is outlined.

2 Related Works

The work presented in this paper is closely related to context-aware recommendation and random partition on tree structure. In the following, we introduce the related works to serve as background for our solution.

2.1 Context-Aware Recommendation

In general, there are three types of integration method [2]: (1)contextual pre-filtering method; (2) contextual post-filtering method; and (3) contextual modeling method. In contrast to the previous two methods, the contextual modeling method uses all the contextual and user–item information simultaneously to make predictions. More recent works have focused on the third method [10, 13, 24, 27].

Karatzoglou et al. [10] proposed Multiverse Recommendation model in which the different types of context are considered as additional dimensions in the representation of the data as a tensor. The factorization of this tensor leads to a compact model of the data which can be used to provide context-aware recommendations. However, for real-world scenarios its computational complexity is too high. Rendle [24] showed that Factorization Machines (FM) model can be applied to context-aware recommendation because that a wide variety of context-aware data can be transformed into prediction task using real-valued feature vectors. Nguyen et al. [16] developed a nonlinear probabilistic algorithm for context-aware recommendation using Gaussian processes which is called Gaussian Process Factorization Machines (GPFM). GPFM is applicable to both the explicit feedback setting and the implicit feedback setting. Currently, the most recent approach in terms of prediction accuracy is COT [13] model, which represented the common semantic effects of contexts as a contextual operating tensor and represents a context as a latent vector. Then, to model the semantic operation of a context combination, it generates contextual operating matrix from the contextual operating tensor and latent vectors of contexts. Thus latent vectors of users and items can be operated by the contextual operating matrices. However, its computational complexity is also too high.

2.2 Random Partition on Tree Structure

Fan et al. [5] proposed Random Decision Trees which are applicable for classification and regression to partition the rating matrix and build ensemble. Each time, according to the feature and threshold which were selected randomly, the instances at each intermediate nodes are partitioned into two parts. Zhong et al. [28] proposed Random Partition Matrix Factorization (RPMF), based on a tree structure constructed by using an efficient random partition technique, which explore low-rank approximation to the current sub-rating matrix at each node. RPMF combines the predictions at each node (non-leaf and leaf) on the decision path from root to leaves. Liu et al. [14] handled contextual information by using random decision trees to partition the original user–item–rating matrix such that the ratings with similar contexts are grouped. Matrix factorization was then employed to predict missing ratings of users for items in the partitioned sub-matrix.

3 Preliminaries

In this section, we briefly review Factorization Machines (FM) which is closely related to our work.

The notations used in this paper are summarized in Table 1.

3.1 Factorization Machines

Factorization Machines (FM), proposed by Rendle [21], is a general predictor which can mimic classical models like biased MF [12], SVD++ [11], PITF [25] or FPMC [23]. The model equation for FM of degree $d = 2$ is defined as:

$$\hat{y}(x_i) = \omega_0 + \sum_{j=1}^{p} \omega_j x_{i,j} + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} \langle <\mathbf{v}_j, \mathbf{v}_{j'} \rangle x_{i,j} x_{i,j'}, \tag{1}$$

and

$$\langle \mathbf{v}_j, \mathbf{v}_{j'} \rangle := \sum_{k=1}^{f} v_{j,k} \cdot v_{j',k}, \tag{2}$$

where the mode parameters Θ that have to be estimated are:

$$\omega_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{V} \in \mathbb{R}^{f \times p}. \tag{3}$$

A row vector $\mathbf{v_i}$ of $\mathbf{V}$ represents the ith variable with f factors. $f \in \mathbb{N}_0^+$ is the dimensionality of the factorization.

The model equation of a factorization machine in Eq. (1) can be computed in linear time $O(f * p)$ because the pairwise interaction can be reformulated:

Table 1 Definition of notations

Notation	Description
R	Training set
R_{dj}	jth training subset at dth level
n_i	Number of the context C_i
m	Number of the contextual variables
f	Dimensionality of latent factor vectors
k	Number of clusters in k-means method
fun	Similarity function in k-means method
ω_0	Global bias
ω_i	Strength of the ith variable.
$\mathbf{v_i}$	Factor vector of the ith variable
S	Structure of tree
N	Number of trees
h	Height of tree
$leastT$	Number of least support tuples at leaf node

Table 2 An example of training set of FM model

	Users			Items				Locations				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	1	0	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	1	0	1	0	2
x_6	0	0	1	0	1	0	0	0	0	0	1	3

$$\sum_{j=1}^{p}\sum_{j'=j+1}^{p}\langle \mathbf{v}_j, \mathbf{v}_{j'}\rangle x_{i,j}x_{i,j'} = \frac{1}{2}\sum_{k=1}^{f}\left(\left(\sum_{j=1}^{p} v_{j,k}x_{i,j}\right)^2 - \sum_{j=1}^{p} v_{j,k}^2 x_{i,j}^2\right)$$

Table 2 shows an example of input formation of training set. Here, there are $|U| = 3$ users, $|I| = 4$ items, $|L| = 4$ locations, which are binary indicator variables.

$$U = \{u_1, u_2, u_3\}$$
$$I = \{i_1, i_2, i_3, i_4\}$$
$$L = \{l_1, l_2, l_3, l_4\}$$

The first tuple x_1 means that user u_1 consumed i_1 at l_1 and rated it as 4 stars. For simplicity, we only consider categorical features in the paper. Table 3 shows the model parameters learned from the training set which is shown in Table 2.

3.2 Extensions to FM

There are a lot of extensions to FM model. Freudenthaler et al. [6] presented simple and fast structured Bayesian learning for FM model. Rendle [22] scaled FM to relational data. Hong et al. [8] proposed co-FM to model user interests and predicted individual decisions in twitter. Qiang et al. [20] exploited ranking FM for microblog retrieval. Loni et al. [15] presented 'Free lunch' enhancement for collaborative filtering with FM. Oentaryo et al. [17] predicted response in mobile advertising with hierarchical importance-aware FM. Cheng et al. [3] proposed a Gradient Boosting Factorization Machine (GBFM) model to incorporate feature selection algorithm with FM into a unified framework. To the best of our knowledge, there is no extension to FM model integrated into random decision trees such as to exploit the universal context-aware recommendations.

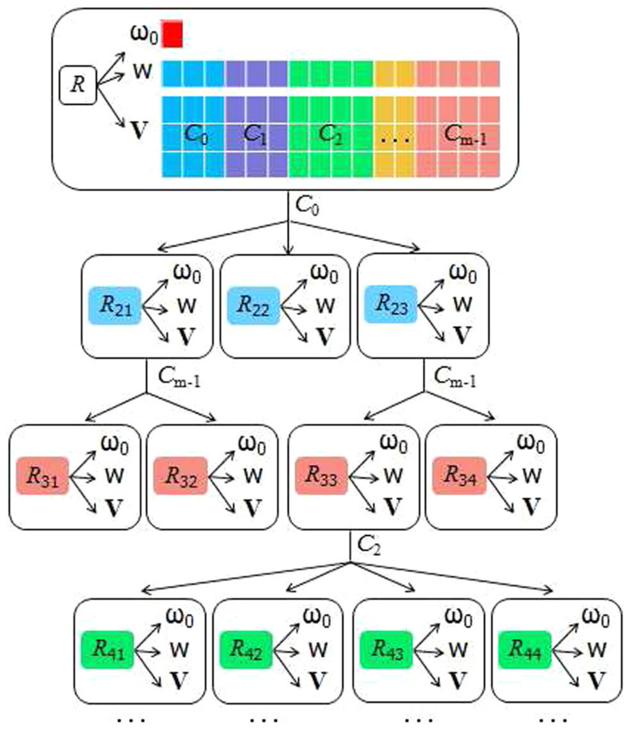

Fig. 1 Random decision trees (one tree)

4 Random Partition Factorization Machines

The intuition is that there are similar rating behaviors among users under the same or similar contextual environments. Motivated by Zhong et al. [28], We describe the proposed Random Partition Factorization Machines (RPFM) for context-aware recommendations.

4.1 Algorithm Description

In order to efficiently take advantage of different contextual information, we adopt the idea the random decision trees algorithm.

The rational is to partition the original training set R such that the tuples generated by the similar users, items

Table 3 An example of parameters' values of FM model

	Users			Items				Locations			
w_0	1.86										
w	0.81	0.22	0.80	0.49	0.05	1.15	0.05	1.10	0.51	−0.10	0.49
V	0.03	0.06	0.03	−0.03	0.01	−0.06	0.03	−0.06	−0.08	−0.01	−0.03
	−0.03	−0.07	−0.02	0.00	0.01	0.10	0.00	0.10	0.13	−0.02	0.01
	0.03	0.03	0.02	0.01	0.01	−0.07	0.01	−0.07	−0.09	0.04	0.01
	−0.02	−0.02	−0.02	0.00	0.00	−0.07	−0.01	−0.06	−0.07	0.03	0.01

or contexts are grouped into the same node. Tuples in the same cluster are expected to be more correlated each other than those in original training set *R*. The main flow can be found in Fig. 1 and Algorithm 2.

To begin with, there is an input parameter *S*, the structure of decision trees, which can be generated by Algorithm 1 and determined by cross-validation. The parameter *S* includes contexts for partition at each level, numbers of clusters at each node. The maximal depth of trees can be inferred from the parameter *S*. For instance, if the value of *S* is 'C2:4,C3:6,C1:10,C0:5', the meaning is: (1) at the root node of decision trees, the *R* can be divided into four groups by using *k*-means method according to the similarity between factor vectors of context C_2. Subsequently, the set at each node of 2nd, 3rd and 4th level of decision trees can be, respectively, divided into six, ten and five groups according to the similarity between factor vectors of context C_3, C_1 and C_0 using *k*-means method. (2) The maximal depth of each tree is five because there are four intermediate levels and one terminal level.

Algorithm 1 GenerateTreesStructure

```
Input: Depth of decsion trees: $h$, Numbers of Each Context: $n_0,...,n_{m-1}$
Output: Structure of Decision Trees
1: Initialize contextual information set $A$;
2: for $i$=0 to $m-1$ do
3:     $A(i)=C_i$;
4: end for
5: $j$=0;
6: for $i$=0 to $m-1$ do
7:     Select a context $C_r$ from $A$ randomly;
8:     Select $k \in [2, n_r]$ randomly such that the set at each node of $(i+1)$th level of decision
       trees will be divided into $k$ groups;
9:     Add $C_r$ and $k$ to $S$;
10:    if $j >= h$ then
11:        break;
12:    end if
13:    $A = A \setminus C_r$;
14:    $j$++;
15: end for
16: return $S$;
```

Algorithm 2 RPFM

```
Input: Training Set: $R$, Dimensionality of Latent Factor Vectors: $f$, Number of Trees: $N$,
    Structure of Trees: $S$, Number of Least Support Instances at the Leaf Node: $leastL$,
    Similarity Function in $k$-means method: $fun$, Learning Rate: $\eta$, Regularization Values:
    $\lambda$
Output: Tree Ensemble with Model Parameters and Cluster Information at Each Node.
1: Get maximum depth of each tree to $h$ according to $S$;
2: for $i = 1$ to $N$ do
3:     Build tree $T_i$;
4:     for $d = 2$ to $h$ do
5:         Get number of clusters to $k$ and context for partition to $C_r$ using $S$ and current
           level $d$-$1$;
6:         Learn the parameters $\Theta$ using Eq.(1) at each node of the current level $d$-$1$;
7:         Partition R into $k$ clusters using context $C_r$ and matrix $\mathbf{V}$ in $\Theta$ by taking advantage
           of $k$-means method;
8:         for $j = 1$ to $k$ do
9:             if the size of cluster $j$ is less than $leastL$ then
10:                Let the current node as leaf node;
11:                break;
12:            end if
13:        end for
14:        if $j > k$ then
15:            for $j = 1$ to $k$ do
16:                Decompose $R_{dj}$ recursively;
17:            end for
18:        end if
19:    end for
20: end for
21: return $\{T_i\}_{i=1}^{N}$;
```

At each node, we learn the model parameter using FM model.

$$\frac{\lambda_2}{2}\|\mathbf{w}\|^2+\frac{\lambda_3}{2}\|\mathbf{V}\|^2\hat{\omega}_0,\hat{\mathbf{w}},\hat{\mathbf{V}}$$
$$=arg\min_{\omega_0,\mathbf{w},\mathbf{V}}\sum_{i=1}^{|R|}(y(x_i)-\hat{y}(x_i))^2+\lambda\sum_{j=1}^{p}\|\mathbf{V}_j-\mathbf{V}_j^{pa}\|^2 \qquad (4)$$

where $\| * \|$ is the Frobenius norm and $\mathbf{V}^{\mathrm{pa}}$ is the latent factor matrix at parent node. The parameter λ controls the extent of regularization. Equation (4) can be solved using two approaches: (1) stochastic gradient descent (SGD) algorithms, which are very popular for optimizing factorization models as they are simple, work well with different loss functions. The SGD algorithm for FM has a linear computational and constant storage complexity [21]. (2) Alternating least-squares (ALS) algorithms that iteratively solves a least-squares problem per model parameter and updates each model parameter with the optimal solution [24]. Here, $\mathbf{V}$ is $f \times p$ matrix of which f is the dimensionality of factor vectors and $p = n_0 + n_1 + \ldots n_{m-1}$. n_i is the number of context C_i, m is the number of contextual variables. For simplicity, we denote user set as C_0 and item set as C_1. Each of the $f \times n_i$ sub-matrix is the latent representation of context C_i, as shown in Table 3. The smaller the distance among the factor vectors of context C_i, the greater the similarity.

To partition the training set R, we extract the context and the number of clusters according to the tree structure S and current level. We group the similar latent vectors of context C by making use of the k-means method, In Table 3, suppose we get the context C_1 (i.e., Item) and number of clusters $k = 2$ according to input parameter S. Then the initial cluster central points selected randomly are i_1 and i_2. Subsequently, the generated clustering result could be $\{i_1, i_3, i_4\}$ and $\{i_2\}$. Lastly, the training set in the current node can be divided into two groups according to the clustering result of context C_1 (i.e., Item) and the value of C_1 (i.e., Item) of tuples. In other words, the current node has two children nodes. The subset of one chid node includes the tuples whose value of C_1 (i.e., Item)$\in \{i_1, i_3, i_4\}$, the remaining tuples are assigned to the other children node.

The partition process stops once one of following conditions is met: (1) the height of a tree exceeds the limitation which can be inferred from the given tree structure parameter S; (2) the number of tuples at each child node of current node is less than the number of least support tuples *leastL*.

During training, the function of each non-leaf node is to separate training set by making use of the clustering result of special context, such that the tuples in the subset have more impact each other. However, leaf nodes are responsible for prediction.

Note that in different decision trees, the training set is divided differently because that initial k cluster central points are selected randomly at each node of decision trees.

During prediction, for a given case x_i in the test set, we transfer it from root node to leaf node at each tree using the clustering information of each non-leaf node. For instance, the value of S is 'C1:2, C0:3, C2:4' and a test case $x_i = \{u_3, i_1, l_2\}$ corresponding to Table 2. Thus from the root node, the x_i would be transferred to node (e.g., R_{23}) which include i_1 at second level. Then from the node R_{23}, the x_i would be transferd to node (e.g., R_{33}) which include u_3 at third level. Subsequently, from the node R_{33}, the x_i would be transferd to node (e.g., R_{41}) which include l_2 at fourth level. At the target leaf node, the rating can be predicted by taking advantage of Eq. (1) and the parameters learned by the training subset. To the end, the predictions from all trees are combined to obtain the final prediction as shown in Eq. (5)

$$\hat{y}(x_i) = \frac{\sum_{t=1}^{N}\hat{y}_t(x_i)}{N}, \qquad (5)$$

where $\hat{y}_t$ means the prediction of the tuple x_i at tth decision tree, N denotes the number of decision trees.

After partitioning the original training set, the tuples at each leaf node have the more influence on each other. So, the FM model at each leaf node can achieve high quality recommendation. By combining multiple predictions from different decision trees, all subsets in which the tuples are more correlated are comprehensively investigated, personalized and accurate context-aware recommendations can be generated.

4.2 Discussion

We discuss the relationship between the proposed RPFM and other state-of-the-art random partition-based methods.

- *Relation to RPMF* Zhong et al. [28] proposed RPMF works by applying a set of local decomposition processes on sub-rating matrices. There are some differences between RPMF and our proposed RPFM. First of all, RPMF explores a basic MF model to factorize the user–item rating matrix. However, RPFM factorizes the user–item–context interactions using FM model. Secondly, the decision trees in RPMF are binary trees created by selecting a latent factor from U, V and a splitting point randomly, while that in RPFM are irregular trees generated by k-means method where k initial cluster central points are selected randomly. Thirdly, the depth of decision trees in RPMF can be very large in theory, while that in RPFM is limited by the number of contextual variables. Finally, during prediction, for a given user–item pair, RPMF obtain a partial prediction at each node on the path from the root to leaf node on each decision tree. However, our proposed RPFM make a partial prediction only at the leaf node of each decision tree for a given user–item–context interaction tuple. So, RPMF spend more time in prediction than RPFM.

- *Relation to SoCo* Liu et al. [14] proposed SoCo to improve recommendation quality by using contexts and social network information. Here, we only pay attention to the relation between SoCo without social information and RPFM. Firstly, in SoCo contextual information c_r, used to separate data at each level of each tree, is selected randomly. Then the training data at each intermediate node are partitioned according to the value of c_r. However, the tree structure in RPFM is determined by the input parameter S and training subset is generated according to similarity of latent factor vectors of selected context c_r. Second, the prediction is made by the basic MF model in SoCo. However, our proposed RPFM makes prediction by taking advantage of FM model. It is worth mentioning that some contextual information which can improve recommendation quality may be lost in SoCo when the depth of tree is less than the number of contextual variables. For instance, the node R_{22} in Fig. 1 has no child node because the number of tuples at the node R_{22} is less than the number of least support tuples. If matrix factorization is performed, the contextual information at node R_{22} can not be taken advantage. However, our proposed RPFM can do it. Third, both the users and items cannot be used to split training set in SoCo. In other words, the number of tuples at leaf nodes in SoCo may be still enormous.

5 Experiments

In this section, we empirically investigate whether our proposed RPFM can achieve better performance compared with other state-of-the-art methods on three benchmark datasets. First we describe the datasets and settings in our experiments, then report and analyze the experiment results.

5.1 Datasets

We conduct our experiments on three datasets: the Adom. dataset [1], the Food dataset [18] as well as the Yahoo! Webscope dataset.

The Adom. dataset [1] contains 1757 ratings by 117 users for 226 movies with many contextual information. The rating scale rang from 1 (hate) to 13 (absolutely love). However, there are missing values in some tuples. After removing the tuples containing missing values, there are 1464 ratings by 84 users for 192 movies in Adom. dataset. We keep 5 contextual information: withwhom, day of the week, if it was on the opening weekend, month and year seen (Table 4).

Table 4 Data set statistics

Dataset	Users	Items	Context dim	Ratings	Scale
Adom.	84	192	5	1464	1–13
Food	212	20	2	6360	1–5
Yahoo!	7642	11,915	2	221,367	1–5

The Food dataset [18] contains 6360 ratings (1–5 stars) by 212 users for 20 menu items. We select 2 context variables. One context variable captures how hungry the user is: normal, hungry and full. The second one describes if the situation in which the user rates is virtual or real to be hungry.

The Yahoo! Webscope dataset contains 221,367 ratings (1–5 stars), for 11,915 movies by 7,642 users. There is no contextual information. However, the dataset contains user's age and gender features. Just like [24], we also follow [10] and apply their method to generate modified dataset. In other words, we modify the original Yahoo! dataset by replacing the gender feature with a new artificial feature $C \in \{0, 1\}$ that was assigned randomly to the value 1 or 0 for each rating. This feature C represents a contextual condition that can affect the rating. We randomly choose 50% items from the dataset, and for these items we randomly pick 50% of the ratings to modify. We increase (or decrease) the rating value by one if $C = 1 (C = 0)$ if the rating value was not already 5 (1).

5.2 Setup and Metrics

We assess the performance of the models by conducting a fivefold cross-validation and use the most popular metrics: the mean absolute error (MAE) and root mean square error (RMSE), defined as follows:

$$\text{MAE} = \frac{\sum_{(x_i, y_i) \in \Omega_{\text{test}}} |y_i - \hat{y}(x_i)|}{|\Omega_{\text{test}}|} \tag{6}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{(x_i, y_i) \in \Omega_{test}} (y_i - \hat{y}(x_i))^2}{|\Omega_{\text{test}}|}} \tag{7}$$

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

5.3 Performance Comparison

We first conduct some experiments to assess the performance of proposed RPFM with different similarity measure function in *k*-means method. Then we compared the performance of proposed RPFM with state-of-the-art context-aware methods.

5.3.1 What's the Better Method of Similarity Function?

The proposed RPFM algorithm takes advantage of *k*-means method to partition the training set. So the tuples in the each training subset are more impact each other. As we know, there are many metrics to measure the similarity among the tuples, for instance, Euclidean distance (*Euclid*), Cosine-based similarity (*Cosine*), correlation-based similarity (*Pearson*), adjusted Cosine-based similarity (*adjCosine*), etc. As shown in Table 5, there are some different performance under the different similarity measure function. However, the difference of performance is not significance. In the following sections, we thus report performances using Euclidean distance.

5.3.2 Comparison to Factorization-Based Context-Aware Methods

To begin with, we determine the structure of decision trees, i.e., input parameters *S*, by Algorithm 1. The parameters are 'C2:2,C6:2,C5:2,C3:3,C0:2,C4:5,C1:5', 'C3:3,C2:2, C0:5,C1:4' and 'C3:2,C2:2,C0:2,C1:2' for Adom., Food and Yahoo! dataset, respectively. Then, we select 0.01 as the values of learning rate and regularization.

Table 5 Performance comparison in terms of different similarity function

Dataset	Metric	RPFM			
		Euclid	Cosine	Pearson	adjCosine
Adom.	RMSE	2.642	**2.640**	2.643	2.649
	MAE	2.039	**2.032**	2.054	2.049
Food	RMSE	**1.022**	1.042	1.035	1.038
	MAE	**0.786**	0.814	0.800	0.813
Yahoo!	RMSE	**0.911**	0.933	0.928	0.926
	MAE	**0.617**	0.626	0.629	0.621

Bold numbers are the best performance in terms of different similarity function for each dataset

- *FM* [21] is easily applicable to a wide variety of context by specifying only the input data and achieves fast runtime both in training and prediction.
- *Multiverse* Recommendation [10] is a contextual collaborative filtering model using *N* dimensional tensor factorization. In Multiverse Recommendation, different types of context are considered as addition dimensions in the representation of the data as tensor. The factorization of this tensor leads to a compact model of the data which can be used to provide context-aware recommendations.
- *COT* [13] represents the common semantic effects of contexts as a contextual operating tensor and represents a context as a latent vector. Then, contextual operating matrix from the contextual operating tensor and latent vectors of contexts was generated so as to model the semantic operation of a context combination.

Dimensionality of latent factor vectors is one of the important parameters. Though latent factor vectors' dimensionality of various contexts can be different in Multiverse and COT. In order to compare with FM and our proposed RPFM, we just take into account the equal dimensionality of latent factor vectors of various contexts. The scale of three datasets is different, so we run models with $f \in \{2,3,4,5,6,7\}$ over Adom. dataset, $f \in \{2,4,6,8,10,12\}$ over Food dataset and $f \in \{5,10,15,20,25,30\}$ over Yahoo! dataset. Figures 2 and 3 show the result of FM, Multiverse, COT and RPFM over the three real-world datasets.

We notice that in all experiment scenarios, dimensionality of latent factor vectors in RPFM is not sensitive and RPFM is more accurate than other recommendation models. These results show that in homogeneous environment which can be obtained by applying random decision trees to partition the original training set, users have similar rating behavior.

High computational complexity for both learning and prediction is one of the main disadvantages of Multiverse and COT. This make them hard to apply for larger dimensionality of latent factor vectors. In contrast to this, the computational complexity of FM and RPFM is linear.

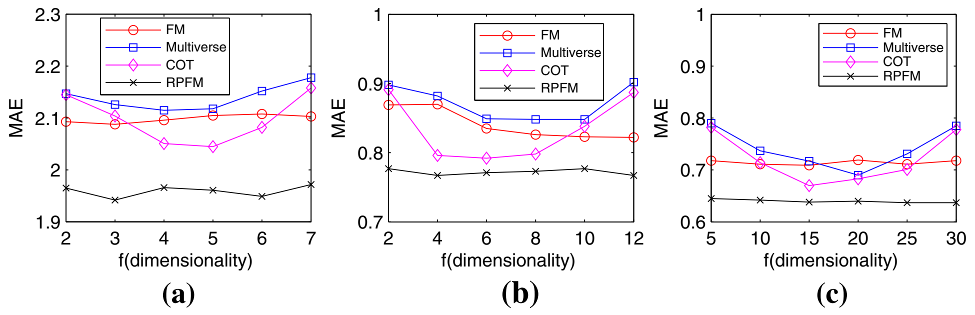

Fig. 2 MAE over three datasets with different dimensionality of latent factor vectors. **a** Adom. dataset, **b** food dataset, **c** Yahoo! dataset

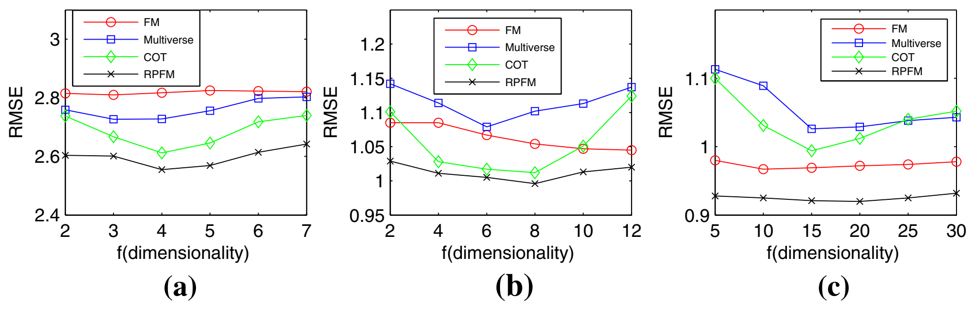

Fig. 3 RMSE over three datasets with different dimensionality of latent factor vectors. **a** Adom. dataset, **b** food dataset, **c** Yahoo! dataset

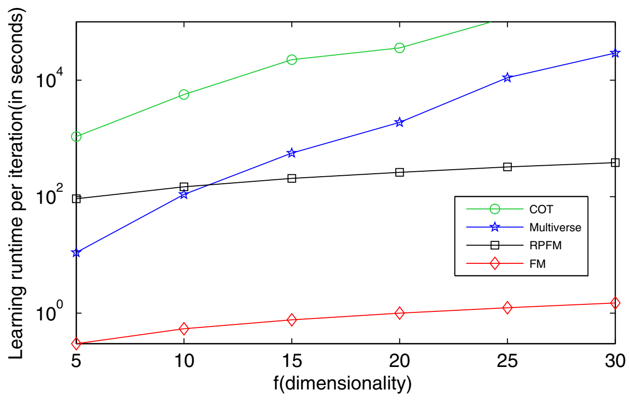

Fig. 4 Learning runtime in seconds for one iteration over the whole training set (in log-*y* scale) over Yahoo! dataset with different latent dimensions

In order to compare the runtime of various models, we do experiment on Yahoo! dataset for one full iteration over whole training set. Figure 4 shows that the learning runtime of RPFM is faster than that of Multiverse and COT with increasing the dimensionality, however, slower than that of FM which is obvious because RPFM generates an ensemble which reduces the prediction error.

5.3.3 *Comparison to Random Partition-Based Context-Aware Methods*

- *RPMF* [28] adopted a random partition approach to group similar users and items by taking advantage of decision trees. The tuples at each node of decision trees have more impact each other. Then matrix factorization is applied at each node to predict the missing ratings.
- *SoCo* [14] explicitly handle contextual information which means SoCo partitions the training set based on the values of real contexts. SoCo incorporate social network information to make recommendation. There are not social network information in our selected datasets, so we just consider SoCo without social network information.

Both the number and depth of trees have important impact on the decision tree-based prediction methods. Because of space limitations, we just report the experimental result over Food dataset.

As shown in Fig. 5, we observe that RPFM achieves the best performance compared with RPMF and SoCo. And we notice that MAE/RMSE decreases with increasing number of trees, which means more trees produces higher accuracy. However, when the number of trees increases to around 3, improvements on prediction quality become negligible. We thus conclude that even a small number of trees are sufficient for decision tree-based models.

The depth of trees which is one of the input parameters in RPMF can be very large because it can select a latent factor from *U*, *V* and a splitting point randomly at each intermediate node during building the decision trees. Here, we define the maximal depth of trees as five in RPMF. In SoCo, the maximal depth of trees equals the number of contextual variables excluding user and item. Specially, the

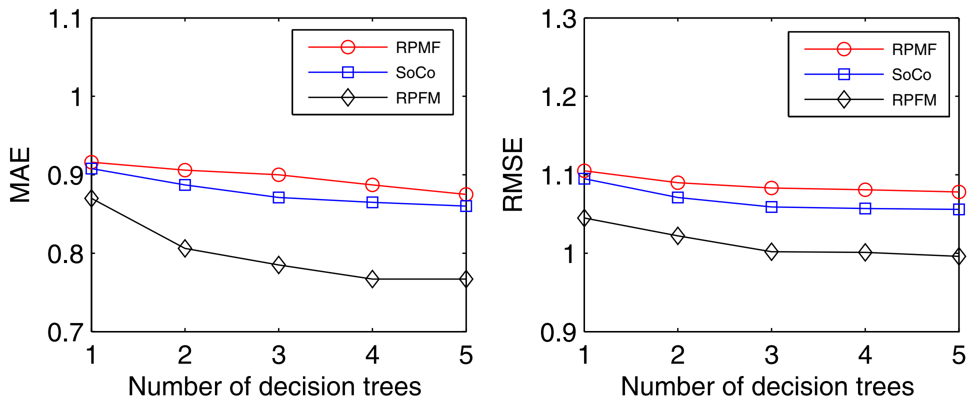

Fig. 5 Impact of number of trees over Food dataset

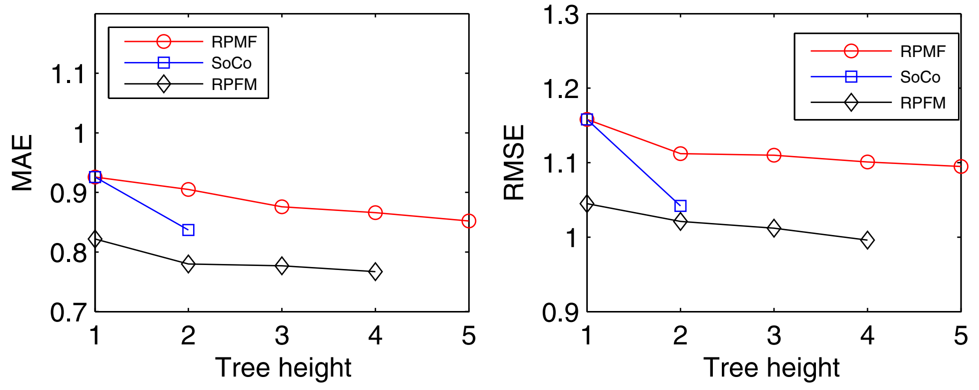

Fig. 6 Impact of depth of trees over Food dataset

maximal depth of trees over Food dataset is two in SoCo. However, both user and item can be considered as contextual variables in RPFM. Then the maximal depth of trees over Food dataset is four in RPFM. Figure 6 shows that the deeper of trees, the better prediction quality, and RPFM outperforms RPMF and SoCo in terms of MAE and RMSE.

6 Conclusion and Future Work

In this paper, we propose Random Partition Factorization Machines (RPFM) for context-aware recommendations. RPFM adopts random decision trees to partition the original training set using *k*-means method. Factorization machines (FM) is then employed to learn the model parameters at each node of the trees and predict missing ratings of users for items under some specific contexts at leaf nodes of the trees. Experimental results demonstrate that RPFM outperforms state-of-the-art context-aware recommendation methods.

There are several directions for future work on RPFM. First, RPFM adopts the *k*-means method to partition the training set. There are many cluster methods [7] such as BIRCH, ROCK, Chameleon, DBSCAN. Some of them may be achieve better performance. Second, manipulation at each node in training phase, such as clustering, partition and learning parameters, can be parallelized. Third, there are many floating point arithmetic at leaf nodes in prediction which will spend much time. While GPU hold powerful capacity of floating point arithmetic, it can be taken advantage to accelerate the prediction.

Acknowledgements 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. 2014AA 015204) and NSFC under the Grant Nos. 61272137, 61202114, 61532021 and NSSFC (No. 12&ZD 220), and the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (15XNLQ06). It was partially done when the authors worked in SA Center for Big Data Research in RUC. This Center is funded by a Chinese National 111 Project Attracting.

References

1. Adomavicius G, Sankaranarayanan R, Sen S, Tuzhilin A (2005) Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans Inf Syst TOIS 23(1):103–145
2. Adomavicius G, Tuzhilin A (2011) Context-aware recommender systems. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender systems handbook, pp. 217–253. Springer, Berlin
3. Cheng C, Xia F, Zhang T, King I, Lyu MR (2014) Gradient boosting factorization machines. In: Proceedings of the 8th ACM conference on recommender systems. ACM, pp 265–272
4. Das AS, Datar M, Garg A, Rajaram S (2007) Google news personalization: scalable online collaborative filtering. In: Proceedings of the 16th international conference on World Wide Web. ACM, pp 271–280
5. Fan W, Wang H, Yu PS, Ma S (2003) Is random model better? On its accuracy and efficiency. In: ICDM 2003, IEEE. pp 51–58
6. Freudenthaler C, Schmidt-Thieme L, Rendle S (2011) Bayesian factorization machines
7. Han J, Kamber M, Pei J (2011) Data mining: concepts and techniques: concepts and techniques. Elsevier, Amsterdam
8. Hong L, Doumith AS, Davison BD (2013) 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, pp 557–566
9. Jannach D, Zanker M, Felfernig A, Friedrich G (2010) Recommender systems: an introduction. Cambridge University Press, Cambridge
10. Karatzoglou A, Amatriain X, Baltrunas L, Oliver N (2010) Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In: Proceedings of the fourth ACM conference on Recommender systems. ACM, pp. 79–86

11. Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: SIGKDD2008. ACM, pp 426–434
12. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems, vol 8. IEEE, New York
13. Liu Q, Wu S, Wang L (2015) Cot: contextual operating tensor for context-aware recommender systems. In: Twenty-ninth AAAI conference on artificial intelligence
14. Liu X, Aberer K (2013) 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 pp 781–802
15. Loni B, Said A, Larson M, Hanjalic A (2014) 'Free lunch' enhancement for collaborative filtering with factorization machines. In: Proceedings of the 8th ACM conference on recommender systems. ACM, pp 281–284
16. Nguyen TV, Karatzoglou A, Baltrunas L (2014) Gaussian process factorization machines for context-aware recommendations. In: SIGIR2014. ACM, pp. 63–72
17. Oentaryo RJ, Lim EP, Low JW, Lo D, Finegold M (2014) 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, pp 123–132
18. Ono C, Takishima Y, Motomura Y, Asoh H (2009) Context-aware preference model based on a study of difference between real and supposed situation data. Springer, Berlin, p. 102–113
19. Page L, Brin S, Motwani R, Winograd T (1999) The PageRank citation ranking: Bringing order to the web
20. Qiang R, Liang F, Yang J (2013) Exploiting ranking factorization machines for microblog retrieval. In: Proceedings of the 22nd ACM international conference on conference on information and knowledge management. ACM, pp 1783–1788
21. Rendle S (2010) Factorization machines. In: 2010 IEEE 10th international conference on data mining (ICDM). IEEE, pp 995–1000
22. Rendle S (2013) Scaling factorization machines to relational data. In: Proceedings of the VLDB endowment, vol. 6. VLDB Endowment, pp 337–348
23. Rendle S, Freudenthaler C, Schmidt-Thieme L (2010) Factorizing personalized markov chains for next-basket recommendation. In: Proceedings of the 19th international conference on World wide web. ACM, pp 811–820
24. Rendle S, Gantner Z, Freudenthaler C, Schmidt-Thieme L (2011) Fast context-aware recommendations with factorization machines. In: SIGIR2011. ACM, pp 635–644
25. Rendle S, Schmidt-Thieme L (2010) Pairwise interaction tensor factorization for personalized tag recommendation. In: Proceedings of the third ACM international conference on web search and data mining. ACM, pp. 81–90
26. Wang S, Du C, Zhao K, Li C, Li Y, Zheng Y, Wang Z, Chen H (2016) Random partition factorization machines for context-aware recommendations. In: International conference on web-age information management. Springer, Berlin, pp 219–230
27. Yin H, Cui B, Chen L, Hu Z, Huang Z (2014) A temporal context-aware model for user behavior modeling in social media systems. In: SIGMOD 2014. ACM, pp 1543–1554
28. Zhong E, Fan W, Yang Q (2012) Contextual collaborative filtering via hierarchical matrix factorization. In: SDM. SIAM, pp 744–755