Soft Computing (2022) 26:4755–4770
https://doi.org/10.1007/s00500-022-06925-z

DATA ANALYTICS AND MACHINE LEARNING

CFDIL: a context-aware feature deep interaction learning for app recommendation

Qingbo Hao[1,2] · Ke Zhu[2,3,4,5] · Chundong Wang[1,2] · Peng Wang[6] · Xiuliang Mo[1,2] · Zhen Liu[6]

Accepted: 19 February 2022 / Published online: 16 March 2022

Abstract
The rapid development of mobile Internet has spawned various mobile applications (apps). A large number of apps make it difficult for users to choose apps conveniently, causing the app overload problem. As the most effective tool to solve the problem of app overload, app recommendation has attracted extensive attention of researchers. Traditional recommendation methods usually use historical usage data to explore users' preferences and then make recommendations. Although traditional methods have achieved certain success, the performance of app recommendation still needs to be improved due to the following two reasons. On the one hand, it is difficult to construct recommendation models when facing with the sparse user–app interaction data. On the other hand, contextual information has a large impact on users' preferences, which is often overlooked by traditional methods. To overcome the aforementioned problems, we proposed a context-aware feature deep interaction learning (CFDIL) method to explore users' preferences and then perform app recommendation by learning potential user–app relationships in different contexts. The novelty of CFDIL is as follows: (1) CFDIL incorporates contextual features into users' preferences modeling by constructing novel user and app feature portraits. (2) The problem of data sparsity is effectively solved by the use of dense user and app feature portraits, as well as the tensor operations for label sets. (3) CFDIL trains a new deep network structure, which can make accurate app recommendation using the contextual information and attribute information of users and apps. We applied CFDIL on three real datasets and conducted extensive experiments, which shows that CFDIL outperforms the benchmark methods.

Keywords Feature portrait · Context-aware · Interaction · App recommendation

✉ Chundong Wang
michael3769@163.com

Qingbo Hao
haoqingbo4546@163.com

1 Tianjin Key Laboratory of Intelligence Computing and Novel Software Technology, Tianjin University of Technology, Tianjin, China

2 School of Computer Science and Engineering, Tianjin University of Technology, Tianjin, China

3 Engineering Research Center of Learning-Based Intelligent System, Ministry of Education, Tianjin, China

4 Key Laboratory of Huang-Huai-Hai Smart Agricultural Technology, Ministry of Agriculture and Rural Affairs, Beijing, P. R. China

5 College of information science and engineering, Shandong Agricultural University, Tai'an, China

6 Graduate School of Engineering, Nagasaki Institute of Applied Science, Nagasaki, Japan

1 Introduction

With the vigorous development of the mobile Internet, the number of mobile applications (app) has increased dramatically, which provides great convenience to people's production and life. According to statistics,[1] as of 2020, there are more than 3.3 million and 2.1 million apps published on Google Play and App store, respectively. Facing the huge amount of apps, users often unable to accurately find the app that really meets their needs, i.e., users cannot solve the overload of apps. Therefore, it is of great significance to help users to personalize and accurately select apps that meet their needs.

As the most effective tool to solve the overload problem, recommender systems are widely used in news media, online shopping and social networking sites. Traditional recommen-

1 https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.

dation method leverages users' historical usage information to explore their preferences, and then make recommendations. The current mainstream method not only uses the historical user's feedback on apps, such as user's rating, comment, frequency of use, but also introduces the additional information of users or apps, such as the users' own attributes, apps' version information, category information. As a result, the recommendation performance can be further improved by building recommendation model based on the above information.

In recent years, deep neural network techniques have achieved great success in computer vision and natural language processing. Many researchers attempt to capture the relationships between users and items using deep neural networks to improve the performance of recommendation model. Even though progress has been gained, most deep recommendation models mine users' preferences through user-item interactions. The effects of spatiotemporal information on users' preferences have not been considered, i.e., research on the space-time laws of users and apps is not enough.

In conclusion, the existing recommendation methods have two problems as follows: (1) context information has much influence on users' selections to apps. However, existing methods considered only additional information about users and apps, not context information. (2) In the real world, user–app interaction data are very sparse. It is difficult to build recommendation model using spare data. In order to address these challenges, we proposed a context-aware feature deep interaction learning for app recommendation, called CFDIL.

CFDIL constructs feature portraits of users and apps, respectively, using context information and attribute information of users or apps. These feature portraits can describe exactly attributes and interactive features of users or apps, which is helpful for mining users' preferences in specific contexts. By introducing convolutional neural network and factorization machine to extract in-depth features of users and apps, CFDIL constructs deep learning framework based on these feature portraits to explore potential interactive features of users and apps in special contexts. Moreover, feature portraits enrich characterization data of users and apps, meanwhile, CFDIL processes sparse label set using tensor decomposition, which can effectively avoid the adverse effects on our model of sparse data to improve the performance of the recommendation method. The main contributions of this paper are as follows.

1. CFDIL proposes a method to construct contextual features of users and apps, and integrates the contextual features with the features of users or apps to form a feature portrait, respectively. Feature portraits can accurately describe the characteristics of users, apps and user–app interactions. Incorporating contextual information in feature portrait provides powerful information support for mining users' preferences of using apps in specific contexts.
2. CFDIL trains a novel deep network framework for feature portraits. By introducing convolutional neural networks (CNN) and factorization machine (FM), CFDIL effectively extracts the deep features of users and apps to explore the potential interactions in specific contextual conditions, which can provide more accurate recommendations for users.
3. The feature portraits of users and apps effectively enrich the representation data of users and apps, and make the model input denser. At the same time, CFDIL uses tensor factorization technique to process the label set for model back-propagation and weight updates. The use of dense feature portrait and label effectively avoid the adverse effects of sparse data on model learning.
4. We deployed CFDIL on three real datasets and implemented a large number of experiments. The experimental results show that CFDIL achieves state-of-the-art performance.

2 Related works

In this section, we introduce the related work of this paper. First, we introduce the classification of traditional app recommendation methods and then introduce the app recommendation methods based on deep learning.

2.1 The traditional app recommendation methods

The traditional app recommendation methods mainly include the following categories.

– Collaborative filtering-based recommendation method (CF-based method).
  CF-based method is a similarity-oriented recommendation method. This method is based on the assumption that a target user has similar preferences with users who have similar historical item experiences. Therefore, the most important part in CF-based method is to calculate the similarity between users or apps. The basic similarity construction method in CF-based method is based on user–app interaction matrix to calculate the similarity of users or apps. For instance, Kim et al. (2013) identify the most similar social members of target users based on semantic relations between apps, and then make app recommendations. Yankov et al. (2013) identify and analyze the relationship between apps in the apps ecosystem. Xia et al. (2014) leverage the app description text to calculate the similarity between apps, and then make app recommendations. Similar to Xia et al. (2014), Hao et al. (2016)

also use the description of app to calculate the similarity between apps. Liu and Wu (2016) leverage users' log of using apps to design a latent factor-based collaborative filter method. Hu et al. (2018) leverage the idea of user-based collaborative filtering to make recommendations. CF-based recommendation method has been widely used in many fields, including app recommendation, because of its simple logic and easy implementation. However, in the real world, user–app interaction matrix unable to reflect users' preferences due to it is extremely spare, which leads to poor performance. Aiming at the problem of data sparsity in app recommendation, the model-based app recommendation method is proposed.

– Model-based app recommendation method.
  The representative method among model-based recommendation methods is matrix factorization-based method (MF-based method). The basic idea of this method is as follows. First, MF-based method constructs an original user–app interaction matrix. The elements in the original matrix are the feedback information of a specific user for a specific app, such as rating, usage frequency. The matrix is a sparse matrix, which is similar with the user–app matrix in CF-based method. Then, the sparse user–app original matrix is processed into a non-empty matrix by matrix factorization technique. The non-empty elements in this matrix are considered to be a user's preference for an app. For example, Liu et al. (2013) give the introduction about MF-based app recommendation. Lin et al. (2014) leverage the probabilistic matrix factorization to explore users' preferences. Zhu et al. (2014a) leverage latent dirichlet allocation (LDA) model to map the interaction contextual information between users and apps into low-dimensional space and then make recommendation. This method is the same as MF-based recommendation. Yao et al. (2017) construct a user–app version rating matrix, and use matrix factorization method to explore users' preferences. Although MF-based method can resist the adverse effects of data sparsity on recommendation, the low order vectors generated in the processing of matrix factorization has no clear physical meaning, which leads to poor interpretability. This also leads to the lack of personalization in MF-based method. At the same time, due to the fact that apps are easy to developed, and new apps are constantly generated, and usually need to be added to the user–app interaction matrix for retraining, which also leads to the lack of scalability of MF-based method.
– The additional information-based recommendation method.
  There is lots of additional information in interaction between users and apps. For example, the contextual information of users using the app (Zhu et al. 2014a, b; Pu et al. 2018; Wang et al. 2016), user comments on apps (Zheng et al. 2014; Fu et al. 2013), app version information (Yao et al. 2017), app permission information (Liu et al. 2015), app description information (Chen et al. 2015), etc. The rich additional information can effectively supplement the user–app interaction matrix, so as to improve the performance of app recommendation.

2.2 Deep learning-based recommendation method

In recent years, deep learning technique has achieved great success in image recognition, natural language processing, speech recognition and other fields because of its excellent nonlinear expression ability, which can automatically learn the potential relationships in features. Deep learning technique emphasizes learning from massive data, which solves the problem that traditional machine learning algorithms are difficult to deal with high-dimensional, heterogeneous and noisy data.

At the same time, researchers also explore the application of deep learning technique in recommender system. For example, due to deep learning technique has powerful ability to mine the potential interaction features, Cheng et al. (2016), Guo et al. (2017), Shan et al. (2016) based on the idea of features interaction to explore the combination of different features between users and items, in order to mine users' preferences. The difference of these methods is that Cheng et al. (2016) explore the influence of the depth and width combination between users and apps on target users' choice. Guo et al. (2017) use the factorization machine to mine the low-level interaction information between users and app features, and use DNN to mine the deep interaction information between features, so as to achieve the purpose of users' preference mining. Unlike the above two methods, Shan et al. (2016) directly embed the features of users and apps, MLP network to reduce a lot of artificial feature engineering.

Harada et al. (2019) propose CNCF to recommend game apps for users, which leverages contextual information to enhance the recommendation performance. Kim et al. (2016) leverage CNN model to extract users' preferences from their comments on items. Xu et al. (2019) incorporate contextual information into deep learning model to explore users' preferences. Bobadilla et al. (2020) proposed a deep neural architecture based on classification. Notably, its collaborative filtering method can be generalized to most of the existing recommender systems. Liang et al. (2020) explore the features of user–app interactions which models the interactions of features from different views through the attention mechanism.

3 Preliminaries

In this section, we give the general definition of the problem to be studied in this paper at first, and then give the motivation of CFDIL.

3.1 Problem definition

The goal of app recommendation is to recommend apps for target users, which meet their preferences under a specific contextual condition. Without loss of generality, for a given user set U and an app set A, our task is to make an app recommendation list R_u for u, $u \in U$, where $R_u = \{a|u, a \in A, u \in U\}$. The a in the recommendation list R_u conforms the preferences of user u.

The recommendation is based on the probability of the target user's preference for an app under a specific context. Therefore, to recommend apps for users, we first need to calculate the probability of the preferences of all apps in A for the target user u under specific context conditions, and then select the top-k apps with the highest preference probability for recommendation. In this process, the prediction of the target users' preferences probability for a given app is particularly critical, which directly determines the accuracy of the recommendation.

Next, we give the motivation of the proposed CFDIL.

3.2 Motivation

Most of the existing app recommendation methods use user–app interaction vectors to explore user's preferences, and then make recommendations. By combing real datasets, we found that user–app interactions show highly aggregated characteristics in spatiotemporal dimensions. However, the existing recommendation methods often ignore this point, so we hope to integrate the spatiotemporal information into feature matrix to improve the accuracy of recommendation. In this paper, we refer to the combination of time and location information as contextual information.

We visualize the spatial characteristics of user–app interactions at a given time slot in the form of a heat map. We leverage a scatter chart to show the temporal characteristics of user–app interactions. Figure 1 (a) shows the heat map of the number of times a user used apps in a certain time slot and (b) shows the heat map of the number of times an app was used in a certain time slot. Figure 2 (a) shows the number of times a user used apps in different time periods and (b) shows the number of times an app was used in different time slots. It is easy to find that the temporal and spatial interaction between users and apps are always aggregative, which indicates that contextual information has strong relevance to users and apps.

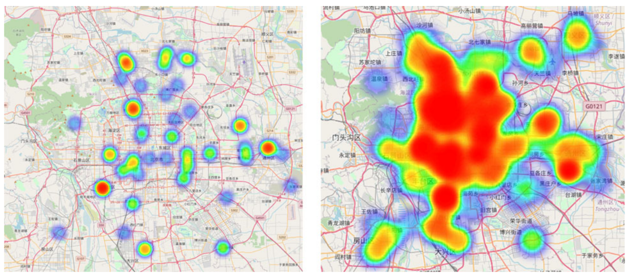


Fig. 1 Left: **a** A user's historical location when using apps, from 13:00 p.m. to 15:00 p.m. Right: **b** the historical usage location of an app, from 8:00 a.m. to 10:00 a.m

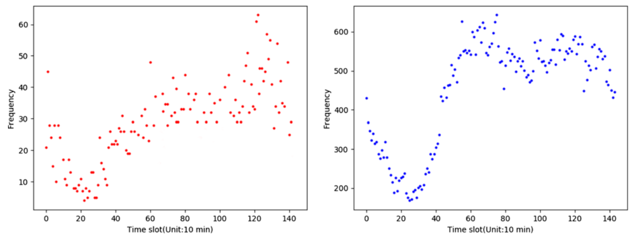


Fig. 2 Left: **a** The number of times a user used apps in different time slots. Right: **b** the number of times an app was used in different time slots

We present several examples to explain context information. For a user, his/her daily life is regular in the long term, no matter he/she is a worker, retiree or teenager. Such regularity can be reflected by context of users using apps. For example, a worker often signs in an app at the workplace and uses food-ordering apps at lunchtime; a retiree often reads news using an app after breakfast. For an app, it always appears in a context accord with its function, i.e., apps also have their regularities. For example, food-ordering apps are often used at lunchtime, and Metro apps are often used in subway stations. Therefore, we construct their feature portraits for users and apps to show their temporal and spatial regularities. We believe that introducing context information to our model will be helpful to improve the accuracy of preference prediction.

The existing app recommendation methods are based on user–app interaction vector for preference modeling. It should be noted that the user–app interaction information in the real world is extremely sparse, and the recommendation model is difficult to effectively explore user preferences by using sparse data. Inspired by the above discussion, the incorporation of contextual information can enrich and supplement the feature matrices of users and apps to alleviate the problem of data sparseness. Based on the aforementioned consideration, we proposed a context-aware feature deep interaction learning (CFDIL) method to improve the app recommendation performance.

Table 1 The descriptions of symbols used in this paper

Symbols	Descriptions
U	The set of users
A	The set of mobile applications
T	The set of time slots
u	A target user, $u \in U$
a	An target app, $a \in A$
t	A time slots, $t \in T$
P_t^u	The feature portrait of u in t
P_t^a	The feature portrait of a in t
$\text{Lab}_{u,a,t}$	Number of times user u interacts with app a in time slot t
F_t^u	The fields set of P_t^u
F_t^a	The fields set of P_t^a
R_u	The app recommendation list for user u

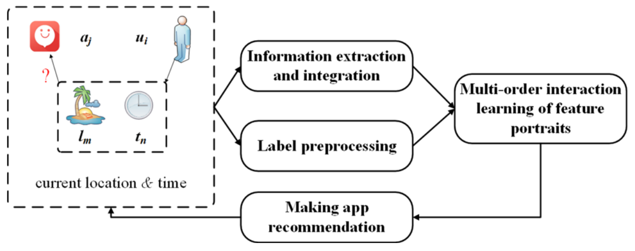

a_j
u_i
?
l_m
t_n
current location & time
Information extraction and integration
Label preprocessing
Multi-order interaction learning of feature portraits
Making app recommendation

Fig. 3 The overall framework of CFDIL. CFDIL consists of four main parts: information extraction and integration, label processing, multi-order interaction learning of feature portraits, making app recommendation

3.3 Symbols

To facilitate the clear presentation of this paper, we give the notations and descriptions of the symbols used in CFDIL, as shown in Table 1.

4 Proposed work

4.1 Framework

In this section, we first introduce the framework of CFDIL and then introduce each module of CFDIL in detail.

The overall flow of CFDIL is shown in Fig. 3, which consists of four main parts.

1. Information extraction and integration. In this part, CFDIL first extracts and constructs the feature portraits of users and apps, which both contain two parts: (1) the own attribute information of users and apps and (2) the contextual information of users and apps (spatial characteristic in a certain time slot).
2. Label processing. CFDIL constructs a three-dimensional tensor with user, app and time as coordinates. The elements in the tensor are the number of interactions between users and apps in a certain time slot. To further explore users' preference for untouched apps and to solve the problem of label data sparsity, this section performs tensor factorization on labels.
3. Multi-order interaction learning of feature portraits. CFDIL constructs new networks by Factorization machine (FM) and convolutional neural network (CNN) to learn multi-order and deep potential interaction features of users and apps. Multi-order learning is performed on user and app feature portraits to effectively explore users' preferences.
4. Making app recommendation. Based on the trained model, CFDIL can accurately predict target users' preferences in specific contexts and complete app recommendations.

Next, we will describe the key technologies used in each stage in detail.

4.2 Information extraction and integration

This part is to construct users and apps feature portraits, and mainly contains two sub-steps, information extraction and information integration. Figure 4 shows the overall construction process of app portraits and user portraits. Next, we describe these two steps in detail, respectively.

4.2.1 Information extraction

We mainly extract two kinds of information from users and apps, attribute information and contextual information. The detailed extraction method is as follows.

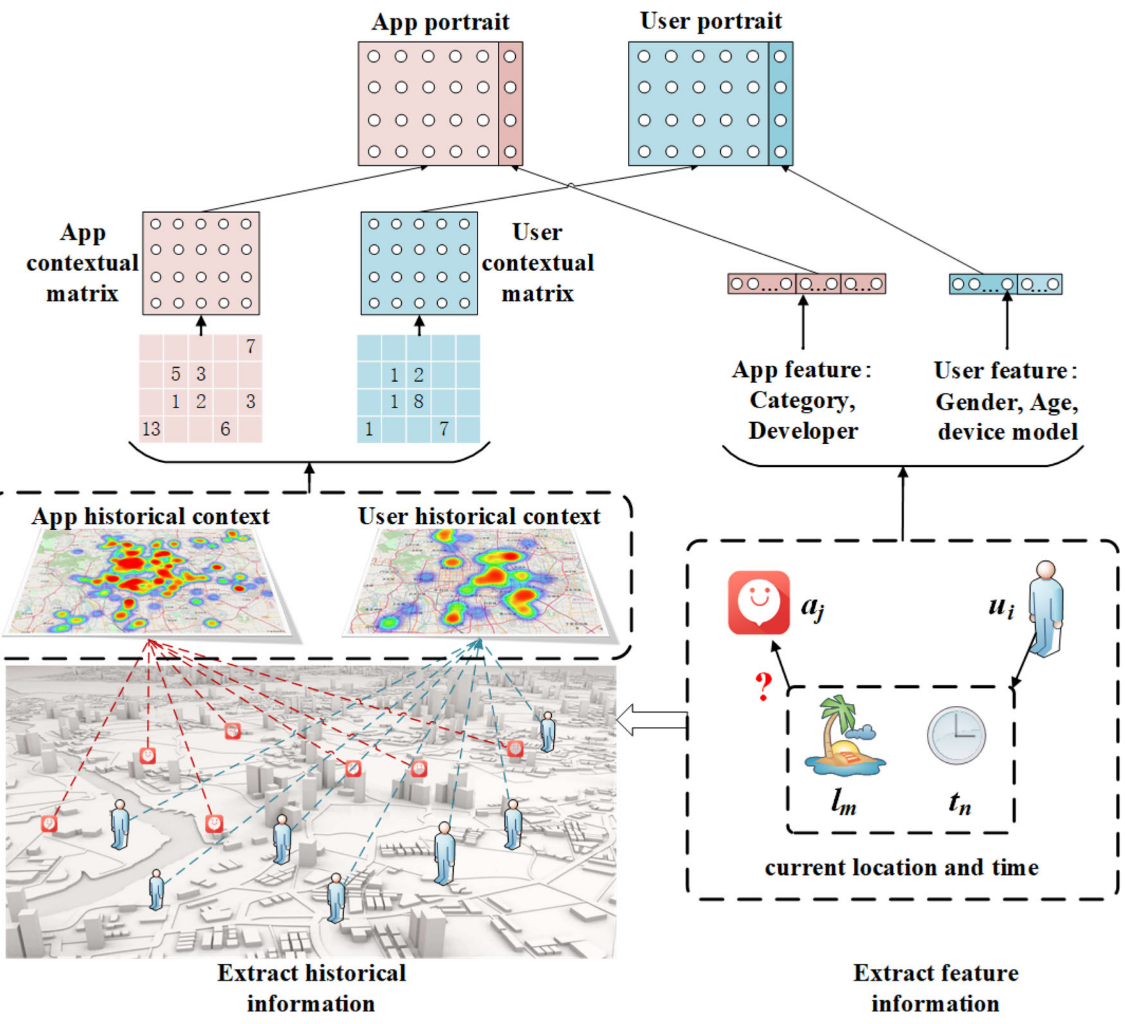

App portrait
User portrait
App contextual matrix
User contextual matrix
App feature: Category, Developer
User feature: Gender, Age, device model
App historical context
User historical context
a_j
u_i
?
l_m
t_n
current location and time
Extract historical information
Extract feature information

Fig. 4 Details about information extraction and integration

1. According to Chen et al. (2015), the attribute information of users and apps has great impact on users' preferences. Therefore, we select users' attribute information, such as gender, age and device model, and apps' attribute information, such as category, and developer, a total of five kinds of attribute information.

   In order to facilitate the training of the subsequent model, we represent the attribute information of users and apps with the same dimension and length vector. In this paper, we map the attribute information of users (Table 3) or apps (Table 4) into an attribute vector VA, $VA \in \mathbb{R}^{200\times 1}$, respectively.
2. We also extract the contextual information of users and apps, in addition to their own attribute information.

   The contextual information mainly refers to time and location information, which can be found in Table 5. We use the following steps to extract contextual information of users and apps.

   – We first split the day into 7 time periods $T = \{00:00-06:00,\ 06:00-09:00,\ 09:00-11:00,\ 11:00-13:00,\ 13:00-17:00,\ 17:00-19:00,\ 19:00-24:00\}$. We split the geographic region which contains of all interaction between users and apps into a matrix VC, $VC \in \mathbb{R}^{200\times 199}$ according to longitude and latitude, i.e., we map this geographic region into a matrix which contains 200×199 geographic cells.
   – For each app and user in a certain time slot, we construct app contextual matrix and user contextual matrix, respectively. For an app, we count the historical number of times the app has been used by all users, and fill the value into the corresponding cell in VC. For a user, we use the same method to construct the user contextual matrix. The difference is that the

elements in user contextual matrix are the historical number of times the user used apps.

4.2.2 Information integration

For each app and user in each time slot, we integrate the attribute information and contextual information of users and apps, respectively, to construct the portrait of apps and users. For an app a, we connect the app's contextual matrix VC with its attribute vector VA to form an app feature portrait P_t^a, $P_t^a \in \mathbb{R}^{200\times200}$, $t \in T$, $a \in A$, as shown in Fig. 4. Similarly, for a user u, we use user's contextual matrix VC and user's attribute vector VA of u to construct the user's feature portrait P_t^u, $P_t^u \in \mathbb{R}^{200\times200}$, $t \in T$, $u \in U$.

4.3 Label preprocessing

The data used to train app recommendation model are $\{P_t^u, P_t^a, \text{Lab}_{u,a,t}\}$, $t \in T$, $u \in U$, $a \in A$, where P_t^u is the feature portrait of user u in t, P_t^a is the feature portrait of app a in time slot t, and $\text{Lab}_{u,a,t}$ is the historical interaction number between user u and app a in t. Generally speaking, $\text{Lab}_{u,a,t}$ is used as a label for model training. However, there are two problems for $\text{Lab}_{u,a,t}$ applied to train app recommendation model:

1. $\text{Lab}_{u,a,t}$ only represents the selection result of user u in a given context, and cannot equivalently represent user's preference. For example, $\text{Lab}_{u,a,t} = 0$ means that u did not use a during t, but this does not mean that u does not like a. Because user u may not be aware of the existence of app a.
2. For a large number of apps, people will only use a few apps. The vast majority of $\text{Lab}_{u,a,t}$ is 0, i.e., the label data are extremely sparse. The sparse label data make the recommendation model unable to fully perceive the positive feedback label data, which leads to the weak generalization ability of the model recommendation.

To solve the above two problems, we leverage tensor factorization technique to process $\text{Lab}_{u,a,t}$. The detailed method is as follows:

1. We construct a tensor LA with user, app and time slot as coordinates. The elements in LA is $\text{Lab}_{u,a,t}$, which is the historical number of times user u interacts with app a in time slot t.
2. We use the same method as Zhu et al. (2021) to decompose LA. The principle of the decomposition process is shown in Fig. 5. After decomposition, we get a new tensor LA^* with the same size as the original tensor LA. The difference is that the data in LA^* is non-sparse.
3. We use the elements of tensor LA^* to update the value of $\text{Lab}_{u,a,t}$, and then use the updated value of $\text{Lab}_{u,a,t}$ to represent u's preference label for a in t.

4.4 Multi-order interaction learning of feature portraits

CFDIL constructs user's portraits P_t^u and app's portraits P_t^a, and processes the sparse label data $\text{Lab}_{u,a,t}$. In this part, we will describe how CFDIL uses P_t^u, P_t^a and $\text{Lab}_{u,a,t}$ for multi-order interactive learning in detail. CFDIL mainly contained two parts, FM part and CNN part. The overall framework of CFDIL is shown in Fig. 6. Next, we will describe how CFDIL performs multi-order interactive learning in detail.

4.4.1 FM model

CFDIL uses FM to learn low-order features interaction between users and apps, as shown in the FM part in Fig. 6.

The input data of FM are a two-tuple which formally expressed as $\{F_t^u \cup F_t^a, \text{Lab}_{u,a,t}\}$, where F_t^u and F_t^a are the field sets of P_t^u and P_t^a. F_t^u=$\{f_{t,n}^u, n \in N\}$, N is the number of attribute of P_t^u. F_t^a=$\{f_{t,m}^a, m \in M\}$, M is the number of attribute of P_t^a. In this paper, N is 4, F_t^u is {gender, age, device model, user contextual matrix}; M is 3, F_t^a is {category, developer, app contextual matrix}. $\text{Lab}_{u,a,t}$ is the label of $F_t^u \cup F_t^a$, which is obtained in Sect. 4.3.

We use an objective function to learn the low-order interaction features of users and apps, which is expressed as Eq. 1:

$$\widehat{y}_f = \omega_0 + \sum_{i=1}^{M+N} \omega_i x_i + \sum_{i=1}^{M+N} \sum_{j=i+1}^{M+N} < V_i, V_j > x_i x_j \qquad (1)$$

where x_i and x_j are the fields of apps and users portraits, $x_i, x_j \in \{F_t^u \cup F_t^a\}$; $\omega_0, \omega_i, V_i, V_j$ are model parameters, ω_0 is the global bias, ω_i is the weight of the ith variable, V_i and V_j are the implicit matrix of parameters. $\omega_0 + \sum_{i=1}^{M+N} \omega_i x_i$ describes the first-order features of the sample in the form

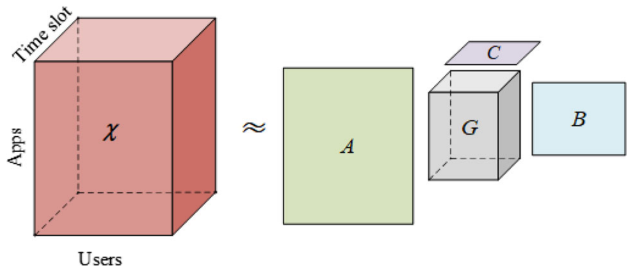


Fig. 5 Tensor decomposition of LA

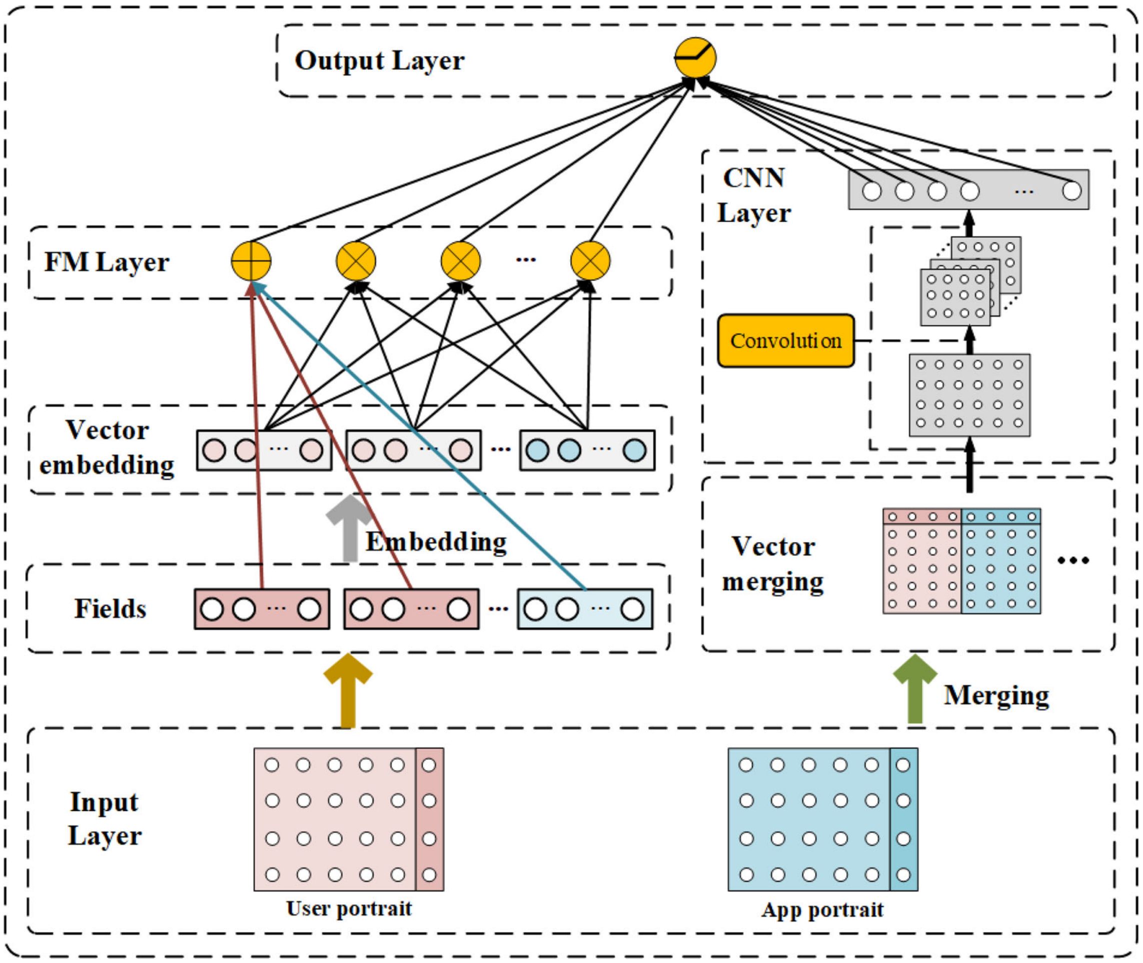


Fig. 6 Details about multi-order interaction learning of feature portraits. CFDIL uses FM to learn low-order interaction features between users and apps, and uses CNN to learn the higher-order interaction features between users and apps

of linear regression, and $\sum_{i=1}^{M+N}\sum_{j=i+1}^{M+N} < V_i, V_j > x_i x_j$ describes its second-order combinatorial features, which are used to learn the interaction features between users and apps in different contexts. Since the FM algorithm uses all univariate and two-by-two feature interactions, it can effectively learn the low-order features interaction between users and apps. In Eq. 1, $< \cdot, \cdot >$ denotes the dot product of two vectors of dimension k, which is expressed as Eq. 2

$$< V_i, V_j > : = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f} \tag{2}$$

$< V_i, V_j >$ is the cross-weights of features x_i and x_j. $v_{i,f}$ and $v_{j,f}$ are the elements in V_i and V_j, respectively. k is the dimension of the implicit parameter. In order to reduce the time complexity of the model, the quadratic term of FM can be simplified according to Eq. 3.

$$\begin{aligned}
&\sum_{i=1}^{M+N}\sum_{j=i+1}^{M+N} < V_i, V_j > x_i x_j \\
&= \frac{1}{2}\sum_{i=1}^{M+N}\sum_{j=1}^{M+N} < V_i, V_j > x_i x_j - \frac{1}{2}\sum_{i}^{M+N} < V_i, V_j > x_i x_j \\
&= \frac{1}{2}\left(\sum_{i=1}^{M+N}\sum_{j=1}^{M+N}\sum_{f=1}^{k} v_{i,f} \cdot v_{j,f} x_i x_j - \sum_{i=1}^{M+N}\sum_{f=1}^{k} v_{i,f} \cdot v_{i,f} x_i x_i\right) \\
&= \frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{M+N} v_{i,f} x_i\right)\left(\sum_{j=1}^{M+N} v_{j,f} x_j\right) - \sum_{i=1}^{M+N} v_{i,f}^2 v_i^2\right) \\
&= \frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{M+N} v_{i,f} x_i\right)^2 - \sum_{i=1}^{M+N} v_{i,f}^2 v_i^2\right)
\end{aligned} \tag{3}$$

In Eq. 3, $\sum_{i=1}^{M+N} v_{i,f}^2 v_i^2$ is a constant value, so we only need the nonzero terms of x_i to train the objective function of Eq. 3. Therefore, the time complexity of the objective function of FM is $O(k(M+N))$, and FM model can quickly extract the low-order interactive features.

We transform recommendation task into exploring the probability of a target user's preference for an app, i.e., into a regression problem, so we select the loss function $\text{loss}(\widehat{y}_f, y_f) = (\widehat{y}_f - y_f)^2$.

In addition, we use stochastic gradient descent to train the parameters, and the gradient formula of the parameters is shown in Eq. 4.

$$\frac{\partial \widehat{y}_f(x)}{\partial \theta} = \begin{cases} 1, & \text{if } \theta \text{ } is \text{ } \omega_0 \\ x_i, & \text{if } \theta \text{ } is \text{ } \omega_i \\ x_i \sum_{j=1}^{n} v_{j,f} x_j - v_{i,f} x_i^2, & \text{if } \theta \text{ } is \text{ } v_{i,f} \end{cases} \tag{4}$$

In summary, we choose FM to explore the low-order interactive features between users and apps, because FM has the following advantages. (1) The complexity of FM model is linear, which can effectively train sparse user–app interaction data. (2) The unique structure of FM makes the interactive learning of low-order features between users and apps more reasonable.

4.4.2 CNN model

CNN model is mainly used to learn the higher-order interaction features of user portraits P_t^u and app portraits P_t^u, as shown in the CNN part in Fig. 6. We choose CNN to extract the high-order interactive feature between users and apps, because CNN has the following three advantages. (1) The data of P_t^u and P_t^a are in matrix format, which is the same as the single-channel image data. CNN networks have a great advantage in processing image format data. (2) CNN can effectively explore deep features in image format data by using convolutional operations. (3) CNN uses a shared convolution kernel mechanism to reduce the complexity of model training. As a result, CNN can efficiently process the high-dimensional users and apps portrait data.

The input data of CNN are a two-tuple which formally expressed as $\{P_t^u \parallel P_t^a, \text{Lab}_{u,a,t}\}$, where $P_t^u \parallel P_t^a$ indicates that P_t^u and P_t^a in the same t-time slot are stitched up and down. $\text{Lab}_{u,a,t}$ is the label of $P_t^u \parallel P_t^a$, and the value of $\text{Lab}_{u,a,t}$ is equal to that described in Sect. 4.3.

The input layer of CNN is the start of the interactive feature learning of P_t^u and P_t^a. The weight is learned through the hidden layer, and the nonlinear segmentation ability of the network is enhanced with the help of the excitation function. By learning parameters and information transfer layer by layer, CNN can effectively learn the high-order interactive features of P_t^u and P_t^a. The main structure of CNN used for our experiments is:

Input layer − > convolutional layer − > activation layer − > ⋯ − > pooling layer − > ⋯ − > fully connected layer.

It should be noted that we performed an average pooling operation near the middle convolutional layers, which reduced features by half. The learning process of CNN is shown in Eq. 5.

$$x^{(l+1)} = \text{ReLu}\,(W^{(l)} x^{(l)} + b^{(l)}) \tag{5}$$

where l represents the lth layer of the neural network. $x^{(l)}$ is the output of lth layer. $W^{(l)}$ and $b^{(l)}$ are the model parameters and deviations for lth layer. $x^{(l+1)}$ is the output of lth layer, and also used as the input of $l+1$th layer. In order to avoid the inefficiency of error back-propagation and to avoid the problem of gradient explosion, ReLu is used as the activation function in our model.

Since we consider user–app recommendation problem as a regression problem, we adopt the mean square error (MSE) as the loss function in the convolution neural network, expressed by Eq. 6.

$$\text{MSE} = \frac{1}{M} \sum_{i=1}^{M} \left(y_c^{(i)}, \widehat{y}_c^{(i)} \right)^2 \tag{6}$$

4.5 App recommendation

CFDIL uses FM to explore the low-order interaction features of users and apps from the feature fields of $\{F_t^u \cup F_t^a, \text{Lab}_{u,a,t}\}$, and uses CNN to explore the high-order interaction features of users and apps from $\{P_t^u \parallel P_t^a, \text{Lab}_{u,a,t}\}$. We weighted and summed the learning results of FM and CNN according to Eq. 7 as the final probability result of CFDIL.

$$\widehat{y}_i = \epsilon_0 \widehat{y}_f + \epsilon_1 \widehat{y}_c \tag{7}$$

where ϵ_0 and ϵ_1 are model parameters, satisfying $\epsilon_0 + \epsilon_1 = 1$. In the subsequent experiments in this paper, setting the values of ϵ_0 and ϵ_1 to 0.65 and 0.35, respectively, will get the best recommendation performance.

After the CFDIL model training is completed, we follow the steps below to recommend apps to users in specific contexts. (1) We construct a feature profile P_t^u for a target user u in time slot t. (2) We construct feature portraits P_t^a of all candidate apps in time slot t. (3) We input P_t^u and P_t^a into the trained CFDIL network and then get the recommendation probability of each app for user u. (4) We sort all candidate apps according to the recommended probability. The top N apps are selected based on the ranking to form the final recommendation list.

5 Experiments

In this section, we deploy CFDIL on a real-world dataset to verify the recommendation performance of CFDIL. First, we give the default experimental settings and then evaluate the performance of CFDIL from different perspectives.

5.1 Experiments settings

In this section, we first introduce the dataset of our experiment, and then give the evaluation metrics. Finally, we introduce the baseline methods for comparison.

5.1.1 Dataset

The experimental data are log files of users using apps in real-world scenarios in three cities, Beijing, Shanghai and Guangzhou. The datasets contain a total of 8198 users, 2671 apps, and 405,837 user–app usage log records. Each app is used at least 10 times, and each user has at least 10 app usage logs. Each user has an average of 49.5 logs. The detailed information of the dataset is shown in Table 2. For example, a user (User_id:2007, Gender: Male, Age: 23, Device: XiaoMi) uses a game app (App_id:147, Category: Game, Developer: Tencent) in a specific context (Time_stamp: 2018-07-15 20: 14, longitude: 113.287, latitude: 23.139). All this information is contained in our dataset, as shown in Tables 3, 4 and 5.

In our experiments, we split the experimental data into training set, validation set and test set. The ratio of these three parts is 7:2:1. The training set is used to train CFDIL. The validation set is used to adjust the hyper-parameters of the model, including the number of layers in the CFDIL's neural network, the size and number of convolution kernels. The test set is used to verify the recommendation performance and generalization ability of CFDIL. During the test, we take apps used by users without using recommender systems as standard, to evaluate the recommendation performance.

Table 2 The data structure of the dataset

Dataset	Beijing	Shanghai	Guangzhou
User	2572	2721	2905
App	852	891	928
Record	99,815	138,568	167,454
Duration	168 Hours	168 Hours	168 Hours

Table 3 The user attribute information

User_id	Gender	Age	Device
2007	Male	23	XiaoMi
2122	Female	29	OPPO
2345	Female	31	HuaWei
1098	Male	34	Apple

Table 4 The app attribute information

App_id	Category	Developer
147	Game	Tencent
302	News	Sina
457	Social	Tencent
722	Shopping	Alibaba

Table 5 The interaction between users and apps in different contexts

User_id	App_id	Time stamp	Longitude	Latitude
2007	147	2018-07-15 20:14	113.287	23.139
2147	722	2018-07-16 10:07	113.289	23.136
2141	318	2018-07-11 11:54	113.257	23.155
1922	097	2018-07-18 14:21	113.286	23.1299

5.1.2 Metric

We recommend an app list for target users and evaluate it with two evaluation metrics: precision and recall. We recommend a top-N recommendation list to users, and the length of the recommendation list is N, so we choose Precisi-on@N and Recall@N to measure the proposed method. However, precision and recall are two related metrics, and when one goes down, it causes the other to go up. In order to consider the two metrics synthetically, we use the $F_\alpha -$ measure-@N to measure the recommendation quality.

Precision@N and Recall@N are defined as follows:

$$\text{Precision}@N = \frac{TP}{N} \tag{8}$$

$$\text{Recall}@N = \frac{TP}{M} \tag{9}$$

where TP is the intersection between the recommendation list and the ground truth, N is the length of the recommendation list, and M is the length of the ground truth.

F_α−measure@N is defined as follows:

$$F_\alpha\text{−measure}@N = (1+\alpha^2)\frac{\text{Precision} \times \text{Recall}}{\alpha^2\text{Precision} + \text{Recall}} \tag{10}$$

where precision and recall are the results in Eqs. 8 and 9, respectively; α is used to balance recall and precision. Here, we choose 1 as the value of α, which means that recall and precision are equally important.

To get the best hyper-parameters of CFDIL, we use mean absolute error (MAE) and root mean squared error (RMSE) to adjust the model parameters. MAE and RMSE are two indicators that are widely used to measure the accuracy of recommender system. The detailed definitions of these two indicators are as follows:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{T} (\widehat{y}_i - y_i)^2}{T}} \tag{11}$$

$$\mathrm{MAE} = \frac{\sum_{i=1}^{T} |\widehat{y}_i - y_i|}{T} \tag{12}$$

where T in RMSE and MAE is the number of records in the validation set. $\widehat{y}_i$ is the ith prediction value of the model, and y_i is the corresponding true value of the ith position.

5.1.3 Benchmark methods

We use the following methods to measure the performance of CFDIL.

UCF (User-based Collaborative Filtering Method). UCF is a collaborative filtering method oriented to user vector similarity which is calculated based on users' rating data. We adapt UCF to our problem by the usage frequency of an app to explore users' preferences and generate the recommendation list. We leverage frequency information to find users who are similar to the target user and recommend apps that similar users have used but the target user has not used, to target users.

ICF (Item-based Collaborative Filtering method). ICF is a collaborative filtering method oriented to app vector similarity. We adapt ICF to our problem by using the following methods. First, we carry out matrixing process on the dataset according to the frequency of an app to form a user–app usage matrix. Then, we extract the app vector from the matrix and use the cosine similarity coefficient to build the app similarity model. Finally, we leverage the app similarity model to generate an app recommendation list for target users.

MF (Matrix Factorization). MF recommendation method is a classic model-based method in the recommendation domain. We input the user–app usage matrix by matrix factorization to obtain a non-empty matrix that contains the same information as the original matrix.

TF (Tensor Factorization). The aim of the model is to compute the factors for the user $U^{n\times d}$, item $A^{r\times d}$ and context $C^{l\times d}$ matrices using historical usage data. TF method is also an MF-based recommendation method.

Similar to the MF method, we use the frequency of an app according to certain contextual information to denote the app usage information of a target user and construct a user–app-context tensor. Then, we use TF method to obtain a recommendation list. Here, the elements in the tensor are the app usage information at a certain time.

DNN (Deep Neural Networks). DNN model is a classic deep learning method. DNN model has strong nonlinear expression ability, which can learn the complex potential interactions between users and apps. We apply DNN to app recommendation. First, the portrait information of users and apps are used as the input of DNN. Then the interaction information of a target user and a target app is used as the label of model training. Finally, a DNN-based app recommendation model is trained.

CNN (Convolutional Neural Networks). CNN is also a classic deep learning method. We apply CNN model to app recommendation. In order to train the app recommendation model based on CNN, we take the portraits of users and apps as the input of CNN model and take the interaction information of users and apps in a specific contextual information as the output of CNN model.

DeepFM. DeepFM model is a novel deep learning-based recommendation model. The idea of this recommendation model is feature cross recommendation, i.e., leveraging the combination of different features of users and items to predict the recommendation probability. It combined by two deep models. One part is FM (factorization machine), which mainly learns the low-order features interaction between users and items. Another is deep model, which learns the high-order features interaction between users and items. We apply DeepFM to app recommendation by inputting the feature information of users and apps, and train a DeepFM model by using the interaction information as output label.

5.2 Performance comparison

Figure 7 shows the precision, recall and F-measure values of recommendation results of CFDIL and benchmark methods in 3 different city datasets and in different recommendation list lengths. We can get the following conclusions from the results.

1. The recommendation performance of ICF and UCF is unacceptable. This is because the interaction data between users and apps is sparse, which hinders the effective construction of users and apps vectors by these two methods. Sparse vectors of users and apps lead to poor performance. Besides, these two methods do not consider the impact of contextual information on users' choice of apps.
2. The recommendation performance of MF and TF is better than UCF and ICF, but their recommendation results are still poor. This is because MF and TF have relative advantages in combating sparsity, which can help the model deal with sparse user and app vectors more effectively. However, since these two models are naturally not personalized, their recommendation lack generalization capabilities. In addition, the result of TF is better than that of MF because TF considers context information.
3. Deep learning-based models show good recommendation performance. Among them, CNN has better recommendation performance than DNN. Both CNN and DNN use deep models to extract the deep latent interac-

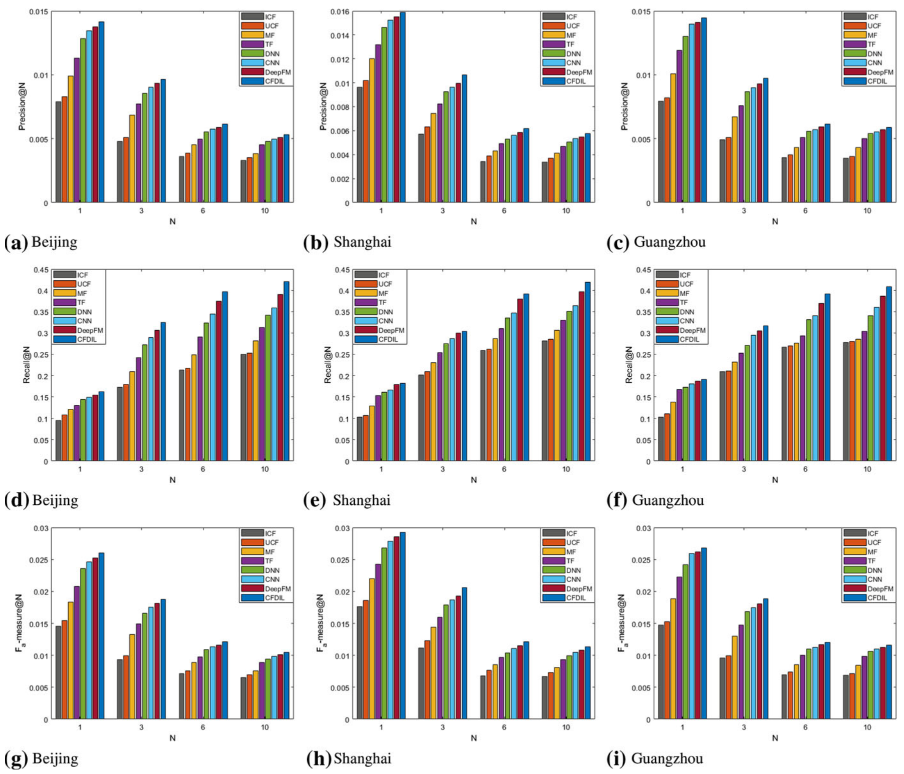


Fig. 7 The comparisons of recommendation performance between CFDIL and benchmark methods. There are seven benchmark methods: ICF, UCF, MF, TF, DNN, CNN, and DeepFM

tions between users and apps. DNN leverages fully connected method to learn deep interaction information between users and apps, while CNN uses convolution kernels to extract deep interaction information between users and apps. DNN's fully connected structure makes a lot of unnecessary information in user and app matrix to be added to the interaction process. Excessive information doping interferes with the ability of DNN to mine users' preferences. CNN uses convolution kernels to effectively eliminate interference information, so that the model can mine users' preferences more efficiently. The performance of DeepFM is better than DNN and CNN. In addition to using the deep part to learn high-order feature interactions, DeepFM also uses FM part to learn low-order interaction information between users and apps. Thus, the disadvantages of the above two methods are eliminated by DeepFM.

4. The recommendation performance of CFDIL is better than all benchmark methods. The reasons are as follows: (1) CFDIL considers contextual information in the matrix construction process, which helps the model to judge users' preference. (2) The FM part in CFDIL fully expresses the low-order interaction features between users and apps. (3) The CNN part of CFDIL effectively explores interaction features of between users and apps under contextual conditions.

5.3 The model hyper-parameters

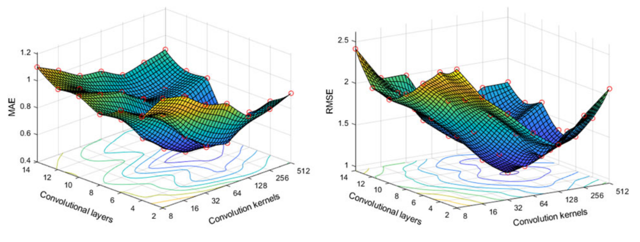
In this section, we mainly show some exploration of CNN parameters in CFDIL. We leverage datasets from three cities, Beijing, Shanghai and Guangzhou, to determine the number of layers of CFDIL and the number of convolution kernels in each layer.

Inspired by He et al. (2016), the size of CFDIL convolution kernels used in convolution layers is 3×3. The number of convolution kernels is set according to the following principle: when the portrait size is halved, the number of convolution kernels should be doubled to ensure the complexity of learning.

We apply CFDIL on three cities dataset and set the convolutional layers of CFDIL to {2, 4, 6, 8, 10, 12, 14} and set the initial convolution kernels to {8, 16, 32, 64, 128, 256, 512}. We use MAE and RSME to evaluate the model performance to determine the hyper-parameters of CFDIL. The hyper-parameters that minimize MAE and RSME are the best.

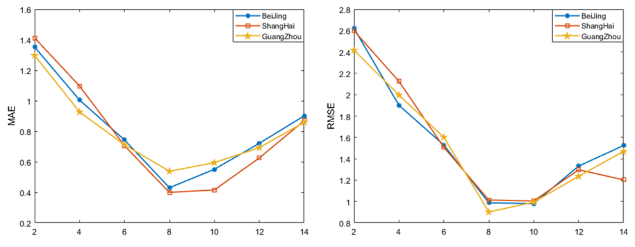
We use the dataset of Beijing City as the representative to show the experimental results, as shown in Fig. 8. As can be seen from Fig. 8, CFDIL performs best when the number of convolutional layers is 8 and the initial number of convolution kernels is 128.

In addition, we use Fig. 9 to show the performance of CFDIL with different number of convolutional layers when



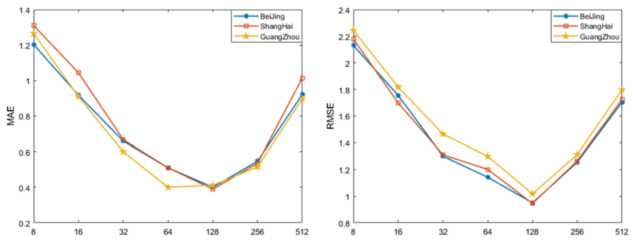
(a) MAE-Beijing **(b)** RMSE-Beijing

Fig. 8 CFDIL's 3D fitting diagram on the number of convolution kernels and convolutional layers. There are 49 real experimental results in the red circle, and the others are fitting results



(a) MAE **(b)** RMSE

Fig. 9 CFDIL with different number of convolutional layers



(a) MAE **(b)** RMSE

Fig. 10 CFDIL with different initial number of convolution kernels

the initial number of convolution kernels is 128. We use Fig. 10 to show the performance of CFDIL with different initial number of convolution kernels when the number of convolutional layers is 8.

5.4 Ablation experiment

5.4.1 Impact of contextual information

We use the feature matrices of users and apps as the input of CFDIL and train a new model named CFDIL-I. The difference between CFDIL-I and CFDIL is that CFDIL-I does not consider the contextual matrices of users and apps. Then we compare the Precision@N of CFDIL and CFDIL-I to judge the validity of the proposed contextual matrices of users and apps.

Figure 11 shows the Precision@N results between CFDIL and CFDIL-I. It can be seen from the figure that the Precision@N of CFDIL is significantly better than these of CFDIL-I under all recommendation lists. This is because CFDIL constructs user and app portraits that fully consider contextual information, which can explore users' preferences more accurately.

5.4.2 Impact of CNN

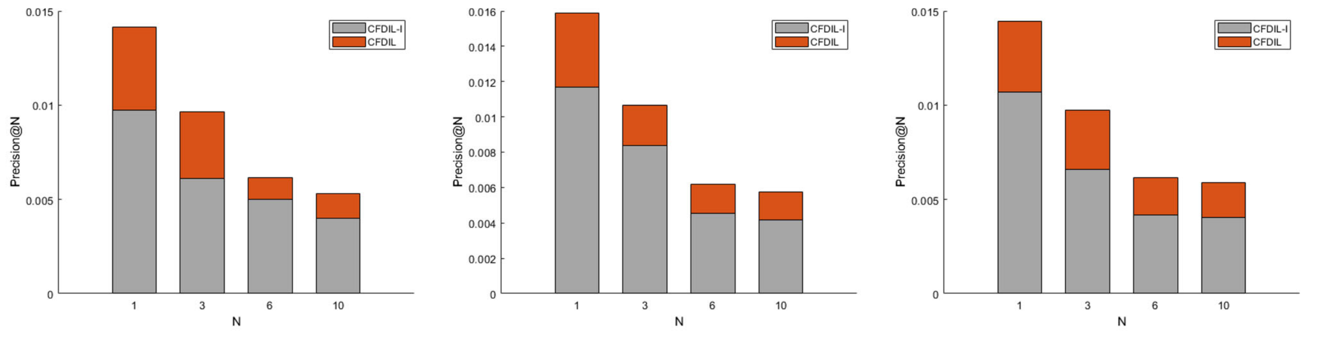
First, we use portraits of users and apps as input and eliminate the CNN part of CFDIL to get a new model, named CFDIL-C. Then we compare the Precision@N of CFDIL and CFDIL-C on three cities datasets to judge the effectiveness of the CNN part of CFDIL.

Figure 12 shows the experimental results. It can be seen from the experimental results, the Precision@N of CFDIL are better than these of CFDIL-C. The main reason is as follows. CNN has a strong ability to extract features from two-dimensional data, which has been verified in the field of image processing. The user and app portraits we construct are mainly composed of two-dimensional geographic information generated by interaction between users and apps in a specific time slot. This data structure is consistent with the image data structure. CNN can effectively extract interaction features from the two-dimensional data by using convolution kernel mechanism. The experimental results show that CNN can effectively extract the portrait features of users and apps, and efficiently mine users' preferences under specific contextual conditions.

5.4.3 Impact of FM

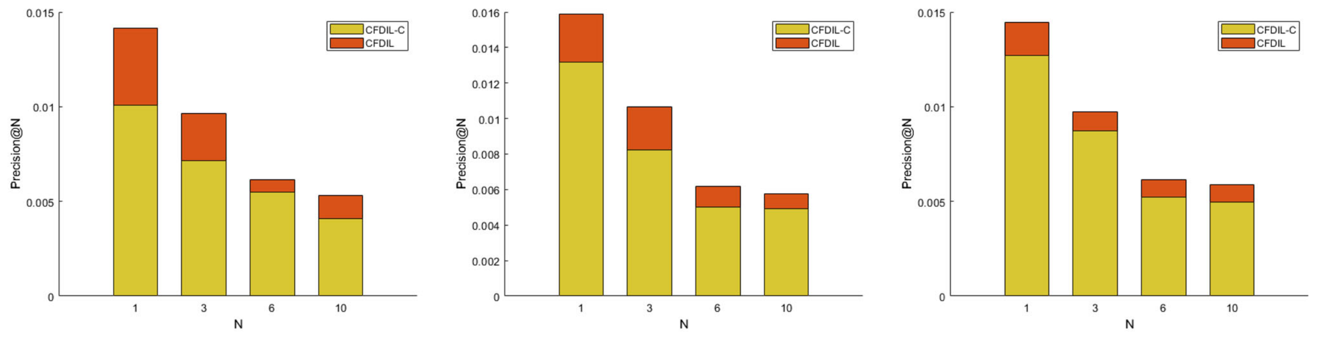
First, we use portraits of users and apps as input and eliminate the FM part in CFDIL to get a new model, named CFDIL-F. Then, we compare the Precision@N of CFDIL and CFDIL-F to judge the effectiveness of the FM part of CFDIL.

The experimental results show that CFDIL is more effective than CFDIL-F, which indicates that the FM part of CFDIL also plays an important role in mining users' preferences. The network structure of FM is similar to the FM part of DeepFM model that has been successful in CTR field. The difference between the two FMs is the type of problem to be solved. The application field of DeepFM is click-through rate prediction, which is a classification problem. The FM part we designed is mainly to help CFDIL predict users' preference probability, which is a regression problem. The role of FM part is to help the model obtain the low-order feature cross information of users and apps effectively, so that the model can comprehensively consider the low-order cross features and high-order potential interaction features of users and apps.



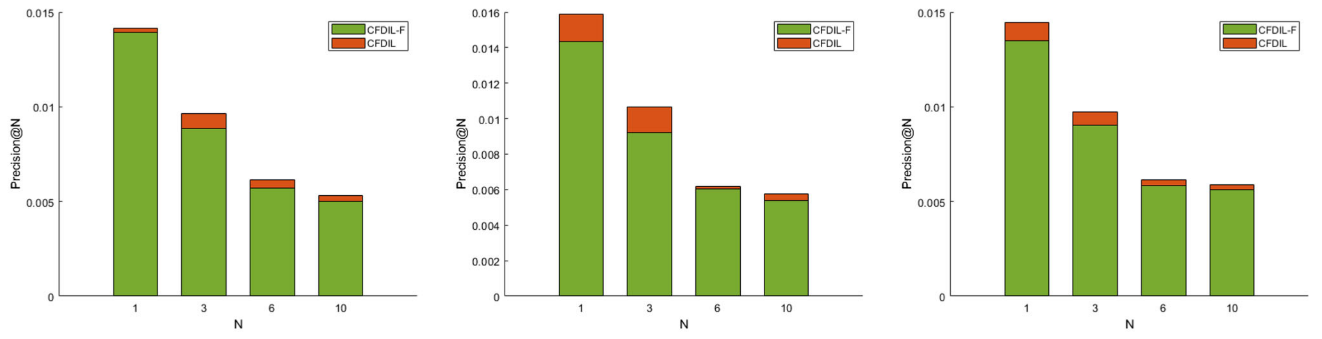
(a) Beijing (b) Shanghai (c) Guangzhou

Fig. 11 Impact of contextual information



(a) Beijing (b) Shanghai (c) Guangzhou

Fig. 12 Impact of CNN



(a) Beijing (b) Shanghai (c) Guangzhou

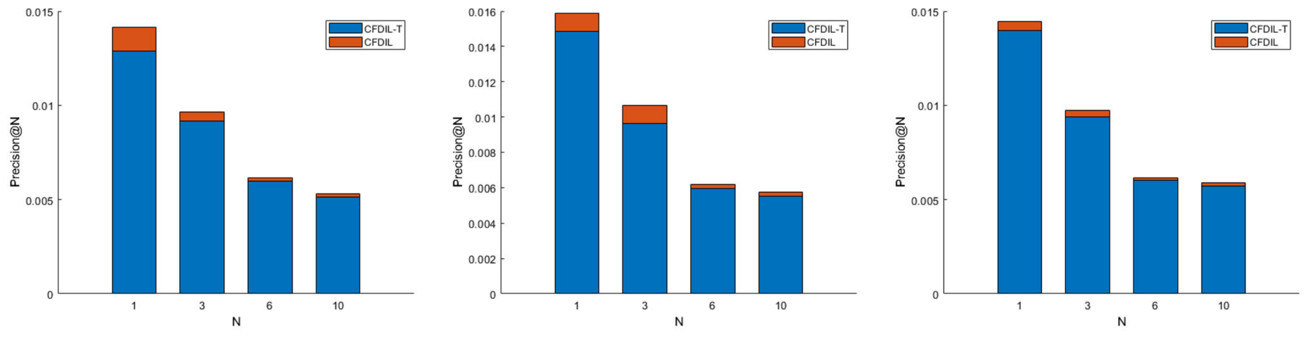
Fig. 13 Impact of FM

5.4.4 Impact of TF

The tensor factorization model in CFDIL is used to process label data. We eliminate the tensor factorization model in CFDIL to get a new model, named CFDIL-T. The difference between CFDIL and CFDIL-T is that the label data of CFDIL-T is extremely unbalanced and sparse. We compare the Precision@N of CFDIL and CFDIL-T to judge the validity of the tensor model in CFDIL for label processing.

The experimental results are shown in Fig. 14. It can be seen from Fig. 14 that the performance of CFDIL is better than CFDIL-T. The experimental results show that the proposed tensor model in CFDIL can effectively handle sparse label data.

The sparse label data of user–app interactions make the training data extremely unbalance. As a result, CFDIL-T cannot fully receive positive user–app feedback data during the training process. The tensor model added in CFDIL can make the label data in user–app-context tensor smoother. The tensor model in CFDIL decomposes the label data, so that even if a user has not touched an app (the original label is 0), the label data of the corresponding user and app will get a nonzero value. The nonzero elements in the user–app-context tensor represent the probability that the user will use the app in a spe-



(a) Beijing (b) Shanghai (c) Guangzhou

Fig. 14 Impact of TF

cific context. Tensor factorization processing enables CFDIL effectively deal with extremely unbalanced user labels, thus improving the performance of app recommendation.

6 Conclusion and future work

In this paper, we proposed the recommendation framework for mobile apps based on contextual feature profiling (CFDIL). As far as we know, this is the first attempt to use contextual feature portraits to explore deep user–app interactions. CFDIL uses contextual feature matrices and the features of users and apps to form feature portraits. Based on these portraits, a deep network framework is trained to provide more accurate recommendations for users, by using decomposers and convolutional neural networks to mine the multi-order interaction between users and apps under specific contextual conditions. We conducted extensive experiments on real-world datasets to prove the effectiveness of CFDIL and the value of each step.

CFDIL constructs feature portraits of users and apps, respectively, using context information and attribute information of users or apps. These feature portraits including space-time laws and attribute features of users and apps. Through a series of experiments, it was proved that these feature portraits are very effective for exploring users' preferences in specific contexts. In CFDIL, FM network can learn shallow features of users and apps, and CNN is more suitable for deep mining of interactive features, which is beneficial to improve the performance of recommendation model. The introduction of feature portraits, combined with the TF processing of labels can resolve the problem of data sparsity to some extent. In addition, CFDIL mines the features of real long-term interactions through feature portraits, which can avoid the problem of unfair recommendations (D'Angelo et al. 2019).

However, there still exist the following shortcomings in CFDIL: (1) CFDIL depends on feature portraits of users and apps to complete recommendation. But these portraits are macro and stable, and sudden events were unable to display. For example, people need to work at home due to the sudden outbreak of the COVID-19, which cannot be reflected in these portraits. This is because users have never done this activity in this context, making the performance of CFDIL not good enough. However, with the frequency of home office increases, this feature will gradually emerge to improve recommendation performance. How to tackle the problem of hysteresis in CFDIL will be our research direction in future. Our preliminary idea is that we introduce attention mechanism to emphasize the importance of recent behavior to shorten the hysteresis of CFDIL as possible. But we know the introduction of attention mechanism will emphasize sudden events, which may have a detrimental effect on the robustness of the model. Therefore, it is important to find the balance between the two. (2) We learn user–app interactions by context information. But the essence of these interactions is diverse instead of single. For example, users play games selectively in free time (active), but it is necessary to punch in (passive). Active choice can better reflect users' preferences; passive choice is more stable. However, this distinction is not drawn in CFDIL, but should be treated equally. All of this need us to do further research and exploration.

Acknowledgements This study was supported by the Joint Funds of the National Natural Science Foundation of China (No. U1536122); Science and Technology Commission Major Special Projects of Tianjin, China (No. 15ZXDSG X00030); Tianjin Municipal Education Commission, China (No. 2021YJSB252).

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by QH, KZ and CW. QH and KZ contribute the same to this article. The first draft of the manuscript was written by QH and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding The authors have not disclosed any funding.

Data availability Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Human participants or animals This article does not contain any studies with human participants or animals performed by any of the authors.

References

Bobadilla J, Ortega F, Gutiérrez A, Alonso S (2020) Classification-based deep neural network architecture for collaborative filtering recommender systems. Int J Interact Multimed Artif Intell 6(1):68–77

Chen N, Hoi SC, Li S, Xiao X (2015) SimApp: a framework for detecting similar mobile applications by online kernel learning. In: Proceedings of the eighth ACM international conference on web search and data mining, pp 305–314

Cheng HT, Koc L, Harmsen J, Shaked T, Chandra T, Aradhye H, Anderson G, Corrado G, Chai W, Ispir M, et al (2016) Wide & deep learning for recommender systems. In: Proceedings of the 1st workshop on deep learning for recommender systems, pp 7–10

D'Angelo G, Palmieri F, Rampone S (2019) Detecting unfair recommendations in trust-based pervasive environments. Inf Sci 486:31–51

Fu B, Lin J, Li L, Faloutsos C, Hong J, Sadeh N (2013) Why people hate your app: making sense of user feedback in a mobile app store. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 1276–1284

Guo H, Tang R, Ye Y, Li Z, He X (2017) Deepfm: a factorization-machine based neural network for ctr prediction. arXiv preprint arXiv:1703.04247

Hao Y, Wang Z, Xu X (2016) Global and personal app networks: characterizing social relations among mobile apps. In: 2016 IEEE International Conference on Services Computing (SCC), IEEE, pp 227–234

Harada S, Taniguchi K, Yamada M, Kashima H (2019) Context-regularized neural collaborative filtering for game app recommendation. In: RecSys (late-breaking results), pp 16–20

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

Hu J, Liang J, Kuang Y, Honavar V (2018) A user similarity-based top-n recommendation approach for mobile in-application advertising. Expert Syst Appl 111:51–60

Kim D, Park C, Oh J, Lee S, Yu H (2016) Convolutional matrix factorization for document context-aware recommendation. In: Proceedings of the 10th ACM conference on recommender systems, pp 233–240

Kim J, Kang S, Lim Y, Kim HM (2013) Recommendation algorithm of the app store by using semantic relations between apps. J Supercomput 65(1):16–26

Liang T, Zheng L, Chen L, Wan Y, Philip SY, Wu J (2020) Multi-view factorization machines for mobile app recommendation based on hierarchical attention. Knowl Based Syst 187:104821

Lin C, Xie R, Guan X, Li L, Li T (2014) Personalized news recommendation via implicit social experts. Inf Sci 254:1–18

Liu B, Kong D, Cen L, Gong NZ, Jin H, Xiong H (2015) Personalized mobile app recommendation: Reconciling app functionality and user privacy preference. In: Proceedings of the eighth ACM international conference on web search and data mining, pp 315–324

Liu CL, Wu XW (2016) Large-scale recommender system with compact latent factor model. Expert Syst Appl 64:467–475

Liu Q, Ma H, Chen E, Xiong H (2013) A survey of context-aware mobile recommendations. Int J Inf Technol Decis Mak 12(01):139–172

Pu C, Wu Z, Chen H, Xu K, Cao J (2018) A sequential recommendation for mobile apps: what will user click next app? In: 2018 IEEE international conference on web services (ICWS). IEEE, pp 243–248

Shan Y, Hoens TR, Jiao J, Wang H, Yu D, Mao J (2016) Deep crossing: web-scale modeling without manually crafted combinatorial features. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 255–262

Wang Y, Yuan NJ, Sun Y, Zhang F, Xie X, Liu Q, Chen E (2016) A contextual collaborative approach for app usage forecasting. In: Proceedings of the 2016 ACM international joint conference on pervasive and ubiquitous computing, pp 1247–1258

Xia X, Wang X, Li J, Zhou X (2014) Multi-objective mobile app recommendation: a system-level collaboration approach. Comput Electr Eng 40(1):203–215

Xu Y, Zhu Y, Shen Y, Yu J (2019) Leveraging app usage contexts for app recommendation: a neural approach. World Wide Web 22(6):2721–2745

Yankov D, Berkhin P, Subba R (2013) Interoperability ranking for mobile applications. In: Proceedings of the 36th international ACM SIGIR conference on research and development in information retrieval, pp 857–860

Yao Y, Zhao WX, Wang Y, Tong H, Xu F, Lu J (2017) Version-aware rating prediction for mobile app recommendation. ACM Trans Inf Syst (TOIS) 35(4):1–33

Zheng X, Ding W, Xu J, Chen D (2014) Personalized recommendation based on review topics. SOCA 8(1):15–31

Zhu H, Chen E, Xiong H, Yu K, Cao H, Tian J (2014) Mining mobile user preferences for personalized context-aware recommendation. ACM Trans Intell Syst Technol (TIST) 5(4):1–27

Zhu H, Liu C, Ge Y, Xiong H, Chen E (2014) Popularity modeling for mobile apps: a sequential approach. IEEE Trans cybern 45(7):1303–1314

Zhu K, Xiao Y, Zheng W, Jiao X, Sun C, Hsu CH (2021) Incorporating contextual information into personalized mobile applications recommendation. In: Soft computing. https://doi.org/10.1007/s00500-021-05988-8