CFDIL: A Context-aware Feature Deep Interaction Learning for App Recommendation

Qingbo Hao (haoqingbo4546@163.com)
Tianjin University of Technology

Ke Zhu
Tianjin University of Technology

Chundong Wang
Tianjin University of Technology

Peng Wang
Nagasaki Institute of Applied Science

Xuliang Mo
Tianjin University of Technology

Zhen Liu
Nagasaki Institute of Applied Science

Research Article

Keywords: Feature portrait, Context-aware, Interaction, App recommendation

Posted Date: October 27th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-742592/v1

License: 😊 This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License
CFDIL: A Context-aware Feature Deep Interaction Learning for App Recommendation

Qingbo Hao¹,² · Ke Zhu²,³ · Chundong Wang¹,² · Peng Wang⁴ · Xiuliang Mo¹,² · Zhen Liu⁴

Received: date / Accepted: date

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, the app recommendation has attracted extensive attention of researchers. Traditional recommendation methods usually use historical data of apps used by users to explore their preferences, and then make an app recommendation list for users. Although the traditional app recommendation methods have achieved certain results, 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 traditional app recommendation models when facing with the sparse user-app interaction data. On the other hand, contextual information has a large impact on users’ app usage preferences, which is often overlooked by traditional app recommendation methods. To overcome the aforementioned problems, we proposed a Context-aware Feature Deep Interaction Learning (CFDIL) method to explore user 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 a novel user and app feature portrait. (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 showed that CFDIL outperformed the benchmark method.

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

1 Introduction

With the vigorous development of the mobile Internet, the number of mobile applications (app) has increased dramatically, and these apps have provided great convenience to people’s production and life. According to statistics¹, 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, recommendation systems are widely used in news media, online shopping and social networking sites. In app recommendation, the traditional recommendation method leverages user’s historical information of using apps to explore

¹ https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/
user’s preferences, and then make recommendations to users based on their preferences. The current mainstream method not only use the historical user’s feedback on apps, such as the user’s rating, comment, frequency of use, etc., but also introduce the additional information about users or apps, such as the users’ own attributes, apps’ version information, category information, and so on. As a result, the recommendation performance can be further improved by building recommendation model based on the above information.

However, the existing methods now facing the following problems: (1) In the real world, users’ data of using apps is extremely sparse compared to the huge number of apps. It is difficult to construct liner recommendation models by using the sparse data. (2) The contextual information has a large influence on users’ preferences, while existing methods only consider the additional information of users and apps, but do not fully consider the influence of contextual information on users’ app usage preferences.

To address the above challenges, we proposed a contextual feature deep interaction learning method, called CFDIL. CFDIL integrates contextual information to model the characteristics of users or apps. CFDIL constructs a novel deep network architecture to explore the multi-order of interaction between users and apps in different contextual information conditions, to make app recommendation accurately. 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 users and apps own and interaction characteristics with each other. 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).

The 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 recommendation. 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 recommendation. 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, due to the extremely sparse user-app interaction matrix, the users or apps matrix constructed based on the sparse matrix often unable to accurately reflect the user’s app preference, which leads to the 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, and so on.
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 of using apps. Zhu et al. (2014a) leverage LDA (Latent Dirichlet Allocation) model to map the interaction contextual information between users and apps into low-dimensional space, and then make recommendation. This idea is the same as the MF-based recommendation. Yao et al. (2017) construct a user-app version rating matrix, and use the 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 the poor interpretability of this method. 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 the 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. (2014b, a); 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 for items. 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 of apps. 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 app for users, which leverages the contextual information to enhance the recommendation performance. Kim et al. (2016) leverage the CNN model to extract users’ preferences of items from users’ comments on items. Xu et al. (2019) incorporate the contextual information into the deep learning model to explore users’ preferences of using apps. Liang et al. (2020) explore the features interaction for app recommendation 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 the 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. Fig. 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. Fig. 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.

We use the real-world scenes to illustrate the above phenomenon. For a user, due to the regularity of his or her daily activity trajectory, the contextual information about the user’s app usage also coincides with the user’s activity spatiotemporal. For instance, users use navigation apps while driving to work, do not use entertainment apps during work hours, and use the food ordering apps during lunch time. As for an app, it will always appear in the context that applies to its function. For example, takeaway apps are always used at meal times, and gaming apps are always used during residential breaks. Incorporating contextual information into preferences modeling will help improve the accuracy of user 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.

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.

| 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, \( t \in T \) |
| \( P^u \) | The feature portrait of user \( u \) in time slot \( t \) |
| \( P^a \) | The feature portrait of app \( a \) in time slot \( t \) |
| Lab_{u,a,t} | number of times user \( u \) interacts with app \( a \) in time slot \( t \) |
| \( F^u \) | The fields set of \( P^u \) |
| \( F^a \) | The fields set of \( P^a \) |
| \( R_u \) | The app recommendation list for user \( u \) |

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.
CFDIL: A Context-aware Feature Deep Interaction Learning for App Recommendation

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. Fig. 4 shows the overall construction process of app portrait and user portrait. 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.

1. According to Chen et al. (2015), the users and apps own attribute information has great impact on users’ app usage 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 or apps into a 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 using their own attribute information. The contextual information mainly refers to time and location information. 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 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 $200 \times 199$ 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 is 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 $P_{at}, P_{at} \in \mathbb{R}^{200 \times 200}, t \in T, a \in A$, 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 $P_{ut}, P_{ut} \in \mathbb{R}^{200 \times 200}, t \in T, u \in U$. 
4.3 Label preprocessing

The data used to train app recommendation model is \( \{P_u^{\text{pt}}, P_a^{\text{pat}}, \text{Lab}_{u,a,t}\}, t \in T, u \in U, a \in A \), where \( P_u^{\text{pt}} \) is the feature portrait of user \( u \) in time slot \( t \), \( P_a^{\text{pat}} \) 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 the user’s app 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 is extremely sparse. The sparse label data makes 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 \( \text{LA} \) with user, app and time slot as coordinates. The elements in \( \text{LA} \) 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 \( \text{LA} \). The principle of the decomposition process is shown in Fig. 5. After decomposition, we get a new tensor \( \text{LA}^* \) with the same size as the original tensor \( \text{LA} \). The difference is that the data in \( \text{LA}^* \) is non-sparse.
3. We use the elements of tensor \( \text{LA}^* \) 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 \( P_u^{\text{pt}} \) and app’s portraits \( P_a^{\text{pat}} \), and processes the sparse label data \( \text{Lab}_{u,a,t} \). In this part, we
will describe in detail how CFDIL uses $P^u_i$, $P^a_i$ and Lab$_{u,a,t}$ for multi-order interactive learning. CFDIL mainly contained two parts, FM part and CNN part. The overall framework of CFDIL is shown in Fig. 6. Next, we will describe in detail how CFDIL performs multi-order interactive learning.

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 is a two-tuple which formally expressed as $\{F^u_i \cup F^a_i, Lab_{u,a,t}\}$, where $F^u_i$ and $F^a_i$ are the field sets of $P^u_i$ and $P^a_i$, $F^u_i = \{f_{u_i}^n, n \in N\}$. $N$ is the number of attribute of $P^u_i$, $F^a_i = \{f_{a_i}^m, m \in M\}$. $M$ is the number of attribute of $P^a_i$. In this paper, $N$ is 4, $F^u_i$ is \{gender, age, device model, user contextual matrix\}; $M$ is 3, $F^a_i$ is \{category, developer, app contextual matrix\}. Lab$_{u,a,t}$ is the label of $F^u_i \cup F^a_i$, 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:

$$\hat{y}_f = \omega_0 + \sum_{i=1}^{M+N} \sum_{j=i+1}^{M+N} f_{i,j} x_i x_j$$

(1)

Where $x_i, x_j \in \{F^u_i \cup F^a_i\}$; $\omega_0$, $\omega_i$, $V_i$, $V_j$ are model parameters, $\omega_0$ is the global bias, $\omega_i$ is the weight of the $i$-th variable, $V_i$ and $V_j$ are the implicit matrix of parameters, $\sum_{i=1}^{M+N} \omega_i x_i$ describes the first-order features of the sample in the form of linear regression, and $\sum_{i=1}^{M+N} \sum_{i=j+1}^{M+N} V_i V_j x_i x_j$ 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, $\cdot \cdot \cdot \cdot \cdot \cdot \cdot$ denotes the dot product of two vectors of dimension $k$, which is expressed as Eq. 2.

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

(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.

$$\sum_{i=1}^{M+N} \sum_{j=i+1}^{M+N} V_i V_j x_i x_j$$

(3)
\[
= \frac{1}{2} \sum_{f=1}^{k} \left( \sum_{i=1}^{M+N} v_i f x_i - \sum_{i=1}^{M+N} v_i f^2 x_i^2 \right)^2
\]

In Eq. 3, \( \sum_{i=1}^{M+N} v_i f x_i \) is a constant value, so we only need the non-zero 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 \( \text{loss}(\hat{y}_f, y_f) = (\hat{y}_f - y_f)^2 \). In addition, we use stochastic gradient descent to train the parameters, and the gradient formula of the parameters is shown in Eq. 4.

\[
\frac{\partial \hat{y}_f(x)}{\partial \theta} = \begin{cases} 
1, & \text{if } \theta = \omega_0 \\
 x_i, & \text{if } \theta = \omega_i \\
x_i \sum_{j=1}^{M+N} v_j f x_j - v_i x_i^2, & \text{if } \theta = v_i f
\end{cases}
\]

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 the FM model is linear, which can effectively train sparse user-app 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

The CNN model is mainly used to learn the higher-order interaction features of user portraits \( P^u \) and app portraits \( P^a \), 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 \( P^u \) and \( P^a \) is 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 is a two-tuple which formally expressed as \( \{ P^u \parallel P^a, \text{Lab}u,a,t \} \), where \( P^u \parallel P^a \) indicates that \( P^u \) and \( P^a \) in the same t-time slot are stitched up and down. \( \text{Lab}u,a,t \) is the label of \( P^u \parallel P^a \), 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 \( P^u \) and \( P^a \). The weight is learned through the hidden layer, and the non-linear 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 \( P^u \) and \( P^a \). The main structure of CNN used for the paper’s 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.

\[
x^{l+1} = \text{ReLU}(W^{(l)}x^{l}) + b^{(l)}
\]

Where \( l \) represents the l-th layer of the neural network. \( x^{l+1} \) is the output of l-th layer. \( W^{(l)} \) and \( b^{(l)} \) are the model parameters and deviations for l-th layer. \( x^{l+1} \) is the output of l-th 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 this paper.

Since we consider the 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.

\[
\text{MSE} = \frac{1}{M} \sum_{i=1}^{M} \left( y_i - \hat{y}_i \right)^2
\]

4.5 App recommendation

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

\[
\hat{y}_i = \epsilon_0 \hat{y}_f + \epsilon_1 \hat{y}_c
\]

where \( \epsilon_0 \) and \( \epsilon_1 \) are model parameters, satisfying \( \epsilon_0 + \epsilon_1 = 1 \). In the subsequent experiments in this paper, setting the values of \( \epsilon_0 \) and \( \epsilon_1 \) 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 \( P^u \) for a target user \( u \) in time slot t. (2) We construct feature portraits \( P^a \) of all candidate apps in time slot t. (3) We input \( P^u \) and \( P^a \) 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 experiment dataset, and then give the evaluation metrics. Finally, we introduce the baseline methods for comparison.

5.1.1 Dataset

Table 2 The data structure of the dataset

| Dataset       | Beijing | Shanghai | Guangzhou |
|---------------|---------|----------|-----------|
| User          | 2,572   | 2,721    | 2,905     |
| Record        | 99,815  | 138,568  | 167,454   |
| Duration      | 168 Hours | 168 Hours | 168 Hours |

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 8,198 users, 2,671 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. The real database used in this paper mainly contains three tables: the user attribute information table, the app attribute information table, and the interaction between users and apps in different contexts. Table 3, Table 4 and Table 5 show samples of the aforementioned data.

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 | User_id | Time stamp | Longitude | Latitude |
|---------|---------|------------|-----------|----------|
| 2007    | 147     | 2018-07-15 10:14 | 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 |

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. We use the training set to train CFDIL, and use the validation set 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. We use the test set to verify the recommendation performance and generalization ability of CFDIL. In the test set, the app actually used by the user is used as the ground truth.

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 Precision@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 $F_a – measure@N$ to measure the recommendation quality.

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

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

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. $F_a – measure@N$ is defined as follows:

F_a – measure@N = \left(1 + \alpha^2\right) \frac{\text{Precision} \times \text{Recall}}{\alpha^2 \text{Precision} + \text{Recall}} \tag{10}

Where precision and recall are the results in Eq. 8 and Eq. 9, respectively; $\alpha$ is used to balance recall and precision. Here we choose 1 as the value of $\alpha$, 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 information system. The detailed definitions of these two indicators are as follows:
We use the following methods to measure the recommendation performance CFDIL we proposed.

**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 using the frequency with which users use the 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 that users use the 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). The 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 nonempty 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 $C^{s\times d}$ matrices using historical usage data. The TF method is also an MF-based recommendation method. The principle of TF is the same as MF.

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

**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

Fig. 7 shows the precision, recall and F-measure values of recommendation results of CFDIL and benchmark methods in 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 the poor recommendation 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 the context information.
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.

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 interactions between users and apps. DNN leverages fully connected method to learn the deep interaction information between users and apps, while CNN uses convolution kernels to extract the deep interaction information between users and apps. DNN’s fully connected structure makes a lot of unnecessary information in the user and app matrix to be added to the interaction process. Excessive information doping interferes with the ability of DNN to mine users’ preferences. The CNN uses convolution kernels to effectively eliminate interference information, so that the model can mine user preferences more efficiently. The recommendation 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 the FM part to learn the 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 the user’s app 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 the features of interactive information 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 $3 \times 3$. 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 the initial number of convolutional layers 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 MAE and RSME of CFDIL and CFDIL-I to judge the validity of the proposed contextual matrices of users and apps.

Fig. 11 shows the MAE and RSME results between CFDIL and CFDIL-I. It can be seen from the figure that the MAE and RSME of CFDIL are significantly better than those of CFDIL-I under all of the recommendation list. This is because CFDIL constructs user and app portraits that fully consider contextual information, which can explore users’ app usage 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 MAE and RSME of CFDIL and CFDIL-C on three cities datasets to judge the effectiveness of the CNN part of CFDIL.

Fig. 12 shows the experimental results. It can be seen from the experimental results, the MAE and RSME of CFDIL are better than those 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 feature interaction information 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 MAE and RSME 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 user preferences. The network structure of FM is similar to the FM part of DeepFM model that has been successful in the 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 for an app, which is a regression problem. The role of the 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 feature and high-order potential interaction feature of users and apps.

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 MAE and RSME 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 interaction between users and apps 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 the 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 non-zero value. The non-zero elements in the user-app-context tensor represent the probability that the user will use the app in a specific context. Tensor factorization processing enables CFDIL to effectively deal...
with extremely unbalanced user labels, thus improving the performance of the 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 the deep user-app interactions. CFDIL uses contextual feature matrices and the features of users and apps to form a feature portrait. Based on the feature portrait, 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. In addition, the construction of feature portraits and the tensor decomposition operation of labels effectively avoid the adverse effects of sparse matrix on the recommendation model. We conducted extensive experiments on real-world datasets to prove the effectiveness of CFDIL and the value of each step.

In the future, we plan to expand our work in the following two areas. (1) Different situations have temporal characteristics and how to deal with the temporal characteristics of the situation. (2) Users interact with the app through context. The nature of this interaction is different (e.g. users selectively play games in their leisure time and users clocking in at work is inevitable), and how to reflect and handle the different nature of the interaction.

7 Acknowledgments

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)

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.
Authorship 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.

References

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
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:170304247
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 Systems with Applications 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. The Journal of Supercomputing 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. Knowledge-Based Systems 187:104821
Lin C, Xie R, Guan X, Li L, Li T (2014) Personalized news recommendation via implicit social experts. Information Sciences 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 Systems with Applications 64:467–475
Liu Q, Ma H, Chen E, Xiong H (2013) A survey of context-aware mobile recommendations. International Journal of Information Technology & Decision Making 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. Computers & Electrical Engineering 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 Transactions on Information Systems (TOIS) 35(4):1–33
Zheng X, Ding W, Xu J, Chen D (2014) Personalized recommendation based on review topics. Service Oriented Computing and Applications 8(1):15–31
Zhu H, Chen E, Xiong H, Yu K, Cao H, Tian J (2014a) Mining mobile user preferences for personalized context-
aware recommendation. ACM Transactions on Intelligent Systems and Technology (TIST) 5(4):1–27
Zhu H, Liu C, Ge Y, Xiong H, Chen E (2014b) Popularity modeling for mobile apps: A sequential approach. IEEE transactions on cybernetics 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, pp https://doi.org/10.1007/s00500-021-05988-8
The SVJOUR3 document class users guide
 Version 3.2 – for Springer journals

© 2007, Springer Heidelberg
All rights reserved.

8 May 2007

Contents

1 Introduction .......................... 1
   1.1 Overview .......................... 2
   1.2 Using PostScript fonts ............ 2

2 Initializing the class ................. 2

3 The article header .................... 4
   3.1 The title ........................ 4
   3.2 Authors .......................... 4
   3.3 Addresses ........................ 5
   3.4 Footnotes to the title block ..... 5
   3.5 Changing the running heads ...... 6
   3.6 Typesetting the header .......... 6

4 Abstract, keywords, MSC, PACS, and CR codes ......... 6

5 Theorem-like structures .......... 7
   5.1 Predefined environments ........ 7
   5.2 Defining new structures ......... 8

6 Additional commands ................ 9

1 Introduction

This documentation describes the SVJOUR3 \LaTeX\ document class. It is not intended to be a general introduction to \LaTeX\ or \BLaTeX\. For this we refer to [2] and [3].
SVJOUR3 was derived from the \LaTeX2ε article.cls, based on \TeX version 3.141 and \LaTeX2ε. Hence text, formulas, figures and tables are typed using the standard \LaTeX2ε commands. The standard sectioning commands are also used.

The main differences to the standard article class are the presence of additional high-level structuring commands for the article header, new environments for theorem-like structures, and some other useful commands.

Please always give a \texttt{\label} where possible and use \texttt{\ref} for cross-referencing. Such cross-references will be converted to hyper-links in the electronic version. The \texttt{\cite} and \texttt{\bibitem} mechanism for bibliographic references is also obligatory.

1.1 Overview

The documentation consists of this document—which describes the whole class (i.e. the differences to the article.cls)—and a ready-to-use template to allow you to start writing immediately.

1.2 Using PostScript fonts

Springer journals produced in \TeX are typeset using the PostScript\textsuperscript{1} Times fonts for the main text. As the use of PostScript fonts results in different line and page breaks than when using Computer Modern (CM) fonts, we encourage you to use our document class together with the psnfss package \texttt{mathptmx}. This package makes all the necessary font replacements to show you the page make-up nearly as it will be printed. Ask your local \TeXpert for details. PostScript previewing is possible on most systems. On some installations, however, on-screen previewing may be possible only with CM fonts.

If, for technical reasons, you are not able to use the PS fonts, it is also possible to use our document class together with the ordinary Computer Modern fonts. Note, however, that in this case line and page breaks will change when we \reTeX your file with PS fonts, making it necessary for you to check them again carefully once you receive the proofs from the printer.

2 Initializing the class

To use the document class, enter

\texttt{\usepackage{svjour3}}

\textsuperscript{1}PostScript is a trademark of Adobe.
at the beginning of your article. The first option \(\langle\text{format}\rangle\) is required and should be set according to the journal for which you are planning to submit a contribution. Three formats are available. The format is pre-set in the template, but choose the one that suits your specific journal if there is no journal-specific template available for your journal.

\begin{itemize}
\item \textbf{[twocolumn]}
\item \textbf{[smallcondensed]} (default)
\item \textbf{[smallextended]}
\end{itemize}

available \texttt{format} options (samples not in full-scale)

There is one general option \(\langle\text{glov3}\rangle\) that is auto-activated if no special option for the particular journal exists or is given. This option causes \LaTeX{} to read in the class option file \texttt{svglov3.clo} (part of the package). Do not try to use those options of the old SVJOUR classes version 1 and 2 as these are not suitable for SVJOUR3—you will get a class error, tops.

Other options, valid for every journal, are

\begin{itemize}
\item \textit{draft} to make overfull boxes visible,
\item \textit{final} the opposite, and
\item \textit{referee} required to produce a hardcopy for the referee with a special layout (bigger interline spacing).
\end{itemize}

The next four additional options control the automatic numbering of figures, tables, equations, and theorem-like environments. The fifth option described below disables the “Springer” theorems (see also Sect. 5). The last option describes the natbib package.

\begin{itemize}
\item \textit{numbook} “numbering like the standard book class”—prefixes all the numbers mentioned above with the section number,
\item \textit{envcountsect} the same for theorem like environments only,
\item \textit{envcountsame} uses one counter for all theorem-like environments,
\item \textit{envcountreset} resets the theorem counter(s) every new section,
\end{itemize}
nospthms

use it only if you want to suppress all Springer theorem-like environments (see Sect. 5) and use the theorem environments of original \LaTeX{} package or other theorem packages instead. (Please check this with your editor.)

natbib

handles reference entries in the author-year system (with or without \BibTeX) by using the natbib package of Patrick W. Daly. It can be found at the Comprehensive \TeX\ Archive Network (CTAN...tex-archive/macros/latex/contrib/supported/natbib/), see \[4, 5, 6\].

If a journal contains articles in languages other than English the class provides two options \"[deutsch]\" and \"[francais]\" that automatically translate supplied texts or phrases given from \LaTeX{}.

There may be additional options for a specific journal—please refer to the extra documentation or to the template file.

As an example, we show how to begin a document for a two-column journal produced in draft mode:

\documentclass[twocolumn,draft]{svjour3}

3 The article header

In this section we describe the usage of the high-level structuring commands for the article header. Header in this context means everything that comes before the abstract.

3.1 The title

The commands for the title and subtitle of your article are

\begin{verbatim}
\title{(your title)}
\subtitle{(your subtitle)}
\end{verbatim}

You can also dedicate your article to somebody by specifying

\begin{verbatim}
\dedication{(dedication)}
\end{verbatim}

3.2 Authors

Information about the authors is provided with
\author{{\authorname /\andauthorname}}

If there is more than one author, the names should be separated by \and. To make this clear, we provide an example:

\author{John B. Doe \and Sally Q. Public \and Joe A. Smith}

3.3 Addresses

Address information is marked with

\institute{{\addressinfo /\andaddressinfo}}

If there is more than one address, the entries are separated by \and.

As the address of the author appears as a footnote on the first page of your article, the author name is to be repeated in the address information with an \at depicting the affiliation. Addresses should be contained in one line, using commas to separate the parts of the address. In addition, you can use

\email{{\emailadd}}

to provide an email address within \institute.

If there are authors appearing with different addresses the affiliations can be indicated with the same author listed “\at” (i.e. before) each particular address in the \institute{...} field—authors in such lists (read: at the same address) should again be separated by an \and.

To continue the example above, we could say

\institute{J.B. Doe
\at Doe Institute, 281 Prime Street, Daisy Town, NA 02467, USA\Tel.: +127-47-678901, Fax: +127-47-678907 \and
J.B. Doe \and S.Q. Public
\at Public-Enterprises \and
J.A. Smith
\at Smith University, \email{smith@smith.edu}}

3.4 Footnotes to the title block

If footnotes to the title, subtitle, author’s names or institute addresses are needed, please code them with
immediately after the word in the corresponding field. Please note that these footnotes are not marked—they will appear above the address information at the bottom of the first page, enclosed in rules.

### 3.5 Changing the running heads

Normally the running heads—if present in the specific journal—are produced automatically by the \maketitle command using the contents of title and author. If the result is too long for the page header (running head) the class will produce an error message and you will be asked to supply a shorter version. This is done using the syntax

\[\textit{titlerunning}\{(text)\}\]
\[\textit{authorrunning}\{(first author et al.)}\]

These commands must be entered before \maketitle.

### 3.6 Typesetting the header

Having entered the commands described in this section, please format the heading with the standard \maketitle command. If you leave it out, the work done so far will produce no text.

### 4 Abstract, keywords, MSC, PACS, and CR codes

The environment for the abstract is the same as in the standard article class. To insert keywords, a “Mathematics Subject Classification” (MSC), “Physics and Astronomy Classification Scheme” (PACS), or “ACM Computing Classification” (CR) codes you should use

\[\textit{keywords}\{(keywords)\}\]
\[\textit{subclass}\{(MSC codes)\}\]
\[\textit{PACS}\{(PACS codes)\}\]
\[\textit{CRclass}\{(CR codes)\}\]

at the end—but still inside—of the abstract environment; the individual words or codes should be separated by and.

Some journals published in other languages than English repeat those elements in translation at the end of the header material before the actual article starts.
5 Theorem-like structures

5.1 Predefined environments

In the SVJOUR3 document class the functions of the standard \texttt{newtheorem} command have been enhanced to allow a more flexible font selection. All standard functions though remain intact (e.g. adding an optional argument specifying additional text after the environment counter). To typeset environments such as definitions, theorems, lemmas or examples, we have predefined the environments in the list below. Note that the font selection of environment heading vs. its body font is depicted in this list with

- \texttt{environment name} = \texttt{bold heading italic text body}
- \texttt{environment name} = \texttt{bold heading normal text body}
- \texttt{environment name} = \texttt{italic heading normal text body}

Unnumbered environments will be produced by: \texttt{claim} and \texttt{proof}.

Numbered environments will be produced by: \texttt{theorem, proposition, lemma, corollary, definition, exercise, problem, solution, remark, note, case, conjecture, example, property, and question}.

The syntax is exactly the same as described in [3, Sect. 3.4.3]:

\begin{(environment) }[(name)]
\end{(environment) }

where the optional \texttt{name} is often used for the common name of the theorem:

\begin{theorem} [Church, Rosser]
\end{theorem}

Sometimes the automatic braces around the optional argument are unwanted (e.g. when it consists only of a reference made with \texttt{cite}). Then you can wrap the whole theorem-like structure in a \texttt{theopargself} environment. It suppresses the braces and gives you complete control over the optional argument, e.g.:
5.2 Defining new structures

For cases where you do not find an appropriate predefined theorem-like structure above, we provide two mechanisms to define your own environment. Use

\texttt{\textbackslash \texttt{spdefaulttheorem}\{\texttt{env name}\}\{\texttt{caption}\}\{\texttt{cap \ font}\}\{\texttt{body font}\}}

to define an environment compliant with the selected class options (see Sect. 2) and designed as the predefined Springer theorem-like environments.

Continuative commands you can use here are

\texttt{\textbackslash \texttt{spnewtheorem}\{\texttt{name}\}\{\texttt{label text}\}\{\texttt{numbered \ within}\}\{\texttt{label \ font}\}\{\texttt{body \ font}\}}
\texttt{\textbackslash \texttt{spnewtheorem}\{\texttt{name}\}\{\texttt{numbered \ like}\}\{\texttt{label text}\}\{\texttt{label \ font}\}\{\texttt{body \ font}\}}

There is also a starred version, without optional arguments, which provides a theorem environment without numbers. Here \texttt{name} is the name of the environment, \texttt{label text} is the text to be typeset as heading, and the \texttt{label font} and \texttt{body font} are the font for the label text and the theorem body.

If you use the \texttt{numbered \ within} argument, the new structure will be numbered within the specified sectional unit—if you specify \texttt{numbered \ like}, it shares its numbering sequence with the referenced structure.

For instance, the predefined environments \texttt{theorem} and \texttt{proof} are defined as

\texttt{\textbackslash \texttt{spnewtheorem}\{\texttt{theorem}\}\{\texttt{Theorem}\}\{\texttt{bf}\}\{\texttt{it}\}}
\texttt{\textbackslash \texttt{spnewtheorem}\{*\texttt{proof}\}\{\texttt{Theorem}\}\{\texttt{it}\}\{\texttt{rm}\}}

whereas one could define a theorem-like structure \texttt{algorithm}, numbered within the current section as

\texttt{\textbackslash \texttt{spnewtheorem}\{\texttt{algorithm}\}\{\texttt{Algorithm}\}\{\texttt{section}\}\{\texttt{bf}\}\{\texttt{rm}\}}

It is also possible to skip all theorem features of the SVJOUR3 document class (see Sect. 2) and/or to use the \texttt{theorem} package shipped with \LaTeX\ 2ε (see [1] for a complete description) or the \texttt{amsthm} package of \LaTeX\ to define new theorem environments. But note that once you use them you should not change the predefined structures.
6 Additional commands

We provide some additional useful commands which you can use in your manuscript. The first is the \texttt{acknowledgements} environment

\begin{acknowledgements}
...
\end{acknowledgements}

which is usually used as the last paragraph of the last section.

The next is an enhancement of the standard \texttt{caption} command used inside of \texttt{figure} environments to produce the legend. The added command

\sidecaption

can be used to produce a figure legend beside the figure. To activate this feature you have to enter it as the very first command inside the \texttt{figure} environment

\begin{figure}
\sidecaption
\resizebox{0.3\hsize}{!}{\includegraphics*{figure.eps}}
\caption{A figure}
\end{figure}

If there is not enough room for the legend the normal \texttt{caption} command will be used. Also note that this works only for captions that come \textit{after} the included images.

We also have enhanced the \texttt{description} environment by an optional parameter, which lets you specify the largest item label to appear within the list. The syntax now is

\begin{description}[⟨largelabel⟩]
...
\end{description}

The texts of all items are indented by the width of \texttt{largelabel} and the item labels are typeset flush left within this space. Note: The optional parameter will work only two levels deep.

The often missed command

\begin{verbatim}
\qed
\end{verbatim}

yields the known □ symbol with appropriate spacing to close e.g. a proof, use the new declaration

\smartqed

to move the position of the predefined qed symbol to be flush right (in text mode). If you want to use this feature throughout your article the declaration
must be set in the preamble; otherwise it should be used individually in the relevant environment, i.e. proof.

The last two commands working as markup in

\[ \vec{\langle symbol \rangle} \]
\[ \tens{\langle symbol \rangle} \]

mark vectors (e.g. $\mathbf{S}$, or $\mathbf{S}$) and tensors (e.g. $\mathbf{S}$) respectively.

References

1. Mittelbach F., Goossens, M.: The \LaTeX Companion, 2nd edn. Addison-Wesley, Boston, Massachusetts (2004)
2. Knuth D.E.: The \TeXbook (revised to cover \TeX3). Addison-Wesley, Reading, Massachusetts (1991)
3. Lamport L.: \LaTeX: A Document Preparation System, 2nd edn. Addison-Wesley, Reading, Massachusetts (1994)
4. \TeX Users Group (TUG), http://www.tug.org
5. Deutschsprachige Anwendervereinigung \TeX e.V. (DANTE), Heidelberg, Germany, http://www.dante.de
6. UK \TeX Users’ Group (UK-TuG), http://uk.tug.org