International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-019-00981-y

ORIGINAL ARTICLE

A deep neural networks based recommendation algorithm using user and item basic data

Jian-Wu Bi[1] · Yang Liu[1] · Zhi-Ping Fan[1,2]

Received: 16 July 2018 / Accepted: 10 July 2019

Abstract
User basic data (e.g. user gender, user age and user ID, etc.) and item basic data (e.g. item name, item category, etc.) are important side data that can be used to enhance the performance of recommendation algorithms, whereas attempts concerning this issue are still relatively scarce. In this study, a deep neural networks based recommendation algorithm is proposed where user average rating, user basic data (user gender, user age, user occupation, user ID), item basic data (item name, item category, item ID) and item average rating are used. The main idea of the algorithm is to build a regression model for predicting user ratings based on deep neural networks. For this, according to the user data (user average rating and user basic data) and the item data (items basic data and item average rating), a user feature matrix and an item feature matrix are respectively constructed using the four types of neural network layers [i.e., embedding layer (EL), convolution layer (CL), pooling layer (PL) and fully connected layer (FCL)]. Then, based on the obtained user feature matrix and item feature matrix, a user-item feature matrix is further constructed using a FCL. On this basis, a regression model for predicting user ratings can be trained, and a recommendation list can be generated according to the predicted user ratings. To verify the effectiveness of the proposed algorithm, three experiments are conducted using the real data from the MovieLens website. The results of experiments show that the proposed algorithm not only outperforms the state-of-the-art collaborative filtering (CF) recommendation algorithms but also alleviates the data sparsity problem and cold-start problem that would occur when the state-of-the-art CF recommendation algorithms are used.

Keywords Recommendation algorithm · Deep neural networks · Side data · Collaborative filtering · Sparsity problem · Cold-start problem

1 Introduction

With the emergence and development of the Internet economy, the amount of information on the Internet has grown explosively. Information overload has become an important issue for users and producers. How to make it convenient for consumers to find their own useful information from the massive information, and to make the information producers show their information to the target users in time has become an important task [1]. To this end, recommendation systems, which attracted the attentions of both the academic and application domains, have emerged as an effective mechanism to solve the information overload problem.

The core of recommendation systems is recommendation algorithms, which determine the quality of recommendation results. To date, collaborative filtering (CF) recommendation algorithms are regarded as the most popular and widely used ones in both academia and industry, since they are simple and effective [2, 3]. Although CF recommendation algorithms have been widely studied and applied, two important problems still remain.

1. Sparsity problem: In general, a single user usually has very limited rating information concerning the available items, which leads to the user-item rating matrix very sparse. The sparse user-item rating matrix makes it difficult for CF recommendation algorithms to iden-

✉ Yang Liu
liuy@mail.neu.edu.cn

Jian-Wu Bi
jianwubi@126.com

Zhi-Ping Fan
zpfan@mail.neu.edu.cn

1 Department of Information Management and Decision Sciences, School of Business Administration, Northeastern University, Shenyang 110169, China

2 State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China

Published online: 20 July 2019

tify similar users or items and significantly restricts the recommendation quality [4–6].
2. Cold-start problem: On the one hand, new users do not have any rating record on available items, thus it is difficult for the CF recommendation algorithms to predict the user preferences or calculate the similarities between the new users and current users. As a result, the recommendation list cannot be generated to new users, which is the so-called user cold-start problem [7, 8]. On the other hand, new items do not have any rating records, thus it is difficult for the CF recommendation algorithms to calculate the similarities between the new items and current items. As a result, the new items cannot be recommended to users, which is the so-called item cold-start problem [9–11].

To address the two problems, various types of side (or auxiliary) data have been incorporated into recommendation algorithms. These algorithms can improve the accuracy of recommendation results, and alleviate the data sparse or cold-start problems. The side data that are widely studied and used in the current recommendation algorithms mainly include: social trust relations [12–16], item contents [17, 18], item reviews [19, 20], social networks [21, 22] and users' social tags [23]. However, in some practical cases, these side data cannot be obtained by sellers or analysts, which may limit the uses of these algorithms. Compared with these types of side data, user basic data (e.g. user gender, user age and user ID) and item basic data (e.g. item name, item category) are easier to be obtained in e-commerce websites. User and item basic data contain user features and item features, which can be used to enhance the performance of the recommendation algorithm and to alleviate the data sparsity and cold-start problems. For this reason, some scholars have tried to develop recommendation algorithms using user and item basic data. For example, Tejeda-Lorente et al. [24] proposed a recommender system for researchers based on bibliometrics, where user and item basic data are respectively used to construct user and item features based on the 2-tuple linguistic approach. However, it should be pointed out that, on the one hand, although some recommendation algorithms using user and item basic data have been developed, these algorithms usually convert user and item data directly into user or item features through some rules or methods (e.g., using fuzzy numbers or interval numbers). It should be noted that these direct conversion methods may result in information loss, and can not fully extract the potential or deep user and item features from user and item basic data for recommendations. On the other hand, compared with other researches on recommendation algorithms, research on constructing recommendation systems simultaneous using user and item basic data is still relatively rare. Therefore, to fully mine and extract the potential or deep user and item features from user and item basic data, and alleviate the data sparsity and cold-start problems, it is necessary to further conduct the studies on new recommendation algorithm that can simultaneously integrate user and item basic data to enhance the performance of the recommendation algorithm.

User and item basic data contain many dimensions which can be used to extract user and item features for recommendations. Based on different features extracted from the dimensions, different recommendation algorithms may be obtained, and the different recommendation results can be suggested by the different algorithms. Thus, how to extract and construct effective features from these dimensions for recommendations is the key issue to develop new recommendation algorithms based on user and item basic data. In recent years, deep learning techniques have made important progress and achieved great success in many fields, such as image classification [25], speech recognition [26], natural language processing [27], etc. Since deep learning techniques have great potential in automatically extracting features from data, such techniques have already begun to be applied to the recommendation field [17, 28]. To better extract features from user and item basic data and get better recommendation results, deep learning techniques are used in this study. Specifically, based on user and item basic data, a deep neural networks based recommendation algorithm is proposed to alleviate the sparsity and cold-start problems. In the algorithm, four aspects of data, i.e., user rating data, user basic (demographic) data, item basic data and item rating data, are used to construct the user-item features for ratings prediction. Based on the real data from the MovieLens website, three experiments are conducted to verify the effectiveness of the proposed algorithm.

The main contributions of this paper are summarized as follows:

1. To the best of our knowledge, this paper is the first attempt to simultaneously incorporate user basic data and item basic data into the development of recommendation algorithm based on deep learning. The incorporation of user basic data and item basic data is helpful to enhance the performance of the recommendation algorithm and to alleviate the data sparsity and cold-start problems in e-commerce website.
2. A novel framework for integrating user average rating, user basic data (user gender, user age, user occupation, user ID), item basic data (item name, item category, item ID) and item average rating is proposed. The framework has a good universality and expansibility, and the framework can be served as a basic framework for integrating more kinds of side data into the development of recommendation algorithms.
3. The effectiveness of the proposed algorithm has been verified by three experiments using the MovieLens 1M

data set. The results of experiments indicate that the proposed algorithm not only outperforms the state-of-the-art CF recommendation algorithms, but also alleviates the sparsity problem and cold-start problem that would occur when the state-of-the-art CF recommendation algorithms are used.

It should be noted that, although the proposed algorithm has a better performance in terms of accuracy compared with the state-of-the-art CF recommendation algorithms, the proposed algorithm has a higher time and space complexities since more types of input data and multiple times of iterations are involved. Fortunately, with the advances in computing and large scale computing techniques, this limitation will dissipate over time.

The remainder of this paper is organized as follows. Section 2 briefly reviews the relevant literatures of recommendation algorithms. Section 3 presents the proposed algorithm. In Sect. 4, three experiments are conducted to verify the effectiveness of the proposed algorithm. Finally, Sect. 5 presents the conclusions of this paper.

2 Related work

To date, CF algorithms are the most popular and widely used recommendation algorithms in both academia and industry. According to the mechanism of the algorithms, CF recommendation algorithms can be divided into two categories, i.e., memory-based CF and model-based CF. On the one hand, the main idea of the memory-based CF is to make a recommendation based on the similarities between users or items, in which the similarities are measured according to the interaction information between users and items, such as ratings and clicks [2, 3]. If the result of recommendation is determined based on the similarity between users, then the algorithm is called user-based CF [29, 30]; whereas, if the result of recommendation is determined based on the similarity between items, then the algorithm is called item-based CF [31, 32]. On the other hand, the main idea of the model-based CF is to predict user rating of unrated items by developing models using different data mining algorithms and machine learning algorithms [33–35]. At present, many CF recommendation algorithms have been proposed, among which the most commonly used ones are: user-based algorithm [48], centered user-based algorithm [49], user-based baseline algorithm [50], singular value decomposition algorithm [34], non-negative matrix factorization algorithm [51], slope one algorithm [52], co-clustering [53, 54], etc. Although these CF recommendation algorithms have been widely studied and applied, two well-known problems still remain, i.e., (1) the sparsity problem and (2) cold-start problem. The reasons for the two problems are respectively given below: (1) a single user usually has very limited rating information concerning the available items, which leads to the user-item rating matrix very sparse. The sparse user-item rating matrix makes it difficult for the above CF recommendation algorithms to identify similar users or items and significantly restricts the recommendation quality. (2) On the one hand, new users do not have any rating record on available items, thus it is difficult for the above CF recommendation algorithms to predict the user preferences or calculate the similarities between the new users and current users. As a result, the recommendation list cannot be generated to new users. On the other hand, new items do not have any rating records, thus it is difficult for the above CF recommendation algorithms to calculate the similarities between the new items and current items. As a result, the new items cannot be recommended to users.

To enhance the performance of the CF recommendation algorithms, various types of side (or auxiliary) data have been incorporated into recommendation algorithms. These studies mainly include two categories, i.e., studies on incorporating user side data into recommendation algorithms and studies on incorporating item side data into recommendation algorithms. On the one hand, the studies on incorporating user side data into recommendation algorithms mainly attempt to use user side data, such as users' social trust relations and users' social tags, etc., to enhance the performance of the recommendation algorithms. For example, Chen et al. [36] proposed a recommendation algorithm for user cold-start based on trust and distrust networks. In the algorithm, trustworthy users are first identified through analyzing the web of trust of experienced users. Then, based on reputation scores of the trustworthy users, the rating preferences of the trustworthy users are aggregated to generate recommendation list for new users. Deng et al. [12] proposed a trust-based recommendation algorithm in social networks. In the algorithm, a deep auto encoder is first used to pre-train the initial values of users and items latent features. Then, based on the obtained latent features, an objective function is built by considering users' characteristics, trusted friends' recommendations and trust propagation. By minimizing the value of objective function, the recommendation results can be obtained. Xia et al. [37] proposed a recommendation algorithm based on social tags. In the algorithm, social tags are first used to construct user interest and product features by a user interest model. Then, the interest model is optimized by clustering social tags to find out the products that user are interested. These obtained products can be recommended to users. On the other hand, the studies on incorporating item side data into recommendation algorithms mainly attempt to use item side data, such as item content, item reviews and item image, etc., to enhance the performance of recommendation algorithms. For example, Zhao et al. [22] proposed a product recommendation algorithm based on the product adopter (the actual user of the product) information mined from online reviews. In the algorithm, product adopters are extracted from online reviews and categorized into different demographic categories by a bootstrapping approach. Products and users are then respectively represented by a distribution on the demographic categories. The two distributions are iteratively updated by a graph-based method. By incorporating the obtained two distributions into a matrix factorization approach, product recommendation can be made. Kim et al. [17] proposed a context-aware hybrid recommendation algorithm by integrating convolutional neural network into probabilistic matrix factorization. In the algorithm, document latent vectors are extracted from the description documents of items using a convolutional neural network. Both the obtained document latent vectors and user-item rating matrix are used to build the probabilistic matrix factorization for rating prediction. Wei et al. [28] proposed two recommendation algorithms by combining stacked denoising auto-encoder with time SVD++ for item cold-start problem. In the algorithms, content features of the items are extracted from items contents using the stacked denoising auto-encoder. Two models are constructed based on time SVD++ to take both content features and user-item rating matrix into ratings prediction.

It can be seen from the existing studies that, to alleviate the sparsity and cold-start problems, various types of side data have been incorporated into recommendation algorithms so as to enhance the performance of CF recommendation algorithms. The side data that are widely studied and used in the current recommendation algorithms mainly includes: social trust relations [12–14, 38], item contents [17, 18], item reviews [19, 20, 39] and users' social tags [23], etc. However, in some practical cases, these side data cannot be obtained by sellers or analysts. The absence of side data limits the uses of these algorithms. Compared with the above side data, user basic data (e.g. user gender, user age and user ID) and item basic data (e.g. item name, item category) are easier to be obtained in e-commerce websites, which can be used to enhance the performance of the recommendation algorithm and alleviate the data sparsity and cold-start problems. However, till now, there are few attempts to simultaneously incorporate user and item basic data into recommendation algorithm. Thus, to alleviate the data sparsity and cold-start problems and enhance the performance of the recommendation algorithm, new recommendation algorithms integrating user and item basic data are needed.

3 Recommendation algorithm

The framework of the proposed recommendation algorithm is shown in Fig. 1. The main idea of the recommendation algorithm is to build a regression model using neural networks for predicting user ratings. In the algorithm, four aspects of data, i.e., user rating data, user basic data, item basic data and item rating data are used to construct the user-item features. It should be noted that the proposed recommendation algorithm is to conduct item recommendation for the consumers or users of e-commerce websites. Although it may be difficult for ordinary users or researchers to obtain relevant information about users and items, it would be easier for e-commerce websites or third-party platforms to obtain the required basic information of the algorithm through the registration information of users and items in the database. Therefore, the proposed algorithm in this paper has a wide application background. To construct the user-item features through the four aspects of data, four types of neural network layers are adopted, i.e., embedding layer (EL), convolution layer (CL), pooling layer (PL) and fully connected layer (FCL). Specifically, the framework of the proposed recommendation algorithm is composed of four parts, i.e.,

- Part 1: Representation of user features.
- Part 2: Representation of item features.
- Part 3: Representation of user-item features.
- Part 4: Training the model.

In Part 1, according to the user average rating and user basic data, the user features are constructed using the EL and FCL. In Part 2, according to the item basic data and item average rating, the item features are constructed using the EL, CL, PL and FCL. Then, in Part 3, based on the obtained user features and item features, the user-item features are further constructed using FCL. On this basis, in Part 4, a regression model for predicting user ratings is trained, and a recommendation list can be generated according to the predicted user ratings. To describe the proposed recommendation algorithm clearly, a brief introduction of the layers used in the recommendation algorithm is first illustrated in Sect. 3.1. Then, the processes of feature representation concerning the four aspects of data are given in Sect. 3.2. Finally, the processes of training the model and making recommendations are described in Sect. 3.3.

3.1 The layers used in the recommendation algorithm

To illustrate the layers used in the recommendation algorithm clearly, the inputs of the algorithm are first introduced. The inputs of the algorithm include user average rating, user gender, user age, user occupation, user ID, item name, item category, item ID and item average rating. In these inputs, only user average rating and item average rating are numerical data that can be directly used for feature representation. The values of the rest six types of inputs are categorical data or text data that cannot be directly used for mathematical

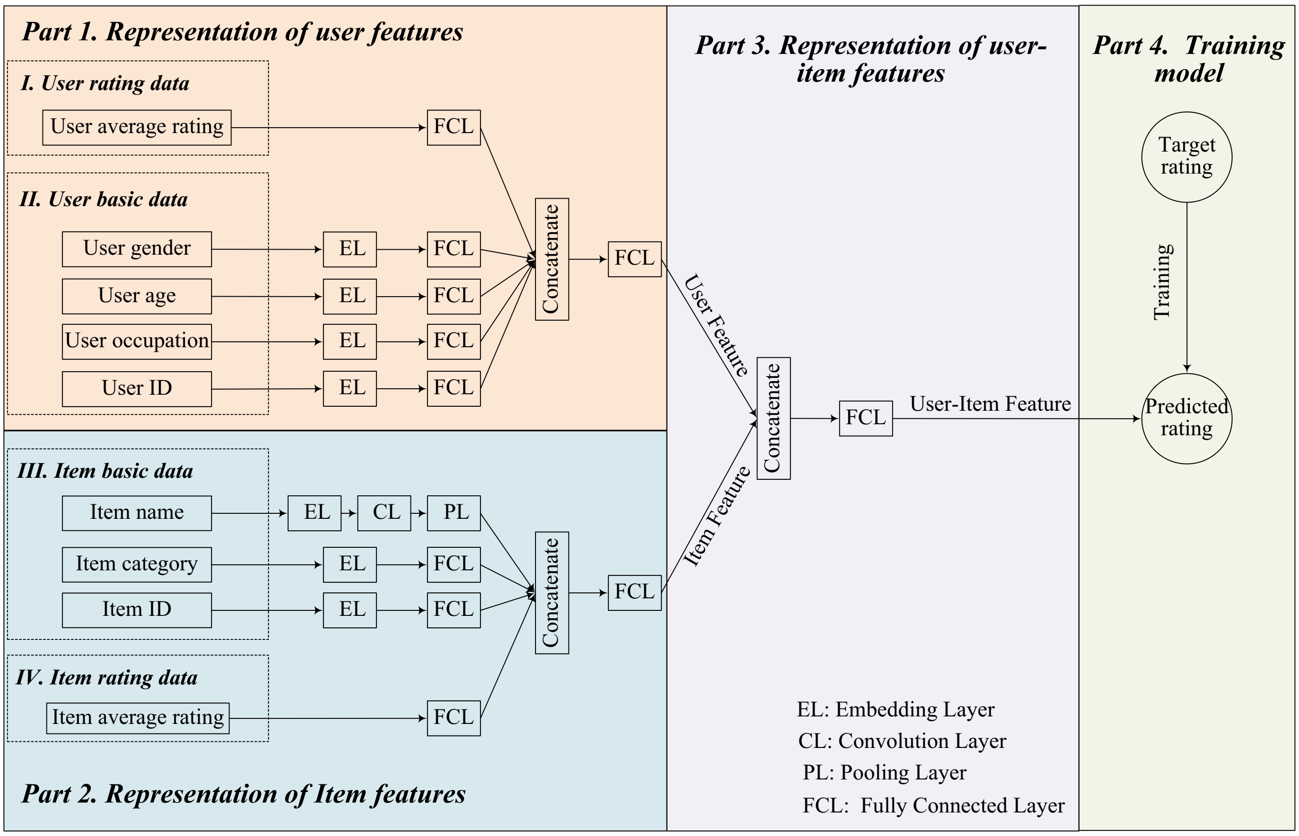

Fig. 1 The framework of the proposed recommendation algorithm

operations. To represent the values of the six types of inputs, these inputs are first respectively transformed into one-hot vectors using one-hot encoding. Let $x_i \in \mathbb{R}^I$ denote the one-hot vector of ith symbol (include words and numbers) in each one of the six types of inputs, where I is the number of unique symbols in each one of the six types of inputs. Here, x_i is a binary vector, where the value of ith element is 1 and the rests are 0. Let us consider the following two film names: "Aladdin and the King of Thieves" and "Lion King". Using the one-hot encoding, each unique symbol in the two film names is transformed into a one-hot vector, as shown in Table 1.

1. Embedding layer.

   The embedding layer maps each of the one-hot vectors into a d-dimensional dense vector. Let $e_i \in \mathbb{R}^d$ denote the d-dimensional dense vector of x_i. Vector e_i can be obtained by Eq. (1), i.e.,

$$e_i = W_\mathrm{E} x_i \tag{1}$$

   where $W_\mathrm{E} \in \mathbb{R}^{d \times I}$ is a weight matrix obtained by training a neural network according to the study of Mikolov et al. [40]. It should be noted that several algorithms, such as continuous bag of words and skip-gram, can be used to train the embeddings for mapping each of the one-hot vectors into a d-dimensional dense vector. In this study, the skip-gram is used to train embeddings for such transformation, details of the skip-gram can be found in literature [40].

   Take the one-hot vectors in Table 1 as an example. By the embedding layer, each of the one-hot vectors in Table 1 maybe transformed into a 3-dimensional dense vector, as shown in Table 2.

Table 1 The one-hot vector of the two film names

Symbol	One-hot vector
Aladdin	$x_1 = (1,0,0,0,0,0,0)$
and	$x_2 = (0,1,0,0,0,0,0)$
the	$x_3 = (0,0,1,0,0,0,0)$
King	$x_4 = (0,0,0,1,0,0,0)$
of	$x_5 = (0,0,0,0,1,0,0)$
Thieves	$x_6 = (0,0,0,0,0,1,0)$
Lion	$x_7 = (0,0,0,0,0,0,1)$

2. Convolution layer.

   In this study, the convolution layer is used to extract item features from item names. The inputs of this layer are item names and the outputs of the layer are contextual feature vectors of the item names [41]. Specifically, based on the obtained dense vector e_i, an item name with I symbols can be represented as a matrix $E \in \mathbb{R}^{I\times d}$, where e_i is the ith line in the matrix, $i = 1, 2, \ldots, I$. Let $E_{i:i+h-1} \in \mathbb{R}^{h\times d}$ denote a window of h words constructed by $e_i, e_{i+1}, \ldots, e_{i+h-1}$, $1 \le h \le I + 1 - i$. Let c_i denote the contextual feature of e_i generated from $E_{i:i+h-1}$. Then, c_i can be determined by Eq. (2):

$$c_i = f(W_C * E_{i:i+h-1} + b) \tag{2}$$

where f denotes a non-linear activation function, such as sigmoid and rectified linear unit (ReLU); $W_C \in \mathbb{R}^{h\times d}$ denotes the weight matrix of a filter; $*$ denotes the convolution operator; $b \in \mathbb{R}$ is a bias term for W_C.

Let $c \in \mathbb{R}^{I-h+1}$ denote a contextual feature vector of an item names with I symbols generated from $E_{i:i+h-1}$. Based on the obtained c_i, c can be further represented by Eq. (3).

$$c = [c_1, \ldots, c_i, \ldots, c_{I-h+1}] \tag{3}$$

It is necessary to point out that a window size with a fixed number of words captures one type of contextual features. To capture multiple types of contextual features, multiple window sizes with different number of words are used in this study.

Take the dense vector in Table 2 as an example. In the example, we set $h = 2$ and choose the ReLU as the activation function, i.e., $f(u) = \max(0, u)$. Besides, we suppose the $W_C = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \end{bmatrix}$ and $b = 0$. Then, c_1, c_2 and c_3 can be calculated as follows:

3. Pooling layer.

The purpose of the pooling layer is to extract important features from the output of the convolution layer and to construct a fixed length feature vector for each item name [17]. As described above, using the convolution layer, an item name with I symbols generated from $E_{i:i+h-1}$ can be represented as a contextual feature vector $c \in \mathbb{R}^{I-h+1}$. Obviously, the contextual feature vector $c \in \mathbb{R}^{I-h+1}$ has a variable length since the number of symbols (i.e., I) in different item names may be different. A variable length of $c \in \mathbb{R}^{I-h+1}$ makes it difficult to be processed by the following layers. Therefore, max-pooling operation, which is used to capture the most important feature, is applied on the contextual feature vector $c \in \mathbb{R}^{I-h+1}$. Let $p \in \mathbb{R}$ denote the output of the pooling layer concerning the contextual feature vector $c \in \mathbb{R}^{I-h+1}$. Then, p can be obtained by Eq. (4).

$$p = \max\{c_1, \ldots, c_i, \ldots, c_{I-h+1}\} \tag{4}$$

4. Fully connected layer.

In this study, the fully connected layer is used to learn features from the previous layers or fuse different types of features. Suppose the inputs and outputs of the fully connected layer are $x = (x_1, \ldots x_i, \ldots x_I)$ and $y = (y_1, \ldots y_j, \ldots y_J)$, respectively. Then, y can be calculated by Eq. (5):

$$y = f(W_F \bullet x + b_F) \tag{5}$$

where f denotes a activation function, such as sigmoid or ReLU; $W_F \in \mathbb{R}^{I\times J}$ denotes the weight matrix; "$\bullet$" denotes the multiplication between matrices; $b_F \in \mathbb{R}^I$ is a bias term.

$$\begin{aligned} c_1 &= \max(0, 0.1 \times 0.31 - 0.2 \times 0.01 + 0.3 \times 0.19 + 0.4 \times 0.51 - 0.5 \times 0.81 - 0.6 \times 0.36) = \max(0, -0.33) = 0, \\ c_2 &= \max(0, 0.1 \times 0.51 - 0.2 \times 0.81 - 0.3 \times 0.36 + 0.4 \times 0.19 - 0.5 \times 0.71 + 0.6 \times 0.92) = \max(0, 0.054) = 0.054, \\ c_3 &= \max(0, 0.1 \times 0.19 - 0.2 \times 0.71 + 0.3 \times 0.92 + 0.4 \times 0.21 - 0.5 \times 0.12 + 0.6 \times 0.03) = \max(0, 0.195) = 0.195. \end{aligned}$$

Table 2 The 3-dimensional dense vector concerning the one-hot vector

Symbol	One-hot vector	3-dimensional dense vector
Aladdin	x_1=(1,0,0,0,0,0,0)	e_1=(0.31, −0.01, 0.19)
and	x_2=(0,1,0,0,0,0,0)	e_2=(0.51, −0.81, −0.36)
the	x_3=(0,0,1,0,0,0,0)	e_3=(0.19, −0.71, 0.92)
King	x_4=(0,0,0,1,0,0,0)	e_4=(0.21, −0.12, 0.03)
of	x_5=(0,0,0,0,1,0,0)	e_5=(0.41, −0.22, −0.13)
Thieves	x_6=(0,0,0,0,0,1,0)	e_6=(0.35, −0.18, 0.09)
Lion	x_7=(0,0,0,0,0,0,1)	e_7=(−0.13, 0.42, 0.63)

3.2 Feature representation

In this section, the processes of feature representation are described, where representation of user features, representation of item features and representation of user-item features are respectively given in Sects. 3.2.1, 3.2.2 and 3.2.3.

3.2.1 Representation of user features

Let $R \in \mathbb{R}^{n\times m}$ denote the user-item rating matrix of n users and m items. The process of constructing the user features is shown in Fig. 2. In Fig. 2, *DF* and *DS* respectively denote the output dimensions of the first FCL and the second FCL. It can be seen from Fig. 2 that two aspects of user data, i.e.,

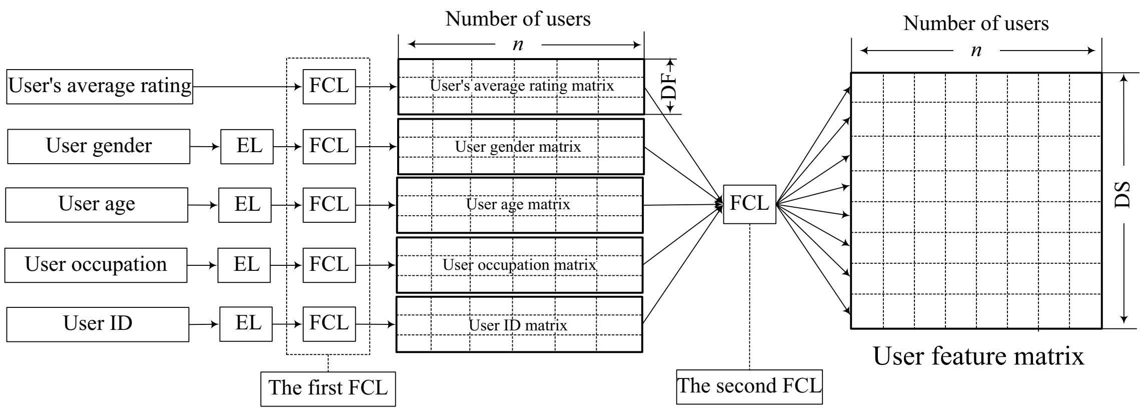

Fig. 2 The process of constructing the user features

user rating data and user basic data, are used to construct the user features, where the user basic data include four types of data, i.e., user gender, user age, user occupation and user ID. Therefore, five types of user data are used to construct the user features. According to the five types of user data, a user feature matrix can be constructed. Specifically, a user average rating matrix can be constructed according to the user average rating data by the first FCL. A user gender matrix, a user age matrix, a user occupation matrix and a user ID matrix can be respectively constructed according to the user gender data, user age data, user occupation data and user ID data by the EL and the first FCL. Then, according to the obtained five matrixes, the user feature matrix can be obtained by the second FCL. The detail process of constructing the user features are given as follows.

1. Representation of the user rating data.

Since the user average rating can reflect the behavior characteristics of the user on providing rating or evaluation, the user average rating can be used to represent user rating data. Specifically, the average rating of each user is first measured by calculating the average value of historical ratings provided by the single user. Then, the features concerning user average ratings can be constructed by a fully connected layer.

2. Representation of user basic data.

In this study, the four types of user basic data, i.e., user gender, user age, user occupation and user ID, are used to construct the user features. The process of feature representation for each type of data is respectively given below.

(a) Representation of user gender.

Users with the same gender are more likely to have similar preferences than users with different genders. Therefore, user gender is selected as a type of user features. Specifically, the user gender data are transformed into one-hot vectors using one-hot encoding. Then, the obtained one-hot vectors are inputted into the embedding layer. By the embedding layer, the dense vectors concerning user gender can be obtained. Then, features concerning user gender can be constructed by inputting the obtained dense vectors into the fully connected layer.

(b) Representation of user age.

Users with similar age are more likely to have similar preferences than those with a larger age gap, thus user age is considered as a type of user features. The process of constructing features concerning user age is the same as that of constructing features concerning user gender.

(c) Representation of user occupation.

Users with similar occupation are more likely to have similar preferences than those with different occupations, thus user occupation is selected as a type of user features. The process of constructing features concerning user occupation is the same as that of constructing features concerning user gender.

(d) Representation of user ID.

A user ID is unique for the user. As we know, the definition approaches of user ID in different websites could be different. In some websites, the user ID is defined by user himself according to his own preference, whereas the user ID in some websites could be assigned by the websites according to some mark rules. If the user IDs are assigned by the websites, then some special marks may embed in the IDs, which could represent different user levels (i.e., Common, Group or VIP) different ages, different regions, and different genders, etc. For the first kind of websites, user

ID may play an unimportant role in the process of recommendation, and may be considered as unrelated factors. For the second kind of websites, user ID may play an important role in the process of recommendation. In addition, previous studies [42, 43] have shown that ID is helpful for recommendation. Therefore, to consider both cases in the process of model training, we take user ID into account to construct the recommendation algorithm.

According to the obtained five types of features, the final user features can be constructed by a fully connected layer.

3.2.2 Representation of item features

The process of constructing the item features is shown in Fig. 3. It can be seen from Fig. 3 that two aspects of item data, i.e., item basic data and item rating data, are used to construct the item features, where the item basic data include three types, i.e., item name, item category and item ID. Therefore, four types of item data are used to construct the item features. According to the four types of item data, an item feature matrix can be constructed. Specifically, an item average rating matrix can be constructed through the item average rating data by the first FCL. Besides, an item name matrix, an item category matrix and an item ID matrix can be respectively constructed through the item name data, item category data and item ID data by the EL and the first FCL. Then, according to the obtained four matrixes, the item feature matrix can be obtained by the second FCL. The detail process of constructing the item features is given as follows.

1. Representation of item basic data.

In this study, three types of the item basic data, i.e., item name, item category and item ID, are used to construct the item features. The process of feature representation for each type of item basic data is respectively given below.

(a) Representation of item name.

Item name is an important feature of an item and can be easily collected. Items with similar names may have higher similarity than items with different names. Thus, users are more likely to show the same preference for these items with similar names. For example, if a user likes the movie “Captain America: The First Avenger”, then the probability that he will like “Captain America: The Winter Soldier” will be higher. Therefore, the item name is considered as a type of item features to generate recommendation. The convolutional neural network is used to process the item names. Specifically, item names are transformed into one-hot vectors using one-hot encoding. Then, the obtained one-hot vectors are inputted into the embedding layer. By the embedding layer, the dense vectors concerning item names can be obtained. Further, the dense vectors are processed using a convolutional layer to extract contextual features. On this basis, the pooling layer is used to extract important features from the extract contextual features and to construct a fixed length feature vector for each item name.

(b) Representation of item category.

Item category is another important feature of an item, and it can also be easily collected. Compared with items with different categories, items with similar categories have a higher degree of similarity. Thus, users are more likely to show similar preferences for items with similar categories. For example, if a user likes science fiction movies, then the probability that he would like another science fiction movie will be greater than that of a horror movie. Therefore, the item category is considered as a type of item features to generate recommendation. The process of constructing features concerning item category is the same as that of constructing features concerning user gender.

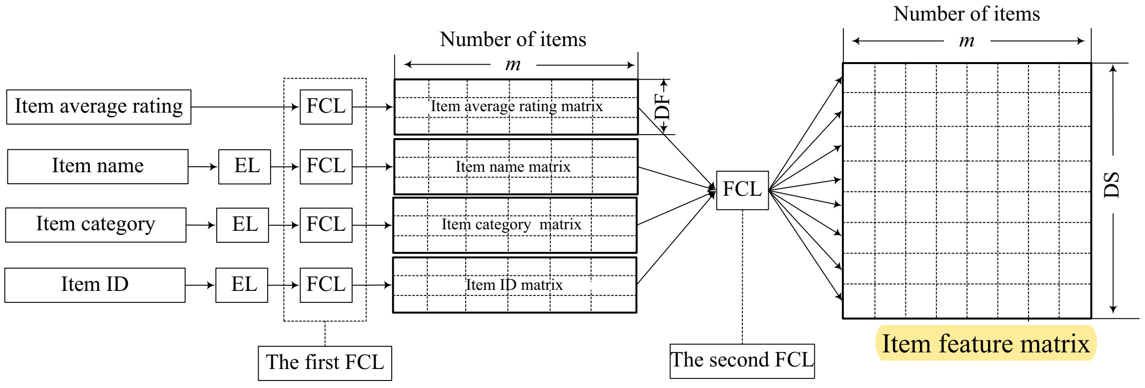

Fig. 3 The process of constructing the item features

(c) Representation of item ID.

An item ID is unique for the item [42, 43]. Similar with that we have discussed in the user ID, the definition approaches of item ID in different websites could be different. If the website defines item ID according to some mark rules, then some special marks could be embedded in the item IDs. Different item IDs may represent items with different categories, brands and origins, etc. In addition, previous studies [42, 43] have shown that ID is helpful for recommendation. Thus, in this paper, the item ID is considered and the process of constructing features concerning item ID is the same as that of constructing features concerning user ID.

2. Representation of item rating data.

An item average rating reflects the item's popularity, which is a type of important item data. Thus, item average rating is selected to represent item rating data. The process of constructing features concerning item average rating is the same as that of constructing features concerning user average rating.

According to the obtained four types of item features, the final item features can be constructed by a fully connected layer.

3.2.3 Representation of user-item features

According to the obtained user feature matrix and item feature matrix, the user-item feature matrix can be constructed. The process of constructing the user-item feature matrix is shown in Fig. 4, where *DT* denote the output dimensions of the third FCL.

In Fig. 4, based on the third FCL, a $n \times NS$ user feature matrix and a $m \times NS$ item feature matrix are converted into a $n \times m \times DT$ user-item feature matrix, where each sub-cuboid represents a feature of a user concerning an item. For example, the shaded sub-cuboid in Fig. 4 represents the feature of user n concerning item m, where DT is also the dimensions of the feature.

3.3 Training the model and making recommendations

The training of the model is to determine the weight matrix and bias term in each layer by minimizing the root mean squared error (MSE) between the actual rating and prediction rating based on the training data. The classical back propagation algorithm [44, 45] and stochastic gradient descent algorithm [46] are used to determine the weight matrix and bias term. Additionally, the dropout technique is applied to prevent the algorithm from overfitting [47].

In accordance with the trained model, user's rating score concerning the items that have not been rated by the user can be predicted. According to the obtained predicted scores, a recommendation list can be generated for a specific user by recommending the items with a higher predicted score for the user. For example, if the predicted scores of five films for a consumer are respectively 2.9, 3.1, 2.4, 4.8 and 1.9, then the film corresponding to the predicted score 4.8 can be recommended to the consumer.

4 Experiments

In this section, to evaluate the performance of the proposed algorithm, three experiments are conducted based on the data from the MovieLens website, where the experimental

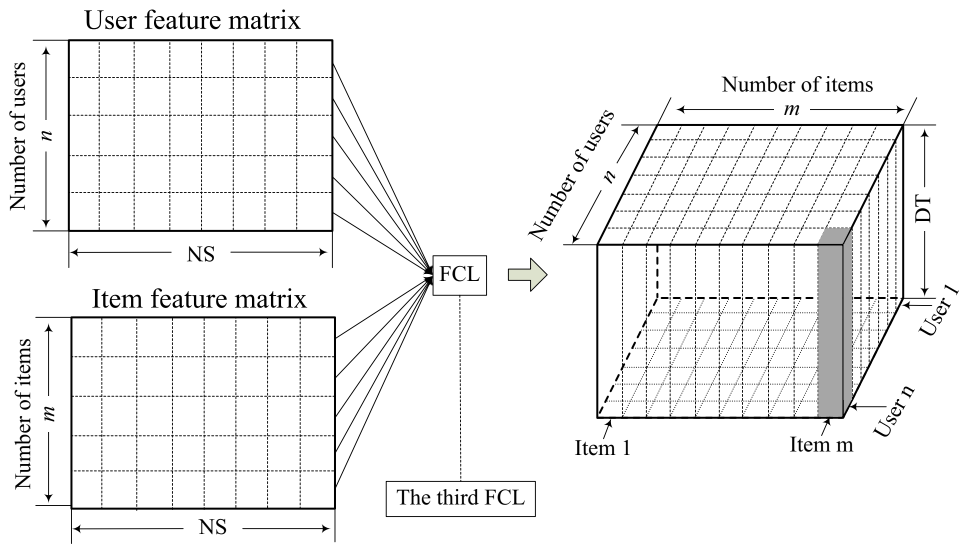

Fig. 4 The process of constructing the user-item features

setting and experimental results are respectively given in Sects. 4.1 and 4.2.

4.1 Experimental setting

4.1.1 Data

The MovieLens 1M data set is used in this study, which can be obtained from the MovieLens website (https://grouplens.org/datasets/movielens/1m/). The data set contains 1,000,209 rating records concerning 6040 users and 3952 movies, where each user has rated at least 20 movies using scores from 1 to 5. In addition, this data set also provides user basic data and movie basic data. The user basic data includes user ID, user gender, user age and user occupation, and the movie basic data includes movie ID, movie title and movie genre.

4.1.2 Compared algorithms

To verify the effective of the proposed algorithm, seven state-of-the-art CF recommendation algorithms are selected as baselines in the experiment. A brief description of each one of the seven algorithms is given below.

1. User-based (UB): The idea of the algorithm is to calculate the similarities between the users, then the rating scores of items that have not been rated are predicted based on the obtained similarities and the corresponding rating records [48].
2. Centered user-based (CUB): The CUB is a collaborative filtering algorithm based on the UB. The main difference between the CUB and UB is that CUB takes into account the mean ratings of each user [49].
3. User-based baseline (UBB): The UBB is also a collaborative filtering algorithm based on the UB. The main difference between the UBB and the UB is that UBB takes into account a baseline rating [50].
4. Singular value decomposition (SVD): This algorithm is to decompose the rating matrix into three matrices based on the singular value decomposition method. The obtained matrices are used to predict the rating scores of items [34].
5. Non-negative matrix factorization (NMF): NMF is a collaborative filtering algorithm that very similar to SVD. In the NMF algorithm, the rating matrix is decomposed using non-negative matrix factorization, where user and item factors are kept positive [51].
6. Slope one (SO): SO is an item-based collaborative filtering algorithm, which makes predictions using the average difference between the ratings of one item and another for users who have rated both [52].
7. Co-clustering (CC): The main idea of this algorithm is to simultaneously obtain user and item neighborhoods through the weighted co-clustering algorithm. The obtained neighborhoods and the individual biases of the users are used to generate predictions [53, 54].

4.1.3 Parameter setting

The parameters used in our algorithm in the experiment are given as follows.

The dimension of embedded layer is 64, i.e., $d = 64$; The window sizes in the convolution layer is 2, 3, 4 and 5, i.e., $h \in \{2, 3, 4, 5\}$; The number of filters in the convolution layer is 8; The learning rate for updating weights and biases is 0.0001; The batch size for training the model is 128; The dropout keep rate for preventing the algorithm from overfitting is 0.5; The output dimensions of the first, second and third FCL are 128 (i.e., $DF = 128$), 256 (i.e., $DS = 256$) and 128 (i.e., $DT = 128$), respectively.

4.1.4 Evaluation metric

The root mean squared error (RMSE), which is widely used in recommendation systems, is adopted to evaluate the performance of the experimental results. RMSE can be calculated by Eq. (6):

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}(r_{ij} - \hat{r}_{ij})^2} \tag{6}$$

where r_{ij} is the actual rating, $\hat{r}_{ij}$ is the prediction rating, N is the total number of ratings. Obviously, the smaller the RMSE is, the better the performance of the algorithm will be.

Additionally, the sparsity rate (SR) is to evaluate the sparsity of the user-item rating matrix. The SR is defined by the following Eq. (7), i.e.,

$$\mathrm{SR} = 1 - \frac{\text{total number of ratings}}{\text{number of users} \times \text{number of items}}. \tag{7}$$

Obviously, a high SR indicates that there are fewer user ratings available in the user-item rating matrix, providing less information for recommendations.

4.1.5 Experimental environment

The experiments are performed on a PC with a 64 GB RAM and a 3.10 GHz, Intel i7-7920HQ 8-Core CPU, using Windows 7 operating system. In the experiments, Anaconda (https://www.anaconda.com/) is used, which is an open source distribution of the Python language for data science and machine learning applications. In addition, the

Tensorflow, a popular framework to implement the deep learning module, is adopted to implement the proposed deep learning based recommendation algorithm.

4.2 Experimental results

We evaluate the performance of the proposed algorithm from the following three cases, i.e., (1) algorithm efficiency under the sparsity case, (2) algorithm efficiency under the user cold-start case and (3) algorithm efficiency under the item cold-start case. The experimental results of the three cases are respectively given in Sects. 4.2.1, 4.2.2 and 4.2.3.

4.2.1 Algorithm efficiency under the sparsity case

To verify the validity of the algorithm, we compare our algorithm with the existing state-of-the-art algorithms under different data sparsity rate (SR) conditions. Following the study of Pan et al. [55], we control the sparsity of the data by changing the size of the training set. Specifically, we randomly select from 10 to 90% of the ratings from the user-item rating matrix as the training set and the rest as the test set. Here, take 10% as an example to illustrate the calculation process of SR. There are 1,000,209 ratings in the use-item matrix, and 10% of the data is selected as the training set, i.e., 100,020.9. According to Eq. (7), the data sparsity can be calculated as follows:

$$SR = 1 - \frac{100020.9}{6040 \times 3952} = 1 - 0.42\% = 99.58\%.$$

The SR corresponding to the size of different training sets is shown in Table 3.

The RMSE of the proposed algorithm and baselines under different SR are shown in Table 4. From Table 4, we can see that the proposed algorithm is better than the existing state-of-the-art algorithms under different SR conditions. Compared with the existing CF algorithms, the proposed recommendation algorithm has a better performance under different data sparse conditions, which may be due to the following reasons: (1) Usually, a single user has only assigned ratings with respect to several items among a large number of available items, which leads to the problem that the user-item rating matrix is highly sparse. It is necessary to point out that, based on the sparse user-item rating matrix, it is difficult for the existing CF recommendation algorithms to identify similar users or items and to obtain high quality recommendation results. Conversely, in the proposed algorithm, a regression model is trained according to the assigned ratings, which can be used to predict the user ratings on other items. Thus, the proposed algorithm is relatively less affected by data sparse problem. (2) The existing CF recommendation algorithms only use the user-item rating matrix, do not make full use of the potential features of users and items. On the contrary, the proposed algorithm not only utilizes the user-item rating matrix, but also user basic data and item basic data. Thus, the proposed algorithm can fully learn the potential features of users and items for recommendation.

In addition, to further analyze the time complexity of different algorithms, the execution time of each algorithm is recorded, as shown in Table 5. Similarly, to further analyze the space complexities of different algorithms, the usages of memory and central processing unit (CPU) of each algorithm are also recorded, which are shown in Table 6. It can be seen from Tables 5 and 6, compared with the existing

Table 3 The SR corresponding to the size of different training sets

Train size	10%	20%	30%	40%	50%	60%	70%	80%	90%
SR	99.58%	99.16%	98.74%	98.32%	97.90%	97.49%	97.07%	96.65%	96.23%

Table 4 The RMSE of the proposed algorithm and baselines under different SR

Train size	UB	CUB	UBB	SVD	NMF	SO	CC	The proposed algorithm
90%	0.9168	0.9253	0.8910	0.8639	0.9144	0.9053	0.9145	0.8394
80%	0.9236	0.9304	0.8961	0.8764	0.9161	0.9076	0.9169	0.8461
70%	0.9292	0.9325	0.8988	0.8822	0.9206	0.9079	0.9206	0.8650
60%	0.9351	0.9347	0.9018	0.8926	0.9230	0.9088	0.918	0.8683
50%	0.942	0.9377	0.9056	0.9021	0.9264	0.9108	0.9241	0.8700
40%	0.9507	0.9418	0.9105	0.9141	0.9332	0.9143	0.9295	0.8990
30%	0.9600	0.9461	0.9161	0.9257	0.9446	0.9201	0.9489	0.9049
20%	0.9727	0.9554	0.9258	0.9345	0.9708	0.9363	0.9664	0.9188
10%	1.0103	0.9914	0.9586	0.9471	1.0230	1.0005	0.9974	0.9289

Table 5 The execution time (s) of the proposed algorithm and baselines under different SR

Train size	UB	CUB	UBB	SVD	NMF	SO	CC	The proposed algorithm
90%	104.792	108.198	110.978	52.428	50.748	45.328	27.363	4587
80%	135.241	146.573	149.686	45.658	45.676	64.905	23.626	4541
70%	162.240	176.014	177.713	40.833	41.782	77.595	23.078	3704
60%	180.815	189.043	194.002	37.813	37.257	84.920	20.497	3582
50%	185.577	196.255	209.416	33.572	32.788	86.360	18.683	3246
40%	181.833	187.144	210.923	28.819	28.121	83.448	17.068	2618
30%	169.381	178.503	201.752	25.398	23.675	75.049	14.275	2352
20%	138.841	156.136	166.461	20.268	18.793	58.555	12.712	1888
10%	89.032	104.937	116.263	15.637	14.848	34.816	11.180	1312

Table 6 The average usages of memory and CPU of different algorithms

Algorithm	UB	CUB	UBB	SVD	NMF	SO	CC	The proposed algorithm
Memory usage (%)	15	25	29	16	16	16	14	41
CPU usage (%)	29	31	30	29	28	30	29	35

CF algorithms, the proposed algorithm has higher time and space complexities. This may be due to the following reasons: (1) there are more types of input data in the proposed algorithm, which requires more steps for data preprocessing; (2) to better extract user and item features and fully fit the data, multiple iterations are used in the proposed algorithm, which also lead to the higher time and space complexities.

4.2.2 Algorithm efficiency under the user cold-start case

At present, the existing state-of-the-art CF algorithms are based on users' historical ratings of items. However, new users do not have any rating record on items, so it is difficult for the existing algorithms to predict user preferences and make a corresponding recommendation, which is the so-called user cold-start problem. An important purpose of this study is to solve the user cold-start problem.

The inputs of the proposed algorithm mainly include four aspects of data, i.e., user average rating, user basic data, item basic data and item average rating. Because new users do not have any rating record on items, the average rating of a new user cannot be obtained. Therefore, for user cold-start problem, only three aspects of data, i.e., user basic data, item basic data and item average rating are selected as the inputs of the proposed algorithm.

To verify the performance of the proposed algorithm in the user cold-start case, we set up four different numbers of new users, i.e., 10, 50, 100, and 200, in this experiment. Since the seven compared algorithms introduced in Sect. 4.1.2 cannot be used to generate predictions for new users, we use the average rating of the item as the baseline in this experiment. The RMSE of the proposed algorithm and the baseline concerning the different number of new users are shown in Fig. 5. From Fig. 5, we can see that the RMSE of the proposed algorithm is significantly smaller than the RMSE of baseline with different numbers of new users. Thus, with respect to the user cold-start case, the proposed algorithm is significantly better than the usage of average rating as the baseline. It should be noted that

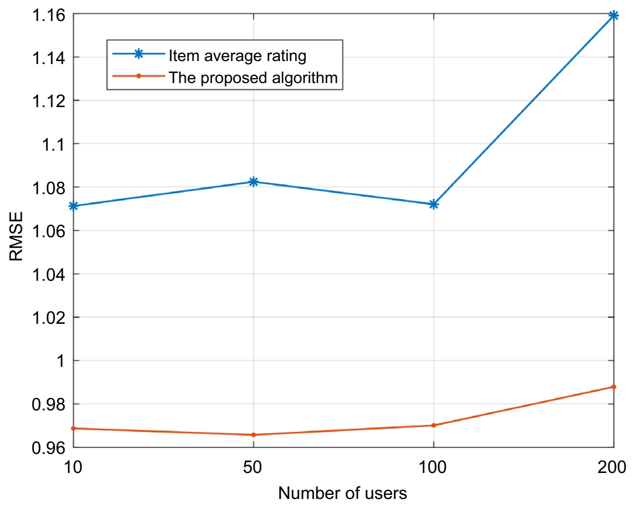

Fig. 5 The RMSE of the proposed algorithm and the baseline concerning different number of new users

the proposed algorithm can fully learn the users' features (including new users without any rating record on items) from user basic data, item basic data and item average rating by using deep learning, which makes the proposed algorithm can alleviate the user cold-start problem to some extent.

4.2.3 Algorithm efficiency under the item cold-start case

At present, the existing state-of-the-art CF algorithms are based on users' historical ratings of items. However, new items do not have any rating records, so it is difficult for the existing algorithms to generate recommendation for new items, which is the so-called item cold-start problem. Another important purpose of this study is to solve the item cold-start problem.

The inputs of the proposed algorithm mainly include four aspects of data, i.e., user average rating, user basic data, item data and item average rating. Because new items do not have any rating record, new item average rating cannot be obtained. Therefore, for item cold-start problem, only three aspects of data, i.e., user average rating, user basic data and item basic data are selected as the inputs of the proposed algorithm.

To verify the performance of the proposed algorithm in the item cold-start case, we set up four different numbers of new items, i.e., 10, 50, 100, and 200, in this experiment. Since the seven compared algorithms introduced in Sect. 4.1.2 cannot be used to generate predictions for new items, following the study of Huang et al. [56], we use the user average rating as the baseline in this experiment. The RMSE of the proposed algorithm and the baseline concerning the different number of new items are shown in Fig. 6.

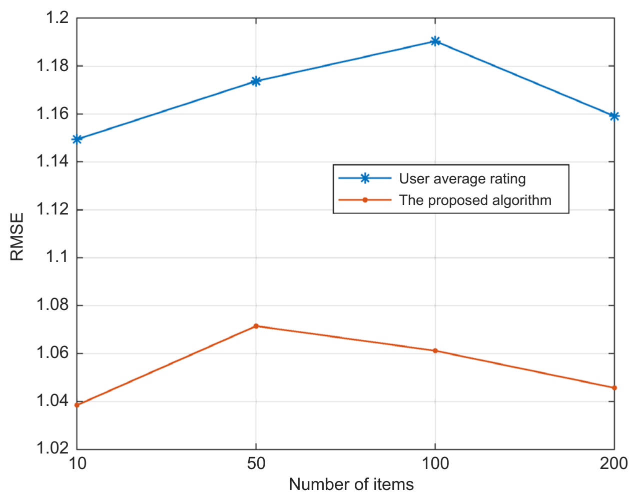

Fig. 6 The RMSE of the proposed algorithm and the baseline concerning different number of new items

From Fig. 6, we can see that the RMSE of the proposed algorithm is significantly smaller than the RMSE of the baseline with different numbers of new items. Thus, with respect to the item cold-start case, the proposed algorithm is significantly better than the usage of average rating as the baseline. It should be noted that the proposed algorithm can fully learn the items' features (including new items that do not have any rating records) from user basic data, item basic data and user average rating by using deep learning, which makes the proposed algorithm can alleviate the item cold-start problem to some extent.

5 Conclusions

Although various types of side (or auxiliary) data have been incorporated into recommendation algorithms, such as social trust relations, item contents, item reviews, social networks and users' social tags, there is no attempt to simultaneously incorporate user basic data and item basic data into the development of recommendation algorithm. In this paper, a deep neural networks based recommendation algorithm is proposed where user average rating, user basic data (user gender, user age, user occupation, user ID), item basic data (item name, item category, item ID) and item average rating are used. The core of the recommendation algorithm is to build a regression model for predicting user ratings based on neural networks. To construct the regression model, four types of neural network layers (i.e., EL, CL, PL and FCL) are used to represent the user data (user average rating and user basic data) and the item data (item basic data and item average rating) into user features and item features, respectively. And then, the user-item features are further constructed for training the regression model. The proposed recommendation algorithm is evaluated by conducting experiments on the MovieLens data set. The experimental results show that the proposed algorithm is better than the existing state-of-the-art algorithms under different SR conditions. In addition, with respect to the user cold-start case, the proposed algorithm is significantly better than the usage of item average rating as the baseline; with respect to the item cold-start case, the proposed algorithm is significantly better than the usage of user average rating as the baseline.

The study also has some limitations, which may serve as avenues for future research. First, although the proposed algorithm has a better performance in terms of accuracy, the proposed algorithm has a higher time and space complexities since more types of input data and multiple times of iterations are involved. Fortunately, with the advances in computing and large scale computing techniques, this limitation will dissipate over time. Second, the user-item rating matrix data is not fully utilized, only the user average rating and the item average rating are used as a part of inputs to construct the user-item features. In the future, based on the framework of the current study, some models such as matrix factorization, can be used to mine more information from the user-item rating matrix to generate more accurate recommendation results. In addition, online reviews, as a new data source, contain a wealth of information, such as customers' concerns, sentiments and opinions [57, 58], which is helpful in developing recommendation systems. Therefore, if online reviews can be integrated into the proposed framework, more accurate recommendation results can be expected.

Acknowledgements This work was partly supported by the National Science Foundation of China (Project Nos. 71771043 and 71871049), Foundation of China Scholarship Council (No. 201706080090), the Fundamental Research Funds for the Central Universities, China (Project No. N170605001) and the 111 Project (B16009).

References

1. Tang X, Xu Y, Geva S (2018) Factorization-based primary dimension modelling for multidimensional data in recommender systems. Int J Mach Learn Cybern. https://doi.org/10.1007/s13042-018-0816-7
2. Huang Z, Zeng D, Chen H (2007) A comparison of collaborative-filtering recommendation algorithms for e-commerce. IEEE Intell Syst 22(5):68–78
3. Linden G, Smith B, York J (2003) Amazoncom recommendations: item-to-item collaborative filtering. IEEE Internet Comput 1:76–80
4. Cai Y, Leung HF, Li Q, Min H, Tang J, Li J (2014) Typicality-based collaborative filtering recommendation. IEEE Trans Knowl Data Eng 26(3):766–779
5. Kim HN, Ji AT, Ha I, Jo GS (2010) Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation. Electron Commerce Res Appl 9(1):73–83
6. Guo G, Qiu H, Tan Z, Liu Y, Ma J, Wang X (2017) Resolving data sparsity by multi-type auxiliary implicit feedback for recommender systems. Knowl Based Syst 138:202–207
7. Ahn HJ (2008) A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. Inf Sci 178(1):37–51
8. Bouras C, Tsogkas V (2016) Assisting cluster coherency via n-grams and clustering as a tool to deal with the new user problem. Int J Mach Learn Cybern 7(2):171–184
9. Kim HN, El-Saddik A, Jo GS (2011) Collaborative error-reflected models for cold-start recommender systems. Decis Support Syst 51(3):519–531
10. Peng F, Lu X, Ma C, Qian Y, Lu J, Yang J (2017) Multi-level preference regression for cold-start recommendations. Int J Mach Learn Cybern 9(11):1–14
11. Pereira ALV, Hruschka ER (2015) Simultaneous co-clustering and learning to address the cold start problem in recommender systems. Knowl Based Syst 82:11–19
12. Deng S, Huang L, Xu G, Wu X, Wu Z (2017) On deep learning for trust-aware recommendations in social networks. IEEE Trans Neural Netw Learn Syst 28(5):1164–1177
13. Fang H, Guo G, Zhang J (2015) Multi-faceted trust and distrust prediction for recommender systems. Decis Support Syst 71:37–47
14. Guo G, Zhang J, Yorke-Smith N (2015) Leveraging multiviews of trust and similarity to enhance clustering-based recommender systems. Knowl Based Syst 74:14–27
15. Guo G, Zhang J, Yorke-Smith N (2016) A novel recommendation model regularized with user trust and item ratings. IEEE Trans Knowl Data Eng 28(7):1607–1620
16. Guo G, Zhang J, Thalmann D, Yorke-Smith N (2014) Leveraging prior ratings for recommender systems in e-commerce. Electron Commerce Res Appl 13(6):440–455
17. Kim D, Park C, Oh J, Yu H (2017) Deep hybrid recommender systems via exploiting document context and statistics of items. Inf Sci 417:72–87
18. Shu J, Shen X, Liu H, Yi B, Zhang Z (2018) A content-based recommendation algorithm for learning resources. Multimed Syst 24(2):163–173
19. Liu H, He J, Wang T, Song W, Du X (2013) Combining user preferences and user opinions for accurate recommendation. Electron Commerce Res Appl 12(1):14–23
20. Ma Y, Chen G, Wei Q (2017) Finding users preferences from large-scale online reviews for personalized recommendation. Electron Commerce Res 17(1):3–29
21. Zhao Z, Yang Q, Lu H, Weninger T, Cai D, He X, Zhuang Y (2018) Social-aware movie recommendation via multimodal network learning. IEEE Trans Multimed 20(2):430–440
22. Zhao WX, Wang J, He Y, Wen JR, Chang EY, Li X (2016) Mining product adopter information from online reviews for improving product recommendation. ACM Trans Knowl Discov Data 10(3):29
23. Nanopoulos A, Rafailidis D, Symeonidis P, Manolopoulos Y (2010) Musicbox: Personalized music recommendation based on cubic analysis of social tags. IEEE Trans Audio Speech 18(2):407–412
24. Tejeda-Lorente A, Porcel C, Bernabé-Moreno J, Herrera-Viedma E (2015) REFORE: a recommender system for researchers based on bibliometrics. Appl Soft Comput 30:778–791
25. Chan TH, Jia K, Gao S, Lu J, Zeng Z, Ma Y (2015) PCANet: a simple deep learning baseline for image classification? IEEE Trans Image Process 24(12):5017–5032
26. Noda K, Yamaguchi Y, Nakadai K, Okuno HG, Ogata T (2015) Audio-visual speech recognition using deep learning. Appl Intell 42(4):722–737
27. Young T, Hazarika D, Poria S, Cambria E (2018) Recent trends in deep learning based natural language processing. IEEE Comput Intell Mag 13(3):55–75
28. Wei J, He J, Chen K, Zhou Y, Tang Z (2017) Collaborative filtering and deep learning based recommendation system for cold start items. Expert Syst Appl 69:29–39
29. Breese JS, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th conference on uncertainty in artificial intelligence, pp 43–52
30. Herlocker JL, Konstan J A, Borchers A, Riedl J (1999) An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval, pp 230–237
31. Deshpande M, Karypis G (2004) Item-based top-n recommendation algorithms. ACM Trans Inf Syst 22(1):143–177
32. Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web, pp 285–295
33. Tsai CF, Hung C (2012) Cluster ensembles in collaborative filtering recommendation. Appl Soft Comput 12(4):1417–1425

34. Sarwar B, Karypis G, Konstan J, Riedl J (2000) Application of dimensionality reduction in recommender system-a case study. In: Proceedings of the ACM WebKDD workshop
35. De Campos LM, Fernández-Luna JM, Huete JF, Rueda-Morales MA (2010) Combining content-based and collaborative recommendations: a hybrid approach based on Bayesian networks. Int J Approx Reason 51(7):785–799
36. Chen CC, Wan YH, Chung MC, Sun YC (2013) An effective recommendation method for cold start new users using trust and distrust networks. Inf Sci 224:19–36
37. Xia X, Zhang S, Li X (2010) A personalized recommendation model based on social tags. In: Database technology and applications (DBTA), 2010 2nd international workshop, pp 1–5
38. Guo G, Zhang J, Zhu F, Wang X (2017) Factored similarity models with social trust for top-N item recommendation. Knowl Based Syst 122:17–25
39. Qiu L, Gao S, Cheng W, Guo J (2016) Aspect-based latent factor model by integrating ratings and reviews for recommender system. Knowl Based Syst 110:233–243
40. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv:13013781
41. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
42. He X, Liao L, Zhang H, Nie L, Hu X, Chua TS (2017) Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web, pp 173–182
43. Zhang L, Luo T, Zhang F, Wu Y (2018) A recommendation model based on deep neural network. IEEE Access 6:9454–9463
44. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323:533–536
45. Werbos PJ (1994) The roots of backpropagation: from ordered derivatives to neural networks and political forecasting. Wiley, New York
46. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:14126980
47. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
48. Konstan JA, Miller BN, Maltz D, Herlocker JL, Gordon LR, Riedl J (1997) GroupLens: applying collaborative filtering to usenet news. Commun ACM 40(3):77–87
49. Zheng VW, Cao B, Zheng Y, Xie X, Yang, Q (2010) Collaborative filtering meets mobile recommendation: a user-centered approach. In: AAAI, pp 236–241
50. Koren Y (2010) Factor in the neighbors: scalable and accurate collaborative filtering. ACM Trans Knowl Discov Data 4(1):1–24
51. Luo X, Zhou M, Xia Y, Zhu Q (2014) An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Trans Ind Inform 10(2):1273–1284
52. Lemire D, Maclachlan A (2005) Slope one predictors for online rating-based collaborative filtering. In: Proceedings of the 2005 siam international conference on data mining, pp 471–475
53. Banerjee A, Dhillon I, Ghosh J, Merugu S, Modha DS (2007) A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. J Mach Learn Res 8:1919–1986
54. George T, Merugu S (2005) A scalable collaborative filtering framework based on co-clustering. In: 5th IEEE international conference on data mining, pp 1–8
55. Pan Y, Wu D, Olson DL (2017) Online to offline (O2O) service recommendation method based on multi-dimensional similarity measurement. Decis Support Syst 103:1–8
56. Huang TCK, Chen YL, Chen MC (2016) A novel recommendation model with Google similarity. Decis Support Syst 89:17–27
57. Liu Y, Bi JW, Fan ZP (2017) Ranking products through online reviews: a method based on sentiment analysis technique and intuitionistic fuzzy set theory. Inf Fusion 36:149–161
58. Bi JW, Liu Y, Fan ZP, Zhang J (2019) Wisdom of crowds: conducting importance-performance analysis (IPA) through online reviews. Tourism Manag 70:460–478