Broad Recommender System: An Efficient Nonlinear Collaborative Filtering Approach

Ling Huang, *Member, IEEE,* Can-Rong Guan, Zhen-Wei Huang, Yuefang Gao, Yingjie Kuang, Chang-Dong Wang, *Member, IEEE*, and C. L. Philip Chen, *Fellow, IEEE*

Abstract—Recently, Deep Neural Networks (DNNs) have been widely introduced into Collaborative Filtering (CF) to produce more accurate recommendation results due to their capability of capturing the complex nonlinear relationships between items and users. However, the DNNs-based models usually suffer from high computational complexity, i.e., consuming very long training time and storing huge amount of trainable parameters. To address these problems, we propose a new broad recommender system called Broad Collaborative Filtering (BroadCF), which is an efficient nonlinear collaborative filtering approach. Instead of DNNs, Broad Learning System (BLS) is used as a mapping function to learn the complex nonlinear relationships between users and items, which can avoid the above issues while achieving very satisfactory recommendation performance. However, it is not feasible to directly feed the original rating data into BLS. To this end, we propose a user-item rating collaborative vector preprocessing procedure to generate low-dimensional user-item input data, which is able to harness quality judgments of the most similar users/items. Extensive experiments conducted on seven benchmark datasets have confirmed the effectiveness of the proposed BroadCF algorithm.

Index Terms—Recommender system, Broad learning system, Collaborative filtering, Neural network

1 Introduction

In this era of information explosion, the rich choice of online services on mobile leads to an increasingly heavy role for recommender systems. The basic idea of recommender systems is to recommend the most suitable items to users based on their hidden preferences found from historical behavioral data [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]. In recommender systems, collaborative filtering is one of the most classical algorithms, which has been widely used in many real-world problems [11], [12], [13], [14], [15].

Traditional collaborative filtering is user-oriented or item-oriented, which relies mainly on the ratings of similar users or items to predict unknown ratings. Among various collaborative filtering techniques, Matrix Factorization (MF) [16], [17], [18], [19] is the most popular approach, which operates by learning a latent space to represent users and items, and mapping users and items into a common space. In the common space, the recommender system predicts a personalized ranking of a set of items for each user based on the similarity between users and items. However, in practical applications, scoring data is prone to long-tail distribution [20], which can lead to the sparsity problem in MF methods. To overcome the sparsity problem, several methods have been proposed [21]. For example, adding social relations to MF or using invisible feedback [22].

Deep Neural Networks (DNNs) have developed rapidly in the past few years, which have been widely explored and shown good results in various fields such as computer vision and natural language processing. DNNs have also been introduced into the field of recommender systems [23], [24], [25], [26], [27], [28]. In the traditional matrix decomposition, the relationships between users and items are usually assumed to be linear, which is however not true in real-world applications. To better learn complex nonlinear relationships between users and items, Xue *et al.* proposed a Deep Matrix Factorization (DMF) [24] that uses a dual-path neural network instead of the linear embedding operation used in ordinary matrix decomposition. DNNs are able to approximate any function, so they are well suited for learning complex matching functions. For example, He et al. proposed NeuMF under the Neural Collaborative Filtering (NCF) framework [26], which takes the connection of user embedding and item embedding as the input and uses the Multi-Layer Perceptron (MLP) model for prediction. Deng *et al.* proposed DeepCF [29] which is a unified deep collaborative filtering framework integrating representation learning and matching function learning. However, the DNNs-based models requires a large number of iterative training processes that consume time and computational resources, and therefore cannot be applied to large-scale data and are very time consuming. As shown in our experiments, due to the out-of-memory error, most DNNs-based models fail to generate results when dealing with an Amazon dataset containing 429622 users, 23966 items, and 583933 ratings on a server with an Intel Core i9-10900 CPU, GeForce RTX 3090, and 256GB of RAM. And even for the relatively small

- *Ling Huang, Can-Rong Guan, Zhen-Wei Huang, Yuefang Gao, and Yingjie Kuang are with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China. huanglinghl@hotmail.com, guancanrong@stu.scau.edu.cn, huangzhenwei@stu.scau.edu.cn, gaoyuefang@scau.edu.cn, kuangyj@scau.edu.cn.*
- *Chang-Dong Wang is with School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China, Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, China, and Guangdong Province Key Laboratory of Computational Science, Guangzhou, China. changdongwang@hotmail.com.*
- *C. L. Philip Chen is with School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. philip.chen@ieee.org.*

datasets that can generate results, most DNNs-based models are time-consuming. In [28], a BP Neural Network with Attention Mechanism (BPAM) has been proposed to mitigate this problem by integrating the attention mechanism into the BP network. However, the gradient-based optimization still consumes long time.

To address the above problems, this paper proposes a novel broad recommender system, called Broad Collaborative Filtering (BroadCF), which is an efficient nonlinear collaborative filtering approach. Different from the existing DNNs-based methods that feed the user-item rating data into DNNs, the proposed BroadCF method feeds the user-item rating data into Broad Learning System (BLS) [30]. As shown in the literature [30], [31], BLS is also able to approximate any function, and its general approximation capability has been theoretically analyzed at the beginning of its design and proved by rigorous theoretical proof. In particular, BLS can map the original samples to a discriminative feature space by using random hidden layer weights according to any continuous probability distribution [30], [31]. This random mechanism provides a fast training process for BLS. That is, only the weights from the hidden layer to the output layer need to be trained by a pseudo-inverse algorithm. Therefore, BroadCF does not need to consume so much training time compared with the DNNs-based models, and is more suitable for larger datasets due to the small number of stored parameters.

However, it is not a trivial task to feed the user-item rating data into BLS. The DNNs-based methods take the original user/item rating vectors (or their direct concatenation) as input, which is suitable because the DNNs-based methods learn the latent factor vectors by stacking several hidden layers, the dimensions of which usually decrease monotonously from the input layer to the output layer. Different from the DNNs-based methods, BLS needs to broaden the original input data into the mapped features and enhanced features with several-order magnitude larger dimensions. Therefore, it is not feasible to take the original user/item rating vectors (or their direct concatenation) as input for BLS. Inspired by the previous work [28], this paper proposes a user-item rating collaborative vector preprocessing to generate low-dimensional user-item input data, which is able to harness quality judgments of the most similar users/items.

The main contributions of this work are summarized as follows.

- A new broad recommender system called BroadCF is proposed to overcome the problems of long training time and huge storage requirements in the DNNs-based recommendation models.
- A new data preprocessing procedure is designed to convert the original rating data into the form of (Rating collaborative vector, Rating one-hot vector) as input and output of BroadCF.
- Extensive experiments are conducted on seven real datasets to demonstrate the effectiveness of the model. The results show that BroadCF significantly outperforms other state-of-the-art algorithms and significantly reduces training time consumption and storage costs.

The rest of this paper is organized as follows. In Section 2, we will review the related work. In Section 3, we will introduce the problem definition and preliminaries. In Section 4, we will describe in detail the proposed BroadCF method. In Section 5, we will report the experimental results along with convincing analysis. Finally, we will conclude the paper in Section 6.

2 Related Work

2.1 Shallow Learning based Collaborative Filtering Methods

Collaborative filtering is one of the most widely adopted recommendation algorithms [32]. Early collaborative filtering in its original form is user-oriented or item-oriented [32], which relies primarily on the ratings of similar users/items to predict unknown ratings. Many efforts have been made in developing variants of CF from different perspectives [1], [3], [33], [34], [35], [36]. In [33], Bell and Koren propose to learn the interpolation weights from rating data as a global solution, which leads to an optimization problem of CF and improves the rating prediction performance. In [22], Koren propose to extend SVD-based latent factor model to the SVD++ model by adding a second set of item factors to model the item-item similarity. Another early attempt is [34], in which the temporal information is considered for improving the accuracy. Instead of relying on the co-rated items, Patra et al. propose to utilize all rating information for searching useful neighbors of the target user from the rating matrix [35]. In order to recommend new items and niche items to users, based on the observation of the Rogers' innovation adoption curve [37], Wang et al. propose an innovator-based collaborative filtering algorithm by designing a new concept called innovators [1], which is able to balance serendipity and accuracy. For more literature reviews of the shallow learning based collaborative filtering methods, please refer to the survey papers [21], [38]. Despite the wide applications, the conventional shallow learning based CF methods assume the linear relationship between users and items, which is however not true in the real-world applications.

2.2 Deep Learning based Collaborative Filtering Methods

Recently, DNNs have been widely used in collaborative filtering to learn the complex mapping relationships between users and items due to their ability to approximate any continuous function. For example, in [24], Xue et al. propose a new matrix decomposition model based on a neural network structure. It maps users and items into a common low-dimensional space via a nonlinear projection. He et al. propose a neural network structure to model the latent features of users and items, and design a general framework for collaborative filtering based on neural networks, called Neural Collaborative Filtering (NCF) [26]. Three examples are developed, namely Generalized Matrix Decomposition (GMF), Multi-Layer Perceptron (MLP), and Neural Matrix Factorization (NeuMF). In [29], Deng et al. propose a unified deep collaborative filtering framework integrating representation learning and matching function learning. In [39], Xue et al. propose a deep item-based

TABLE 1
Summary of the main notations.

$\mathcal{U}$, $\mathcal{V}$	User set, item set
$\mathbf{R} \in \mathbb{R}^{\lvert\mathcal{U}\rvert \times \lvert\mathcal{V}\rvert}$	Rating matrix
$r_{u,i}$	Rating of user u to item i
$\hat{r}_{u,i}$	Predicted rating of user u to item i
k	Pre-specified number of nearest users of each user u
$\mathbf{n}^u \in \mathbb{R}^{1\times k}$	KNU index vector of user u, i.e. the index vector storing the indexes of k nearest users (KNU) of user u
$\mathbf{n}^u_j$	Index of the j-th nearest user of user u
$\mathbf{p}^u_i \in \mathbb{R}^{1\times k}$	KNU rating vector of user u on item i, i.e. the ratings of the k nearest users of user u on item i
$\bar{\mathbf{p}}^u_i \in \mathbb{R}^{1\times k}$	Post-processed KNU rating vector of user u on item i
l	Pre-specified number of nearest items of each item i
$\mathbf{n}^i \in \mathbb{R}^{1\times l}$	LNI index vector of item i, i.e. the index vector storing the indexes of l nearest items (LNI) of item i
$\mathbf{n}^i_j$	Index of the j-th nearest item of item i
$\mathbf{q}^u_i \in \mathbb{R}^{1\times l}$	LNI rating vector of item i from user u, i.e. the ratings of user u to the l nearest items of item i
$\bar{\mathbf{q}}^u_i \in \mathbb{R}^{1\times l}$	Post-processed LNI rating vector of item i from user u
$\mathbf{x}^u_i \in \mathbb{R}^{1\times(k+l)}$	User-item rating collaborative vector for the given user-item pair composed of user u and item i
d_y	Maximum rating, i.e. dimension of user-item rating one-hot vector
$\mathbf{y}^u_i \in \mathbb{R}^{1\times d_y}$	User-item rating one-hot vector for the given user-item pair composed of user u and item i
$\mathcal{D}$	Training set of the proposed BroadCF algorithm
$\mathbf{X} \in \mathbb{R}^{\lvert\mathcal{D}\rvert \times (k+l)}$	Input matrix of BLS
$\mathbf{Y} \in \mathbb{R}^{\lvert\mathcal{D}\rvert \times d_y}$	Output matrix of BLS
d_z	Dimension of each mapped feature group
$\mathbf{Z}_j \in \mathbb{R}^{\lvert\mathcal{D}\rvert \times d_z}$	The j-th mapped feature matrix
n	Number of the mapped feature groups
ϕ_j	The j-th nonlinear feature mapping function
d_h	Dimension of each enhanced feature group
$\mathbf{H}_j \in \mathbb{R}^{\lvert\mathcal{D}\rvert \times d_h}$	The j-th enhanced feature matrix
m	Number of the enhanced feature groups
ξ_j	The j-th nonlinear feature enhancement function
$\mathbf{W} \in \mathbb{R}^{(nd_z+md_h)\times d_y}$	Learnable weight matrix of BLS

collaborative filtering method, which takes into account of the higher-order relationship among items. For more literature reviews of the deep learning based collaborative filtering methods, please refer to the survey paper [40]. However, due to the large number of trainable parameters and complex structures involved, the DNNs-based models usually suffer from high computational complexity, i.e., consuming very long training time and storing huge amount of trainable parameters.

In this paper, to address the above problems, we propose a new neural network based recommender system called Broad Collaborative Filtering (BroadCF). Instead of DNNs, Broad Learning System (BLS) is used as a mapping function to learn the complex relationships between users and items, which can avoid the above issues while achieving very satisfactory recommendation performance.

3 Problem Definition and Preliminaries

In this section, we will present the problem definition and preliminaries. For clarity, Table 1 summarizes the main notations used in this paper.

3.1 Problem Definition

Suppose that in the recommender system we are given a user set $\mathcal{U}$ and an item set $\mathcal{V}$ respectively. Following [29], a user-item rating matrix $\mathbf{R} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{V}|}$ is constructed from users' ratings on items as follows,

$$\mathbf{R}_{u,i} = \begin{cases} r_{u,i}, & \text{if user } u \text{ has rated item } i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $r_{u,i}$ denotes the rating of user u on item i.

The goal of recommendation algorithms is to estimate the missing ratings in the rating matrix $\mathbf{R}$. The model-based methods generate the predicted rating $\hat{r}_{u,i}$ of user u to item i as follows,

$$\hat{r}_{u,i} = f(u, i|\Theta) \tag{2}$$

where f denotes the mapping function that maps the model input, e.g. the ratings of neighbor users or items, to the predicted rating of the corresponding user-item pair composed of user u and item i by utilizing the model parameters Θ [29].

The mapping function of DNNs-based CF inherits the nonlinear representation learning capability of DNNs, and thus it is able to learn the complex nonlinear relationships between users and items. However, due to the large number of trainable parameters involved, the complex structure and the continuous iterative training process, the DNNs-based models often suffer from high computational complexity, i.e., consuming very long training time and storing huge amount of trainable parameters. Although a recent work called BPAM [28] has shed light on mitigating the problem of training time consumption by integrating attention mechanisms into BP networks. However, the gradient-based optimization still takes some time. In this paper, instead of DNNs, a lightweight neural network called broad learning system (BLS) [30] is used as a mapping function to learn the complex relationships between users and items, which can avoid the above issues while achieving very satisfactory recommendation performance.

3.2 Preliminaries

Broad Learning System (BLS) is a lightweight neural network that can approximate any function [30], which has been widely used in many different applications [41], [42], [43], [44]. It is based on a random vector functional linked neural network (RVFLNN) [45]. In BLS, the original data are firstly mapped into the mapped features by random weights, which are stored in feature nodes. Next, the mapped features are further mapped by random weights to obtain enhanced features, which are used for extensive scaling. Finally, the parameter normalization optimization problem is solved using the ridge regression approximation to obtain the final network weights.

The advantage of BLS is that it can use random hidden layer weights to map the original samples to a discriminative feature space tensed by these vectors. The parameters of the hidden layer nodes can be randomly generated according to any continuous probability distribution, the rationale of which has been theoretically proven. This random mechanism can quickly provide a fast training process for BLS. It solves the serious training time consuming problem suffered by the DNNs-based models. More importantly, BLS retains the powerful mechanism of randomly generating the weights of hidden layer nodes based on any continuous probability distribution. Specifically, the system can be updated incrementally in the

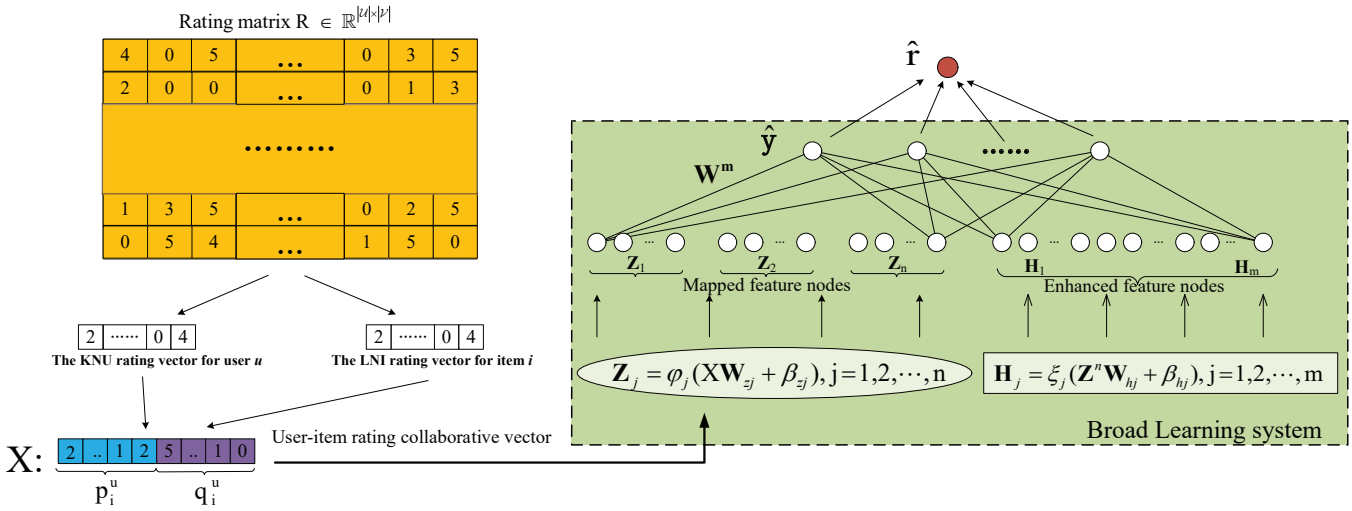


Fig. 1. Illustration of the proposed BroadCF algorithm.

face of newly added samples and hidden nodes, without the need of rebuilding the entire network from scratch. As a result, BLS requires a small number of parameters to be stored, which can solve the problem of huge storage requirements arising from the DNNs-based models. Most importantly, the feature layer to the output layer in BLS is directly connected, and the structure is very simple. Therefore, even without a large number of samples in the recommender system, prediction performance can be ensured.

4 Broad Collaborative Filtering

In this section, we will describe in detail the proposed Broad Collaborative Filtering (BroadCF) method. First of all, we will describe how to transform the original user-item rating data into the form of (Rating collaborative vector, Rating one-hot vector) for each user-item pair, which will be taken as input and output to the broad learning system. Then, we will describe the broad learning part. Finally, we will describe the rating prediction procedure and summarize the propose BroadCF method. For clarity, Figure 1 illustrates the main idea of the proposed BroadCF method.

4.1 Data Preprocessing

4.1.1 User-Item Rating Collaborative Vector

As shown in Eq. (2), for a user-item pair (u, i), the model-based recommendation methods aim to establish a mapping function from the user-item input data to the user-item rating. A simple type of user-item input data is the direct concatenation of the user rating vector $\mathbf{R}_{u,:}$ and the item rating vector $\mathbf{R}_{:,i}$. It has been widely adopted in the DNNs-based methods, i.e., the DNNs-based methods learn the latent factor vectors from the original user/item rating vectors. However, it does not work well in the case of BLS. This is because, different from the DNNs-based methods, which stack several hidden layers with decreasing layer sizes, BLS requires broadening the original input features into the mapped features and enhanced features, resulting in several-order magnitude larger dimension. In addition, directly utilizing extremely sparse user rating vector and item rating vector may cause the overfitting issue. Inspired by the previous work [28], [46], a user-item rating collaborative vector preprocessing is proposed to generate the low-dimensional user-item input data, which is able to harness quality judgments of the most similar users/items.

First of all, for each user u, a set of k nearest users (KNU) is obtained by calculating the cosine similarity between the rating vectors of user u and other users. Let $\mathbf{n}^u = [\mathbf{n}_1^u, \mathbf{n}_2^u, \cdots, \mathbf{n}_k^u]$ denote the KNU index vector of user u, i.e., the j-th nearest user of user u can be expressed as $\mathbf{n}_j^u, \forall j = 1, \cdots, k$. Then for each user-item pair composed of user u and item i, a KNU rating vector $\mathbf{p}_i^u \in \mathbb{R}^{1\times k}$ of user u on item i is obtained from the rating matrix $\mathbf{R}$ with each entry $\mathbf{p}_i^u[j]$ being defined as follows,

$$\mathbf{p}_i^u[j] = \mathbf{R}_{\mathbf{n}_j^u, i} \quad \forall j = 1, \cdots, k. \tag{3}$$

That is, $\mathbf{p}_i^u[j]$ is the rating value of user $\mathbf{n}_j^u$ to item i.

In the above procedure, it is possible that some entries $\mathbf{p}_i^u[j]$ are zeros, i.e., user $\mathbf{n}_j^u$ has not yet rated item i. In [28], the mean value of the ratings of user $\mathbf{n}_j^u$ is used to fill the missing rating, which however, treats all the items equally while failing to utilize the rating of other similar users. To this end, we propose a new strategy that fills the missing rating with the rating of the nearest user of $\mathbf{n}_j^u$ to item i, i.e.,

$$\bar{\mathbf{p}}_i^u[j] = \begin{cases} \mathbf{R}_{u^*,i} & \text{if } \mathbf{p}_i^u[j] = 0 \\ \mathbf{p}_i^u[j] & \text{otherwise} \end{cases} \tag{4}$$

where u^* denotes the nearest user of user $\mathbf{n}_j^u$ who has rated item i.

Similarly, for each item i, a set of l nearest items (LNI) can be obtained by calculating the cosine similarity between the rating vectors of item i and other items. Let $\mathbf{n}^i = [\mathbf{n}_1^i, \mathbf{n}_2^i, \cdots, \mathbf{n}_l^i]$ denote the LNI index vector of item i, i.e., the j-th nearest item of item i can be expressed as $\mathbf{n}_j^i, \forall j = 1, \cdots, l$. Then for each user-item pair composed of user u and

item i, a LNI rating vector $\mathbf{q}_i^u \in \mathbb{R}^{1\times l}$ of user u on item i is obtained from the rating matrix $\mathbf{R}$ with each entry $\mathbf{q}_i^u[j]$ being defined as follows,

$$\mathbf{q}_i^u[j] = \mathbf{R}_{u,\mathbf{n}_j^i} \quad \forall j = 1, \cdots, L. \tag{5}$$

That is, $\mathbf{q}_i^u[j]$ is the rating value of user u to item $\mathbf{n}_j^i$.

Like before, the following strategy is used to fill the missing entries

$$\bar{\mathbf{q}}_i^u[j] = \begin{cases} \mathbf{R}_{u,i^*} & \text{if } \mathbf{q}_i^u[j] = 0 \\ \mathbf{q}_i^u[j] & \text{otherwise} \end{cases} \tag{6}$$

where i^* denotes the nearest item of item $\mathbf{n}_j^i$ rated by user u.

After obtaining $\bar{\mathbf{p}}_i^u$ and $\bar{\mathbf{q}}_i^u$, they are concatenated to form the user-item rating collaborative vector, i.e.,

$$\mathbf{x}_i^u = [\bar{\mathbf{p}}_i^u | \bar{\mathbf{q}}_i^u] \in \mathbb{R}^{1\times(k+l)} \tag{7}$$

Notice that the dimension of the input data $\mathbf{x}_i^u$, i.e. $k+l$, is relatively small compared with that of the direct concatenation of the user rating vector $\mathbf{R}_{u,:}$ and the item rating vector $\mathbf{R}_{:,i}$, i.e. $|\mathcal{U}| + |\mathcal{V}|$.

4.1.2 User-Item Rating One-Hot Vector

In addition, for each user-item pair (u, i) with known rating in the training rating matrix, i.e. $\mathbf{R}_{u,i} \neq 0$, the user-item rating $r_{u,i}$ is transformed into a user-item rating one-hot vector $\mathbf{y}_i^u \in \mathbb{R}^{1\times d_y}$ with

$$\mathbf{y}_i^u[j] = \begin{cases} 1 & \text{if } j = r_{u,i} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where d_y is the maximum rating, e.g., 5.

Finally, for each user-item pair (u, i) with known rating in the training rating matrix, the user-item rating collaborative vector $\mathbf{x}_i^u$ and the user-item rating one-hot vector $\mathbf{y}_i^u$ are taken as the input and output of a training sample $(\mathbf{x}_i^u, \mathbf{y}_i^u)$. That is, the training set of the proposed BroadCF method is

$$\mathcal{D} = \{(\mathbf{x}_i^u, \mathbf{y}_i^u) | \mathbf{R}_{u,i} \neq 0\}. \tag{9}$$

4.2 Broad Learning

In this subsection, we will describe the mapping function f that maps the rating collaborative vector $\mathbf{x}_i^u$ into the predicted rating one-hot vector $\mathbf{y}_i^u$, which will be achieved by BLS. According to [30], BLS consists of three modules, namely mapped feature layer, enhanced feature layer, and output layer.

4.2.1 Mapped Feature Layer

Given the training set $\mathcal{D} = \{(\mathbf{x}_i^u, \mathbf{y}_i^u) | \mathbf{R}_{u,i} \neq 0\}$, the input matrix of BLS is formed as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{i_1}^{u_1} \\ \mathbf{x}_{i_2}^{u_2} \\ \vdots \\ \mathbf{x}_{i_{|\mathcal{D}|}}^{u_{|\mathcal{D}|}} \end{bmatrix} \in \mathbb{R}^{|\mathcal{D}|\times(k+l)} \tag{10}$$

Algorithm 1 Broad Collaborative Filtering (BroadCF)

Training part:

1: **Input:** Training rating matrix $\mathbf{R}$; number of nearest users k; number of nearest items l; number of mapped feature groups n; mapped feature dimension d_z; number of enhanced feature groups m; enhanced feature dimension d_h.
2: Obtain the k nearest users of each user u.
3: Obtain the l nearest items of each item i.
4: **for all** training user-item pair with $\mathbf{R}_{u,i} \neq 0$ **do**
5: Generate user-item rating collaborative vector $\mathbf{x}_i^u$ via Eq. (7).
6: Generate user-item rating one-hot vector $\mathbf{y}_i^u$ via Eq. (8).
7: **end for**
8: Generate the training set $\mathcal{D}$ via Eq. (9).
9: Generate the input matrix $\mathbf{X}$ and output matrix $\mathbf{Y}$ via Eq. (10) and Eq. (11) respectively.
10: Generate the mapped feature matrices $\mathbf{Z}^n$ and enhanced feature matrices $\mathbf{H}^m$ via Eq. (13) and Eq. (15) respectively.
11: Calculate the trainable weight matrix $\mathbf{Y}$ via Eq. (17).

Output: The trainable weight matrix $\mathbf{W}$.

Testing part:

1: **Input:** A user-item pair (u, i) with unknown rating.
2: Generate user-item rating collaborative vector $\mathbf{x}_i^u$ via Eq. (7).
3: Feed $\mathbf{x}_i^u$ into the Broad Learning procedure to generate the user-item rating strength vector $\hat{\mathbf{y}}_i^u$.
4: Convert $\hat{\mathbf{y}}_i^u$ into the predicted rating $\hat{r}_i^u$ via Eq. (18).
5: **Output:** The predicted rating $\hat{r}_i^u$.

where $|\mathcal{D}|$ is the number of training samples, i.e., there are $|\mathcal{D}|$ user-item pairs $(u_1, i_1), (u_2, i_2), ..., (u_{|\mathcal{D}|}, i_{|\mathcal{D}|})$. The output matrix of BLS is formed as

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_{i_1}^{u_1} \\ \mathbf{y}_{i_2}^{u_2} \\ \vdots \\ \mathbf{y}_{i_{|\mathcal{D}|}}^{u_{|\mathcal{D}|}} \end{bmatrix} \in \mathbb{R}^{|\mathcal{D}|\times d_y}. \tag{11}$$

In the mapped feature layer, the input matrix $\mathbf{X}$ is transformed into the mapped feature matrix $\mathbf{Z}_j \in \mathbb{R}^{|\mathcal{D}|\times d_z}$ as follows

$$\mathbf{Z}_j = \phi_j(\mathbf{X}\mathbf{W}_{z_j} + \boldsymbol{\beta}_{z_j}), \quad j = 1, 2, \cdots, n \tag{12}$$

where d_z denotes the dimension of each mapped feature group, n denotes the number of the mapped feature groups, and ϕ_j is the j-th nonlinear feature mapping function. In our experiments, ReLu is used as the nonlinear feature mapping function. In the above procedure, $\mathbf{W}_{z_j} \in \mathbb{R}^{(k+l)\times d_z}$ and $\boldsymbol{\beta}_{z_j} \in \mathbb{R}^{|\mathcal{D}|\times d_z}$ are randomly generated matrices. In this way, the mapped feature matrices can be obtained, i.e.,

$$\mathbf{Z}^n = [\mathbf{Z}_1 | \mathbf{Z}_2 \cdots | \mathbf{Z}_n] \in \mathbb{R}^{|\mathcal{D}|\times n d_z}. \tag{13}$$

TABLE 2
Statistics of the seven datasets.

Datasets	#Users	#Items	#Ratings	Sparsity	Scale
ml-la	610	9,724	100,836	98.30%	[1, 5]
A-Digital-Music	1000	4,796	11,677	99.90%	[1, 5]
A-Grocery-and-Gourmet-Food	2,500	7,527	11,532	99.96%	[1, 5]
A-Patio-lawn-and-Garden	5,000	5,400	12,429	99.98%	[1, 5]
A-Automotive	5,000	6,596	12,665	99.98%	[1, 5]
A-Baby	6,000	5,428	17,532	99.98%	[1, 5]
A-Instant-Video	429,622	23,966	583,933	99.98%	[1, 5]

4.2.2 Enhanced Feature Layer

After obtaining $\mathbf{Z}^n$, the enhanced feature matrix $\mathbf{H}_j \in \mathbb{R}^{|\mathcal{D}|\times d_h}$ can be calculated as follows,

$$\mathbf{H}_j = \xi_j(\mathbf{Z}^n\mathbf{W}_{h_j} + \boldsymbol{\beta}_{h_j}), \quad j = 1, 2, \cdots, m \tag{14}$$

where d_h denotes the dimension of each enhanced feature group, m denotes the number of the enhanced feature groups, and ξ_j is the j-th nonlinear feature enhancement function. In our experiments, ReLu is used as the nonlinear feature enhancement function. In the above procedure, $\mathbf{W}_{h_j} \in \mathbb{R}^{nd_z\times d_h}$ and $\boldsymbol{\beta}_{h_j} \in \mathbb{R}^{|\mathcal{D}|\times d_h}$ are randomly generated matrices. In this way, the enhanced feature matrices can be obtained, i.e.,

$$\mathbf{H}^n = [\mathbf{H}_1|\mathbf{H}_2\cdots|\mathbf{H}_m] \in \mathbb{R}^{|\mathcal{D}|\times md_h}. \tag{15}$$

4.2.3 Output Layer

After obtaining the mapped feature matrices $\mathbf{Z}^n$ and the enhanced feature matrices $\mathbf{H}^m$, the output matrix $\mathbf{Y} \in \mathbb{R}^{|\mathcal{D}|\times d_y}$ can be predicted as

$$\mathbf{Y} = [\mathbf{Z}^n|\mathbf{H}^m]\,\mathbf{W} \tag{16}$$

where $\mathbf{W} \in \mathbb{R}^{(nd_z+md_h)\times d_y}$ is the trainable weight matrix that maps the concatenated mapped and enhanced features into the output $\mathbf{Y}$.

In the training procedure, the only trainable weight matrix $\mathbf{W}$ can be easily solved by using the ridge regression approximation of pseudoinverse $[\mathbf{Z}^n|\mathbf{H}^m]^+$, i.e.,

$$\mathbf{W} = [\mathbf{Z}^n|\mathbf{H}^m]^+\,\mathbf{Y}. \tag{17}$$

4.3 Rating Prediction and Algorithm Summary

In the testing procedure, for each user-item pair (u, i) with unknown rating, the rating can be predicted by feeding the user-item rating collaborative vector $\mathbf{x}_i^u \in \mathbb{R}^{1\times(k+l)}$ into the broad learning procedure, which outputs the predicted user-item rating strength vector $\hat{\mathbf{y}}_i^u \in \mathbb{R}^{1\times d_y}$. Finally, the predicted rating $\hat{r}_{u,i}$ of user u to item i is calculated as follows,

$$\hat{r}_{u,i} = \sum_{j=1}^{s} \frac{\hat{\mathbf{y}}_i^u[j] - \min(\hat{\mathbf{y}}_i^u)}{\max(\hat{\mathbf{y}}_i^u) - \min(\hat{\mathbf{y}}_i^u)}\hat{\mathbf{y}}_i^u[j] \tag{18}$$

where $\hat{\mathbf{y}}_i^u[j]$ denotes the j-th entry of vector $\hat{\mathbf{y}}_i^u$, and $\min(\hat{\mathbf{y}}_i^u)$ and $\max(\hat{\mathbf{y}}_i^u)$ denote the minimum and maximum values of vector $\hat{\mathbf{y}}_i^u$.

For clarity, the whole procedure of BroadCF is summarized in Algorithm 1.

5 Experiments

5.1 Experimental Settings

5.1.1 Datasets

The experiments are conducted on seven real-world publicly available datasets obtained from the following two main sources.

1) MovieLens[1]: The MovieLens dataset contains rating data of multiple movies by multiple users, and also contains movie metadata information and user attribute information. This dataset is often used as the testing dataset for evaluating the performance of a recommender system. In our experiment, the ml-latest dataset (*abbr.* ml-la) is adopted.
2) Amazon[2]: This dataset is an updated version of the Amazon review dataset for 2018 [47], which contains reviews (ratings, text, helpful polls), product metadata (descriptions, category information, price, brand and image features) and links (also view/also buy charts). In particular, six datasets, namely A-Digital-Music, A-Grocery-and-Gourmet-Food, A-Patio-Lawn-and-Garden, A-Automotive, A-Baby, and A-Instant-Video, are adopted.

The statistics of the above seven datasets are summarized in Table 2. From the table, it can be seen that one dataset (i.e. A-Instant-Video) is relatively very large. Each dataset is randomly split into the training set and testing set with ratio 3:1 for each user. A quarter of the samples in the training set are randomly selected as the validation set to tune the appropriate hyper-parameters for all the methods.

5.1.2 Evaluation Measures

We utilize the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) to evaluate the performance of the prediction results. They are calculated as follows,

$$\begin{aligned} RMSE &= \sqrt{\frac{1}{|\mathcal{T}|}\sum_{(u,i)\in\mathcal{T}}(r_{u,i} - \hat{r}_{u,i})^2} \\ MAE &= \frac{1}{|\mathcal{T}|}\sum_{(u,i)\in\mathcal{T}}|(r_{u,i} - \hat{r}_{u,i})| \end{aligned} \tag{19}$$

where $\hat{r}_{u,i}$ denotes the predicted rating value, $r_{u,i}$ is the ground-truth rating, and $\mathcal{T}$ denotes the set of the tested ratings. Smaller values of RMSE and MAE indicate the better performance. When comparing the prediction performance of BroadCF with that of baseline, the improvement percent is calculated as

$$\text{Improvement Percent} = \frac{\text{Baseline Result} - \text{BroadCF Result}}{\text{Baseline Result}} \times 100\% \tag{20}$$

That is, the error reduction percentage compared with the baseline. In addition, the running cost, including training time, testing time, and trainable parameter storage cost is also used as evaluation criteria.

1. https://grouplens.org/datasets/movielens/
2. http://jmcauley.ucsd.edu/data/amazon/

TABLE 3
The comparison results on prediction performance: The mean value of RMSE obtained by running each algorithm five times on each dataset.

Methods	ml-la	A-Digital-Music	A-Grocery-and-Gourmet-Food	A-Patio-Lawn-and-Garden	A-Automotive	A-Baby	A-Instant-Video
PMF	1.0016	2.2170	3.2144	3.0658	3.2621	2.6281	NA
NeuMF	1.9113	2.8638	2.9396	2.8667	2.7982	2.9844	1.2749
DMF	1.6325	3.6161	4.1642	4.1030	4.1488	3.2546	NA
DeepCF	1.9029	2.1101	2.2909	2.0957	2.1786	1.8420	NA
BPAM	0.8484	1.2509	1.7281	1.9805	1.9805	0.8678	2.4521
BroadCF	0.7982	0.7291	0.7422	0.8973	0.7720	0.8025	0.6364

TABLE 4
The comparison results on prediction performance: The mean value of MAE obtained by running each algorithm five times on each dataset.

Methods	ml-la	A-Digital-Music	A-Grocery-and-Gourmet-Food	A-Patio-Lawn-and-Garden	A-Automotive	A-Baby	A-Instant-Video
PMF	0.7525	1.6670	2.9038	2.6038	2.8330	2.1097	NA
NeuMF	1.5190	1.9742	2.0531	2.0718	1.9759	2.2517	1.0138
DMF	1.2297	3.4477	3.9433	3.8417	3.9055	3.8351	NA
DeepCF	1.5995	2.2104	2.0480	1.9768	1.9848	1.8652	NA
BPAM	0.6570	1.5800	1.2425	1.5440	0.6790	1.4900	1.9529
BroadCF	0.4822	0.3176	0.2414	0.3282	0.2542	0.2802	0.1961

5.2 Comparison Results

5.2.1 Baselines

We compare the proposed BroadCF approach with the following five state-of-the-art approaches.

- **PMF**[3] [36] is a classical CF approach that considers only latent factors and uses matrix decomposition to capture the linear relationship between users and items.
- **DMF**[4] [24] uses a dual-path neural network instead of the linear embedding operation used in ordinary matrix decomposition.
- **NeuMF**[5] [26] takes the connection of user embedding and item embedding as the input and uses the Multi-Layer Perceptron (MLP) model for prediction.
- **DeepCF**[6] [29] integrates a unified deep collaborative filtering framework with representation learning and matching function learning.
- **BPAM**[7] [28] is a neighborhood-based CF recommendation framework that captures the global influence of a target users nearest users on their nearest target set of users by introducing a global attention matrix.

The five state-of-the-art methods are tuned using the parameters suggested by the authors. For the PMF model, the number of factors for both users and items is set to 30. The regularization parameters for users and items are set to 0.001 and 0.0001, respectively. For the NeuMF model, the number of predictors for the generalized matrix decomposition part and the embedding sizes for users and items are set to 8 and 16, respectively. For the DMF model, the number of predictors for both users and items is set to 64. For the DeepCF model, the size of the MLP layer representing the learning part is set to [512, 256, 128, 64], and the same settings as DMF are used in the matching function learning part. For these deep models, i.e., DMF, NeuMF, and DeepCF, the batch size is set to 256. For the BPAM model, we adjust the number of nearest users and items in [3, 5, 7, 9]. For the BroadCF algorithm, we set the number of both nearest users and nearest items as 5, i.e., $k = l = 5$, the number of both mapped feature groups and enhanced feature groups as 25, i.e., $n = m = 25$, and the mapped feature dimension and the enhanced feature dimension as 10 and 15 respectively, i.e. $d_z = 10, d_h = 15$. All experiments are implemented in Python and run on a server with an Intel Core i9-10900 CPU, GeForce RTX 3090, and 256GB of RAM. For the DNNs-based methods, including DMF, NeuMF and DeepCF, GPUs are used, while for the other methods, including PMF, BPAM and BroadCF, no GPU is used. The code of BroadCF is publicly available at https://github.com/BroadRS/BroadCF.

3. https://github.com/xuChenSJTU/PMF
4. https://github.com/RuidongZ/Deep_Matrix_Factorization_Models
5. https://github.com/ZJ-Tronker/Neural_Collaborative_Filtering-1
6. https://github.com/familyld/DeepCF
7. https://github.com/xiwd/BPAM

5.2.2 Comparison on Prediction Performance

The comparison results on accuracy in terms of RMSE and MAE are shown in Table 3 and Table 4 respectively. In this experiment, we run each algorithm 5 times on each dataset and report the average performance over the 5 runs. Out-of-memory errors occur when training the PMF, DMF and DeepCF models on the largest dataset, i.e. A-Instant-Video. This is because the three models require the user-item rating matrix as input data for the models, but for the A-Instant-Video dataset, the entire rating matrix takes up to $|\mathcal{U}| \cdot |\mathcal{V}| \cdot 32$ bits of memory, which is approximately equal to 308.64 GB, exceeding the 256 GB memory of the experimental machine. As a result, no result is reported in the corresponding entries of the comparison result tables (i.e. NA). According to the two tables, we have the following key observations.

Overall, the RMSE and MAE values obtained by the proposed BroadCF algorithm are much smaller than those by other algorithms, including the classical CF method (PMF), three DNNs-based algorithms (DMF, NeuMF and DeepCF), and one BP neural network-based method (BPAM). In particular, compared with PMF, the proposed BroadCF algorithm achieves at least 20.31% and 35.92% improvements in terms of

TABLE 5
The comparison results on running cost: The training time (in seconds) consumed by each algorithm on each dataset.

Methods	ml-la	A-Digital-Music	A-Grocery-and-Gourmet-Food	A-Patio-Lawn-and-Garden	A-Automotive	A-Baby	A-Instant-Video
PMF	98.24	202.92	1132.87	4072.20	4311.29	1100.17	NA
NeuMF	6013.53	109.42	75.38	68.71	80.47	101.84	48219.60
DMF	2462.42	62.70	85.75	123.25	104.01	209.50	NA
DeepCF	2059.59	202.80	306.60	362.40	366.40	501.00	NA
BPAM	159.36	128.72	143.30	125.70	130.27	137.80	11416.20
BroadCF	38.00	2.60	2.49	3.37	5.08	7.06	123.22

TABLE 6
The comparison results on running cost: The testing time (in seconds) consumed by each algorithm on each dataset.

Methods	ml-la	A-Digital-Music	A-Grocery-and-Gourmet-Food	A-Patio-Lawn-and-Garden	A-Automotive	A-Baby	A-Instant-Video
PMF	0.03	0.01	0.01	0.01	0.01	0.01	NA
NeuMF	301.40	118.53	189.72	331.24	464.19	450.14	10825.30
DMF	21.11	7.95	29.69	95.61	73.82	126.90	NA
DeepCF	427.09	67.44	102.65	120.89	94.47	171.50	NA
BPAM	0.76	0.11	0.13	0.20	0.19	0.28	7.13
BroadCF	1.18	0.08	0.08	0.10	0.14	0.19	6.01

TABLE 7
The comparison results on running cost: The number of trainable parameters to be stored during the training procedure of each algorithm on each dataset.

	ml-la	A-Digital-Music	A-Grocery-and-Gourmet-Food	A-Patio-Lawn-and-Garden	A-Automotive	A-Baby	A-Instant-Video
PMF	310,080	173,910	300,840	312,030	347,910	342,900	NA
NeuMF	392,601	234,521	403,761	418,681	466,521	459,801	18,038,161
DMF	12,327,168	33,519,232	40,030,464	48,012,672	20,921,728	29,942,272	NA
DeepCF	39,266,060	97,444,766	115,947,300	137,582,762	62,738,578	86,977,048	NA
BPAM	43,615	31,500	78,750	157,500	157,500	189,000	7,518,367
BroadCF	3,125	3,125	3,125	3,125	3,125	3,125	3,125

RMSE and MAE, which is a relatively significant performance improvement. The main reason is that PMF only captures the linear latent factors, i.e. the linear relationship between users and items, while BroadCF is able to capture the nonlinear relationship between users and items. Compared with the three DNNs-based algorithms, BroadCF obtains at least 50.08% and 60.79% improvements in terms of RMSE and MAE, which is a more significant performance improvement. The main reason may be that, although both the DNNs-based algorithms and BroadCF are able to capture the nonlinear relationship between users and items, the DNNs-based algorithms easily suffer from the overfitting issue, i.e., given the very large number of trainable parameters, the testing performance cannot be improved compared with the training performance. As a contrast, compared with BPAM, which is a lightweight neural network based method, the performance of BroadCF is still better than that of BPAM, although the performance improvement is not such significant. As will be shown in the comparison on running cost, the number of trainable parameters in BPAM is relatively small compared with those of the DNNs-based algorithms, although still larger than that of BroadCF. Overall, the comparison results in terms of prediction performance have confirmed the effectiveness of the proposed BroadCF algorithm.

5.2.3 Comparison on Running Cost

Apart from the prediction performance, the proposed BroadCF algorithm has superiority on running cost. To this end, this subsection compares the running cost of BroadCF with the baselines, including the training time, testing time, and trainable parameter storage cost.

Table 5 and Table 6 report the training time and the testing time (in seconds) consumed by each algorithm on each dataset. According to the two tables, we can see that the proposed BroadCF algorithm is very efficient in terms of running time. In particular, from Table 5, BroadCF consumes several-magnitude shorter running time compared with the baselines. On the ml-la dataset, BroadCF is twice faster than the fastest baseline namely PMF, and it consumes about 23% running time by BPAM. Most importantly, compared with the three DNNs-based models, BroadCF consumes less than 2% running time on the ml-la dataset. On the six Amazon datasets, the four neural network based baselines namely NeuMF, DMF, DeepCF and BPAM, are faster than PMF. This is mainly due to that the neural network models take the rating vectors, which is relatively efficient in the case of relatively sparse dataset with small number of ratings. However, BroadCF is still much faster than the four neural network based baselines. This is mainly due to that the optimization problem of BroadCF is solved by using the ridge regression approximation of pseudoinverse without need of iterative training process. From Table 6, we can see that although BroadCF is not the fastest algorithm in terms of testing time, it is very close to the fastest baseline namely PMF. This is mainly due to that, BroadCF requires calculating the mapped features and enhanced features, in addition to the multiplication by the trainable weight matrix, while PMF only needs to calculate the matching score of the user/item latent factors. Nevertheless, BroadCF is still much

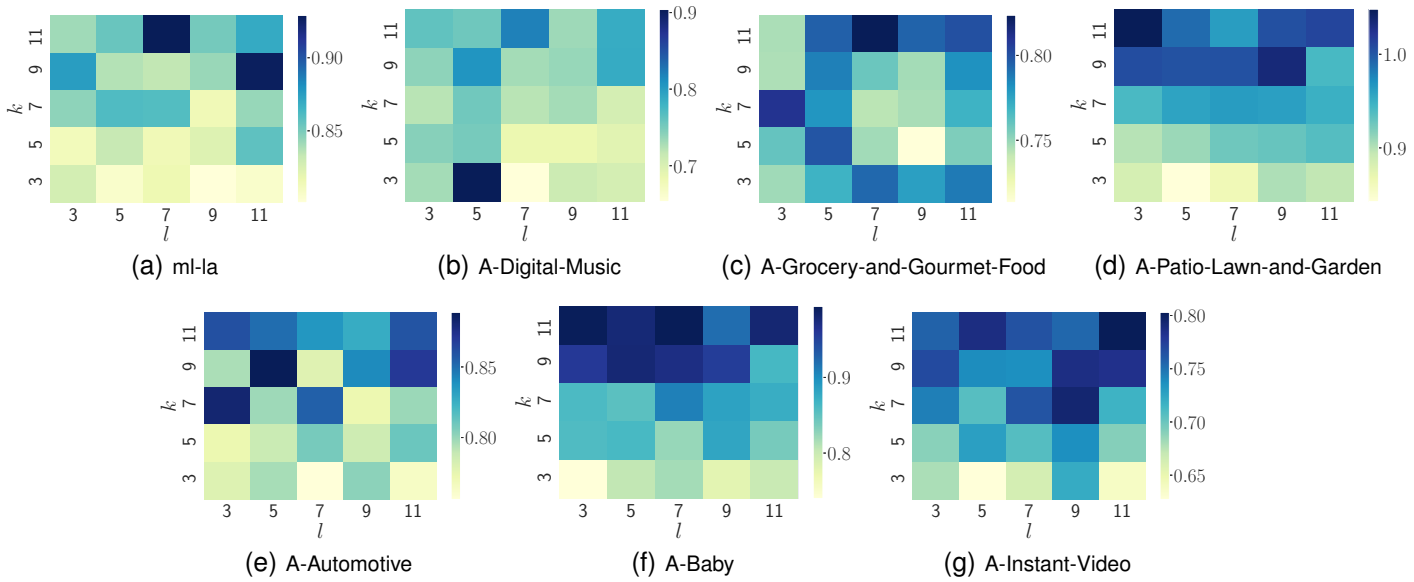


Fig. 2. Hyper-parameter analysis: The RMSE values obtained by BroadCF with varying number of nearest users k and number of nearest items l.

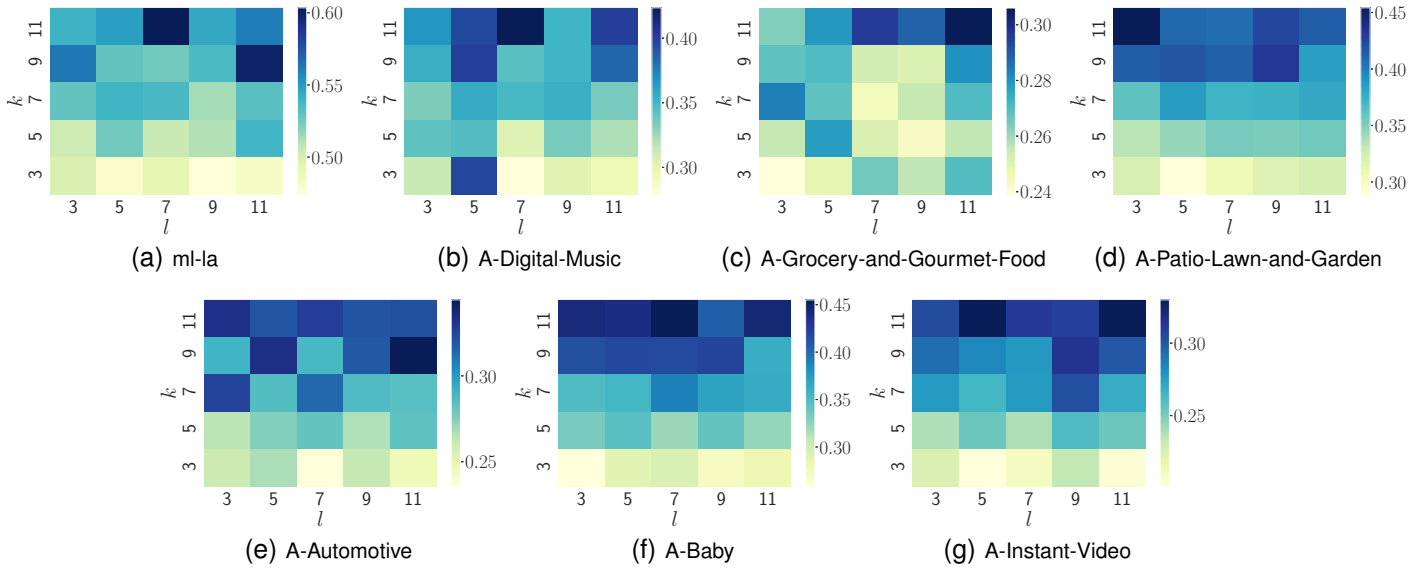


Fig. 3. Hyper-parameter analysis: The MAE values obtained by BroadCF with varying number of nearest users k and number of nearest items l.

faster than the four neural network based baselines namely NeuMF, DMF, DeepCF and BPAM. The above comparison results on training time and testing time have confirmed the efficiency of the proposed BroadCF algorithm from the perspective of running time.

We also demonstrate the efficiency of the proposed BroadCF algorithm from the perspective of storage requirements. Following [46], Table 7 reports the number of trainable parameters to be stored during the training procedure of each algorithm on each dataset. From the table, we can see that, the proposed BroadCF algorithm is several-magnitude more efficient than all the baselines. This is mainly due to that the only trainable parameters to be stored in BroadCF are the weight matrix $\mathbf{W} \in \mathbb{R}^{(nd_z+md_h)\times d_y}$, which is independent of the input data size and dimension. On the contrary, all the baselines need to store much more trainable parameters, especially for the DNNs-based models. This result also confirms the fact that BroadCF requires several-magnitude less training parameters, even less than the linear model namely PMF. Despite the relatively small number of trainable parameters, BroadCF can still well capture the nonlinear relationship between users and items and therefore generates much better results as shown in the prediction performance comparison. This is mainly due to the advantage achieved by mapping the original features into the mapped features and then into the enhanced features via random matrices plus nonlinear activation functions, which can be regarded as learning nonlinear representations from

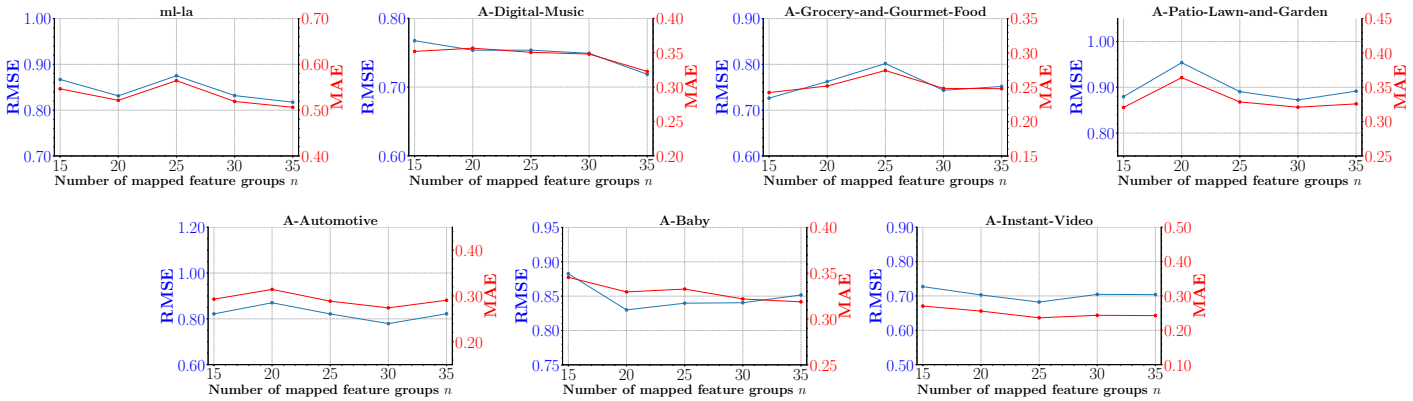


Fig. 4. Hyper-parameter analysis: The RMSE and MAE values obtained by BroadCF with different number of mapped feature groups n.

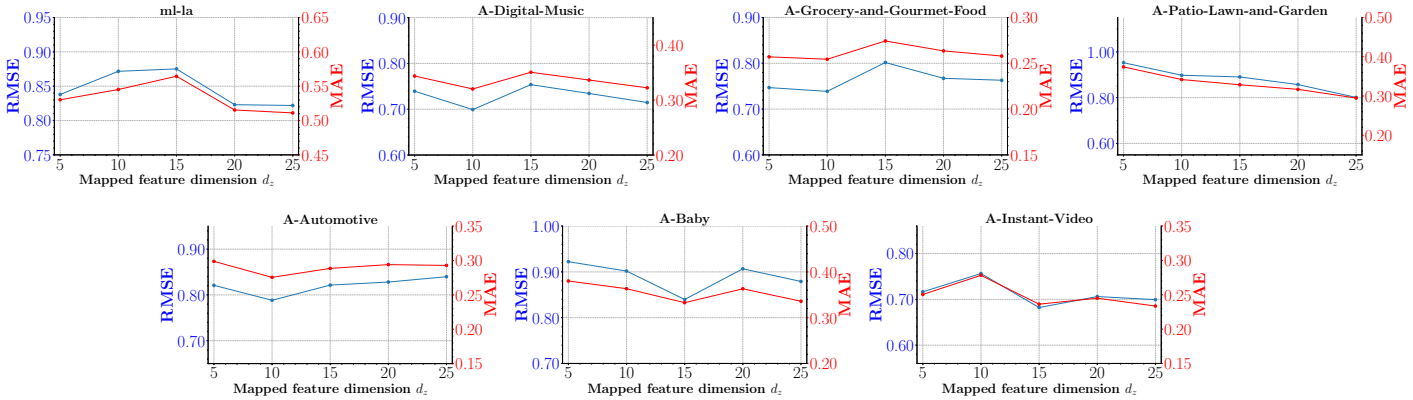


Fig. 5. Hyper-parameter analysis: The RMSE and MAE values obtained by BroadCF with different mapped feature dimension d_z.

different views. Overall, the above comparison results on running cost have confirmed the efficiency of the proposed BroadCF algorithm from the perspective of trainable parameter storage requirement.

5.3 Hyper-Parameter Analysis

In this section, we will analyze the impact of six hyper-parameters on the performance of the proposed BroadCF method, i.e., number of nearest users k, number of nearest items l, number of mapped feature groups n, mapped feature dimension d_z, number of enhanced feature groups m, and enhanced feature dimension d_h. When analyzing one parameter, the other parameters are fixed.

5.3.1 Number of Nearest Users/Items

First of all, we will analyze the impact of the number of nearest users/items, namely k and l, on the performance of the proposed BroadCF method. To this end, we run BroadCF by setting k and l in the range of [3, 5, 7, 9, 11], and report the prediction performance in terms of RMSE and MAE in Fig. 2 and Fig. 3 respectively. From the figure, we can observe that a relatively stable performance is achieved when using different k and l on most datasets. That is, the variances of the RMSE and MAE values in the two figures are relatively small as shown in the heatmaps. However, a general trend is that when increasing k and l, the performance would slightly degenerate expect on the ml-la dataset. This is mainly due to the long-tail distribution, i.e., most of users/items have very few ratings. Setting a relatively large number of nearest users/items would mistakenly introduce more noises, i.e., filling the missing entries by means of the nearest rating via Eq. (4) and Eq. (6). And the exception on the ml-la dataset has confirmed this analysis. That is, ml-la is a relatively dense dataset, in which the users/items contain more ratings. Considering the trade-off between the running efficiency and the accuracy, we set k and l to 5 in the experiments.

5.3.2 Hyper-Parameters in Feature Mapping

Secondly, we will analyze the impact of the hyper-parameters in feature mapping module, including the number of mapped feature groups n and the mapped feature dimension d_z. In the proposed BroadCF algorithm, the feature mapping module plays the role of mapping the original input matrix composed of user-item collaborative vectors into the mapped feature matrix, which is the first step of capturing the nonlinear relationship between users and items. The two hyper-parameters would affect the performance of this mapping capability. To this end, we run BroadCF by setting n and d_z in the

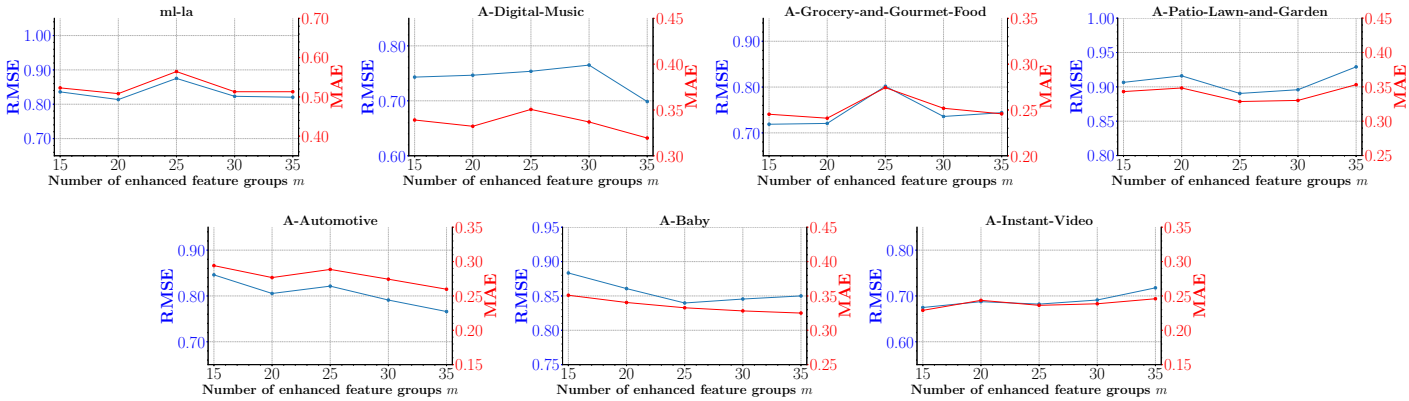


Fig. 6. Hyper-parameter analysis: The RMSE and MAE values obtained by BroadCF with different number of enhanced feature groups m.

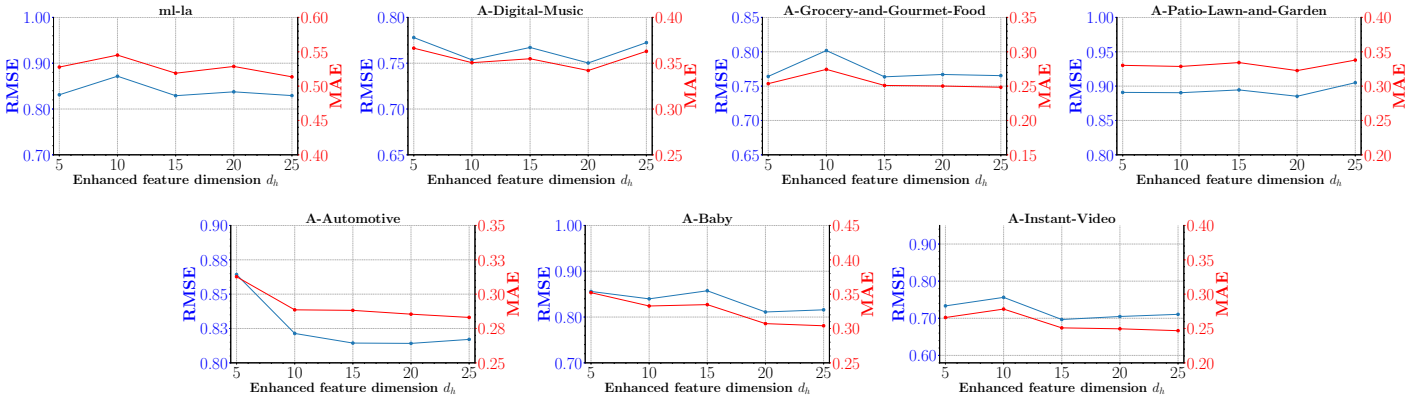


Fig. 7. Hyper-parameter analysis: The RMSE and MAE values obtained by BroadCF with different enhanced feature dimension d_h.

ranges of [15,20,25,30,35] and [5,10,15,20,25] respectively, and report the RMSE and MAE values as a function of n and d_z in Fig. 4 and Fig. 5 respectively. From the two figures, although the best values of n and d_z vary from one dataset to another, the performance variation is relatively small, i.e. smaller than 0.01. This confirms the relatively insensitivity of the proposed BroadCF algorithm to the hyper-parameters in feature mapping. And in our experiments, we set n and d_z to 25 and 10 respectively.

5.3.3 Hyper-Parameters in Feature Enhancement

Similar to the previous subsection, we will also analyze the impact of the hyper-parameters in feature enhancement module, including the number of enhanced feature groups m and the enhanced feature dimension d_h. In the proposed BroadCF algorithm, the feature enhancement module plays the role of mapping the mapped feature matrices into the enhanced feature matrix, which is the second step of capturing the nonlinear relationship between users and items. The two hyper-parameters would affect the performance of this enhancement capability. Similarly, we run BroadCF by setting m and d_h in the ranges of [15,20,25,30,35] and [5,10,15,20,25] respectively, and report the RMSE and MAE values as a function of m and d_h in Fig. 6 and Fig. 7 respectively. From the two figures, although the best values of m and d_h vary from one dataset to another, the performance variation is relatively small, i.e. smaller than 0.01. This confirms the relatively insensitivity of the proposed BroadCF algorithm to the hyper-parameters in feature enhancement. And in our experiments, we set m and d_h to 25 and 15 respectively.

6 Conclusions

In this paper, we have proposed a new neural network based recommender system called Broad Collaborative Filtering (BroadCF). Compared with the existing DNNs-based recommendation algorithms, the proposed BroadCF algorithm is also able to capture the nonlinear relationship between users and items, and hence generate very satisfactory recommendation performance. However, the superiority of BroadCF is that it is very efficient compared with the DNNs-based methods, i.e., it consumes relatively very short training time and only requires relatively small amount of data storage, in particular trainable parameter storage. The main advantage lies in designing a data preprocessing procedure to convert the original rating data into the form of (Rating collaborative vector, Rating one-hot vector), which is then feed into the very efficient BLS for rating prediction. Extensive experiments conducted on seven benchmark datasets have confirmed the superiority

of the proposed model, including both prediction performance and running cost.

ACKNOWLEDGMENT

This work was supported by NSFC under grants 62106079 and 61876193, Natural Science Foundation of Guangdong Province under grant 2020A1515110337, Guangdong Province Key Laboratory of Computational Science at the Sun Yat-sen University under grant 2020B1212060032, and Key Platforms and Scientific Research Projects in Universities in Guangdong Province under grant 2020KCXTD053.

REFERENCES

[1] C.-D. Wang, Z.-H. Deng, J.-H. Lai, and P. S. Yu, "Serendipitous recommendation in e-commerce using innovator-based collaborative filtering," *IEEE Trans. Cybernetics*, vol. 49, no. 7, pp. 2678–2692, 2019.

[2] J. Ma, J. Wen, M. Zhong, W. Chen, and X. Li, "MMM: multi-source multi-net micro-video recommendation with clustered hidden item representation learning," *Data Sci. Eng.*, vol. 4, no. 3, pp. 240–253, 2019.

[3] Q.-Y. Hu, L. Huang, C.-D. Wang, and H.-Y. Chao, "Item orientated recommendation by multi-view intact space learning with overlapping," *Knowledge-Based Systems*, pp. 358 – 370, 2019.

[4] P. Hu, R. Du, Y. Hu, and N. Li, "Hybrid item-item recommendation via semi-parametric embedding," in *IJCAI*, 2019, pp. 2521–2527.

[5] C.-C. Hsu, M.-Y. Yeh, and S.-D. Lin, "A general framework for implicit and explicit social recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2228–2241, 2018.

[6] C. Xu, P. Zhao, Y. Liu, J. Xu, V. S. Sheng, Z. Cui, X. Zhou, and H. Xiong, "Recurrent convolutional neural network for sequential recommendation," in *WWW*, 2019, pp. 3398–3404.

[7] O. Barkan, N. Koenigstein, E. Yogev, and O. Katz, "CB2CF: a neural multiview content-to-collaborative filtering model for completely cold item recommendations," in *RecSys*, 2019, pp. 228–236.

[8] W. Fu, Z. Peng, S. Wang, Y. Xu, and J. Li, "Deeply fusing reviews and contents for cold start users in cross-domain recommendation systems," in *AAAI*, 2019, pp. 94–101.

[9] A. Ferraro, "Music cold-start and long-tail recommendation: bias in deep representations," in *RecSys*, 2019, pp. 586–590.

[10] J. Chen, W. Chen, J. Huang, J. Fang, Z. Li, A. Liu, and L. Zhao, "Co-purchaser recommendation for online group buying," *Data Sci. Eng.*, vol. 5, no. 3, pp. 280–292, 2020.

[11] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T.-S. Chua, "Outer product-based neural collaborative filtering," in *IJCAI*, 2018, pp. 2227–2233.

[12] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," *SIGIR Forum*, vol. 51, no. 2, pp. 227–234, 2017.

[13] J. B. Schafer, D. Frankowski, J. L. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web, Methods and Strategies of Web Personalization*, 2007, pp. 291–324.

[14] D.-K. Chae, J.-S. Kang, S.-W. Kim, and J. Choi, "Rating augmentation with generative adversarial networks towards accurate collaborative filtering," in *WWW*, 2019, pp. 2616–2622.

[15] H. Shin, S. Kim, J. Shin, and X. Xiao, "Privacy enhanced matrix factorization for recommendation with local differential privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1770–1782, 2018.

[16] R. Du, J. Lu, and H. Cai, "Double regularization matrix factorization recommendation algorithm," *IEEE Access*, vol. 7, pp. 139 668–139 677, 2019.

[17] C. Chen, Z. Liu, P. Zhao, J. Zhou, and X. Li, "Privacy preserving point-of-interest recommendation using decentralized matrix factorization," in *AAAI*, 2018, pp. 257–264.

[18] M. Vlachos, C. Dünner, R. Heckel, V. G. Vassiliadis, T. P. Parnell, and K. Atasu, "Addressing interpretability and cold-start in matrix factorization for recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 7, pp. 1253–1266, 2019.

[19] Y. He, C. Wang, and C. Jiang, "Correlated matrix factorization for recommendation with implicit feedback," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 3, pp. 451–464, 2019.

[20] L. Hu, L. Cao, J. Cao, Z. Gu, G. Xu, and D. Yang, "Learning informative priors from heterogeneous domains to improve recommendation in cold-start user domains," *ACM Trans. Inf. Syst.*, pp. 13:1–13:37, 2016.

[21] Y. Shi, M. A. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 3:1–3:45, 2014.

[22] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *KDD*, 2008, pp. 426–434.

[23] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu, "Heterogeneous information network embedding for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 357–370, 2019.

[24] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems." in *IJCAI*, 2017, pp. 3203–3209.

[25] W. Yan, D. Wang, M. Cao, and J. Liu, "Deep auto encoder model with convolutional text networks for video recommendation," *IEEE Access*, vol. 7, pp. 40 333–40 346, 2019.

[26] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*, 2017, pp. 173–182.

[27] H. Liu, F. Wu, W. Wang, X. Wang, P. Jiao, C. Wu, and X. Xie, "NRPA: neural recommendation with personalized attention," in *SIGIR*, 2019, pp. 1233–1236.

[28] W.-D. Xi, L. Huang, C.-D. Wang, Y.-Y. Zheng, and J. Lai, "BPAM: recommendation based on BP neural network with attention mechanism," in *IJCAI*, 2019, pp. 3905–3911.

[29] Z.-H. Deng, L. Huang, C.-D. Wang, J.-H. Lai, and P. S.Yu, "DeepCF: A unified framework of representation learning and matching function learning in recommender system," in *AAAI*, 2019, pp. 61–68.

[30] C. L. P. Chen, Z. Liu, and F. Shuang, "Universal approximation capability of broad learning system and its structural variations," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 4, pp. 1191–1204, 2019.

[31] X. Gong, T. Zhang, C. L. P. Chen, and Z. Liu, "Research review for broad learning system:algorithms, theory, and applications," *IEEE Trans. Cybern.*, 2021.

[32] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[33] R. M. Bell and Y. Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," in *ICDM*, 2007, pp. 43–52.

[34] R. Jia, M. Jin, and C. Liu, "Using temporal information to improve predictive accuracy of collaborative filtering algorithms," in *APWeb*, 2010, pp. 301–306.

[35] B. K. Patra, R. Launonen, V. Ollikainen, and S. Nandi, "A new similarity measure using Bhattacharyya coefficient for collaborative filtering in sparse data," *Knowledge-Based Systems*, pp. 163–177, 2015.

[36] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *NIPS*, 2008, pp. 1257–1264.

[37] C. C. Krueger, *The Impact of the Internet on Business Model Evolution within the News and Music Sectors*. Citeseer, 2006.

[38] Z. Zheng, L. Xiaoli, M. Tang, F. Xie, and M. R. Lyu, "Web service QoS prediction via collaborative filtering: A survey," *IEEE Trans. Serv. Comput.*, 2020.

[39] F. Xue, X. He, X. Wang, J. Xu, K. Liu, and R. Hong, "Deep item-based collaborative filtering for top-n recommendation," *ACM Trans. Inf. Syst.*, vol. 37, no. 3, pp. 33:1–33:25, 2019.

[40] L. Wu, X. He, X. Wang, K. Zhang, and M. Wang, "A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation," *IEEE Trans. Knowl. Data Eng.*, 2022.

[41] B. Sheng, P. Li, Y. Zhang, L. Mao, and C. L. P. Chen, "GreenSea: Visual soccer analysis using broad learning system," *IEEE Trans. Cybern.*, vol. 51, no. 3, pp. 1463–1477, 2021.

[42] J. Du, C.-M. Vong, and C. L. P. Chen, "Novel efficient RNN and lstm-like architectures: Recurrent and gated broad learning systems and their applications for text classification," *IEEE Trans. Cybern.*, vol. 51, no. 3, pp. 1586–1597, 2021.

[43] C. L. P. Chen and B. Wang, "Random-positioned license plate recognition using hybrid broad learning system and convolutional networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 1, pp. 444–456, 2022.

[44] J. Guo, Z. Liu, C. L. P. Chen, T. Zhang, L. Wang, and K. Fan, "An efficient inspection system based on broad learning: Nondestructively estimating cement compressive strength with internal factors," *IEEE Trans. Ind. Informatics*, vol. 18, no. 6, pp. 3787–3798, 2022.

[45] Y.-H. Pao and Y. Takefuji, "Functional-link net computing: Theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, 1992.

[46] C.-D. Wang, W.-D. Xi, L. Huang, Y.-Y. Zheng, Z.-Y. Hu, and J.-H. Lai, "A bp neural network based recommender framework with attention mechanism," *IEEE Trans. Knowl. Data Eng.*, 2020.

[47] R. He and J. J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *WWW*, 2016, pp. 507–517.

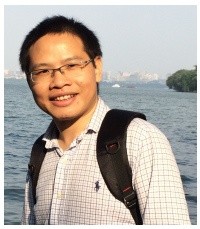
Ling Huang received her Ph.D. degree in computer science in 2020 from Sun Yat-sen University, Guangzhou. She joined South China Agricultural University in 2020 as an associate professor. She has published over 10 papers in international journals and conferences such as IEEE TCYB, IEEE TKDE, IEEE TNNLS, IEEE TII, ACM TKDD, IEEE/ACM TCBB, Pattern Recognition, KDD, AAAI, IJCAI and ICDM. Her research interest is data mining.

Can-Rong Guan received his B.E. degree in computer science in 2021 from South China Agricultural University. He is currently working toward the Master degree at South China Agricultural University. His research interest is data mining.

Zhen-Wei Huang received his B.E. degree in computer science in 2021 from South China Agricultural University. He is currently working toward the Master degree at South China Agricultural University. His research interest is data mining.

Yuefang Gao received the Ph.D. degree in computer science in 2009 from South China University of Technology, Guangzhou, China. She is a visiting scholar at the University of Sydney from March 2016 to April 2017. She joined South China Agricultural University in 2003, where she is currently an associate professor with School of Mathematics and Informatics. Her current research interests include computer vision and machine learning. She has published over 10 scientific papers in international journals and conferences such as TCSVT, Pattern Recognition and ACM MM.

Yingjie Kuang received his M.S. degree in Computer Software and Theory from South China Normal University in 2005. He joined South China Agricultural University in 2005, where he is currently an assistant professor with School of Mathematics and Informatics. His research interest is data mining and its applications.



Chang-Dong Wang received the Ph.D. degree in computer science in 2013 from Sun Yat-sen University, Guangzhou, China. He joined Sun Yat-sen University in 2013, where he is currently an associate professor with School of Computer Science and Engineering. His current research interests include machine learning and data mining. He has published over 70 scientific papers in international journals and conferences such as IEEE TPAMI, IEEE TCYB, IEEE TKDE, IEEE TNNLS, KDD, AAAI and IJCAI. His ICDM 2010 paper won the Honorable Mention for Best Research Paper Awards. He won 2012 Microsoft Research Fellowship Nomination Award. He was awarded 2015 Chinese Association for Artificial Intelligence (CAAI) Outstanding Dissertation. He is an Associate Editor in Journal of Artificial Intelligence Research (JAIR).

C. L. Philip Chen (Fellow, IEEE) received the M.S. degree in electrical engineering from the University of Michigan at Ann Arbor, Ann Arbor, MI, USA, in 1985, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1988.

He is currently a Chair Professor and the Dean of the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. Being a Program Evaluator of the Accreditation Board of Engineering and Technology Education in the USA, for computer engineering, electrical engineering, and software engineering programs, he successfully architects the University of Macaus Engineering and Computer Science programs receiving accreditations from Washington/Seoul Accord through the Hong Kong Institute of Engineers (HKIE), of which is considered as his utmost contribution in engineering/computer science education for Macau as the former Dean of the Faculty of Science and Technology. His current research interests include systems, cybernetics, and computational intelligence.

Dr. Chen received the IEEE Norbert Wiener Award in 2018 for his contribution in systems and cybernetics, and machine learnings. He is also a 2018 Highly Cited Researcher in Computer Science by Clarivate Analytics. He was a recipient of the 2016 Outstanding Electrical and Computer Engineers Award from his alma mater, Purdue University. He was the IEEE Systems, Man, and Cybernetics Society President from 2012 to 2013. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, and the IEEE TRANSACTIONS ON CYBERNETICS. He is currently a Vice President of the Chinese Association of Automation (CAA). He is a member of the Academia Europaea, the European Academy of Sciences and Arts, and the International Academy of Systems and Cybernetics Science. He is a Fellow of IEEE, AAAS, IAPR, CAA, and HKIE.