Computing and Informatics, Vol. 41, 2022, 757–787, doi: 10.31577/cai_2022_3_757

NEW STRATEGY BASED ON RBF NETWORK TO DEVELOP A COLLABORATIVE FILTERING RECOMMENDER SYSTEM

Maryam Mohammadi, Somaye Arabi Naree
Mohammad-Ali Naseri

Faculty of Mathematical Sciences and Computer
Kharazmi University
50 Taleghani Avenue, 1561836314 Tehran, Iran
e-mail: {m.mohammadi, s.arabi}@khu.ac.ir, ma.naseri.uk@gmail.com

Abstract. Collaborative filtering is a popular recommendation algorithm. It predicts user's interests according to the ratings or behaviour of other users in the system. However, the collaborative filtering recommender system suffers from several major limitations including scalability, sparsity, and cold start. In this paper, a collaborative filtering recommendation approach using radial basis function (RBF) network and power method is proposed. The proposed system has offline and online phases. In the offline phase, the sparse user-item rating matrix is completed by using RBF network based on Cover's theorem on the separability of patterns. RBF network learning is done by unsupervised kernel-based fuzzy c-means clustering algorithm for selecting RBF centers, and supervised gradient descend method for selecting RBF weights. In the offline phase, we predict non-rated items of a user. Then the full rating matrix is used to rank all the users. The ranking is done by solving an eigenvalue problem. This paper overcomes the scalability problem by clustering the users, the sparsity problem by completing the sparse rating matrix, and the new user cold start problem by recommending the top rated items of the high-ranked user. The results of the experiments, on the benchmark data sets, show that the proposed system produces high quality recommendation, in terms of accuracy and quality.

Keywords: Recommender systems, collaborative filtering, radial basis function network, power method

1 INTRODUCTION

Recommender systems provide personalized recommendations suiting users' taste by analyzing patterns of their interest in products [1, 2]. In collaborative filtering (CF) recommender system, users who have rated the same items similarly, are considered to have similar preferences [3, 4]. Most CF algorithms are based on either nearest neighbors or matrix factorization [5, 6, 7, 8]. Despite its widespread adoption, CF recommender system suffers from several major limitations including scalability, data sparsity, and cold start [9, 10, 11].

Scalability. Since classical CF recommender systems need to calculate the pairwise similarities among users or items, time complexity of computing similarities increases exponentially with the number of users and the number of items. Given the growing amount of users and items available in E-commerce systems, traditional collaborative filtering algorithms suffer seriously from scalability problems.

Data Sparsity. A modern E-commerce referral system can involve millions of users and items. Even for a very active user, there may be a relatively low proportion of items in electronic commerce systems. Even highly popular items are rated by only a small portion of existing users. Because of the sparsity of available user activity records, it is difficult for CF recommender systems to discover similar users or similar items according to their rating behaviors. Hence the sparsity problem leads to low-correlated users, which results in poor recommendations [12, 13].

Cold Start Problem. Since the CF process does not require extra data on the users or the items, it is qualified of recommending an item without comprehending the item itself [11]. Nevertheless, this advantage leads to the so-called "cold start" problem. The never users are those who have not rated many items, so it is difficult to find other users with similar fondness. Also, newer items are those which have not been rated by many users. Therefore it is challenging to recommend them to anyone.

In the literature of the recommender system, we will see with the rise of the internet, the number of products and contents available to consumers has greatly increased compared to traditional consumer-facing distribution channels such as brick-and-mortar stores, newspapers, etc. Internet services and websites such as E-commerce stores can carry a much more extensive selection of results than the aforementioned physical distribution channels due to restrictions set by physical storage spaces, logistics, and inventory holding prices [14]. However, Schwartz reasons that the raised freedom for what is available for personal consumers can be overwhelming and direct to a mental burden due to the option cost and regret felt after making a wrong choice among the alternatives consumers [15].

In the current years, recommender systems help users find relevant choices among all items, thus alleviating the data overload experienced by users generated by excessive alternatives. According to the definition by Ricci: Recommender systems are software mechanisms and techniques which provide recommendations to users about items that could be of curiosity to those particular users. According to Resnick and Varian, humans rely on recommendations supplied by fellow humans that help make everyday decisions. Also, Resnick and Varian state that recommender systems supply a way to grow this natural and social process without explicit human interaction [16, 17].

Item is a general term for a thing that can be recommended such as a movie or news article. The information needed to furnish recommendations depends on the kind of recommendation algorithm used. The data can consist of mixed data points about the users and the items and exchanges between these user-item pairs that can be recorded to be used as input for a recommender system. In addition, results from a recommendation system are commonly understood as viable recommendations, rather than the result of an explicit search query made by a user, which is the result of an information retrieval system [16, 18].

Video content platform Netflix stated in 2015 that their recommender system is the core to their business. They calculated that supplying personalization and recommendations to their users keeps them up to 12 to 1 billion USD every year by reducing client churn and increasing individual customers' lifetime value [19]. Recommender systems are also used extensively in big E-commerce zones such as Amazon [20]. Although recommender systems have been very prosperous in contemporary years, they pose privacy risks as recommender systems often leverage personal data not just on user demographics but also on preferences that could be highly sensitive if made public. An example of such acute preference could be political opinions that users likely want to keep secret [21].

The vision of collaborative filtering arises from the idea of leveraging the collaborative behaviors of all users for predicting the conduct of the target user (active user). Early approaches perform this opinion by estimating the behavior similarity of users or of items with memory-based models. Thereafter, matrix factorization-based models become dominant which implement the CF idea by collectively discovering the latent spaces that encode user-item relations matrix. These measures can predict users' preferences to some scope. Yet, they suffered from little prediction power due to the conflict between users' complex preferences and the merely linear modeling ability.

Given the explicit complex modeling power of neural networks, collaborative filtering based on neural networks with Radial basis function is used.

Data sparsity and cold start are critical problems in recommender systems for collaborative filtering techniques, particularly for new users and items. There are a variety of hybrid methods to alleviate data sparsity and cold start problems. In the sequel, a brief overview of model-based factorization CF is given and then some existing deep learning-based recommendation methods are discussed [22, 23, 24].

Matrix Factorization. Matrix factorization (MF) is one of the typically used CF methods. Some earlier works explored the content information of items to boost the implementation [25]. Salakhutdinov et al. [26] used restricted Boltzmann machines to carry out CF and Georgiev and Nakov [27] expanded it by combining associations between user-user and item-item. Sainath et al. [28] decreased the number of model parameters and speed up training by using low-rank MF. Yang et al. [29] suggested a recommendation model concentrated on MF that would map users into a low-dimensional latent feature space considering the connection of trust. Silva et al. [30] have proposed Poisson MF-CS which uses factorization of the Poisson matrix to model the importance of users and the contents of items. Ren et al. [31] offered SCVR that item ranks would be predicted based on user ideas and social connections.

Deep Learning. Newly, deep learning models have been operated to improve the implementation of recommender algorithms due to their non-linear modeling faculty. These include NeuMF (Neural Matrix Factorization) and CDAE (Collaborative Denoising Auto-Encoder) [32, 33]. Sainath et al. [28] lessened the number of model parameters and speed up training by using low-rank matrix factorization. Jiang et al. suggested DTR [34] by utilizing a stacked denoising autoencoder (SDAE) to transform the trust information to latent feature space. In [35], a neural network architecture by a denoising autoencoder is proposed founded on the integration of the rating and explicit trust.

This paper overcomes the sparsity problem by completing the sparse user-item rating matrix by using radial basis function neural network (RBFNN) [36], and the scalability problem by clustering the users with kernel-based fuzzy c-means (KFCM) clustering algorithm [37].

The RBF centers are also learned by applying the KFCM clustering method. The original Euclidean norm metric in fuzzy c-means (FCM) method is replaced by a kernel-induced metric in the data space, in KFCM.

The proposed system has two offline and online phases. In the offline phase, the radial basis functions network is used to complete the sparse user-item rating matrix. So the missing ratings for items not yet rated by a user are predicted. Then we rank all the users by finding the Perron vector of the Cosine similarity matrix. In the online phase, the users ranking vector is used to better weight average of deviations from the neighbor's mean and the active users get recommendation based on their likes and dislikes. The new user cold-start problem is solved by recommending the top-rated items of the high-ranked user.

To summarize, our work makes the following contributions:

- We propose a novel framework of RBFNN with a kernel-induced metric in the data space.
- We solve the scalability, sparsity, and cold start problem challenges with the proposed technique based on the RBF network and the strategy that Google adopts for ranking web pages based on the link structure of the web.
- We demonstrate the effectiveness of the proposed method in real-world data and confirm its importance.

The rest of this paper is organized as follows. We present in Section 2 some basic requirements including radial basis function neural network fundamentals and KFCM clustering technique. The proposed recommender system framework is given in Section 3. In Section 4, we provide a running example which aids understanding the underlying ideas and algorithms. Experiments are evaluated in Section 5. The paper ends with a brief conclusion.

2 PRELIMINARIES

2.1 Radial Basis Function Neural Network

The RBFNN is a three-layer (n–p–s) feedforward neural network [38]. The input layer contains n number of neurons. This layer is fully connected to all the neurons in the hidden layer. Each node in the hidden layer uses a radial basis function $\phi(r)$, as its nonlinear activation function. The hidden layer is also fully connected to the output layer. The output layer is a linear combiner mapping the nonlinearity into a new space. The output layer contains s number of neurons. For input $X \in \mathbb{R}^n$, the output of the RBFNN is the approximate function $F(X) = (f_1(X), \dots, f_s(X))^T \in \mathbb{R}^s$, where

$$f_k(X) = \sum_{j=1}^{p} w_{jk}\phi(\frac{\|X - C_j\|}{\sigma}), \quad k = 1, \dots, s \tag{1}$$

where C_j is the prototype or center of the j^{th} node, σ is a positive real number which we call the shape parameter or the width of the RBFs, w_{jk} is the connection weight from the j^{th} hidden unit to the k^{th} output unit, and $\| \cdot \|$ denotes the Euclidean norm.

For a set of m pattern pairs $\{(X_l, Y_l), \quad l = 1, \dots, m\}$, with $X_l \in \mathbb{R}^n$, and $Y_l \in \mathbb{R}^s$, we are going to find a smooth function F such that

$$F(X_l) = Y_l, \quad l = 1, \dots, m. \tag{2}$$

Enforcing the interpolation condition (2) in (1) gives the following linear system of equations

$$Y = \Phi W, \tag{3}$$

where $[Y]_{m\times s} = [Y_1, Y_2, \dots, Y_m]^T$, $[\Phi]_{m\times p} = [\phi(\|X_i - C_j\|)]_{1\le i\le m,\ 1\le j\le p}$, $[W]_{p\times s} = [w_{jk}]_{1\le j\le p,\ 1\le k\le s}$. The most commonly used RBFs $\phi(r)$ are listed in Table 1, where l, β and λ are RBF parameters. In this paper, we use Gaussian RBF, and such an RBFNN is usually called the Gaussian RBFNN. Moreover, the input layer contains n number of neurons, to which the user's rating vector is input, and the output layer also contains $s = n$ number of neurons, to which the user's complete rating vector is output.

The diagram of the RBFNN is depicted in the Figure 1.

Name	$\phi(r)$	Condition
Gaussian	$\exp\left(-\frac{r^2}{2\sigma^2}\right)$	
Multiquadric	$\left(1+\frac{r^2}{\sigma^2}\right)^{\beta/2}$	$\beta \in \mathbb{R}_{\neq 0} \setminus 2\mathbb{N}$
Powers	r^λ	$0 < \lambda \notin 2\mathbb{N}$
Thin-plate splines	$r^{2l}\ln(r)$	$l \in \mathbb{N}$

Table 1. Global RBFs

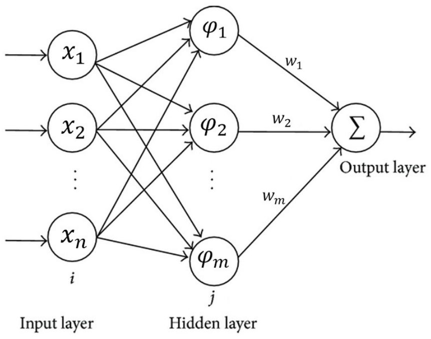


Figure 1. The architecture of the RBFNN

2.2 RBFNN Learning

A neural network adapts itself to a stimulus by making proper parameter adjustments in order to result production of the desired response. This process is called learning or training. RBFNN learning can be formulated as the minimization of the mean squared error function

$$E = \frac{1}{m}\sum_{i=1}^{m} \|Y_i^T - \Phi(i,:)W\|^2 = \frac{1}{m}\|Y - \Phi W\|_F^2 \tag{4}$$

where $\Phi(i,:)$ denotes i^{th} row of the matrix Φ, according to MATLAB notation. RBNN learning requires the determination of the RBF centers and the weights. For some RBFs such as the Gaussian, it is also necessary to determine the smoothness parameter σ [39].

2.2.1 Unsupervised Learning of RBF Centers and Weights

To determine RBF centers, the training set is grouped into appropriate clusters whose prototypes are used as RBF centers. Efficiency of RBF network learning depends largely on the performance of the clustering [40, 41].

After RBF centers and their widths are determined, learning of the weights W is reduced to a linear optimization problem, which can be solved using the least square method leading to pseudo-inverse method or a gradient-descent method. In the pseudo-inverse method, W is trained to minimize the mean squared error (4). This leads to

$$W = \Phi^{\dagger} Y = (\Phi^T \Phi)^{-1} \Phi^T Y$$

where † is the pseudo-inverse of the matrix within [36]. Since the over- or under-determined linear least square system is an ill-conditioned problem, singular value decomposition (SVD) defined as follows is preferred [42].

Definition 1. Let $A \in \mathbb{R}^{N \times M}$ be a matrix with $\operatorname{rank}(A) = r$. The Singular Value Decomposition is defined as

$$U^T A V = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}$$

where $U \in \mathbb{R}^{N \times N}$, and $V \in \mathbb{R}^{M \times M}$ are orthogonal, with their columns being the eigenvectors of AA^T and $A^T A$, respectively. Also $D = \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$ with nonzero singular values $\sigma_1, \ldots, \sigma_r$.

This property that SVD can supply the optimal approximation to the matrix A using three smaller matrices multiplication, can be very helpful in developing recommender systems [43].

2.2.2 Supervised Learning of All Parameters

The simplest solution to the supervised learning of the RBFNN is the gradient descent method. Rewriting the error function (4) gives

$$E = \frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{s} (e_{i,k})^2$$

where

$$e_{i,k} = Y_{i,k} - \sum_{j=1}^{p} W_{jk} \phi \left(\|X_i - C_j\| \right) = Y_{i,k} - \phi(i,:) W(:,k)$$

is the approximation error at the kth output node for the ith sample. By taking the first-order derivative of E with respect to W_{jk} and C_j, we have for $j = 1, \ldots, p$,

$k = 1, \ldots, s$

$$\begin{aligned}
\frac{\partial E}{\partial W_{jk}} &= \frac{\partial}{\partial W_{jk}} \left(\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{s} \left(Y_{i,k} - \phi(i,:)W(:,k)\right)^2 \right) \\
&= \frac{2}{m} \sum_{i=1}^{m} \sum_{k=1}^{s} \left(Y_{i,k} - \phi(i,:)W(:,k)\right) \left(-\phi\left(\|X_i - C_j\|\right)\right) \\
&= -\frac{2}{m} \sum_{i=1}^{m} e_{i,k} \phi\left(\|X_i - C_j\|\right), \\
\frac{\partial E}{\partial C_j} &= \frac{\partial}{\partial C_j} \left(\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{s} \left(Y_{i,k} - \phi(i,:)W(:,k)\right)^2 \right) \\
&= -\frac{2}{m} W_{jk} \sum_{i=1}^{m} \left(Y_{i,k} - \phi(i,:)W(:,k)\right) \left(\frac{\partial}{\partial C_j} \phi\left(\|X_i - C_j\|\right) \right) \\
&= \frac{2}{m} W_{jk} \sum_{i=1}^{m} e_{i,k} \dot{\phi}\left(\|X_i - C_j\|\right) \frac{X_i - C_j}{\|X_i - C_j\|}
\end{aligned}$$

where $\dot{\phi}(\cdot)$ is the first-order derivative of $\phi(\cdot)$.

Now the update rules for center and weight learning is given by

$$\begin{aligned}
W_{jk}(t+1) &= W_{jk}(t) - \eta_1 \frac{\partial E}{\partial W_{jk}}, \quad j = 1, \ldots, p, \quad k = 1, \ldots, s, \qquad (5) \\
C_j(t+1) &= C_j(t) - \eta_2 \frac{\partial E}{\partial C_j}, \quad j = 1, \ldots, p.
\end{aligned}$$

where η_1 and η_2 are learning rates. Initialization can be based on a random selection of the RBF centers from the samples and W as a matrix with small random components.

2.3 Kernel-Based Fuzzy C-Means Clustering Algorithm

KFCM minimizes the following objective function [37]:

$$J_q(m, c) = 2 \sum_{l=1}^{k} \sum_{i=1}^{n} m_{li}^q \left(1 - K(x_i, c_l)\right) \qquad (6)$$

where $K(x, c) = \exp\left(-\frac{\|x-c\|^2}{2\sigma^2}\right)$ is the Gaussian kernel function with $K(x, x) = 1$, k is the number of clusters and selected as a specified value in this paper, n is the number of data points, m_{li} is the degree of membership of x_i in class l which ranges in $[0,1]$, and q is the quantity controlling clustering fuzziness. By minimizing Equation (6) under the constraint $\sum_{l=1}^{k} m_{li} = 1$, we have

$$m_{li} = \frac{\left(1/\left(1-K\left(x_i,c_l\right)\right)\right)^{\frac{1}{q-1}}}{\sum_{l=1}^{k}\left(1/\left(1-K\left(x_i,c_l\right)\right)\right)^{\frac{1}{q-1}}}, \quad l=1,\dots,k, \quad i=1,\dots,n, \tag{7}$$

$$c_l = \frac{\sum_{i=1}^{n}\left(m_{li}^{(t)}\right)^q K(x_i,c_l)x_i}{\sum_{i=1}^{n}\left(m_{li}^{(t)}\right)^q K(x_i,c_l)}, \quad l=1,\dots,k. \tag{8}$$

In the FCM clustering algorithm, it is assumed that the data in a dataset are complete, that is, all of the features of every vector in dataset are known or exist. However, many real data sets such as recommendation engines lack completeness, i.e., one or more of the components in the ratings database are missing, due to the fact that users typically rate only a small proportion of the available items.

According to Equation (8), the similarity between x_i and c_l, is measured by additional weight $K(x_i,c_l)$. When x_i is far from the other data points, i.e., x_i is an outlier, $K(x_i,c_l)$ will be very small, so the weighted sum of data points shall be more robust. Since in recommender systems, a data point with missing ratings is likely to turn into an outlier, the algorithm based on KFCM to cluster in recommender systems is of great importance. The KFCM algorithm for clustering incomplete dataset is given as follows.

Algorithm 1. Clustering incomplete data using KFCM
Input: $X=\{x_1,\dots,x_n\}$, where $x_i\in\mathbb{R}^d$, $K(x,c)$: kernel function, k: number of clusters, q: fuzziness exponent, ε: termination tolerance, N: maximum number of iterations.
Output: $\mathcal{I}=\{I_1,\dots,I_n\}$: set of cluster indices corresponding to points, $\mathcal{C}=\{c_1,c_2,\dots,c_k\}$: set of cluster centers.

```
Initialize set of cluster centers $\{c_1,c_2,\dots,c_k\}$, where $c_i\in\mathbb{R}^d$.
Set initial degree of membership $m_{li}^{(0)}=0$, $l=1,\dots,k$, $i=1,\dots,n$.
for all $t=1,\dots,N$ do
    Update all degree of memberships $m_{li}^{(t)}$ with
        $m_{li}^{(t)} = \frac{\left(1/\left(1-K\left(x_i,c_l\right)\right)\right)^{\frac{1}{q-1}}}{\sum_{l=1}^{k}\left(1/\left(1-K\left(x_i,c_l\right)\right)\right)^{\frac{1}{q-1}}}, \quad l=1,\dots,k,\ i=1,\dots,n.$
    Calculate the missing values (zero entries of $X$) using
        $x_{ij} = \frac{\sum_{l=1}^{k}\left(m_{li}^{(t)}\right)^q K(x_i,c_l)c_{lj}}{\sum_{l=1}^{k}\left(m_{li}^{(t)}\right)^q K(x_i,c_l)}, \quad i=1,\dots,n, \quad j=1,\dots,d.$
    Update all centers $c_l$ with
        $$c_l = \frac{\sum_{i=1}^{n} \left(m_{li}^{(t)}\right)^q K(x_i, c_l) x_i}{\sum_{i=1}^{n} \left(m_{li}^{(t)}\right)^q K(x_i, c_l)}, \quad l = 1, \ldots, k.$$
    If $\max_{l,i} \left| m_{li}^{(t)} - m_{li}^{(t-1)} \right| \leq \epsilon$, stop.
end for
for all $i = 1, \ldots, n$ do
    Set $l^* = \arg\max_l m_{li}$.
    Set $I_i = l^*$.
end for
return $\mathcal{I}$ and $\mathcal{C}$;
```

3 THE PROPOSED RECOMMENDER SYSTEM FRAMEWORK

In this section, we describe the framework of the proposed collaborative filtering recommender system. The proposed system has offline and online phases. It should be noted that we use Matlab notation for the algorithms given in this paper.

3.1 Offline Phase

Let $R = [r_{ij},\ 1 \leq i \leq m,\ 1 \leq j \leq n]$, be the user-item rating matrix, where r_{ij} denotes the rating of user i to item j. The goal of the recommender system in offline phase is to predict the nonrated items. Since the nonrated items are represented by a value of zero, the matrix $R = [r_{ij}]_{m \times n}$ is highly sparse. So, the sparse user-item rating matrix is to be completed and becomes a full rating matrix. Moreover, the correlated users can be found more easily from a full rating matrix. The completing is done using RBFNN.

Remark 1. In this paper, we use Gaussian RBF. The kernel-based fuzzy c-means clustering algorithm is used for selecting RBF centers.The width is fixed according to the spread of centers by $\sigma = \frac{d}{\sqrt{2p}}$, where d is the maximum distance between the selected centers [36]. This choice makes the Gaussian RBF neither too steep nor too flat. The RBF weights are also obtained by the gradient descend method.

The completing algorithm is given as follows.

Algorithm 2. Completing
Input: $R = [r_{ij},\ 1 \leq i \leq m,\ 1 \leq j \leq n]$: sparse user-item rating matrix, ϵ: termination tolerance, η: weight learning rate.
Output: $\widehat{R} = [\widehat{r}_{ij},\ 1 \leq i \leq m,\ 1 \leq j \leq n]$: complete user-item rating matrix.

```
1: Set $\widehat{R} = R$.
2: Set $range = (max\_rating - min\_rating) + 1$.
```

```
3:  Set number of clusters $\tilde{k} = \min k$,  subject to  $\left\lfloor \frac{range}{k} \right\rfloor \le 3$.
4:  for all $j = 1, \ldots, n$ do
5:      Set $X = R(:, j)$, call Algorithm 1. and partition the users into $\tilde{k}$ clusters.
6:      Set cluster indices $I(:, j) = \mathcal{I}$.
7:      Set cluster centers $C(:, j) = \mathcal{C}$.
8:  end for
9:  for all $j = 1, \ldots, n$ do
10:     for all $i = 1, \ldots, \tilde{k}$ do
11:         Set $f = \text{find}(I(:, j) == i)$.
12:         Set $G(f, j) = \phi\left(\|R(f, j) - C(i, j)\|_2\right)$, where $\phi$ is one of the activation func-
            tions in Table 1.
13:     end for
14: end for
15: Set initial weight matrix $[W_0]_{m\times n} = (\max(G) - G)\,./(\max(G) - \min(G))$.
16: Set $W_0 = W_0./\text{sum}(W_0)$.
17: for all $j = 1, \ldots, n$ do
18:     for all $i = 1, \ldots, \tilde{k}$ do
19:         Set $f = \text{find}(I(:, j) == i)$.
20:         Set $F(i, j) = \langle G(f, j), W_0(f, j)\rangle$ according to Equation (1).
21:     end for
22: end for
23: Round entries of $F$ to the nearest integers.
24: Set $F(F < min\_rating) = min\_rating$.
25: Set $F(F > max\_rating) = max\_rating$.
26: Set $[S, T] = \text{find}(R == 0)$.
27: Set $\widehat{R}(S, T) = F(I(S, T), T)$.
28: repeat
29:     Set $W = W_0 + \eta\left(\left(\widehat{R} - R\right) .* Q\right)$.
30:     Set $E = \frac{\|W - W_0\|_1}{\|W\|_1}$.
31:     Set $W_0 = W$.
32: until $E \le \epsilon$.
33: Repeat Steps 17–27 with $W_0 = W$.
34: return $\widehat{R}$.
```

The ratings range from $[min\_rating, max\_rating]$, where min_rating represents dislike and max_rating represents a strong preference. In steps 4–8, users have been clustered to $\tilde{k}$ clusters for each item using KFCM clustering Algorithm 1. In steps 9–13, the Gaussian activation function is obtained for each item corresponding to m number of users and results the matrix G.

The initial normalized weight matrix W_0, ($\|W_0\|_1 = 1$), is calculated in steps 15–16, which indicates that

$$(W_0)_{i,j} = \frac{(\max(G(:, j)) - G(i, j))\,/\,(\max(G(:, j)) - \min(G(:, j)))}{\sum_{i=1}^{m} (\max(G(:, j)) - G(i, j))\,/\,(\max(G(:, j)) - \min(G(:, j)))}.$$

The approximate rating values of nonrated items are computed in steps 17–25 which we substitute the minimum value of the ratings range for the elements less than it and the maximum value of the ratings range for the elements more than it. Then the complete matrix $\widehat{R}$ is obtained in steps 26–27, firstly. But it is used in order to update weight matrix W_0 by using the iterative sequence (5) in gradient descent method in steps 28–32. After the stopping condition $E = \frac{\|W-W_0\|_1}{\|W\|_1} \le \epsilon$ is satisfied, the final weight matrix W is used to get predicted user-item rating matrix in steps 17–27.

3.2 Online Phase

In this phase, we try to recommend favorite items to an active user entering the system through the "*login session*" based on his likes and dislikes. Once the user login the system, he can give his ratings for some randomly selected items. Based on his ratings the recommendation is provided. At first, we rank all the users with the aid of the strategy Google adopts for ranking web pages based on the link structure of the web [44].

Let users are ordered from 1 to m, and l be a particular user. The rank of user l is defined by

$$\mathtt{r}_l = \sum_{k=1}^{m} \mathtt{C}_{lk}\mathtt{r}_k \tag{9}$$

where

$$\mathtt{C}_{lk} = \frac{\sum_{j=1}^{n} \widehat{r}_{lj}\ \widehat{r}_{kj}}{\sqrt{\sum_{j=1}^{n} \left(\widehat{r}_{lj}\right)^2}\sqrt{\sum_{j=1}^{n} \left(\widehat{r}_{kj}\right)^2}} \tag{10}$$

is the Cosine similarity between two users l^{th} and k^{th} corresponding to complete matrix $\widehat{R}$. It should be noted that the ranking formula (9) is a weighted sum of the ranks of the users that have similarity to l^{th} user. Now, consider the Cosine similarity matrix $\mathtt{C} = [\mathtt{C}_{lk},\ 1 \le l \le m,\ 1 \le k \le m]$. Since $\widehat{R}$ is a full rating matrix with positive entries then $0 < \mathtt{C}_{lk} \le 1$. Equation (9) can be represented by the matrix form

$$\lambda\mathtt{r} = \mathtt{Cr}, \quad \lambda = 1,$$

that is $\mathtt{r}$ is an eigenvector of $\mathtt{C}$ with eigenvalue $\lambda = 1$. Users ranking will be well-defined if there exists a unique eigenvalue equal to 1. For this sake, consider the normalized matrix $\bar{\mathtt{C}}$ to be the matrix of dividing each entry of $\mathtt{C}$ in a given column by the sum of the entries of that column.

The matrix $\bar{\mathtt{C}}$ has positive elements, and the sum of elements of each column is 1. So it is a column-stochastic matrix satisfying $e^T\bar{\mathtt{C}} = e^T$ which means that 1 is an eigenvalue of $\bar{\mathtt{C}}$. However, in order to have a unique eigenvalue with eigenvalue 1, we refer to the Perron's theorem given as follows [45]:

Theorem 1. (Perron's Theorem) The following statements are true for any positive matrix $A_{n\times n}$ with spectral radius $\rho(A)$:

- $\rho(A) \in \sigma(A)$, where $\sigma(A)$ is set of eigenvalues of A.
- $\rho(A)$ has geometric and algebraic multiplicities 1.
- There is a unique corresponding eigenvector r satisfying $r > 0$, and $\|r\|_1 = 1$; this is the only eigenvector that is nonnegative.

Therefore, users' ranking can be formulated mathematically as an eigenvalue equation for a certain matrix. So the power method [42] can be applied in order to find the eigenvector $\mathtt{r}$ corresponding to dominant eigenvalue $\lambda = 1$, which ranks users.

In the sequel, we give the algorithmic approach of recommendation in online phase.

Algorithm 3. Recommendation
Input: $\widehat{R} = [\widehat{r}_{ij},\ 1 \le i \le m,\ 1 \le j \le n]$: complete user-item rating matrix, u: active user's ratings to the items, T: number of items to be recommended.
Output: $\widehat{u}$: predicted ratings for the active user, Z: item recommendations for the active user.

```
1:  Set $\widehat{u} = u$.
2:  Set $[\mathtt{C}_{lk},\ 1 \le l \le m,\ 1 \le k \le m]$ to be the Cosine similarity matrix between all
    users corresponding to $\widehat{R}$ with entries given in Equation (10).
3:  Set $\bar{\mathtt{C}}$ to be the matrix of dividing each element of $\mathtt{C}$ in a given column by the
    sum of the elements of that column.
4:  Set users ranking vector $w$ to be the eigenvector corresponding to dominant
    eigenvalue $(\lambda = 1)$ in $\bar{\mathtt{C}}$.
5:  Set $I = \arg\max\ w$ : index of high-ranked user.
6:  Set $u^* = \widehat{R}(I,:)$.
7:  if the active user is a new user and the rating vector is NULL (cold start problem)
    then
8:      Set $Z = \arg\max(u^*, T)$ to be $T$ top rated items of the high-ranked user.
9:  else if the active user is a new user with some rating vector then
10:     Set $v$ to be the Pearson similarity vector between $u$ and all users corresponding
        to $\widehat{R}$:
```

$$v_l = \frac{\sum_{t \in r_l \bigcap u} \left(\widehat{r}_{lt} - \overline{\widehat{r}_l}\right)(u_t - \overline{u})}{\sqrt{\sum_{t \in r_l \bigcap u} \left(\widehat{r}_{lt} - \overline{\widehat{r}_l}\right)^2} \sqrt{\sum_{t \in r_l \bigcap u} (u_t - \overline{u})^2}}, \quad l = 1, \ldots, m.$$

```
11:     Set $P = \text{find}(v >= 0)$ to be the positive neighbouring cluster.
12:     Set $N = \text{find}(v < 0)$ to be the negative neighbouring cluster.
13:     Set $J = \text{find}(u == 0)$ to be the zero rated items for the active user.
14:     Set $s$ to be the Cosine similarity vector between $u$ and all users corresponding
```

```
      to $R$:
$$s_l = \frac{\sum_{t\in r_l \bigcap u} r_{lt}\ u_t}{\sqrt{\sum_{t\in r_l \bigcap u} (r_{lt})^2}\sqrt{\sum_{t\in r_l \bigcap u} (u_t)^2}}, \quad l = 1,\dots,m.$$
15:   Predict the rating of the set of items $J$ for the active user as the weighted
      average of deviations from the positively correlated neighbours' mean
$$\widehat{u}_j = \overline{r_j} + \frac{\sum_{i\in P}\left(\widehat{r}_{ij} - \overline{\widehat{r}_i}\right) s_i\, w_i}{\sum_{i\in P} s_i}, \quad j \in J.$$
16:   Set [sort_val, sort_ind] = sort($\widehat{u}$, ‘descend’).
17:   Set $X$ = sort_ind(sort_val >= 3) to be the set of recommended items based
      on liking.
18:   Predict the rating of the set of items $J$ for the active user based on negatively
      correlated neighbours
$$\widehat{v}_j = \overline{r_j} + \frac{\sum_{i\in N}\left(\widehat{r}_{ij} - \overline{\widehat{r}_i}\right) s_i\, w_i}{\sum_{i\in N} s_i}, \quad j \in J.$$
19:   Set [sort_val, sort_ind] = sort($\widehat{v}$, ’descend’).
20:   Set $Y$ = sort_ind(sort_val >= 3) to be the set of recommended items based
      on disliking.
21:   Set $Z = X - Y$.
22:   if $Z ==$ NULL then
23:     Set $Z = \arg\max(u^*, T)$ to be $T$ top rated items of the high-ranked user.
24:   else if length($Z$) $\geq T$ then
25:     $Z = Z(1 : T)$.
26:   else
27:     Set $\widehat{Z} = \arg\max(u^*, T - \text{length}(Z))$ to be $T - \text{length}(Z)$ top rated items of
        the high-ranked user.
28:     $Z = [Z, \widehat{Z}]$.
29:   end if
30: else if the active user is an existing user with row index $i^*$ and with no additional
    ratings then
31:   Set $\widehat{u} = \widehat{R}(i^*, :)$.
32:   Set [sort_val, sort_ind] = sort($\widehat{u}$, ’descend’).
33:   Set $Z$ = sort_ind(sort_val >= 3) and do steps 22–29.
34: else if the active user is an existing user with additional ratings then
35:   Do steps 10–29.
36: end if
37: return $\widehat{u}$ and $Z$.
```

Theorem 2. Let $\bar{C}$ be the normalized Cosine similarity matrix between all users corresponding to the complete user-item rating matrix $\widehat{R}$. Then the dominant eigenvalue λ_1 is equal to 1. There is a unique corresponding eigenvector w satisfying $w > 0$, and $\|w\|_1 = 1$; this is the only eigenvector that is nonnegative.

Proof. Since $\widehat{R} > 0$ then $\bar{C} > 0$. The matrix $\bar{C}$ is a column-stochastic matrix which has nonnegative elements, and the elements of each column sum up to 1. So it satisfies $e^T P = e^T$ which means that 1 is an eigenvalue of $\bar{C}$. The rest of the statement can be proved using the Perron's Theorem 1. $\square$

Now, according to Google page ranking idea, the ratings vector of the high-ranked user, who has most significant role based on his high similarity to all other users, is calculated in step 6. In steps 7–35, the predicted ratings for the active user as well as item recommendations for him are obtained based on the fact that he is a new user or an existing user. It should be noted that $r_l \bigcap u$ indicates the set of items that are rated by both lth user and the active user u, and $\overline{r_j}$ is the average rating of the jth user defined as

$$\overline{r_j} = \frac{1}{|B_j|} \sum_{l \in B_j} r_{jl}$$

where B_j is the set of items rated by the jth user. $\overline{\widehat{r}_i}$ is also defined as the average rating of the ith user corresponding to the complete user-item rating matrix $\widehat{R}$, similarly. We also used users ranking vector w in order to have a weighted prediction in steps 15 and 18.

For a deeper understanding, the offline and online phase algorithms are shown in the Figure 2.

3.3 Computational Complexity

In this section we analyse the computational complexity of the Algorithms 1, 2, and 3. Let m, n, T, and k be the number of users, items, iteration, and clusters, respectively. We can see that the number of addition-subtraction and multiplication-division operations are equal to $T(2m + 5k + 4)$, and $T(5m^2 + 6k + 3)$, respectively, for the Algorithm 1. Furthermore, the number of addition-subtraction and multiplication-division operations are equal to $(n + 1)k + 2m^2n$, and $7kn + m^2n$, respectively, for the Algorithm 2. Finally, Algorithm 3 requires $9mn + 6n$ and $4mn+8m^2n$ operations for addition-subtraction and multiplication-division, respectively.

Since the number of users is usually greater than the number of items, the dominant term is $Tknm^2$, and so the computational complexity of the proposed recommender system is $O(Tknm^2)$.

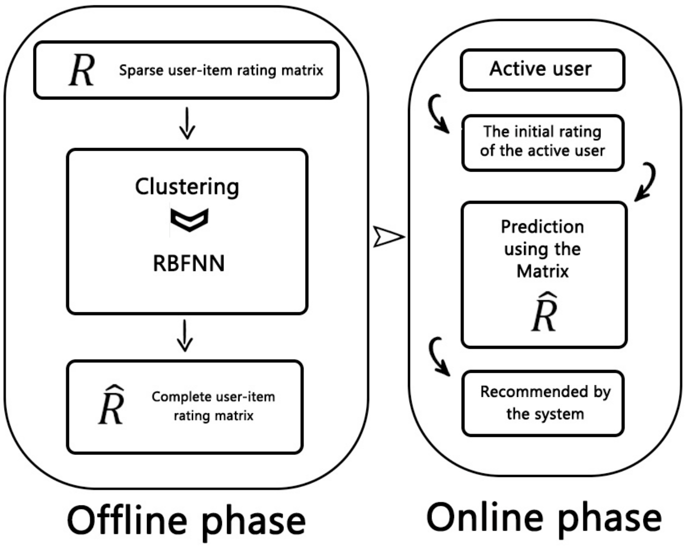

R Sparse user-item rating matrix
Clustering
RBFNN
$\widehat{R}$ Complete user-item rating matrix
Offline phase
Active user
The initial rating of the active user
Prediction using the Matrix $\widehat{R}$
Recommended by the system
Online phase

Figure 2. Framework of the proposed recommender system

4 RUNNING EXAMPLE

In this section, we provide a running example that helps to understand and follow basic ideas and algorithms. We work with the sparse rating matrix shown in Table 2 with 5 users and 10 items, which is taken from [46]. The ratings are in the range 1 (poor) to 5 (good), and the nonrated items are represented by 0.

Offline phase.

Input: sparse user-item rating matrix R.

Output: complete user-item rating matrix $\widehat{R}$.

According to the offline phase, we first need to cluster users based on the rating of each item by KFCM clustering Algorithm 1. As the rating range is 1–5, the number of clusters is 2. The cluster centers for each item are shown in Table 3. The users are grouped in the clusters based on the cluster centers. The cluster to which the users belong is shown in Table 4. The matrix G which includes Gaussian activation function values for the original rating matrix is given in Table 5. The initial weight matrix W_0 is reported in Table 6. Table 7 shows the final weight matrix W based on the Gaussian activation functions for the original rating matrix. Using the Gaussian activation function given in Table 5 and the weights in Table 7, the full rating matrix is calculated by Algorithm 2. The full rating matrix is shown in Table 8. Training error is 0.2880 and the rating matrix is completely filled.

Online phase.

Input: complete user-item rating matrix $\widehat{R}$.

Output: recommendation for users.

Table 9 shows the user rating details and Top 2 recommendation for the Gaussian based complete rating matrix.

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
U1	3	0	0	0	2	0	0	5	0	0
U2	0	1	0	0	4	0	0	0	0	0
U3	1	3	0	0	0	0	1	0	2	1
U4	0	0	4	2	0	0	1	0	0	0
U5	4	0	0	4	0	2	0	0	0	0

Table 2. Original sparse rating matrix with 68 % sparsity

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
Cluster center1	1.1955	1.0063	1.2293	1.4201	1.4201	0.7193	0.4645	1.4820	0.7193	0.4641
Cluster center2	3.6705	3.0999	4.1000	4.0898	4.0898	2.0999	1.0997	5.1000	2.0999	1.0991

Table 3. Cluster centers

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
U1	2	1	1	1	1	1	1	2	1	1
U2	1	1	1	1	2	1	1	1	1	1
U3	1	2	1	1	1	1	2	1	2	2
U4	1	1	2	1	1	1	2	1	1	1
U5	2	1	1	2	1	2	1	1	1	1

Table 4. User cluster index for each item

5 EVALUATION

In this section, we will analyze the accuracy and quality of the predictions and recommendations of our proposed algorithm. Several experiments were performed to test the efficiency of the proposed system using Netflix[1] [47] and MovieLens[2] [48] data sets including MovieLens 100K, MovieLens 1M, and MovieLens 10M. Summary statistics for the datasets used in this paper are shown in Table 10. In the Netflix

[1] `http://www.netflixprize.com/download`

[2] `http://grouplens.org/datasets/movielens`

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
U1	0.9590	0.9101	0.8688	0.8289	0.9692	0.9530	0.9801	0.9991	0.9530	0.9802
U2	0.8755	1.0000	0.8688	0.8289	0.9993	0.9530	0.9801	0.8152	0.9530	0.9802
U3	0.9965	0.9991	0.8688	0.8289	0.8289	0.9530	0.9991	0.8152	0.9991	0.9991
U4	0.8755	0.9101	0.9991	0.9692	0.8289	0.9530	0.9991	0.8152	0.9530	0.9802
U5	0.9900	0.9101	0.8688	0.9993	0.8289	0.9991	0.9801	0.8152	0.9530	0.9802

Table 5. Guassian activation function values

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
U1	0.1309	0.3322	0.2500	0.3148	0.0556	0.2500	0.3333	0	0.2500	0.2500
U2	0.4232	0	0.2500	0.3148	0	0.2500	0.3333	0.2500	0.2500	0.2500
U3	0	0.0034	0.2500	0.3148	0.3148	0.2500	0	0.2500	0	0
U4	0.4232	0.3322	0	0.0556	0.3148	0.2500	0	0.2500	0.2500	0.2500
U5	0.0227	0.3322	0.2500	0	0.3148	0	0.3333	0.2500	0.2500	0.2500

Table 6. Initial weight matrix

dataset, we choose users who have rated at least 2 000 movies. This results in a data set with 2 626 320 ratings, 5 265 movies, and 1 212 users. The ratings are converted into a user-item matrix. The rating range is from 1 to 5, where 1 represents dislike and 5 represents a strong preference. All unrated items have a value of zero.

Proposed algorithms were implemented in MATLAB R2018a on a PC with an Intel (R) Core (TM) i5-6300U, CPU 2.50 GHz, and 8 GB RAM. The parameters used in Algorithm 1 are fuzziness exponent $q = 2$, termination tolerance $\varepsilon = 10^{-3}$, maximum number of iterations $N = 10$. The initial cluster centers were also chosen by the following procedure:

$$h = (max\_rating - min\_rating)/\tilde{k},$$
$$\mathcal{C} = (min\_rating + h/2 : h : max\_rating - h/2)' + 0.1$$

where $min\_rating = 1$ and $max\_rating = 5$.

The parameters used in Algorithm 2 are termination tolerance $\epsilon = 10^{-1}$, and weight learning rate $\eta = 0.1$.

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
U1	0.1309	1.0603	0.9451	0.9779	0.0556	1.0124	1.1174	0	1.0124	1.0341
U2	1.1235	0	0.9451	0.9779	0	1.0124	1.1174	0.9021	1.0124	1.0341
U3	0	0.0034	0.9451	0.9779	0.9779	1.0124	0	0.9021	0	0
U4	1.1235	1.0603	0	0.0556	0.9779	1.0124	0	0.9021	1.0124	1.0341
U5	0.0227	1.0603	0.9451	0	0.9779	0	1.1174	0.9021	1.0124	1.0341

Table 7. Final weight matrix

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
U1	3	3	3	2	2	4	3	5	4	4
U2	2	1	3	2	4	4	3	3	4	4
U3	1	3	3	2	2	4	1	3	2	1
U4	2	3	4	2	2	4	1	3	4	4
U5	4	3	3	4	2	2	3	3	4	4

Table 8. Complete rating matrix with 0 % sparsity

User	Rating	Prediction	Recommendation
New user	No (cold start user)	No	Item 6, 8
New user	$[0,1,0,0,4,0,0,0,0,0]$	$[2,1,4,3,4,2,1,5,2,1]$	Item 5, 8
Existing user (U2)	No new rating	$[2,1,3,2,4,4,3,3,4,4]$	Item 5, 6
Existing user (U2)	$[0,1,0,0,4,0,0,5,0,2]$	$[2,1,4,2,4,2,1,5,2,2]$	Item 5, 8

Table 9. TOP 2 recommendations

5.1 Evaluation Metrics

We use four metrics for evaluating the performance of the proposed recommender system. In this paper we consider the Mean Absolute Error (MAE),

$$\mathrm{MAE} = \frac{1}{K}\sum_{j=1}^{K}\left(\frac{1}{|V_j|}\sum_{x\in V_j}|R(x)-\widehat{R}(x)|\right),$$

and the Root Mean Square Error (RMSE),

$$\mathrm{RMSE} = \frac{1}{K}\sum_{j=1}^{K}\sqrt{\frac{1}{|V_j|}\sum_{x\in V_j}\left(R(x)-\widehat{R}(x)\right)^2}.$$

We analyze the performance of the proposed recommender system according to

Name	Date Range	Rating Scale	Users	Movies	Ratings	Density
MovieLens 100K	1997–1998	1–5	943	1 682	100 000	6.3 %
MovieLens 1M	2000–2003	1–5	6 040	3 706	1 000 209	4.47 %
MovieLens 10M	1995–2009	1–5	69 878	10 681	10 000 054	1.34 %
Netflix	1998–2005	1–5	480 000	17 000	100 000 000	1.178 %

Table 10. Quantitative summary of the ratings datasets used. The sole computed column, Density, represents the percentage of cells in the user-item matrix that contain rating values.

precision, recall, and F-measure as follows:

$$\text{Precision} = \frac{\#\text{tp}}{\#\text{tp} + \#\text{fp}},$$

$$\text{Recall} = \frac{\#\text{tp}}{\#\text{tp} + \#\text{fn}},$$

$$\text{F-measure} = \frac{\#\text{tp}}{\#\text{tp} + \frac{1}{2}(\#\text{fn} + \#\text{fp})}.$$

The parameters needed for computing precision, recall, and F-measure are described in Table 11.

	Recommended	Not Recommended
Used by a user	True-Positive (tp)	False-Negative (fn)
Not used by a user	False-Positive (fp)	True-Negative (tn)

Table 11. The parameters used for defining different metrics

5.2 Experimental Results and Discussion

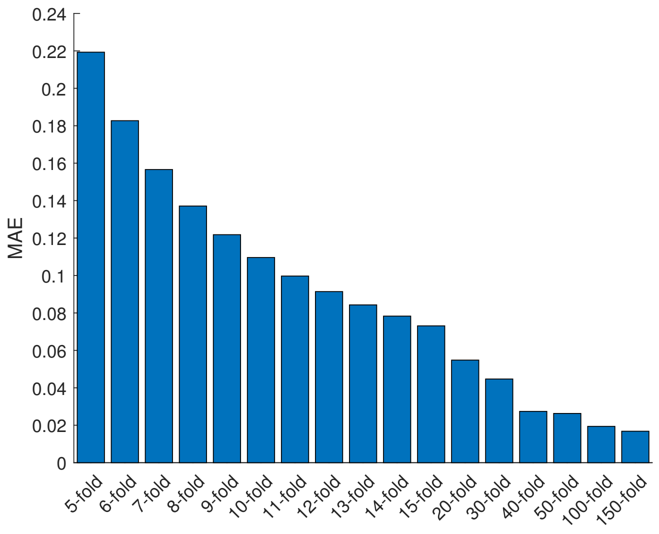
In this section, we will analyze the accuracy of the predictions and recommendations of our proposed method. We also compare our results with some recent successful approaches. MAE of the proposed method for MovieLens 100K and Netflix datasets with different values of K-fold cross-validation, has been depicted in Figure 3. Figure 4 depicts RMSE of the proposed method for MovieLens 100K and Netflix datasets with different values of K. It can be noted from Figures 3 and 4 that the proposed method produces accurate recommendations. Table 12 reports MAE and RMSE of our proposed method for MovieLens 100K and Netflix datasets with different values of K-fold cross-validation. The results are more accurate for MovieLens dataset due to its less sparsity level. Decision support measures including classical information retrieval measures of precision, recall, and F-measure are depicted in Figure 5 for determining how well our proposed recommender system can make predictions of high-relevance items. Precision, recall, and F-measure measures for MovieLens 100K dataset are 0.98, 0.85, and 0.91, respectively. Precision, recall, and F-measure measures for Netflix dataset are 0.89, 0.8, and 0.84, respectively. The MAE and RMSE values of the proposed method are compared with Matlab SVD algorithm as well as three competing matrix factorization methods including Bayesian nonnegative matrix factorization (BNMF) [49], Imputation-based multiplicative update rules (IMULT) [50], and Enhanced SVD (ESVD) [51] for MovieLens 10M dataset with 5-fold cross-validation in Table 13. It can be seen from Table 13 that the proposed method achieves higher accuracy than the others. Figure 6 a) depicts MAE comparison of the proposed method with content-boosted matrix factorization generalized alignment based method (gAB) [52], Extended content-boosted matrix factorization method (ECB) [52], support vector machine (SVM) using polynomial kernel function (SVM-POLY), SVM using radial basis function (SVM-RBF) and multilayered perceptrons (MLP) trained with the back propagation algorithm using sigmoid activation function, for MovieLens 100K dataset with 5-fold cross-validation. The SVM results are computed using Gist Support vector machine and kernel principal components analysis software toolkit (Version 2.0.9). The other methods are modeled in MatLab software. It can be noted from Figure 6 a) that the proposed method leads to more accurate results than the gAB, ECB, and SVM-poly, but less accurate than SVM-RBF and MLP. The decision support measure comparison is shown in Figure 6 b). The observation shows that the proposed method outperforms other methods. Figure 7 depicts computational time comparison of the proposed method with some existing methods for MovieLens 100K dataset with 5-fold cross-validation.

The observations of the comparison of MAE, decision support measure, and computational time are as follows:

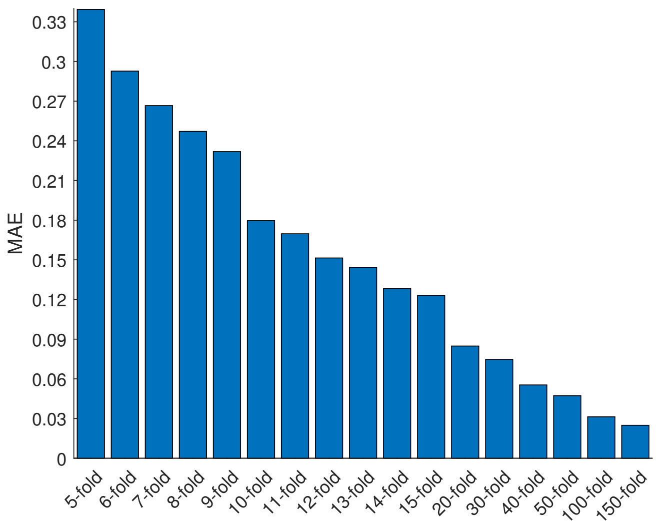
- SVD is the dimensionality reduction method. So, it must outperform other techniques whereas its errors are comparatively high in Table 13.
- ESVD, IMULT, and BNMF have slightly higher 'MAE' and 'RMSE' than the proposed method in Table 13.
- The reason for the low 'MAE' of the proposed method is using the Gaussian RBF and the kernel-based fuzzy c-means clustering algorithm.
- Table 12 and Figures 3 and 4 confirm the fact that the value of errors decreases when the number of folds increases.
- Figure 6 a) also shows that the error of gAb and ECB is greater than the other techniques. In the same figure, the MAE of techniques SVM-RBF and MLP is lower than the proposed method, while the decision support measure of the proposed technique is higher than all the methods in Figure 6 b).
- In Figure 7, all the techniques are at the same level of the computational time in the offline phase, except for SVD with less amount of time. But in the online phase, the proposed method and SVD have the lowest computational time compared to the other methods.

	Movielens		Netflix	
K	MAE	RMSE	MAE	RMSE
3	0.3655	0.6881	0.5318	0.7101
4	0.2741	0.5959	0.4423	0.6312
5	0.2193	0.4330	0.3391	0.5332
6	0.1827	0.3765	0.2928	0.4866
7	0.1566	0.3505	0.2670	0.4505

Table 12. Error values of the proposed method for MovieLens 100K and Netflix datasets

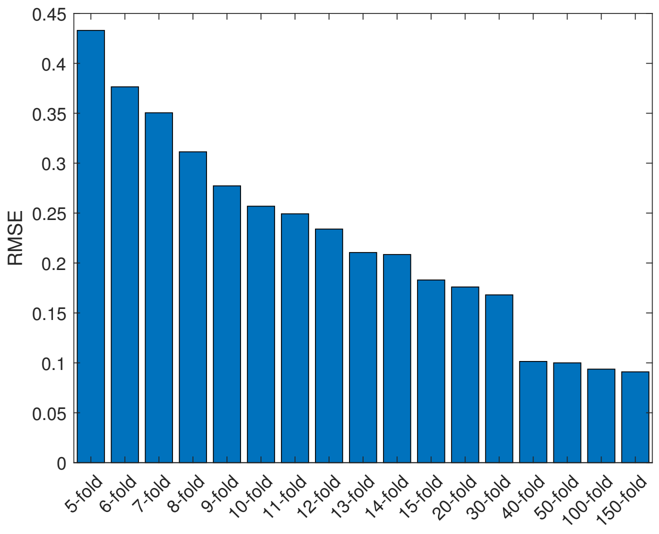


a) MovieLens 100K

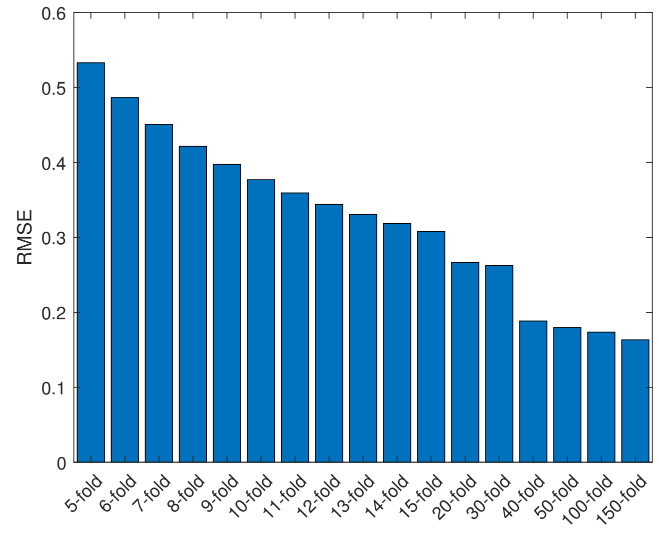


b) Netflix

Figure 3. MAE graph of the proposed method for MovieLens 100K (left) and Netflix (right) datasets with different values of K-fold cross-validation

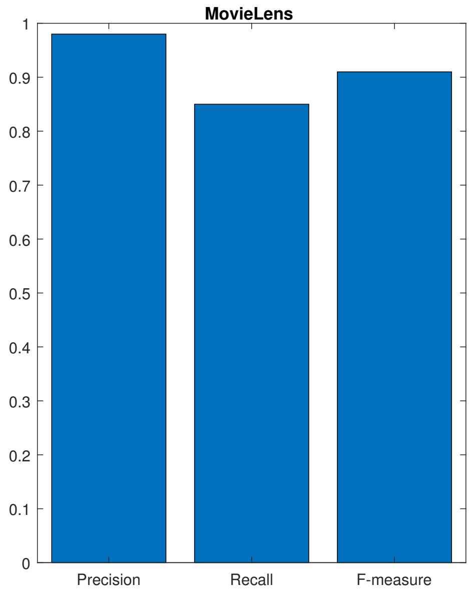


a) MovieLens 100K

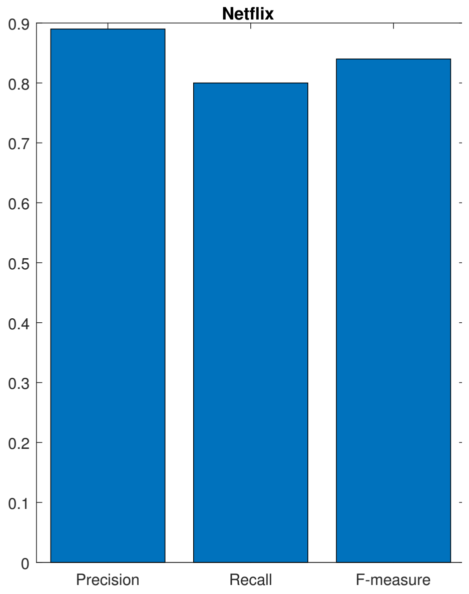


b) Netflix

Figure 4. RMSE graph of the proposed method for MovieLens 100K (left) and Netflix (right) datasets with different values of K-fold cross-validation

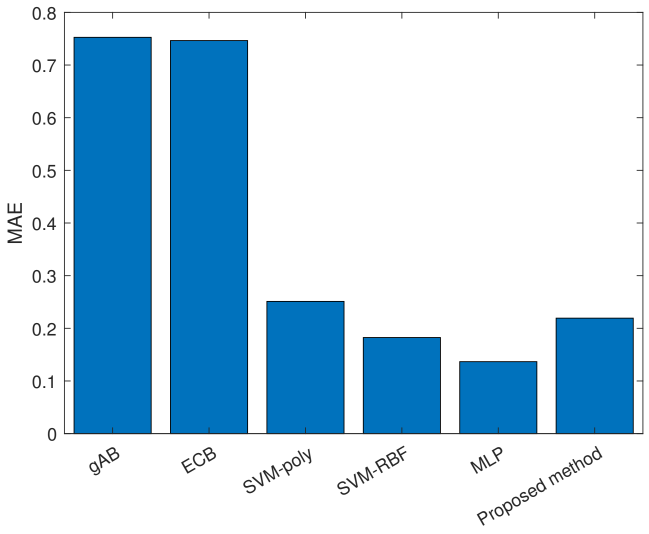


a) MovieLens 100K

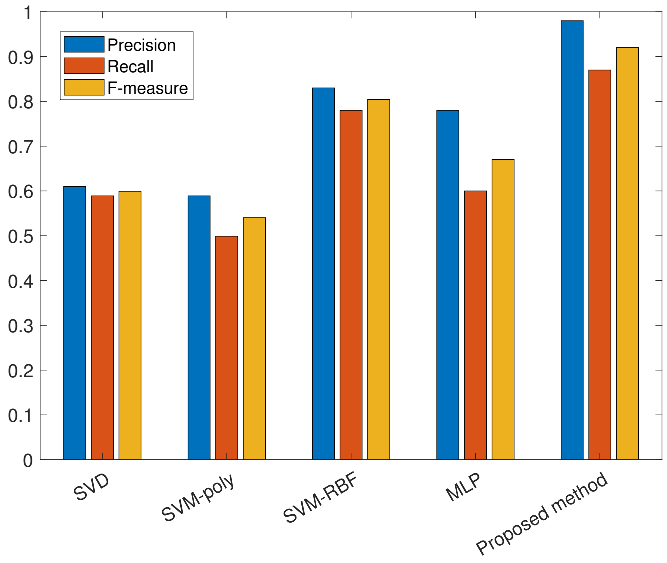


b) Netflix

Figure 5. Decision support measure of the proposed system



a) MovieLens 100K



b) Netflix

Figure 6. MAE (a)) and Decision support measure (b)) comparison of the proposed method for MovieLens 100K dataset with 5-fold cross-validation

MAE				
Proposed method	SVD	ESVD	IMULT	BNMF
0.3011	3.2904	0.9201	0.7094	0.6760

RMSE				
Proposed method	SVD	ESVD	IMULT	BNMF
0.5102	3.7612	0.9615	0.9160	0.9229

Table 13. MAE and RMSE comparison of the proposed method for MovieLens 10M dataset with 5-fold cross-validation

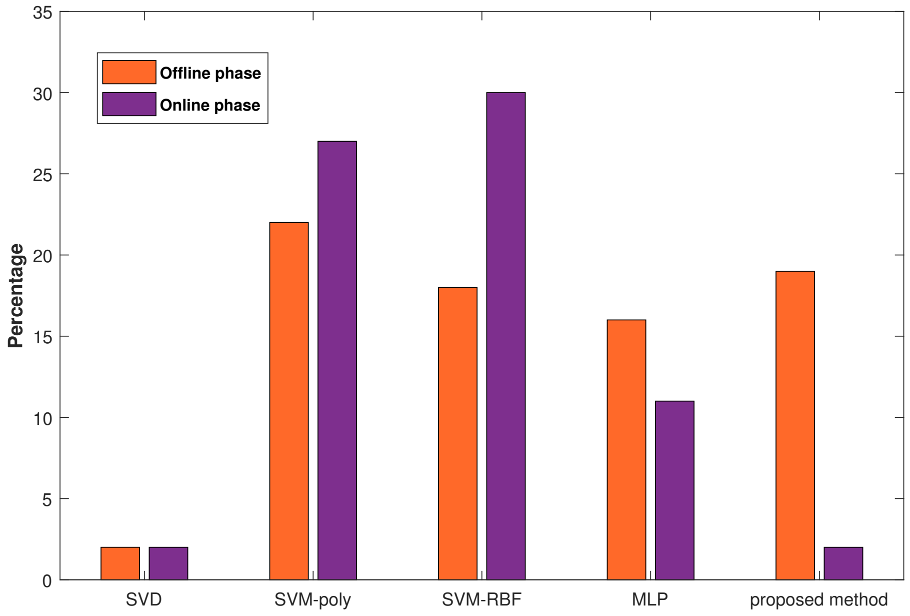


Figure 7. Computational time comparison for MovieLens 100K dataset with 5-fold cross-validation

6 CONCLUSION

In this paper, the radial basis functions network is used for developing a collaborative filtering recommendation approach. The proposed system has offline and online phases. In the offline phase, the sparse user-item rating matrix is completed by using radial basis functions network. Then the full rating matrix is used to rank all the users by solving an eigenvalue problem according to the Google page ranking strategy. In the online phase, users ranking vector is used to better weight the average of deviations from the neighbor's mean and the active users get recommendation based on their likes and dislikes. We overcome the scalability problem by clustering the users, the sparsity problem by completing the sparse rating matrix, and the new user cold-start problem by recommending the top-rated items of the high-ranked user. The effectiveness of the proposed system is supported by some empirical studies on the MovieLens and Netflix datasets.

REFERENCES

[1] Jannach, D.—Zanker, M.—Felfernig, A.—Friedrich, G.: Recommender Systems: An Introduction. Cambridge University Press, 2010, doi: 10.1017/CBO9780511763113.

[2] Varga, E.: Recommender Systems. Chapter 8. Practical Data Science with Python 3, Apress, Berkeley, CA, 2019, pp. 317–339, doi: 10.1007/978-1-4842-4859-1_8.

[3] Lee, S.: Using Entropy for Similarity Measures in Collaborative Filtering. Journal of Ambient Intelligence and Humanized Computing, Vol. 11, 2020, No. 1, pp. 363–374, doi: 10.1007/s12652-019-01226-0.

[4] Feuerverger, A.—He, Y.—Khatri, S.: Statistical Significance of the Netflix Challenge. Statistical Science, Vol. 27, 2012, No. 2, pp. 202–231, doi: 10.1214/11-STS368.

[5] Koren, Y.: Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08), 2008, pp. 426–434, doi: 10.1145/1401890.1401944.

[6] Resnick, P.—Iacovou, N.—Suchak, M.—Bergstrom, P.—Riedl, J.: GroupLens: An Open Architecture for Collaborative Filtering of Netnews. Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW '94), 1994, pp. 175–186, doi: 10.1145/192844.192905.

[7] Breese, J. S.—Heckerman, D.—Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI '98), 1998, pp. 43–52, doi: 10.48550/arXiv.1301.7363.

[8] Feng, L.—Zhao, Q.—Zhou, C.: Improving Performances of Top-N Recommendations with Co-Clustering Method. Expert Systems with Applications, Vol. 143, 2020, Art. No. 113078, doi: 10.1016/j.eswa.2019.113078.

[9] Adomavicius, G.—Tuzhilin, A.: Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, Vol. 17, 2005, No. 6, pp. 734–749, doi: 10.1109/TKDE.2005.99.

[10] Singh, M.: Scalability and Sparsity Issues in Recommender Datasets: A Survey. Knowledge and Information Systems, Vol. 62, 2020, pp. 1–43, doi: 10.1007/s10115-018-1254-2.

[11] Su, X.—Khoshgoftaar, T. M.: A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence, Vol. 2009, 2009, Art. No. 421425, doi: 10.1155/2009/421425.

[12] Balabanović, M.—Shoham, Y.: Fab: Content-Based, Collaborative Recommendation. Communications of the ACM, Vol. 40, 1997, No. 3, pp. 66–72, doi: 10.1145/245108.245124.

[13] Chu, P. M.—Mao, Y. S.—Lee, S. J.—Hou, C. L.: Leveraging User Comments for Recommendation in E-Commerce. Applied Sciences, Vol. 10, 2020, No. 7, Art. No. 2540, doi: 10.3390/app10072540.

[14] Brynjolfsson, E.—Hu, Y.—Simester, D.: Goodbye Pareto Principle, Hello Long Tail: The Effect of Search Costs on the Concentration of Product Sales. Management Science, Vol. 57, 2011, No. 8, pp. 1373–1386, doi: 10.1287/mnsc.1110.1371.

[15] Schwartz, B.: The Paradox of Choice: Why More Is Less. Ecco New York, 2004.

[16] Ricci, F.—Rokach, L.—Shapira, B.: Recommender Systems: Introduction and Challenges. Chapter 1. In: Ricci, F., Rokach, L., Shapira, B. (Eds.): Recommender Systems Handbook. Springer, Boston, MA, 2015, pp. 1–34, doi: 10.1007/978-1-4899-7637-6_1.

[17] Resnick, P.—Varian, H. R.: Recommender Systems. Communications of the ACM, Vol. 40, 1997, No. 3, pp. 56–58, doi: 10.1145/245108.245121.

[18] Burke, R.: Hybrid Web Recommender Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (Eds.): The Adaptive Web. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4321, 2007, pp. 377–408, doi: 10.1007/978-3-540-72079-9_12.

[19] Gomez-Uribe, C. A.—Hunt, N.: The Netflix Recommender System: Algorithms, Business Value, and Innovation. ACM Transactions on Management Information Systems (TMIS), Vol. 6, 2016, No. 4, Art. No. 13, doi: 10.1145/2843948.

[20] Linden, G.—Smith, B.—York, J.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing, Vol. 7, 2003, No. 1, pp. 76–80, doi: 10.1109/MIC.2003.1167344.

[21] Aggarwal, C. C.: Advanced Topics in Recommender Systems. Chapter 13. Recommender Systems, Springer, Cham, 2016, pp. 411–448, doi: 10.1007/978-3-319-29659-3_13.

[22] Jiang, M.—Zhang, Z.—Jiang, J.—Wang, Q.—Pei, Z.: A Collaborative Filtering Recommendation Algorithm Based on Information Theory and Bi-Clustering. Neural Computing and Applications, Vol. 31, 2019, No. 12, pp. 8279–8287, doi: 10.1007/s00521-018-3959-2.

[23] Zhang, F.—Qi, S.—Liu, Q.—Mao, M.—Zeng, A.: Alleviating the Data Sparsity Problem of Recommender Systems by Clustering Nodes in Bipartite Networks. Expert Systems with Applications, Vol. 149, 2020, Art. No. 113346, doi: 10.1016/j.eswa.2020.113346.

[24] Nikolakopoulos, A. N.—Ning, X.—Desrosiers, C.—Karypis, G.: Trust Your Neighbors: A Comprehensive Survey of Neighborhood-Based Methods for Recommender Systems. In: Ricci, F., Rokach, L., Shapira, B. (Eds.): Recommender Systems Handbook. Springer, New York, NY, 2022, pp. 39–83, doi: 10.1007/978-1-0716-2197-4_2.

[25] Koren, Y.: Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. Proceedings of the 14th ACM SIGKDD International

Conference on Knowledge Discovery and Data Mining, 2008, pp. 426–434, doi: 10.1145/1401890.1401944.

[26] Salakhutdinov, R.—Mnih, A.—Hinton, G.: Restricted Boltzmann Machines for Collaborative Filtering. Proceedings of the 24th International Conference on Machine Learning (ICML '07), 2007, pp. 791–798, doi: 10.1145/1273496.1273596.

[27] Georgiev, K.—Nakov, P.: A Non-IID Framework for Collaborative Filtering with Restricted Boltzmann Machines. International Conference on Machine Learning, Proceedings of Machine Learning Research (PMLR), Vol. 28, 2013, pp. 1148–1156.

[28] Sainath, T. N.—Kingsbury, B.—Sindhwani, V.—Arisoy, E.—Ramabhadran, B.: Low-Rank Matrix Factorization for Deep Neural Network Training with High-Dimensional Output Targets. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 6655–6659, doi: 10.1109/ICASSP.2013.6638949.

[29] Yang, B.—Lei, Y.—Liu, J.—Li, W.: Social Collaborative Filtering by Trust. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 39, 2017, No. 8, pp. 1633–1647, doi: 10.1109/TPAMI.2016.2605085.

[30] de Souza da Silva, E.—Langseth, H.—Ramampiaro, H.: Content-Based Social Recommendation with Poisson Matrix Factorization. In: Ceci, M., Hollmén, J., Todorovski, L., Vens, C., Džeroski, S. (Eds.): Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2017). Springer, Cham, Lecture Notes in Computer Science, Vol. 10534, 2017, pp. 530–546, doi: 10.1007/978-3-319-71249-9_32.

[31] Ren, Z.—Liang, S.—Li, P.—Wang, S.—de Rijke, M.: Social Collaborative Viewpoint Regression with Explainable Recommendations. Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM '17), 2017, pp. 485–494, doi: 10.1145/3018661.3018686.

[32] He, X.—Liao, L.—Zhang, H.—Nie, L.—Hu, X.—Chua, T. S.: Neural Collaborative Filtering. Proceedings of the 26th International Conference on World Wide Web (WWW '17), 2017, pp. 173–182, doi: 10.1145/3038912.3052569.

[33] Wu, Y.—DuBois, C.—Zheng, A. X.—Ester, M.: Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM '16), 2016, pp. 153–162, doi: 10.1145/2835776.2835837.

[34] Jiang, Y.—Chen, H.—Yang, B.: Deep Social Collaborative Filtering by Trust. Proceedings of 2018 International Conference on Big Data Technologies (ICBDT '18), 2018, pp. 52–56, doi: 10.1145/3226116.3226121.

[35] Wang, M.—Wu, Z.—Sun, X.—Feng, G.—Zhang, B.: Trust-Aware Collaborative Filtering with a Denoising Autoencoder. Neural Processing Letters, Vol. 49, 2019, No. 2, pp. 835–849, doi: 10.1007/s11063-018-9831-7.

[36] Broomhead, D. S.—Lowe, D.: Multivariable Functional Interpolation and Adaptive Networks. Complex Systems, Vol. 2, 1988, No. 3, pp. 321–355.

[37] Zhang, D. Q.—Chen, S. C.: Clustering Incomplete Data Using Kernel-Based Fuzzy C-Means Algorithm. Neural Processing Letters, Vol. 18, 2003, No. 3, pp. 155–162, doi: 10.1023/B:NEPL.0000011135.19145.1b.

[38] Majdisova, Z.—Skala, V.: Radial Basis Function Approximations: Comparison

and Applications. Applied Mathematical Modelling, Vol. 51, 2017, pp. 728–743, doi: 10.1016/j.apm.2017.07.033.

[39] Wu, Y.—Wang, H.—Zhang, B.—Du, K. L.: Using Radial Basis Function Networks for Function Approximation and Classification. International Scholarly Research Notices, Vol. 2012, 2012, Art. No. 324194, doi: 10.5402/2012/324194.

[40] Du, K. L.—Swamy, M. N. S.: Neural Networks in a Softcomputing Framework. Springer, 2006, doi: 10.1007/1-84628-303-5.

[41] Du, K. L.: Clustering: A Neural Network Approach. Neural Networks, Vol. 23, 2010, No. 1, pp. 89–107, doi: 10.1016/j.neunet.2009.08.007.

[42] Golub, G. H.—Van Loan, C. F.: Matrix Computations. Fourth Edition. Johns Hopkins University, 2012.

[43] Mohammadi, M.—Naree, S. A.—Lati, M.: User-Item Content Awareness in Matrix Factorization Based Collaborative Recommender Systems. Intelligent Data Analysis, Vol. 24, 2020, No. 3, pp. 723–739, doi: 10.3233/IDA-194599.

[44] Eldén, L.: Matrix Methods in Data Mining and Pattern Recognition. Siam, Fundamentals of Algorithms Book Series, 2019.

[45] Meyer, C. D.: Matrix Analysis and Applied Linear Algebra. Siam, Other Titles in Applied Mathematics Book Series, Vol. 71, 2000.

[46] Devi, M. K. K.—Venkatesh, P.: Smoothing Approach to Alleviate the Meager Rating Problem in Collaborative Recommender Systems. Future Generation Computer Systems, Vol. 29, 2013, No. 1, pp. 262–270, doi: 10.1016/j.future.2011.05.011.

[47] Bennett, J.—Lanning, S.: The Netflix Prize. Proceedings of KDD Cup and Workshop (KDDCup '07), 2007, pp. 3–6.

[48] Harper, F. M.—Konstan, J. A.: The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems, Vol. 5, 2016, No. 4, Art. No. 19, doi: 10.1145/2827872.

[49] Hernando, A.—Bobadilla, J.—Ortega, F.: A Non Negative Matrix Factorization for Collaborative Filtering Recommender Systems Based on a Bayesian Probabilistic Model. Knowledge-Based Systems, Vol. 97, 2016, pp. 188–202, doi: 10.1016/j.knosys.2015.12.018.

[50] Ranjbar, M.—Moradi, P.—Azami, M.—Jalili, M.: An Imputation-Based Matrix Factorization Method for Improving Accuracy of Collaborative Filtering Systems. Engineering Applications of Artificial Intelligence, Vol. 46, 2015, pp. 58–66, doi: 10.1016/j.engappai.2015.08.010.

[51] Guan, X.—Li, C. T.—Guan, Y.: Matrix Factorization with Rating Completion: An Enhanced SVD Model for Collaborative Filtering Recommender Systems. IEEE Access, Vol. 5, 2017, pp. 27668–27678, doi: 10.1109/ACCESS.2017.2772226.

[52] Krasnoshchok, O.—Lamo, Y.: Extended Content-Boosted Matrix Factorization Algorithm for Recommender Systems. Procedia Computer Science, Vol. 35, 2014, pp. 417–426, doi: 10.1016/j.procs.2014.08.122.

Maryam Mohammadi is Assistant Professor of applied mathematics (numerical analysis) in the Faculty of Mathematical Sciences and Computer at the Kharazmi University in Tehran, Iran. She received her Master's degree in 2009 and her Ph.D. in 2013, both in numerical analysis at the Isfahan University of Technology. Her main research interests are numerical analysis, numerical linear algebra, numerical solution of integer and fractional partial differential equations, numerical simulation, meshless methods, reproducing kernel Hilbert space methods, radial basis functions methods, matrix methods in data mining and pattern recognition, kernel methods in machine learning, recommender systems and topics related to interaction between numerical analysis and other fields such as engineering and computer science.

Somaye Arabi Naree is Assistant Professor of computer science in the Faculty of Mathematical Sciences and Computer at the Kharazmi University in Tehran, Iran. She received her Ph.D. in 2013 in computer science at the Iran University of Science and Technology. Her main research interests are machine learning, data analysis, recommender systems, numerical linear algebra, matrix methods in data mining and pattern recognition, image processing and other topics related to data science.

Mohammad-Ali Naseri has his Master's degree in applied mathematics (numerical analysis) from the Kharazmi University of Tehran in Iran. Also, he received his Bachelor's degree in applied mathematics from the Vali-e-Asr University of Rafsanjan in Iran. His research interests are machine learning, radial basis functions methods, numerical linear algebra, and matrix methods in data mining. He has recently collaborated with several companies on a multi-website SEO project using numerical linear algebra and data mining methods.