Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-18885-7

Euclidean embedding with preference relation for recommender systems

V Ramanjaneyulu Yannam[1] · Jitendra Kumar[1] · Korra Sathya Babu[2] · Bidyut Kumar Patra[1]

Received: 16 January 2022 / Revised: 8 March 2024 / Accepted: 11 March 2024

Abstract
Recommender systems (RS) help users pick the relevant items among numerous items that are available on the internet. The items may be movies, food, books, etc. The Recommender systems utilize the data that is fetched from the users to generate recommendations. Usually, these ratings may be explicit or implicit. Explicit ratings are absolute ratings that are generally in the range of 1 to 5. While implicit ratings are derived from information like purchase history, click-through rate, viewing history, etc. Preference relations are an alternative way to represent the users' interest in the items. Few recent research works show that preference relations yield better results compared to absolute ratings. Besides, in RS, the latent factor models like Matrix Factorization (MF) give accurate results especially when the data is sparse. Euclidean Embedding (EE) is an alternative latent factor model that yields similar results as MF. In this work, we propose a Euclidean embedding with preference relation for the recommender system. Instead of using the inner product of items and users' latent factors, Euclidean distances between them are used to predict the rating. Preference Relations with Matrix Factorization (MFPR) produced better recommendations compared to that of traditional matrix factorization. We present a collaborative model termed EEPR in this work. The proposed framework is implemented and tested on two real-world datasets, MovieLens-100K and Netflix-1M to demonstrate the effectiveness of the proposed method. We utilize popular evaluation metric for recommender systems as precision@K. The experimental outcomes

Korra Sathya Babu and Bidyut Kumar Patra contributed equally to this work.

✉ V Ramanjaneyulu Yannam
518cs1003@nitrkl.ac.in

Jitendra Kumar
520CS1001@nitrkl.ac.in

Korra Sathya Babu
ksb@iiitk.ac.in

Bidyut Kumar Patra
patrabk@nitrkl.ac.in

1 National Institute of Technology Rourkela, Rourkela, India

2 Indian Institute of Information Technology, Design and Manufacturing, Kurnool, India

Published online: 02 April 2024

show that the proposed model outperforms certain state-of-the-art existing models such as MF, EE, and MFPR.

Keywords Recommender system (RS) · Collaborative filtering (CF) · Cold start problem · Preference relation · Matrix factorization (MF) · Euclidean Embedding (EE)

1 Introduction

Recommender Systems (RS) are software applications and approaches that make recommendations of relevant items that a user would find useful. These suggestions are related to a variety of topics like deciding to buy e-commerce items, what song to listen to next, what news articles to read etc. Users and items are the two basic terms that are often encountered in RS. Users are the customers or consumers while the items may be movies, hotels, or any real-world objects that are consumable by the users. RS typically specializes in a single sort of item like movies, as a result, its basic recommendation mechanism is to generate and deliver relevant recommendations for that specific sort of item [16]. Recommendation engines are built with the help of information about consumers and items. Recommendations are generally personal to the individual, various users or user groups will get different suggestions [18]. Non-personalized recommendations also exist. It is much easier to make suggestions based on their popularity. While these types of non-personalized suggestions may be relevant in some cases, they are not much explored in RS.

Necessity of RS [28] Here, we discussed the necessity of RS in the modern era from both the customer and industry point of view.

- **User point of view**: In this information era, an abundance of data is available on the internet. It is almost impossible to manually choose an item among the numerous available items. We face these types of problems in various real-world scenarios like choosing a hotel, movie, etc. Recommender systems (RS) solve this problem. They suggest items to the user such that the recommended items are relevant and accurate enough to satisfy the user.
- **Business point of view**: In some businesses, RS is crucial since it can produce a large amount of revenue or serve as a method to differentiate a company from its competitors.

Various approaches The two main approaches in which recommender systems recommend items are Content-based Filtering and Collaborative Filtering [30]. A hybrid approach is an ensemble model of the other two methods. In content-based filtering, information about a particular item or user decides the recommendations. In Collaborative Filtering, recommendations are based on the historical relationship among different users and items. Such interactions are generally captured as feedback from users either directly or indirectly. These are known as explicit and implicit feedback. Using this information, ratings are predicted for any given user and item. The main idea behind Collaborative Filtering is, that users will agree now if they agreed previously [1].

There are mainly two different approaches in which collaborative filtering is applied. It is memory-based and model-based. Memory-based techniques are simple compared to model-based techniques. Memory-based approaches are generally user-based and item-based. In a user-based technique, items are recommended to a user such that those items are highly liked by similar users (Fig. 1). In the same way, in an item-based approach, items are suggested such that those items receive similar feedback from the same user [29]. Model-based techniques employ Machine Learning (ML)and Deep-Learning (DL) methodologies to train the

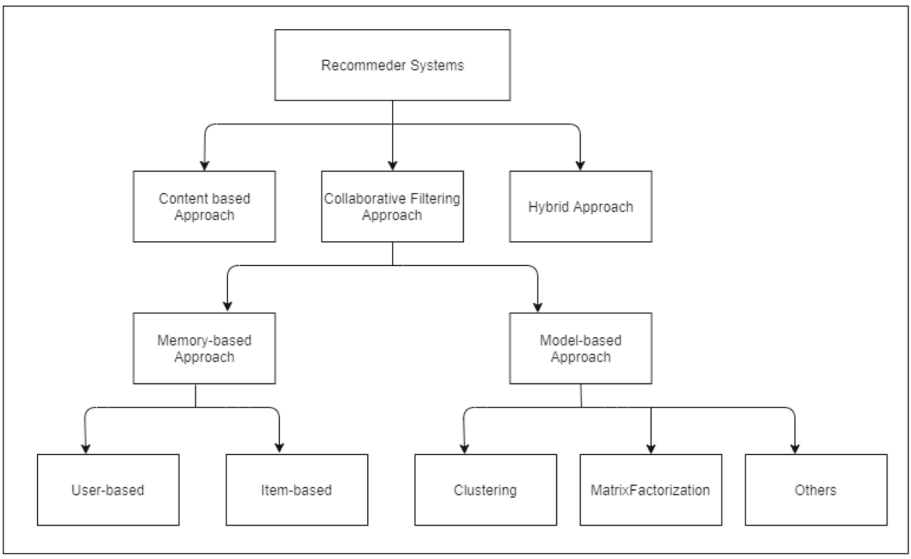

Fig. 1 Categorisation of recommender systems [32]

model. As a result, model-based techniques yield better results compared to memory-based techniques. Model-based methods are generally Matrix Factorization(MF) based methods, clustering techniques, and others.

We consider only the movie recommendation system throughout this work. The application of recommender systems to movies became popular with the Netflix prize competition announcement in 2006 [1]. The winners of this competition used a latent factor model called Matrix Factorization. Rating to an item given by any user is predicted by using the latent factors of the user to that particular item. Euclidean Embedding is an alternative latent factor model that yields similar results as Matrix Factorization. In the Matrix Factorization technique [4], explicit ratings that are in the range of 1 to 5 are used to train the model. Desarkar et al. [2], the absolute ratings which are in the range of 1 to 5 are not used directly. Instead, they are converted into preference relations. This is because the preference relation-based collaborative filtering yields better results compared to that of absolute ratings as mentioned in [2]. In this work, we propose a latent factor technique EEPR for the recommender system in Collaborative Filtering Framework. Here, instead of using absolute ratings to train the model, preference relations are used to predict the rating. Here we try to validate the hypothesis, whether higher-order information like preference relations derived from the item ratings, improves the existing latent factor models. Euclidean distance is measured between any user and item rather than using the dot product. There are many open-source datasets available for building RS like MovieLense-100k, 1M, 10M, 25M, TripAdvisor, Netflix-1M, Yahoo Music etc. In this work we focus only on the movie RS dataset, detailed in Section 5.

The major challenge of the proposed methodology is extracting a dataset from one of the popular datasets due to hardware limitations. This is the reason, the majority of researchers are not working in collaborative filtering based on preference relation. eg. in our case, extracting dataset has taken 3 to 4 days. The major contribution of this paper is discussed below.

[1] https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data

- The training dataset is formulated by using preference relation.
- Euclidean Embedding is applied to the formulated dataset.
- We use two datasets and compare them with existing state-of-the-art.
- We validate the hypothesis, that higher-order information like preference relations derived from the item ratings, improves the recommendation performance of existing latent factor models.

The structure of the rest of the paper is as follows: Literature, which is related to this paper discussed in Section 2. Section 3 explores the background work. Section 4 discusses the proposed methodology. Section 5 explains the experimental setup, evaluation strategy, and used dataset. Results and their analysis are given in Section 6. Conclusion and future work are given in Section 7.

2 Related work

2.1 Collaborative filtering approaches

The collaborative filtering (CF) approach is the most popular methodology that is often employed in the domain of recommendation systems. This method's main goal is to predict the ratings from the already available ratings. CF is usually implemented using two methods. The first one is neighborhood-based methods which are also referred to as Memory-based techniques. The second one is model-based techniques. Neighborhood methods are called memory-based because they require a complete data set of ratings to predict the recommendations. User-based and item-based K-NN are well-known memory-based techniques, as discussed in [9, 17].

The user-based approach predicts ratings based on users who are most liked by the given user, while the item-based approach predicts ratings based on the most similar items to the given items. Many similarity metrics like cosine similarity (COS), Pearson correlation (PCC), constrained Pearson correlation (CPCC), sigmoid PCC, weighted PCC etc. are available in the literature. Hyung et al. [33] proposed a similarity technique named as PIP. They have addressed the cold start user problem. Later, Zheng et al. [35] improve the accuracy compared to PIP in terms of cold start. They have given the solution for the drawbacks of the memory-based similarity technique. The drawbacks of the similarity techniques are the low similarity between two similar users, and the high similarity between two unlike users, ignoring the proportion of common ratings of two users. Sreepada et al. [27] proposed a revisited tendency-based approach (RTCF) to address a new cold start user problem. They introduce a stable neighborhood with user and item tendencies. The authors in [6] treated the recommender system as a classification problem. They have used Naive Bayes Classifier (NBC) to provide the recommendations. NBC is a probabilistic model, where the posterior probability is computed using the prior and likelihood probabilities from the training data set. Explicit ratings provided by the users are considered as features. It is treated as a multi-class classification problem where each class corresponds to an integer value on a rating scale of 1 to 5. The maximum a posterior (MAP) rule is used to predict the class label. The results computed by this model are better compared to that of neighborhood techniques. [2].

On the other side, model-based methods learn the model parameters during the training phase with the help of a training data set that contains ratings (Table 1). Later, in the prediction phase, they predict ratings using the parameters that are learned during the training phase.

[2] https://grouplens.org/datasets/movielens/

Table 1 Literature survey for proposed methodology

S.No	Year	Author	Technique used	Remarks
1	2008	Hyung et al. [33]	PIP	Gives solution for existing memory-based(PCC,CPCC,WPCC,SPCC,COS,ACOS e.t.c) in terms of cold-start and data sparsity.
2	2009	Robert et al. [4]	MF	Gives solution for data sparsity and scalability.
3	2010	NIck et al. [3]	EE	Gives solution for cold-start and scalability. Accuracy improved than MF.
4	2012	Desarkar et al. [5]	MFPR	Introduce preference in latent factor model to improve data sparsity and top-N recommendation.
5	2014	Mariya et al. [34]	BMF	Based on latent factor model. Introduce a lower and an upper bound on every estimated missing element of the rating matrix.
6	2014	Zheng et al. [35]	NHSM	Improving the accuracy in terms of cold start and data sparsity.
7	2016	Hernando et al. [14]	NMF	Improve accuracy in terms of scalability.
8	2018	Sreepada et al. [27]	RTCF	Introduce stable neighbour with user tendency and item tendency.
9	2019	Valdiviezo et al. [6]	NBCF	Increased accuracy of TOP-N recommendation.
10	2021	Nahata et al. [26]	MF with meta data	Improve the accuracy in terms of cold start.
11	2022	Anwar et al. [42]	CF with K-NNBaseline	Gives solutions for cold start and sparsity issues.
12	2021	Chen et al. [43]	DDCF	Captures the preference variations of users and includes the concept of dynamic time decay
13	2021	Thaipisutikul et al. [44]	CReS	Capture users' dynamic preferences by modeling the hierarchical relationships between contexts and items in a particular session.
14	2022	Bobadilla et al. [45]	DeepMF and NCF	Incorporates visual representation of data.
15	2022	Thaipisutikul et al. [46]	ATCRec	Gives solutions for the sequential recommendation problem.
16	2022	Ni et al. [47]	TSEM	Leverages item multimodal auxiliary information to substantially improve recommendation performance.
17	2022	Thaipisutikul et al. [48]	SCR	An Improved Sequential Model for next POI recommendation.
18	2021	Thaipisutikul et al. [49]	LSCAR	Improves recommendation accuracy by capturing contexts that could differently influence current users' decision-making.

Latent factor models are popular and these models belong to model-based approaches. One of the most successful strategies among latent factor models is matrix factorization (MF) [4]. The rating matrix, known as the user-item matrix is factorized into two matrices, namely the item-feature matrix and the user-feature matrix. The user rating to any item is predicted by the dot product of the user's and an item's feature vectors. There are many variations of MF like NMF(Non-Negative Matrix Factorization), and BNMF(Bayesian Non-Negative Matrix Factorization). A rating matrix is factored into two matrices, W and H, using NMF [13]. The values inside these matrices should be greater than zero always. Each user and item in BNMF [14] is associated with a vector of K components. The components of these vectors have probabilistic significance and allow for the identification of groups of users with similar tastes, along with the justification of the recommendations provided by this technique. Another variation of MF is proposed in [8]. DeepMF implements MF in a layered architecture. Besides MF, there are other latent factor models in the literature like Euclidean Embedding. It is an unsupervised learning model [12] used as dimensionality reduction. But most of the time it is used for visualization purposes [11]. It is similar to MF except for the fact that Euclidean distance is used in predicting rating instead of the inner product between the user and item features. The recommendations provided by the EE model are better compared to that of the MF model. This approach also allows for online implementation criteria like adding new users or objects to an existing model [3]. Euclidean Embedding techniques are widely explored with Deep Neural Networks [7].

Nicolas et al. [10], they discuss various types of ratings that are used in recommender systems and show that preference relations give better information than absolute ratings. Later on, Armelle et al. [15] proposes a collaborative model for predicting ratings and gives suggestions using it. The item's position in the preference series of the similarly rated individual user is used to predict its significance. The items on the recommendation list are arranged in decreasing order of their expected utility. Recommendations are generated in the decreasing order of the items' estimated utilities. Desarkar et al. [2], proposed matrix factorization with preference relations (MFPR) in the place of actual ratings. They extended this work in [5] such that the predicted ratings are in the range of 1 to 5.

2.2 Content-based approaches

Content-Based(CB) RS suggests things to the users depending on their previous ratings. To give relevant information to the user, a user profile with certain features of the items must be developed utilizing data mining and other information retrieval methods. In the content-based RS, the similarity of the items that the user is interested in is used to filter things. It either suggests or examines items depending on high-rated items that are comparable to the user's preferences [21]. The problem with this system is that sometimes items are recommended in coincidence.

Pasquale et al. [22] proposed a Content-based RS that builds the profiles for users and items. It works in such a way that the performance gradually increases over time. Implicit feedback is used to implement and evaluate the model. Tewari et al. [23] developed a book RS by using multiple features like quality, book description, etc. They also used the collaboration of other users to recommend books. The model built is offline and has no performance issues. The major problem with this approach is cold start still exists. Nahtal et al. [26] introduced Meta Embedding Deep Collaborative Filtering (MEDCF) for rating prediction in the cold-start scenarios.

2.3 Application of recommender system(RS)

Recommendation System finds huge use in various domains. Some of the examples are discussed below.

- TV program recommendation: Smart TV sets are growing fast. Nowadays, Authorised user is allowed to provide preference on TV. These preferences are used during the recommendation scenarios of TV. Tivo [36] permits authorized users to give preference using remote control. Suitable programs are efficiently recommended using CF. Hyeong et al. [37] gave a unique similarity method that applies raw moment based similarity. It is a memory-based CF to resolve the cold start problem and cost optimization. The software queveo.tv is originated by Barragáns-Martínez et al. [38], which combines the CB approach and item-based CF approach to address the cold start problem. Zimmerman et al. [39] made recommendations based on the implicit and explicit feedback from authorized users. It infers the approximate interest of the user. Govind Kumar Jha et al. [41] proposed a trustworthy approach for recommender systems in the scenario of movie communities. They have developed the model using a machine learning approach and introduced an inversion similarity procedure in order to obtain efficient recommendations.

3 Background

In this section, we discuss the baseline paper for the proposed one based on collaborative filtering namely Matrix Factorization, and Euclidean Embedding in a detailed manner. We also discuss the motivation behind using preference relations. In the later part of the section, the detailed formulation of preference relation with matrix factorization is discussed.

The models under discussion are optimization-based Machine Learning(ML) problems. So, here we explain the steps involved in solving any ML problem in RS.

- **Phase1:** This phase is called data prepossessing. Here, we perform data cleaning and remove unwanted things from the data. We divide the entire dataset into training data and testing data.
- **Phase2:** This phase is called model learning. The model is learned using the training data. The steps involved in this phase are explained with the help of Fig. 2.
- **Phase3:** This phase is recommendation generation. In this phase, we generate the recommendations for the user by taking the input as user weights. We compute the score for all the items that are previously not rated by the user, and then items recommended to the user in the decreasing order of their scores.
- **Phase4:** This phase is evaluation. In this phase, the model is evaluated using the test data. Various evaluation metrics like rmse, mae, precision, and recall are used.

3.1 Matrix factorization (MF) [2, 4]

It is one of the most commonly used latent factor models used in RS. The Netflix Prize Competition (2006) winners used this technique to improve the accuracy of the RS. There are many variations of MF that are being proposed in literature even today. This approach is even extended for group RS. We explain the process of MF in detail along with its advantages and limitations in this section. The diagrammatic representation of the MF process is given in Fig. 3.

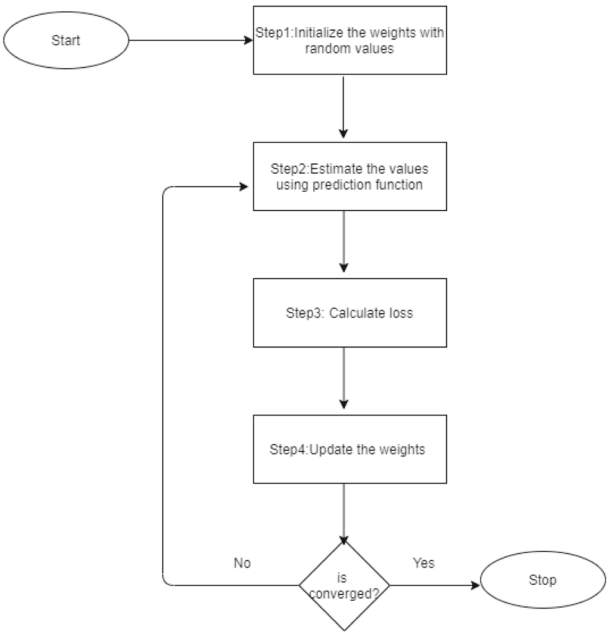

Fig. 2 Flowchart for learning the model using optimization-based machine learning approach

The primary aim of the Matrix Factorization problem is to factorize a rating matrix R into P and Q such that these two matrices can estimate all of R's unknown values. R is a rating matrix that contains the explicit ratings that a user has provided for a particular item in the past. Generally, R is very sparse. P represents user features and Q represents item features. The inner product of P and Q gives the predicted ratings. Consider a scenario in which there are N items and M users in a d-dimensional space, then $R \in \mathbb{R}^{MXN}$, $P \in \mathbb{R}^{MXd}$ and $Q \in \mathbb{R}^{NXd}$. Now, R is factorized in a way such that it is approximately equal to PxQ.

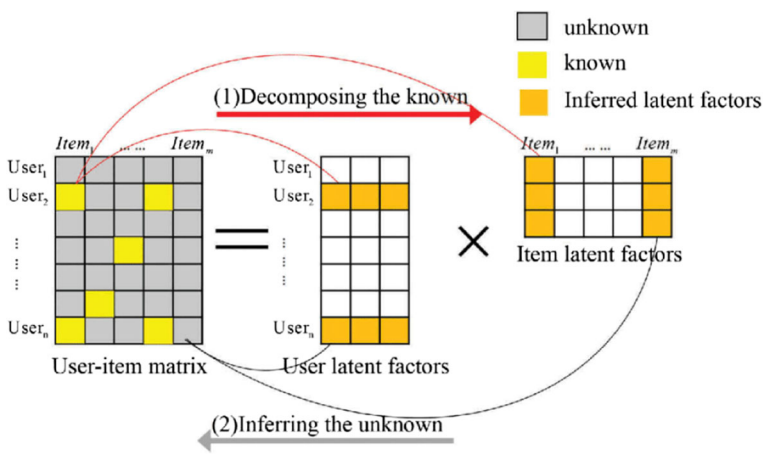

Fig. 3 An overview of MF [19]

Each row of P corresponds to the user vector p_u and quantifies his/her satisfaction levels on the corresponding factors present in an item. Similarly, each row of Q corresponds to an item vector q_i and quantifies the presence of those factors in that item. The dot product of item and user vectors is used to measure the predicted rating for any item i by a given user u using (1).

$$\hat{r}_{ui} = q_i^T p_u \tag{1}$$

The difference between the actual and predicted rating is used to measure the error function using (2)

$$\epsilon_{ui} = r_{ui} - q_i^T p_u \tag{2}$$

The minimization expression is presented by (3).

$$\min_{q,p} \sum_{u,i \in S} (r_{ui} - \hat{r}_{ui})^2 + \lambda(p_u^2 + q_i^2) \tag{3}$$

where λ is regularization variable to avoid overfitting and S is training dataset.

Gradient descent is one way to find matrices P and Q. Solving the above-mentioned minimization function, we get the update (4) and (5).

$$q_i \longleftarrow q_i + \alpha(\epsilon_{ui} p_u - \lambda q_i) \tag{4}$$

$$p_u \longleftarrow p_u + \alpha(\epsilon_{ui} q_i - \lambda p_u) \tag{5}$$

where α is the learning rate.

Advantages of MF The main advantages of MF are:

- It works well with sparse data.
- It is practical to implement.

Limitations of MF [7] The main limitation of MF is the linearity associated with it. Since MF only captures the linear relationship between the items and users, the accuracy of a model is lesser than that of the models that find non-linear patterns in the data. This can be explained with the following example shown in Fig. 4. P_4 is quite closer to P_1, with P_3 and P_2 following closely behind. However, putting P_4 closer to P_1 brings P_4 closer to P_2 than P_3. This is the problem due to linearity.

3.2 Euclidean embedding(EE)

An alternative latent factor model is Euclidean Embedding [3]. All the items and users are placed in an Euclidean space where the distance between them is inversely proportional to their rating. The location of the user defines his/her features, whereas the location of the item represents its features, like genre. As a result, (1) is modified to (6).

$$\hat{r}_{ui} = b_u + b_i + \mu - (x_u - y_i)(x_u - y_i)^T \tag{6}$$

where, b_u represent user bias, b_i represents item bias and μ represents global mean. Here x_u and y_i are user and item vectors in the D-dimensional Euclidean Space. We use squared Euclidean distance because of its computational efficiency. The error function is given in (7).

$$\epsilon_{ui} = r_{ui} - \hat{r}_{ui} \tag{7}$$

The minimization expression is modified as mentioned in (8):

$$\min_{x,y} \sum_{u,i \in S} (r_{ui} - \hat{r}_{ui})^2 + \lambda(x_u - y_i)^2 \tag{8}$$

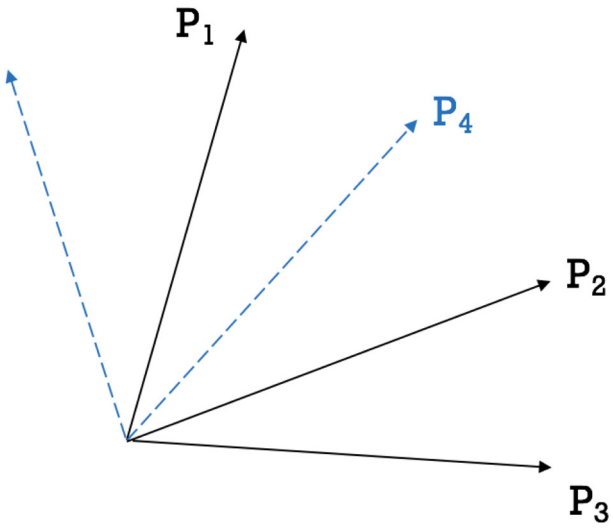

Fig. 4 Linearity issue [7]

where λ is regularization parameter that avoids overfitting and S is training dataset. Here x_u - y_i is included in the regularization term since the distance depends on the relative positions of x_u and y_i rather than individual absolute points. To learn the parameters, we use Stochastic Gradient Descent and the update (9) and (10).

$$x_u \longleftarrow x_u - \alpha(x_u - y_i)(\epsilon_{ui} + \lambda) \tag{9}$$

$$y_i \longleftarrow y_i + \alpha(x_u - y_i)(\epsilon_{ui} + \lambda) \tag{10}$$

3.3 Comparison between EE and MF

In EE, the square of Euclidean distance is used as a similarity component whereas in MF the dot product is used [3]. In EE, the search space is reduced, and hence it can provide faster and better recommendations compared to MF, as shown in Fig. 5.

The main advantage of using EE instead of MF is that the neighbourhood structure of the unified space provides better recommendations than MF. Solving the cold start problem is not trivial in MF, whereas in EE, mapping the new users and items into the existing model is easy. Therefore, it can be used as an online recommender system model.

3.4 Matrix factorization with preference relations [2]

In most of the recommender systems, explicit ratings are used to train and test the model. But preference relations are another way to provide better recommendations. The reason is that a user might give the same rating value to both of the items but may prefer one over the other item. Hence, the best way to recommend an item is using the preference relation rather than the ratings. The following function is used to define the preference relation between any

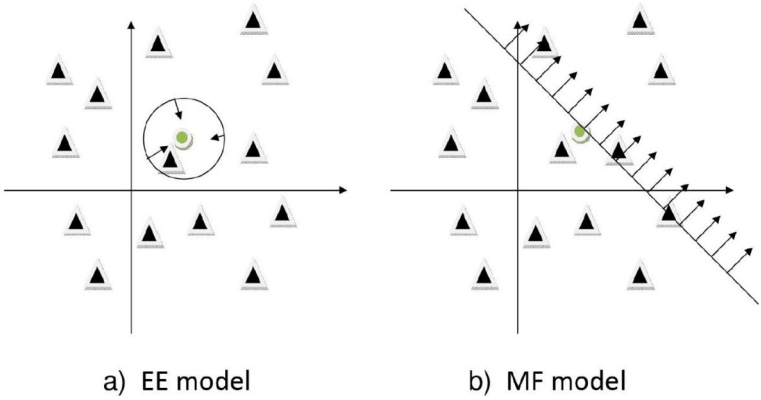

Fig. 5 Comparing EE and MF [3]

item pair(i, j) and the user u in (11).

$$\pi_{u,i,j} = \begin{cases} 1 & \text{if } u \text{ likes } i \text{ more than } j \\ 0.5 & \text{if } u \text{ likes } i \text{ and } j \text{ equally} \\ 0 & \text{if } u \text{ likes } j \text{ more than } i \end{cases} \tag{11}$$

Now, the matrix factorization is implemented with preference relations instead of absolute ratings [2]. The user's preference values for any pair of items (i, j) are $p_u q_i^T$ and $p_u q_j^T$, respectively. If $p_u q_i^T$ is greater than $p_u q_j^T$, the user likes item i more than item j. The following equation can be used to predict the preference relation using (12)

$$\hat{\pi}_{u,i,j} = \frac{e^{p_u (q_i - q_j)^T}}{1 + e^{p_u (q_i - q_j)^T}} \tag{12}$$

Here, we apply the sigmoid function in order to limit the value of $\hat{\pi}_{u,i,j}$ 1 to (13).

$$f(x) = \frac{e^x}{1 + e^x} \tag{13}$$

Figure 6 shows the plot of the sigmoid function whose maximum value is 1.

The error function is calculated using (14).

$$err = \frac{1}{2} \sum_{(u,i,j,\pi(u,i,j)) \in K \& (i<j)} (\pi(u,i,j) - \frac{e^{p_u (q_i - q_j)^T}}{1 + e^{p_u (q_i - q_j)^T}})^2 \tag{14}$$

where K refers to the training dataset.

The minimization expression is modified by (15).

$$\min_{p,q} \frac{1}{2} (err + R(p,q)) \tag{15}$$

where *R(p,q)* is regularization parameter to avoid overfitting. It is defined in (16):

$$R(p,q) = \lambda_p \sum_{u \in U} p_u^2 + \lambda_q \sum_{i \in I} q_i^2 \tag{16}$$

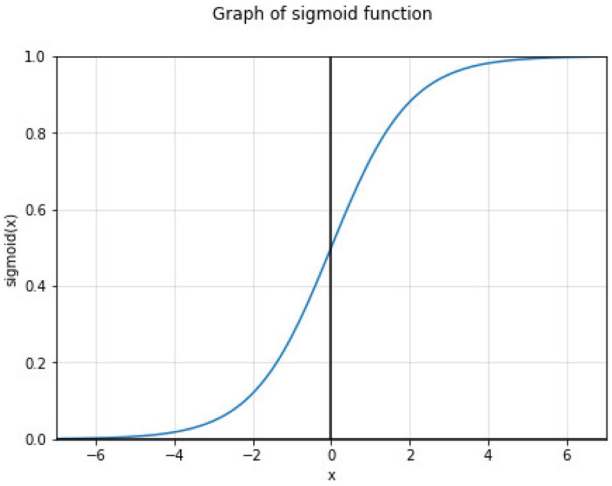

Fig. 6 Plot of sigmoid function [20]

where I and U denote the list of unique items and users, respectively. Like in MF, the goal here is to learn p_u, q_i and q_j. This can be done by using the gradient descent algorithm with the help of the following update in (17), (18), and (19)

$$p_u = p_u + \gamma\left(\frac{err_{uij}\hat{\pi}(u,i,j)(q_i - q_j)}{1 + e^{p_u(q_i - q_j)^T}}\right) + \lambda_p p_u) \tag{17}$$

$$q_i = q_i + \gamma\left(\frac{err_{uij}\hat{\pi}(u,i,j)(p_u)}{1 + e^{p_u(q_i - q_j)^T}}\right) + \lambda_q q_i) \tag{18}$$

$$q_j = q_j - \gamma\left(\frac{err_{uij}\hat{\pi}(u,i,j)(p_u)}{1 + e^{p_u(q_i - q_j)^T}}\right) + \lambda_q q_j) \tag{19}$$

where γ represents the learning rate. After learning the matrices, the next task is recommendation generation. Unlike in MF, items are not directly recommended based on the rating values. Instead, items are recommended in decreasing order based on the scores determined by the following function mentioned in (20), and $x_u(i)$ is the user vector for an item i.

$$x_u(i) = \sum_{j \in I - \{i\}} p_u(q_i - q_j)^T \tag{20}$$

where I denotes list of all unique items.

However, it is unnecessary to compute this scoring function between every pair of possible items. An alternative scoring function is defined in (21) with which we can recommend items:

$$y_u(i) = p_u q_i^T \tag{21}$$

It is observed that both the scoring functions produce exactly the same top −k recommendations. and $y_u(i)$ is user vector for an item i. We prefer to use the latter scoring function since it is computationally cheaper and efficient.

4 Proposed methodology

The details of our proposed work are presented in this section. The proposed work is divided into four phases, namely, Data prep-processing, representation of data, learning the model, making recommendations, and finally, evaluation (Fig. 7).

4.1 Phase-1

Data pre-processing In this step, we perform data cleaning and prepare the data so that the model can learn from the preference values. In general, the current dataset that is available over the internet looks like $< userid, itemid, rating >$. But we need the preference values of each user between pairs of items. The preference value always lies between 0 and 1. So, we need the training dataset in the format $< userid, item_iid, item_jid, preferencevalue >$. Since the dataset doesn't contain the preference relations. Therefore, we derive them using the (11).

User-item representation As mentioned in [6], all items and users are assumed to be in a Euclidean space. In the Euclidean space, if the distance between any user u and any item

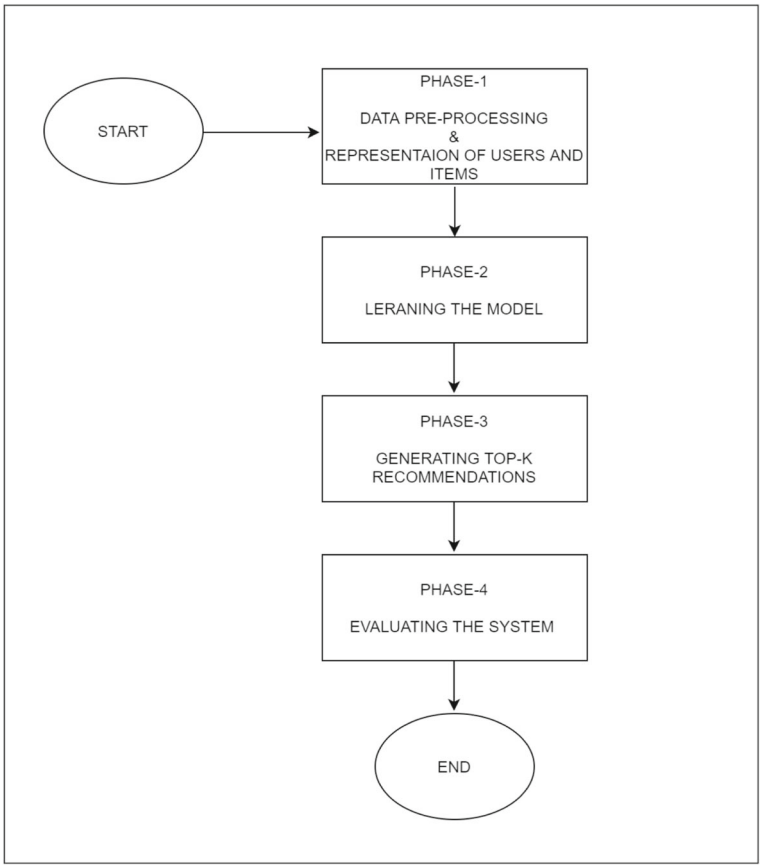

Fig. 7 Flow chart showing the phases in the proposed system approach

i is smaller than the distance between u and any item j, user u prefers item i. x_u, y_i and y_j represents the user u, item i and item j features in euclidean space, respectively.

4.2 Phase-2: model learning with preference relations

The goal of this step is to learn the features of users and items so that the preference relations are predicted.

Step1: Prediction of preference relation can be predicted as mentioned in (22).

$$\hat{\pi}(u, i, j) = \frac{e^{[(x_u - y_j) + (x_u - y_i)][(y_j - y_i)']}}{1 + e^{[(x_u - y_j) + (x_u - y_i)][(y_j - y_i)']}} \tag{22}$$

Here we use the sigmoid function to limit the preference value not to exceeding 1.

Step2: Determining the loss function and minimization expression. The error function is similar to (14), and it is given in (23).

$$err = \frac{1}{2} \sum_{(u,i,j,\pi(u,i,j)) \in K \& (i<j)} (\pi(u, i, j) - \frac{e^{x_u (y_i - y_j)^T}}{1 + e^{x_u (y_i - y_j)^T}})^2 \tag{23}$$

where K is the training dataset.

The optimal values of x_u, y_i and y_j are found using the following minimization expression using (24).

$$\min_{x,y} \frac{1}{2} \sum_{(u,i,j,\pi(u,i,j)) \in K \& (i<j)} (\pi(u, i, j) - \hat{\pi}(u, i, j) + R(x,y)) \tag{24}$$

where *R(x,y)* is the regularization function to reduce the effect of overfitting, and K is the Training set.

Step3: Update Equations. The features of items and users are learned by applying a stochastic gradient descent approach to training data. We get the update Equations for x_u, y_i and y_j using (25), (26), and (27).

$$x_u = x_u + \alpha(\frac{err_{uij} \hat{\pi}(u, i, j)(y_j - y_i)}{1 + e^{[(x_u - y_j) + (x_u - y_i)][(y_j - y_i)']}}) + \lambda_x x_u) \tag{25}$$

$$y_i = y_i - \alpha(\frac{err_{uij} \hat{\pi}(u, i, j)(x_u - y_i)}{1 + e^{[(x_u - y_j) + (x_u - y_i)][(y_j - y_i)']}}) + \lambda_y y_i) \tag{26}$$

$$y_j = y_j + \alpha(\frac{err_{uij} \hat{\pi}(u, i, j)(x_u)}{1 + e^{[(x_u - y_j) + (x_u - y_i)][(y_j - y_i)']}}) + \lambda_y y_j) \tag{27}$$

where λ_x and λ_y are regularization parameters, while α represents the learning rate.

4.3 Phase-3: generating recommendations

After learning the model parameters x_u, y_i and y_j, items are recommended based on the scoring function similar to (20). Here, the score is computed in Euclidean space according to (6). This scoring feature is used to sort items so that the top-most k of those items is provided to the user. Using a scoring function similar to (21), we can improve the process of recommending items. As mentioned in [2], both the scoring functions give exactly the same recommendations. However, the later scoring function is computationally efficient.

4.4 Phase-4: evaluation

The obtained recommendations are evaluated using precision for top-K recommendations. In this methodology, it is not suggested to calculate RMSE and MAE. This is because of the fact that the predicted and actual preferences are not in the same range. The formula for calculating Precision@K is mentioned in the Section 5.

5 Experimental analysis

5.1 Dataset used

We conducted all the experiments on Movielens-100K dataset and Netflix1M dataset (Table 2). The MovieLens-100K dataset contains 943 users and 1682 items, with each user rating minimum of 20 items. A total of 100000 ratings are available for both training and testing. The ratings are in the range of 1 to 5. From the actual Netflix dataset, we used only a portion of it. This reduces the training time. We extracted the ratings in such a way that each user rated a minimum of 20 movies and a maximum of 500 movies. We took the first 1500 movies and removed few ratings such that a total of 1,34,000 ratings are available in the final dataset with 3998 users and 1250 movies. For both datasets, the training data makes up 80% of the overall dataset, while the testing data makes up the remaining 20%. We converted the training datasets into preference relations using (11).

5.2 Experimental setup

We implemented Matrix Factorization with preference relations and EEPR in this work. These experiments are conducted with preference relations that are derived using absolute ratings. The hyper parameters, learning rate α is chosen as 0.01 and regularization parameters λ_p,λ_q are set to 0.001 and 0.0005 respectively. For Matrix Factorization with preference relations while learning rate α is chosen as 0.05 and regularization parameters λ_x,λ_y is set to 0.001 and 0.0005 respectively for EEPR. We compute precision for the top-5 and top-10 recommendations. We used a threshold value of 3 while calculating precision. This process is repeated for different dimensions ranging from 20 to 120 with an interval of 20.

5.3 Evaluation metrics

Here, we provide the details of the metrics used in the model evaluation phase. In RS, especially while implementing the latent factor models, RMSE and MAE are used for evaluation. But, we used precision for the top-K items as the metric. The reason is that the actual values

Table 2 Outline of datasets used

	MovieLens100K	Sample Netflix
# Users	943	3998
# Movies	1682	1250
# Ratings	100000	134000
Rating scale	$1-5$	$1-5$

in the test dataset are in the range of 1 to 5. While the predicted values are not in the range of 1-5. Hence, we sort the items in the decreasing order of their score and will suggest the top-K items. We chose K values as 5 and 10. Precision for top-K [40] items can be found by the following equation:

$$Precision@K = \frac{\#\, top - K\ recommended\ and\ relevant\ items}{\#\, top - K\ recommended\ items} \tag{28}$$

Here, relevant items represent those whose true rating is greater than or equal to the threshold. Recommended items are those generated by the recommender engine. In our experiments, we used three as the threshold.

Table 3 Result on MovieLens-100K dataset

Algorithm	Features	Precision@5	Precision@10
DCFGRS [31]	NA	0.8242	0.8048
MEDCF [26]	20	0.6402	0.6202
	40	0.6612	0.6284
	60	0.7842	0.7468
	80	0.8442	0.8024
	100	0.8572	0.7842
	120	0.8124	0.7924
MF [4]	20	0.8592	0.7579
	40	0.8603	0.7599
	60	0.8607	0.7599
	80	0.8618	0.7601
	100	0.8679	0.7638
	120	0.8666	0.7586
MFPR [5]	20	0.8880	0.7771
	40	0.8884	0.7790
	60	0.8845	0.7819
	80	0.8915	0.7823
	100	0.8897	0.7803
	120	0.8888	0.7797
EE [3]	20	0.8919	0.7801
	40	0.8928	0.7801
	60	0.8936	0.7812
	80	0.8967	0.7823
	100	0.8958	0.7835
	120	0.8949	0.7814
EEPR (Proposed method)	20	0.7908	0.7065
	40	0.7934	0.7069
	60	0.9263	0.8183
	80	0.9372	0.8230
	100	0.9376	0.8233
	120	0.9350	0.8211

Table 4 P-test and t-test on MovieLens-100K

	DCFGRS	MEDCF	MF	MFPR	EE	EEPR
DCFGRS						
MEDCF	0.9309, 0.3737					
MF	10.9473, 6.8932	2.5175, 0.0305				
MFPR	15.9581, 1.9256e-08	3.1925, 0.0096	14.7072, 4.2246e-08			
EE	17.1317, 9.6912e-09	3.3447, 0.0074	19.0821, 3.3954e-09	4.8079, 0.0007		
EEPR(Proposed)	6.1761, 0.0001	4.0631, 0.0022	2.7499, 0.0204	0.0934, 0.9274	0.1203, 0.9065	

6 Results and analysis

Here we provide two tables in which the results of the conducted experiments are recorded. The values of precision for top-5 and top-10 recommendations provided by EEPR and existing techniques on MovieLens-100K dataset in Table 3. Table 5 displays the results on Netflix dataset for different dimensions. As shown in Table 3 precision@5 is 0.6402, 0.8592, 0.8880, 0.8919 for MEDCF, MF, MFPR, EE respectively for feature 20 on MovieLens-100K dataset. The proposed EEPR performance is not good with a value of 0.7908 in a lower dimension only. The proposed method is performing better in higher dimensions and performance is also increasing to 100 number of features. Precision@5 is 0.7842, 0.8607, 0.8845, 0.8936 for MEDCF, MF, MFPR, EE respectively for feature 60. However, our method (EEPR) outperforms the existing method with a value of 0.9263. Similarly, our methods perform better for features 80, 100, 120. Other used metric precision@10 is showing similar results. These results are similar for both used datasets.

From the results Tables 3 and 5, it is observed that the proposed algorithm works better in higher dimensions. Especially for the dimensions greater than 40, EE with preference relations clearly outperforms the MF with preference relations. As shown in Table 3 for the MovieLens-100K dataset, especially in the lower dimensional space EE with preference relations have lesser accuracy. This is due to the dataset constraints. While generating the preference relations from the MovieLens-100K dataset, we did not consider the dense and sparse users separately (Table 4). As shown in Table 5 for the Netflix dataset proposed algorithms work the same as MF with preference even in the lower dimensions. Tables 4 and 6 shows the p-test and t-test.

7 Conclusion and future work

We demonstrated various latent factor models that are popular in Recommender Systems. We also provided the details of Matrix Factorization, Euclidean Embedding, MF with Preference relations, and EE with preference relations techniques with the equations. From the results, we conclude that EE with preference is an alternative latent factor model and is comparable with

Table 5 Result on Netflix-1M Dataset

Algorithm	Features	Precision@5	Precision@10
DCFGRS [31]	NA	0.7982	0.7762
MEDCF [26]	20	0.6824	0.6524
	40	0.7242	0.7024
	60	0.8046	0.7624
	80	0.8254	0.7920
	100	0.8296	0.8024
	120	0.8012	0.7824
MF [4]	20	0.8417	0.7186
	40	0.8514	0.7212
	60	0.8545	0.7235
	80	0.8561	0.7241
	100	0.8565	0.7244
	120	0.8530	0.7230
MFPR [5]	20	0.8974	0.7598
	40	0.8982	0.7604
	60	0.8990	0.7607
	80	0.8992	0.7611
	100	0.8988	0.7596
	120	0.8988	0.7585
EE [3]	20	0.8980	0.7595
	40	0.8980	0.7596
	60	0.8978	0.7599
	80	0.8984	0.7601
	100	0.8982	0.7699
	120	0.8982	0.7594
EEPR (Proposed method)	20	0.8426	0.7309
	40	0.8489	0.7388
	60	0.9227	0.7667
	80	0.9235	0.7776
	100	0.9315	0.7906
	120	0.9319	0.7810

MF with the Preference relations technique. The proposed method clearly outperforms the existing method in higher dimensions as displayed in the results. Hence EE with Preference relations can be used in recommender systems for better recommendations to the user. We also observe that EEPR gives better recommendation results than EE, like MFPR gives better recommendation results than MF. So it can be concluded that higher-order information derived from the first-order information has the potential to improve the recommendation results. However, as a drawback, the proposed model has a greater time complexity compared to MF and EE due to the extraction of preference relations.

Table 6 P-test and t-test on Netflix-1M

	DCFGRS	MEDCF	MF	MFPR	EE	EEPR
DCFGRS						
MEDCF	0.5000, 0.6278					
MF	13.2864, 1.1150	3.0050, 0.0132				
MFPR	26.4704, $1.3654e-10$	4.9002, 0.0006	20.5507, $1.6452e-09$			
EE	26.4082, $1.3975e-10$	4.8815, 0.0006	20.4766, $1.7044e-09$	−3.6453, 0.1309		
EEPR(Proposed)	2.7681, 0.0198	2.4755, 0.0327	0.7987, 0.4429	−0.0589, 0.9541	−0.2524, 0.8058	

Since latent factor models work better for group recommender systems, the proposed work can be implemented for groups. This work can also be extended by using the side information in the dataset to get better recommendations. In this work, we used the traditional machine learning paradigm to attain optimal weights while learning the model. We can attain better precision and recall values if implemented in a deep learning framework. This work can be explored in a group recommendation system(GRS) using group modeling strategies.

Funding This research received no specific grant from any funding agency in the public, commercial sectors.

Declaration of competing interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

1. Shambour Q (2021) A deep learning based algorithm for multi-criteria recommender systems. Knowl-Based Syst 211:106545
2. Desarkar MS, Saxena R, Sarkar S (2012) Preference relation based factorization for recommender systems. International Conference on user modeling, adaptation, and personalization, pp 63–75
3. Khoshneshin M, Street WN (2010) Collaborative filtering via euclidean embedding. Proceedings of the Fourth ACM Conference on Recommender systems, and personalization, pp 87–94
4. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42:30–37
5. Desarkar MS, Sarkar S (2012) Rating prediction using preference relations based matrix factorization. UMAP Workshops
6. Valdiviezo-Diaz P, Ortega F, Cobos E, Lara-Cabrera R (2019) A collaborative filtering approach based on Naïve Bayes classifier. IEEE Access 7:108581–108592
7. He X, Liao L, Zhang H, Nie L, Hu X, Chua T-S (2017) Neural collaborative filtering. Proceedings of the 26th international conference on world wide web, pp 173–182
8. Lara-Cabrera R, González-Prieto Á, Ortega F (2020) Deep matrix factorization approach for collaborative filtering recommender systems. Appl Sci 10:4926
9. Candillier L, Meyer F, Boullé M (2007) Comparing state-of-the-art collaborative filtering systems. International workshop on machine learning and data mining in pattern recognition, pp 548–562
10. Jones N, Brun A, Boyer A (2011) Comparisons instead of ratings: towards more stable preferences. 2011 IEEE/WIC/ACM International conferences on Web intelligence and intelligent agent technology, vol 1, pp 451–456

11. Basalaj W (1999) Incremental multidimensional scaling method for database visualization. Visual Data Exploration and Analysis VI 3643:149–158
12. DeSarbo WS, Howard DJ, Jedidi K (1991) MULTICLUS: a new method for simultaneously performing multidimensional scaling and cluster analysis. Psychometrika 56:121–136
13. Zhang L, Chen Z, Zheng M, He X (2011) Robust non-negative matrix factorization. Front Electr Electron Eng China 6:192–200
14. Hernando A, Bobadilla J, Ortega F (2016) A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model. Knowl-Based Syst 97:188–202
15. Brun, A, Hamad A, Buffet O, Boyer A (2010) Towards preference relations in recommender systems. Preference learning (PL 2010) ECML/PKDD 2010 workshop
16. Villegas NM, Sánchez C, Díaz-Cely J, Tamura G (2018) Characterizing context-aware recommender systems: a systematic literature review. Knowl-Based Syst 140:173–200
17. Thakker U, Patel R, Shah M (2021) A comprehensive analysis on movie recommendation system employing collaborative filtering. Multimed Tools Appl, pp 1–26
18. Forouzandeh S, Berahmand K, Rostami M (2021) Presentation of a recommender system with ensemble learning and graph embedding: a case on MovieLens. Multimed Tools Appl 80(5):7805–7832
19. Cai L, Xu J, Liu J, Pei T (2017) Integrating spatial and temporal contexts into a factorization model-Centrality Prediction in Temporally Evolving Networks for POI recommendation. Int J Geogr Inf Sci 32:1–23
20. Bueno S, Salmeron JL (2009) Benchmarking main activation functions in fuzzy cognitive maps. Expert Syst Appl 36:5221–5229
21. Pereira N, Varma S (2016) Survey on content based recommendation system. Int J Comput Sci Inf Technol 7:281–284
22. Lops P, De Gemmis M, Semeraro G (2011) Content-based recommender systems: state of the art and trends. Recommender systems handbook, pp 73–105
23. Tewari AS, Kumar A, Barman AG (2014) Book recommendation system based on combine features of content based filtering, collaborative filtering and association rule mining. IEEE International advance computing conference (IACC), pp 500–503
24. Anwar T, Uma V (2021) Comparative study of recommender system approaches and movie recommendation using collaborative filtering. Int J Syst Assur Eng Manag 12:426–436
25. Yang Y, Yao H, Li R, Wang S (2021) A collaborative filtering recommendation algorithm based on user clustering with preference types. J Phys: Conf Ser 1848:012–043
26. Nahta R, Meena YK, Gopalani D, Chauhan GS (2021) Embedding metadata using deep collaborative filtering to address the cold start problem for the rating prediction task. Multimed Tools Appl 80:18553–18581
27. Sreepada RS, Patra BK, Chakrabarty A, Chandak S (2018) Revisiting tendency based collaborative filtering for personalized recommendations. Proceedings of the ACM India joint international conference on data science and management of data, pp 230–239
28. Isinkaye FO, Folajimi YO, Ojokoh BA (2015) Recommendation systems: principles, methods and evaluation. Egypt Inform J 16:261–273
29. Ghazarian S, Nematbakhsh MA (2015) Enhancing memory-based collaborative filtering for group recommender systems. Expert systems with applications, pp 3801–3812
30. Bobadilla J, Ortega F, Hernando A, Gutiérrez A (2013) Recommender systems survey. Knowledge-based systems, pp 109–132
31. Yannam VR, Kumar J, Babu KS, Sahoo B (2023) Improving group recommendation using deep collaborative filtering approach. Int J Inf Technol 15:1489–1497
32. Mehta R, Rana K (2017) A review on matrix factorization techniques in recommender. 2nd International conference on communication systems, computing and IT applications (CSCITA) systems, pp 269–274
33. Ahn HJ (2008) A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. Inf Sci 178:37–51
34. Kannan R, Ishteva M, Park H (2014) Bounded matrix factorization for recommender system. Knowl Inf Syst 39:491–511
35. Liu H, Hu Z, Mian A, Tian H, Zhu X (2014) A new user similarity model to improve the accuracy of collaborative filtering. Knowl-Based Syst 56:156–166
36. Ali K, Van Stam W (2004) TiVo: making show recommendations using a distributed collaborative filtering architecture. Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining, pp 394–401
37. Kwon H-J, Hong K-S (2011) Personalized smart TV program recommender based on collaborative filtering and a novel similarity method. IEEE Trans Consum Electron 57:1416–1423

38. Barragáns-Martínez AB, Costa-Montenegro E, Burguillo JC, Rey-López M, Mikic-Fonte FA, Peleteiro A (2010) A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. Inf Sci 180:4290–4311
39. Zimmerman J, Kauapati K, Buczak A, Schaffer D, Gutta S, Martino J (2004) TV personalization system, Personalized Digital Television
40. Alhamid MF, Rawashdeh M, Al Osman H, El Saddik A (2013) Leveraging biosignal and collaborative filtering for context-aware recommendation. Proceedings of the 1st ACM international workshop on multimedia indexing and information retrieval for healthcare, pp 41–48
41. Jha GK, Gaur M, Thakur HK (2022) A trust-worthy approach to recommend movies for communities. Multimedia Tools and Applications, Springer, pp 1–28
42. Anwar Taushif VU, Hussain MI, Pantula M (2022) Collaborative filtering and kNN based recommendation to overcome cold start and sparsity issues: a comparative analysis. Multimed Tools Appl 81(25):35693–35711
43. Chen Y-C, Hui L, Thaipisutikul T (2021) A collaborative filtering recommendation system with dynamic time decay. J Supercomput 77:244–262
44. Thaipisutikul T, Shih TK (2021) A novel context-aware recommender system based on a deep sequential learning approach (CReS). Neural Comput Appl 33:11067–11090
45. Bobadilla J, Dueñas J, Gutiérrez A, Ortega F (2022) Deep variational embedding representation on neural collaborative filtering recommender systems. Appl Sci 12(9):4168
46. Thaipisutikul T (2022) An adaptive temporal-concept drift model for sequential recommendation. ECTI Transactions on Computer and Information Technology (ECTI-CIT) 16(2):222–236
47. Ni J, Huang Z, Hu Y, Lin C (2022) A two-stage embedding model for recommendation with multimodal auxiliary information. Inf Sci 582:22–37
48. Thaipisutikul T, Chen Y-N (2023) An improved deep sequential model for context-aware POI recommendation. Multimed Tools Appl, pp 1–26
49. Thaipisutikul T, Shih TK, Enkhbat A, Aditya W (2021) Exploiting long-and short-term preferences for deep context-aware recommendations. IEEE Trans Comput Soc Syst 9(4):1237–1248