The Journal of Supercomputing (2022) 78:7410–7427
https://doi.org/10.1007/s11227-021-04178-5

Collaborative filtering in dynamic networks based on deep auto-encoder

Shiva Jalali[1] · **Monireh Hosseini**[1]

Accepted: 27 October 2021 / Published online: 11 November 2021

Abstract

The collaborative filtering method in recommender systems can produce suggestions with good quality. However, this method suffers from the issues of cold start and data sparsity. To overcome these problems, other data sources must be used to identify users. As a source of information, social relationships between users can help solve these problems. Recommender systems, on the other hand, are set in environments where data are generated continuously and with high dimension over time. For example, users have new rates and new relationships. These are indicative of the dynamic nature of system data, the changes in which classical models are unable to manage in order to provide appropriate suggestions to users. Then a dynamic model is needed. In this research, a new hybrid dynamic recommender model, which utilizes deep auto-encoder networks, is described to close these gaps. In this model, users' rating information and social relationships between them are used simultaneously to calculate similarity matrices between users at different timestamps. In each timestamp, the similarity matrix is given as input to the deep neural network, in which users are divided into different clusters. Users' new behaviors over time will help update matrices values. The use of social relationships data and the consideration of users' new behaviors over time solves the problems of classical methods, reduces recommending error, and increases user satisfaction. The proposed model is compared with state-of-the-art models; based on evaluation metrics. The results of execution on the real dataset show that the proposed model outperforms them.

Keywords Collaborative filtering · Social network · Dynamic networks · Deep auto-encoder · Clustering

✉ Monireh Hosseini
Hosseini@kntu.ac.ir

Shiva Jalali
shivajalali@email.kntu.ac.ir

[1] Department of Industrial Engineering, K.N.Toosi University of Technology, Tehran, Iran

1 Introduction

Collaborative filtering (CF)—neighborhood-based and model-based approaches—are tools that assist people in making decisions in their daily lives using similarities between users' ratings and the opinions of others. If users in these systems rate a small number of items, the sparsity of their rating matrix will cause a cold start problem and reduce system performance [1]. To tackle this problem, other data sources should be used to identify users and determine their similarities with other users. One solution is to use the social network and relationships among users.

Social networks have turned into a tool for people to communicate, allowing them to communicate in a way that was previously impossible. Classic recommender systems ignored the social data and interaction among users [2] while social relationships among users always express commonalities between them, hence considering these interactions and data can improve the performance of recommender systems. The study of social network-based recommender systems began with the spread of Web 2.0 [3].

Most of the recommender systems are based on algorithms that are designed on the assumption that users' feedback are static and that all dataset are always available [3, 4]. However, there is dynamism in users' behaviors over time. For example, users rate the recommended items or other items of interest. They also connect with new users on social network. This dynamism in the users' behaviors cannot be modeled via classical algorithms because it will cause data loss and reduce system performance. Therefore, there is a need for a system that considers the data dynamism over time and learns and manages users' behaviors to increase the accuracy of recommendation [5–7].

Deep learning methods and their structures [8] are a powerful framework for learning which are very successful in speech recognition, image classification, natural language processing, and data clustering. Nevertheless, the historical and current datasets should be combined for learning in dynamic networks and for predicting the future is a tough challenge. Despite much attention paid to the deep learning in recommender systems [9–12], very little effort has been made to take advantage of the temporal deep learning approach for dynamic recommender systems [13]. Among the deep learning structures, the auto-encoder network has a powerful memory function that fully extracts time series changes for future prediction.

On the other hand, since users tend to relate to people who have similar preferences and choices, these tendencies lead to the formation of communities among them which can be identified by clustering; the views of similar users are further used to provide recommendations. However, if the original data are not well-distributed and has large dimension, it is difficult to achieve satisfactory performance via traditional clustering algorithms. To solve this problem, the original data space can be mapped into a new space that is more suitable for clustering. The auto-encoder network is a good candidate to solve this problem. This network provides a nonlinear mapping by continuous learning from the encoder and decoder. Recently, deep clustering, which learns feature representation for clustering tasks using deep neural networks, has attracted increasing attention for various clustering applications.

In this research, the mentioned features are used for deep auto-encoder networks in the recommender system and a hybrid collaborative filtering approach (combining user rating data and social relationships among them) based on a new deep model of semi-supervised auto-encoder networks is presented. It is recommended to learn the interactions between users and their neighbors up to step 2. In this approach, a multi-layered architecture (as the number of timestamps) is designed; each layer is an auto-encoder, with its input being a similarity matrix of users in each timestamp with time decay and its output clusters of users. The weights and clusters of each layer of architecture will be used in the next layer. The proposed approach and architecture take the dynamism in user data in each timestamp and the historical records of their activities into account. In addition to solving the cold start problem, this approach will increase the performance of the recommender system as of its capability to cluster users with high accuracy.

The innovations of this research are as follows:

- The provision of a dynamic recommender system based on a deep learning approach
- The use a combination of rating data and social relationship data among users and their neighbors up to step 2 as peripheral information to solve the problem of data sparsity
- The calculation of the similarity among users by utilizing the existing information and based on users' activity time
- The use of a deep network layered approach to manage user data dynamically
- The semi-supervised clustering of users in order to consider the opinions of the most similar users to solve the problem of cold start, and consequently, enhancing the accuracy and performance of the system compared to the comparable methods.

In what follows in this paper, in the next section, the related studies are given. In the third and fourth sections, the proposed approach and architecture and the results of its implementation on the selected dataset are provided. In the last section, conclusions are presented.

2 Related work

Dynamic and temporal recommender systems are used in a variety of areas, including such websites as movie recommender [14, 15], music recommender [16], news recommender [17, 18], and e-commerce recommender websites [19]. These systems perform better than static systems. In these systems, temporal deep learning techniques can be used to provide a recommendations tailored to users' rating history at different timestamps. However, despite the benefits of using such techniques, only a few have been used in sequential and dynamic recommender systems. The following are the most important research bodies that have made use of deep learning techniques in presenting temporal recommendations and in dynamic conditions.

Authors in [17] have provided a multivariate temporal deep learning model for predicting news click sequences that combines static long-term and temporary short-term (temporal) user preferences to improve recommending function (TDSSM). This model consists of several components: a deep semantic structural model for modeling the user's static interests, two LSTM-based temporal models for recording the user's daily and weekly time patterns, and a temporal LSTM model for recording the user's overall preferences. These recurrent neural networks are combined using a feed-forward network with complete connections. In paper [20], a model for collaborative filtering sequence based on Gated Recurrent unit networks (C-GRU) is proposed. This model is designed to collect the user's contextual state as a personalized hidden vector from earlier times. In particular, static global hidden factors are used for items, and separate hidden factors are assigned to users depending on their past records. Each event in a user sequence is a unique choice relative to the items set. This research focuses on sequences, regardless of the exact time of the events.

Authors in [21] have provided a recurrent neural network model (DLRec) to predict the likelihood of user feedback on an item at a later time. The proposed neural network consists of two parts: the recurrent part and the non-recurrent part. The recurrent part retains the effect of previous feedback. The non-recurrent part indicates the user's priority. First, information is continuous extracted from the sequences of consumed items. Then, the user-specific features are fed to the feed-forward network independent of the sequence features. Finally, the results of the two methods are combined. In paper [19], a sequential recommender system using recurrent neural networks is presented. These networks have been developed by considering the unique features of the range of recommender systems. This study demonstrates how both user and item features can be integrated into a new architecture of the GRU sequence to suggest next-stage items to individuals.

Authors in [22] have designed an architecture based on a deep learning approach to provide sequential content recommendations (DRNN). To this end, the GRU deep learning technique has been used to map the history of user interaction to a common Euclidean space of user vectors and contents. This mapping makes it possible to recommend the next probable content based on similar patterns. In paper [23] are presented recurrent recommender networks (RRNs) which are able to predict the future behavior. Individuals' RNNs have been trained to model user and item evolution separately. The outputs of both networks are subsequently accompanied by additional auxiliary parameters in order to predict the user ratings.

In [24], a RNN-based method and attention mechanism for recommending in a dynamic environment are presented (ARSE). In this method, the static and dynamic preferences of users over time are used. Moreover, the power of social influence between users is considered. Authors in [25], have proposed a recommender system based on a dynamic-graph-attention network (DGRec). This system uses the preferences of users in each session and the short-term and long-term preferences of their friends. The preferences of users and those of their friends are modeled by recurrent neural networks (RNNs).They are then combined in a graph-attention network to provide recommendations. In [26], an interactive recommender system based on a combination of knowledge graph (KG) and reinforcement learning (RL) is developed to solve the problem of data sparsity (KGQR). The knowledge graph is used to

extract connections between items and find candidate items, and a convolutional network (GCN) graph, based on users' preferences, is utilized to describe their states. The real-time users' feedback to the RL network affects the recommended items.

Modeling users' dynamic behavior improves the accuracy of the recommender systems. Dynamic deep neural networks and mentioned models can model this type of users' behavior well due to their ability to incorporate users' short-term and long-term experiences. But despite that these models take into account the sequential users' information, most of them can only be used to predict their opinions in the one next timestamp and if the stream of data continues and enters the system, the models need to be retrained anew. On the other hand, despite the significance of social network data among users in solving the problem of sparsity of the data matrix and reducing the cold start issue, this data have not been considered in most of these models. The utility of this data, along with users' rating data, can increase the power of relationships among users. On the basis of these connections, more stable and stronger communities are formed among users. Hence, similar users can be identified, and their opinions can be used.

In this research, multi-layer deep auto-encoder networks have been used to converts large-scale data into smaller dimensions of data and cluster users, with each timestamp containing a layer of deep auto-encoder network. The input of each layer of the network is a similarity matrix among users with time decay, which is obtained by using past data of users' ratings and their social relationships. At each timestamp, this network input data are updated. Input data are converted to smaller hidden vector, and clusters from the previous layer are updated to find the most similar users in that timestamp. To provide recommendations, then, the focus is shifted to users' recently-rated items. The use of the proposed method and network has no time limit, and as long as the users' data continues to generate, a new layer of auto-encoder networks can be used without the need to train the previous layers.

A summary of the relevant research bodies, together with their comparisons with the proposed method, is presented in the table below (Table 1).

3 The proposed approach

In this section, we present a dynamic collaborative filtering method using multi-layer deep neural networks (auto-encoders). In this method, a weighted graph with time decay is formed between users in each timestamp. These graphs are formed using users' common rates to items and the social relationships between them. Each graph is the input of a layer of deep auto-encoding network. The task of each layer of the auto-encoder network is to cluster users. The updated weights and clusters from each network layer are considered as initial parameters and initial clusters in the next layer. Two layers of auto-encoder networks are shown in Fig. 1. These layered networks can reflect changes in user behaviors (users' interests in rating and their social relationships) in different timestamps in the output, and in each layer and each timestamp, place the most similar users in the same clusters. This is done to offer recommendations to target users so that the latest user opinions in the target cluster can be used.

Table 1 Comparison of related research with the proposed method

Row	Reference, author and year	The method	Users' data	Dataset
1	[17] Song et al., 2016 (TDSSM)	LSTM	User's news click history	Commercial news portal
2	[20] Ko et al., 2016 (C-GRU)	GRU	Sequences of songs played by a user	LastFM
3	[21] Wu et al., 2016 (DLRec)	RNN	Users' rates to movies	MovieLens
4	[19] Donkers et al., 2017	GRU	Users' rates to movies	MovieLens
5	[22] Soh et al., 2017 (DRNN)	GRU	Per-user resource accesses at Microsoft	Microsoft Web data
6	[23] Wu et al., 2017	RNN	Users' rates to movies	Netflix
7	[24] Sun et al., 2018 (ARSE)	RNN	Users' rates with social relations between them	Epinions, Gowalla
8	[25] Song et al., 2019 (DGRec)	RNN with attention network	Users' comments with social relations between them	Douban, Yelp
9	[26] Zhou et al., 2020 (KGQR)	Reinforcement Learning(RL) with Knowledge graph(KG)	Users' rates to movies	MovieLens, Book-Crossing
10	The proposed method	Multi layers Auto-encoders	Users' rates to items social relations between them	Epinions

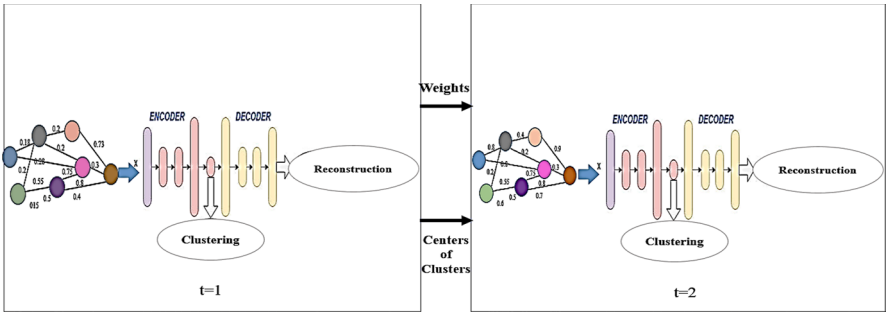

Fig. 1 The proposed framework

3.1 Statement of the problem

3.1.1 Dynamic weighted network

A dynamic weighted network is a set of snapshots $\{G_1, G_2,..., G_T\}$ in the network that the structure of which is $G_t = \{V, E_t, W_t\}$, $t = 1,\ldots T$. In this structure, E_t is the network edges, W_t is the weight of the edges at time t, and T is the total number of timestamps.

3.1.2 Community structure (cluster)

A cluster in a dynamic network is a set of nodes that are densely connected; each cluster has a loose connection with other clusters.

3.1.3 Auto-encoder (AE)

An unsupervised model of three layers: input, hidden, output that reconstructs its input data in the output layer. In general, the middle layer is used as the data embedding layer and highlights the characteristics of the input data [8].

3.1.4 Dynamic network embedding

In each timestamp G_t of the dynamic network and d as dimensions, network embedding converts input data to d-dimensional data, and a timed adaptation as $F = \{F_1, F_2,\ldots, F_T\}$ are obtained where $F_t = X_t H_t$ and $t = 1, 2,\ldots, T$. X_t is the input data, and H_t is a representation of input data with dimensions d at time t that are clustered by the auto-encoder.

3.2 Problem formulation

In this research, in each timestamp, a weighted graph is formed between the users; this is given, as an input, to the deep auto-encoder network in that timestamp in order to cluster the users. Figure 2 shows the proposed research algorithm.

Input: User rating data and social relationships in timestamps 1,…, T
Output: Offer items to each target user in each timestamp

1. Divide input data into train and test sets
2. Create / update similarity matrices between users: direct neighbors, and indirect neighbors within a distance of 2 (matrices M_1 to M_4) based on user input data in the timestamp of the train set.
3. Combine and merge the weights of these matrices into one final matrix (M-final)
4. Provide the final matrix as deep auto-encoder network input and clustering users
5. Determine the cluster of target users in the test set and recommend items based on the users' latest opinions in each cluster
6. Multiply the values of the matrices M1 to M4 by the value of the decay parameter (weights with time decay)
7. Return to line 2 for the next timestamp

Fig. 2 General steps of the proposed method

It was mentioned that the input of each auto-encoder in each timestamp is a weighted graph between users. In the graph between users, the weight on the edges is obtained by weighted sum of the values of 4 matrices (the final similarity matrix in each timestamp). These 4 matrices represent the first- and second-rank neighborhoods in common items and social relationships between users. The combination of these 4 matrices represents the global structure of the network between users and the strength of relationship between them. These matrices are described below.

A) The first matrix $(M_1) = (\text{sim1}(x, y)_{m\times m})$ is a weighted matrix between users (m is the number of users) that is obtained based on their rates on common items and using the cosine similarity coefficient (Eq. 1).

$$\text{sim1}(x,y) = \cos\left(\vec{x},\vec{y}\right) = \frac{\vec{x}.\vec{y}}{\left|\left|\vec{x}\right|\right|_2 \times \left|\left|\vec{y}\right|\right|_2} = \frac{\sum_{i\in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i\in I_{xy}} r_{x,i}^2}\sqrt{\sum_{i\in I_{xy}} r_{y,i}^2}} \tag{1}$$

where I_{xy} is the set of rates given by both users x and y; and $r_{x,i}$ is the user's rating of x to item i.

B) The second matrix $(M_2) = (\text{sim2}(x,y)_{m\times m})$ is a matrix that calculates the similarity between users who have common neighbors. In fact, it obtains the similarity of each user with neighbors with a distance of 2 using the first matrix (M_1). The amount of weight between the user and the second-rank neighbors is based on the average weight of the connected edges (Eq. 2).

$$\text{sim2}(x,y) = \frac{\sum_{z\in U} w_{x,z} w_{z,y}}{n} \tag{2}$$

where z is the common neighbors between user x and y, $w_{x,z}$ represents the edge weight between users x and z, and n shows the number of common users (or the same number as $|z|$).

C) The third matrix $(M_3)=(\text{sim3}(x,y)_{\text{m}\times\text{m}})$ is the matrix obtained from the social relations between users, in which the edge weight is 1 if there is a direct relationship between the two users (Eq. 3).

$$\text{sim3}(x,\ y) = \begin{cases} 1 & \text{if } x, y \text{ have a social relationship} \\ 0 & \text{else} \end{cases} \tag{3}$$

D) The fourth matrix $(M_4)=(\text{sim4}(x,y)_{\text{m}\times\text{m}})$ is acquired from the third matrix (M_3) and obtains the social relations between each user and the neighbors up to step 2 (Eq. 4).

$$\text{sim4}(x,y) = \frac{\sum_{u\in U} w_{\text{x,u}} w_{\text{u,y}}}{n} \tag{4}$$

Where u is the common neighbors (with n numbers) between x and y in the M_3 matrix, and $w_{\text{x,u}}$ indicates the edge weight value (number 1) between users x and u.

In each timestamp, the weighted sum of the values of the mentioned matrices (Eq. 5) is given to the auto-encoder of that layer (timestamp) as the final matrix of relationship between users (M-final matrix) so that the users can be clustered. Matrix coefficients are parameters that control the weight of the overall similarity between users among different similarity matrices.

$$M - \text{final} = M_1 + \alpha * M_2 + \beta * M_3 + \gamma * M_4 \tag{5}$$

One of the problems with conventional clustering methods (e.g., *k*-means) is that if the data have numerous dimensions, they are usually significantly distant from the normal distribution, and it is not possible to obtain suitable initial centers from them. Therefore, these methods cannot properly cluster the data. The use of deep auto-encoder networks results in reduction of data dimensions and better clustering performance [27]. Also, due to the use of each auto-encoder in each timestamp and the use of clusters from the previous step as initial centers in the auto-encoder, the problem of initial centers will also be solved.

Each auto-encoder has one input layer and several hidden layers to convert large-dimensional input data to smaller-sized data. The output of the middle hidden layer is used to cluster the data using the k-means method. Figure 3 shows the structure of each auto-encoder.

In this structure, tansig function [27] is used to map the data of each layer to the next layer (Eq. 6).

$$h_{\text{i}} = f\left(x_{\text{i}}\right) = \frac{1 - e^{-(W_{\text{i}}x_{\text{i}}+b_{\text{i}})}}{1 + e^{-(W_{\text{i}}x_{\text{i}}+b_{\text{i}})}} \tag{6}$$

Moreover, the network output error (e_1) and the clustering error (e_2) are reduced simultaneously (Eq. 7) [27]. The output error of the network is the difference between the input and output of the network (which is the same as the input), and the clustering error is the difference between the output and the nearest center of the cluster (similarity within the cluster).

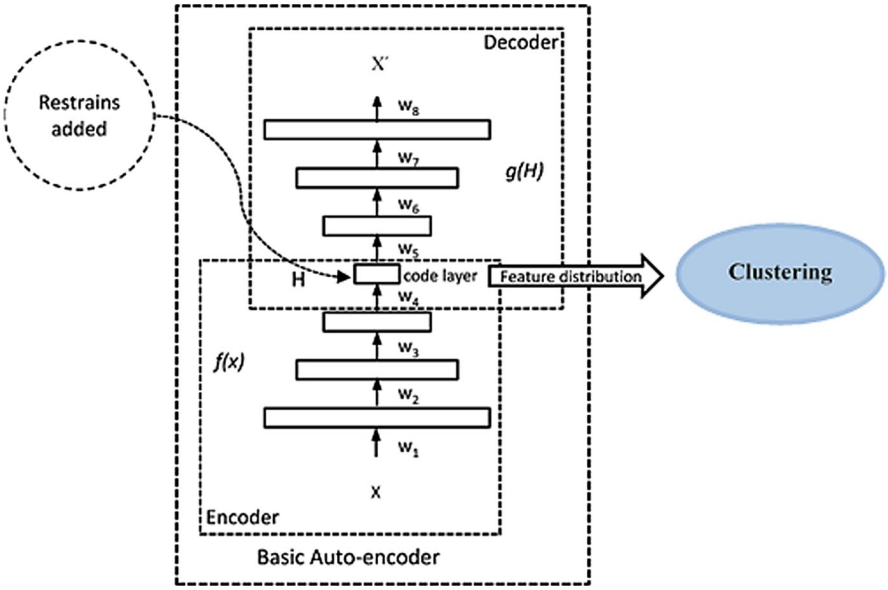

Fig. 3 The structure of each auto-encoder for data clustering [27]

$$e_{\min} = e_1 + e_2 = \frac{1}{N}\sum_{i=1}^{N}\left\|x_i - x_i'\right\|^2 + \lambda.\sum_{i=1}^{N}\left\|f^t\left(x_i\right) - c_i^*\right\|^2$$
$$c_i^* = \min_{c_j^{p-1}}\left\|f^t\left(x_i\right) - c_j^{p-1}\right\|^2 \tag{7}$$

Accordingly, N is the number of train data, $f^{\mathrm{t}}\left(x_{\mathrm{i}}\right)$ represents the map function in the p iteration, c_j^{p-1} shows the center of the cluster j in the p-1 iteration of the training loop, and c_{i}^* is the closest center of cluster to the data that is in the most-inner middle hidden layer.

The initial centers of the clusters in the first auto-encoder are randomly selected from the data at time 1 and updated in the data training process. However, in subsequent timestamps, the clusters obtained in the previous timestamps (middle layer output of the previous auto-encoder) will be used as initial centers.

After clustering the users, the latest opinions of other users in the target user cluster are used to provide a recommendation to the target user via the similarity weight between the target user and others (Eq. 8).

$$p_{\mathrm{u,i}} = \frac{\sum_{v=1}^{k} w_{u,v} * r_{v,i}}{\sum_{v=1}^{k} w_{u,v}} \tag{8}$$

$p_{\mathrm{u,i}}$ is the value of the predicted user's rate to item i, k represents the number of users in the target user cluster, and $\mathrm{w_{u,v}}$ is the similarity weight between the target user and another user (v) in the cluster.

Due to the importance of user rating time for items as well as the time it takes to make social relationships, the similarities between users should diminish over time. Therefore, after passing each timestamp and entering the next one, the values of the mentioned similarity matrices (matrices M_1–M_4) are multiplied by a time decay (Eq. 9 [14]).

$$M = M. * e^{(-1*\Phi)} \tag{9}$$

where ϕ is the amount of time decay parameter considered for user activity data. Then in the next timestamp, if there is another similarity based on new rating data and new social relationships between both users, it is added to the decayed values of these matrices. In fact, the matrices are updated to enter the next auto-encoder. Creating a time decay in the similarity values between users and updating them with new data in each timestamp cause the short-term preferences of users to have more weight in the similarity values between them. Thus, the amount of similarity of target users enjoys a higher value with users who have rated similar items or established social relationships at almost the same time. Also, when using their opinions and recommending items to target users, items that other users have rated in more recent times will be selected. These steps are repeated for each auto-encoder in each timestamp. This process continues as long as there is a stream of user data.

4 Results and analysis

In this section, experiments are conducted on a selected dataset, and the performance of the proposed method is evaluated in comparison with a number of common dynamic methods.

4.1 Selected dataset

In this study, Epinions dataset was used. The features of this dataset are as follows (Table 2):

The dataset is divided into three parts. In each timestamp, the first and major part of the data, including 70% of it, is used as auto-encoder network training data. The second part is to validate and prevent the auto-encoder network from overfitting; it comprises 10% of the data. The third part is to test and measure the performance of the network in data clustering and provide recommendations to the target users. As it is common in deep network research, the training section is filtered from the validation and testing sections.

Table 2 Features of the selected dataset

Dataset	Users	Items	Ratings	Scale	Sparsity	Social relations	Total timestamp
Epinions	22,166	296,277	922,267	[0–5]	0.9964	300,548	11

4.2 Comparable methods

Comparable methods are: User-based CF with exponentially decaying, Deep-learning-based CF with exponentially decaying, Standard GRU-based CF, and DGRec [25].

4.2.1 User-based CF with exponentially decaying

In user-based CF with exponential decaying [28], users' rates will gradually decrease exponentially from the first timestamp (Eq. 10) so that the items that users have recently rated have higher values. In this equation, rate_T is the user rate at time T, rate_t is the user rate at each time 1,… T-1, and ϕ shows the decay factor.

$$\text{rate}_T = \text{rate}_t . * e^{(-1*\Phi)*(T-t)} \tag{10}$$

After that, the similarity between users is calculated based on their decayed ratings. Then, based on the opinions of the most similar users to the target users, items are recommended to them, and the target users' rates are predicted for time T.

4.2.2 Deep-learning-based CF with exponentially decaying

This method is simplified version (sample) of the proposed method in the present paper and is implemented in order to draw comparisons with it. In this sample, using deep neural network, the collaborative filtering method is implemented. To this end, user rating data is multiplied and reduced over time in accordance with Eq. (10) by a decay coefficient. Based on the data obtained, the similarities between users at time T-1 are calculated and placed in a matrix. This matrix is an auto-encoder network input in which users are clustered. Then, the closest cluster to the target user is selected as the target cluster; and by using the ratings of other users in that cluster, the target user's rating values is achieved for the recommended items.

4.2.3 Standard GRU-based CF

Based on the standard Gate Recurrent Unit [29], the collaborative filtering method, with clustering technique, is implemented. In the standard GRU method, T-1 number of GRU are used. In this method, user rating data are given to the GRUs as input from times 1 to T-1, respectively. GRUs outputs are aggregated as a code space for user communication. These code spaces are then clustered, and users are placed in different clusters. To provide recommendations to the target users, the opinions of other users in their clusters are used. Next for the recommended items, rates are predicted and compared with the users' rating matrix at time T.

4.3 Evaluation metrics

To evaluate the algorithms of recommender systems, two evaluation metrics, namely, precision and recall are used. Precision is the fraction of relevant items (TP) among all recommended items (TP + FP). Recall is the fraction of relevant items (TP) to rated items (TP + FN).

Given these values, the precision is calculated as Eq. (11):

$$\text{precision} = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{TP}}{\text{TP}+\text{FP}} \tag{11}$$

The recall criterion is also calculated as Eq. (12):

$$\text{recall} = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{TP}}{\text{TP}+\text{FN}} \tag{12}$$

Mean Average Error (MAE) criterion is used to calculate the amount of error between predicted rates and their actual values. MAE refers to the average of all values of the deviation from the ratings and is calculated as Eq. (13):

$$\text{MAE} = \frac{\sum_{i=1}^{N}|r_i - p_i|}{N} \tag{13}$$

where p_i and r_i are the actual and predicted values of ratings for item i, and N is the total number of predicted rates.

4.4 Implementation results

In this research, in the first experiment, the ratings of 2000 users to 200 items, together with the social relations of these users in 11 timestamps, have been used.

The proposed method is compared with the methods mentioned in Sect. 4.2. The results of these comparisons in the evaluation metrics and the number of clusters 2 to 30 are shown in Table 3. Both methods User-based CF with exponentially decaying and DGRec do not have clustering; therefore, their results have the same values for different clusters. The decay coefficient is considered to be 0.01, and each autoencoder also has 3 nested layers, which the clustering is done in the most-inner hidden layer. The values of MAE, precision, and recall in the proposed method are their average in different timestamps.

The results of the table reveal that the 3 methods based on deep learning in different clusters have better performance than the user-based CF with exponentially decaying. This is because the user-based CF with exponentially decaying method suffers from the problem of sparsity of the rating matrix and uses only the opinions of users which bear the most similarity to the target users.

The deep-learning-based CF method clusters users based on their similarity in the T-1 timestamp. In this method, a network of users with common items are

Table 3 The implementation results of the proposed method and the comparable methods (2000 users and 200 items)

Method	Number of clusters	MAE	Precision	Recall	Intra-cluster distance
User-based CF with exponentially decaying[28]	–	0.58	0.29	0.33	–
DGRec [25]	–	0.25	0.49	0.6	–
Deep-learning-based CF with exponentially decaying	2	0.49	0.35	0.41	0.593
Standard GRU-based CF [29]	2	0.44	0.41	0.47	0.297
Proposed Method	2	0.28	0.55	0.62	0.271
User-based CF with exponentially decaying [28]	–	0.58	0.29	0.33	–
DGRec [25]	–	0.25	0.49	0.6	–
Deep-learning-based CF with exponentially decaying	3	0.46	0.39	0.41	0.484
Standard GRU-based CF [29]	3	0.42	0.5	0.56	0.233
Proposed Method	3	0.25	0.61	0.68	0.145
User-based CF with exponentially decaying [28]	–	0.58	0.29	0.33	–
DGRec [25]	–	0.25	0.49	0.6	–
Deep-learning-based CF with exponentially decaying	5	0.38	0.45	0.48	0.433
Standard GRU-based CF [29]	5	0.37	0.53	0.58	0.112
Proposed Method	5	0.23	0.67	0.73	0.092
User-based CF with exponentially decaying [28]	–	0.58	0.29	0.33	–
DGRec [25]	–	0.25	0.49	0.6	–
Deep-learning-based CF with exponentially decaying	7	0.34	0.52	0.58	0.371
Standard GRU-based CF [29]	7	0.28	0.59	0.6	0.096
Proposed Method	7	0.19	0.78	0.81	0.077
User-based CF with exponentially decaying [28]	–	0.58	0.29	0.33	–
DGRec [25]	–	0.25	0.49	0.6	–
Deep-learning-based CF with exponentially decaying	10	0.36	0.51	0.52	0.392
Standard GRU-based CF [29]	10	0.3	0.58	0.59	0.264
Proposed Method	10	0.27	0.59	0.63	0.102
User-based CF with exponentially decaying [28]	–	0.58	0.29	0.33	–
DGRec [25]	–	0.25	0.49	0.6	–
Deep-learning-based CF with exponentially decaying	15	0.44	0.43	0.49	0.401
Standard GRU-based CF [29]	15	0.41	0.37	0.41	0.340
Proposed Method	15	0.35	0.43	0.54	0.269
User-based CF with exponentially decaying [28]	–	0.58	0.29	0.33	–
DGRec [25]	–	0.25	0.49	0.6	–
Deep-learning-based CF with exponentially decaying	20	0.47	0.4	0.42	0.407
Standard GRU-based CF [29]	20	0.43	0.36	0.39	0.376
Proposed Method	20	0.43	0.39	0.42	0.323
User-based CF with exponentially decaying [28]	–	0.58	0.29	0.33	–
DGRec [25]	–	0.25	0.49	0.6	–
Deep-learning-based CF with exponentially decaying	25	0.48	0.35	0.38	0.436
Standard GRU-based CF [29]	25	0.47	0.34	0.37	0.409
Proposed Method	25	0.46	0.26	0.32	0.380

Table 3 (continued)

Method	Number of clusters	MAE	Precision	Recall	Intra-cluster distance
User-based CF with exponentially decaying [28]	–	0.58	0.29	0.33	–
DGRec [25]	–	0.25	0.49	0.6	–
Deep-learning-based CF with exponentially decaying	30	0.52	0.31	0.34	0.502
Standard GRU-based CF [29]	30	0.49	0.32	0.33	0.464
Proposed method	30	0.5	0.23	0.28	0.426

placed in each cluster. The advantage of this method over the previous method (user-based CF method with exponential decaying) is that users who have common neighbors are also clustered, and their opinions can be used. In the user-based method, however, the only factor is the opinions of other users with common items (direct neighbors) which are used. Therefore, deep-learning-based CF accuracy is higher than the first method. However, since the data are not considered in different timestamps and also due to the sparsity of the rating matrix, users are not clustered properly. Its accuracy, therefore, is less compared to the other three methods.

The standard GRU-based CF, DGRec, and the proposed method have processed user data in different timestamps. The standard GRU-based CF method, like the previous methods, use only rating data and clusters the users based on them (problem of sparsity of the rating matrix). However, both DGRec and the proposed methods does this based on both the rating data and social relationship data (hence addressing the problem of sparsity of the rating matrix). DGRec method uses the opinions of direct friends of the users. In the proposed method, however, the values of similarity between users (based on both common rates and social relations) with direct neighbors and those of up to step 2 are obtained, and strong communication networks between users are recognized. Thus, in users clustering and with the appropriate number of clusters, opinions of more similar users are used. Moreover, due to the use of the timestamp parameter in ratings and user relationships, items that have recently been rated by other users are recommended to target users. In result, the accuracy and validity of the proposed method are higher than the other methods. Also, the most considerable advantage of the proposed method over the other comparable methods is the ability to consider the stream of users' data.

In addition to the above experiments, a series of experiments was performed on bigger data sets (second experiments). The data include 20,000 users (with relationships between them) and 2000 items. Table 4 shows the results of comparison between the proposed method and other methods with respect to the number of mentioned clusters.

Also, on the basis of these results, the accuracy and validity of the proposed method are higher than the other methods.

Table 4 The implementation results of the proposed method and the comparable methods (20,000 users and 2000 items)

Method	Number of clusters	MAE	Precision	Recall	Intra-cluster distance
User-based CF with exponentially decaying[28]	–	0.73	0.21	0.23	–
DGRec [25]	–	0.51	0.32	0.38	–
Deep-learning-based CF with exponentially decaying	2	0.66	0.22	0.3	0.602
Standard GRU-based CF [29]	2	0.58	0.28	0.45	0.377
Proposed Method	2	0.43	0.4	0.5	0.324
User-based CF with exponentially decaying [28]	–	0.73	0.21	0.23	–
DGRec [25]	–	0.51	0.32	0.38	–
Deep-learning-based CF with exponentially decaying	3	0.63	0.27	0.31	0.585
Standard GRU-based CF [29]	3	0.53	0.39	0.46	0.356
Proposed Method	3	0.38	0.46	0.48	0.293
User-based CF with exponentially decaying [28]	–	0.73	0.21	0.23	–
DGRec [25]	–	0.51	0.32	0.38	–
Deep-learning-based CF with exponentially decaying	5	0.59	0.3	0.32	0.552
Standard GRU-based CF [29]	5	0.47	0.41	0.48	0.328
Proposed Method	5	0.34	0.5	0.53	0.232
User-based CF with exponentially decaying [28]	–	0.73	0.21	0.23	–
DGRec [25]	–	0.51	0.32	0.38	–
Deep-learning-based CF with exponentially decaying	7	0.56	0.32	0.35	0.514
Standard GRU-based CF [29]	7	0.39	0.48	0.52	0.297
Proposed Method	7	0.3	0.54	0.58	0.201
User-based CF with exponentially decaying [28]	–	0.73	0.21	0.23	–
DGRec [25]	–	0.51	0.32	0.38	–
Deep-learning-based CF with exponentially decaying	10	0.54	0.33	0.38	0.488
Standard GRU-based CF [29]	10	0.35	0.5	0.55	0.264
Proposed Method	10	0.26	0.6	0.63	0.173
User-based CF with exponentially decaying [28]	–	0.73	0.21	0.23	–
DGRec [25]	–	0.51	0.32	0.38	–
Deep-learning-based CF with exponentially decaying	15	0.51	0.36	0.41	0.437
Standard GRU-based CF [29]	15	0.31	0.54	0.58	0.233
Proposed Method	15	0.24	0.64	0.66	0.126
User-based CF with exponentially decaying [28]	–	0.73	0.21	0.23	–
DGRec [25]	–	0.51	0.32	0.38	–
Deep-learning-based CF with exponentially decaying	20	0.55	0.33	0.38	0.465
Standard GRU-based CF [29]	20	0.35	0.49	0.52	0.267
Proposed Method	20	0.31	0.57	0.62	0.158
User-based CF with exponentially decaying [28]	–	0.73	0.21	0.23	–
DGRec [25]	–	0.51	0.32	0.38	–
Deep-learning-based CF with exponentially decaying	25	0.59	0.3	0.33	0.498
Standard GRU-based CF [29]	25	0.38	0.43	0.46	0.301
Proposed Method	25	0.35	0.51	0.55	0.186

Table 4 (continued)

Method	Number of clusters	MAE	Precision	Recall	Intra-cluster distance
User-based CF with exponentially decaying [28]	–	0.73	0.21	0.23	–
DGRec [25]	–	0.51	0.32	0.38	–
Deep-learning-based CF with exponentially decaying	30	0.63	0.26	0.28	0.544
Standard GRU-based CF [29]	30	0.43	0.37	0.4	0.347
Proposed Method	30	0.38	0.48	0.5	0.225

5 Conclusion

In this paper, a combination of rating data and users' social relationships is used to determine the similarities between them in different timestamps. To this end, a weighted graph is formed between users, in which there is an edge between users who have either given rates to shared items, or there is a social relationship between them, or have common neighbors through the two mentioned cases. After a similarity graph between users is created, the auto-encoder network is used to cluster similar users and provide them with a time-based recommendation. There is an auto-encoder in each timestamp, and each has 3 nested layers to reduce the dimensions of the input data and to cluster users. The results of the comparison showed that the proposed method performs better and more accurately in providing personalized recommendations to users than other state-of-the-art methods, and that as long as there is a stream of data, this method can be used in any timestamp via graph updates.

One of the future research areas in this field is to use other sorts of information, such as user demographic data, to determine the similarity between users and provide them with recommendations. Items can also be clustered based on their features, and similar items can be used when presenting a recommendation to the user.

References

1. Rokach L, Ricci F, Shapira B (eds) (2015) Recommender systems handbook. Springer
2. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749
3. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37
4. Vinagre J, Jorge AM, Gama J (2015) An overview on the exploitation of time in collaborative filtering. Wiley Interdiscip Revi Data Min knowl Discov 5(5):195–215
5. Baldominos A, Albacete E, Saez Y, Isasi P (2014) A scalable machine learning online service for big data real-time analysis. In: 2014 IEEE Symposium on Computational Intelligence in Big Data (CIBD), pp 1–8. IEEE
6. Rana C (2018) An Evolutionary clustering algorithm for handling dynamics of user profiles in recommender systems
7. Jalali S, Hosseini M (2021) Social collaborative filtering using local dynamic overlapping community detection. J Supercomput 1–21

8. Goodfellow I, Bengio Y, Courville A, Bengio Y (2016) Deep learning, vol 1, no 2. Cambridge: MIT press
9. Van den Oord A, Dieleman S, Schrauwen B (2013) Deep content-based music recommendation. Adv Neural Inf Process Syst 26:2643–2651
10. Wang H, Wang N, Yeung DY (2015) Collaborative deep learning for recommender systems. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 1235–1244
11. Hidasi B, Karatzoglou A, Baltrunas L, Tikk D (2015) Session-based recommendations with recurrent neural networks. arXiv preprint
12. Bernhardsson E (2014) Recurrent neural networks for collaborative filtering. Online https://erikbern.com/2014/06/28/recurrent-neural-networks-for-collaborative-filtering.html
13. Da'u A, Salim N (2019) Recommendation system based on deep learning methods: a systematic review and new directions. Artif Intelli Rev 1–40
14. Koren Y (2009) Collaborative filtering with temporal dynamics. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 447–456
15. Xiong L, Chen X, Huang TK, Schneider J, Carbonell JG (2010) Temporal collaborative filtering with bayesian probabilistic tensor factorization. In: Proceedings of the 2010 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, pp 211–222
16. Koenigstein N, Dror G, Koren Y (2011) Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In: Proceedings of the Fifth ACM Conference on Recommender Systems, pp 165–172
17. Song Y, Elkahky AM, He X (2016) Multi-rate deep learning for temporal recommendation. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 909–912
18. Liu J, Dolan P, Pedersen ER (2010) Personalized news recommendation based on click behavior. In: Proceedings of the 15th International Conference on Intelligent User Interfaces, pp 31–40
19. Donkers T, Loepp B, Ziegler J (2017) Sequential user-based recurrent neural network recommendations. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, pp 152–160
20. Ko YJ, Maystre L, Grossglauser M (2016) Collaborative recurrent neural networks for dynamic recommender systems. In: Asian Conference on Machine Learning. PMLR, pp 366–381
21. Wu C, Wang J, Liu J, Liu W (2016) Recurrent neural network based recommendation for time heterogeneous feedback. Knowl Based Syst 109:90–103
22. Soh H, Sanner S, White M, Jamieson G (2017) Deep sequential recommendation for personalized adaptive user interfaces. In: Proceedings of the 22nd International Conference on Intelligent User Interfaces, pp 589–593
23. Wu CY, Ahmed A, Beutel A, Smola AJ, Jing H (2017) Recurrent recommender networks. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pp 495–503
24. Sun P, Wu L, Wang M (2018) Attentive recurrent social recommendation. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp 185–194
25. Song W, Xiao Z, Wang Y, Charlin L, Zhang M, Tang J (2019) Session-based social recommendation via dynamic graph attention networks. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp 555–563
26. Zhou S, Dai X, Chen H, Zhang W, Ren K, Tang R, Yu Y (2020) Interactive recommender system via knowledge graph-enhanced reinforcement learning. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 179–188
27. Song C, Liu F, Huang Y, Wang L, Tan T (2013) Auto-encoder based data clustering. In Iberoamerican congress on pattern recognition. Springer, Berlin, Heidelberg, pp 117–124
28. Ding Y, Li X (2005) Time weight collaborative filtering. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp 485–492
29. Cho K, Van Merriënboer B, Bahdanau D, Bengio Y (2014) On the properties of neural machine translation: encoder-decoder approaches. arXiv preprint