CAAI Transactions on Intelligence Technology

Research Article

Multi-model deep learning approach for collaborative filtering recommendation system

ISSN 2468-2322
Received on 2nd February 2020
Revised on 12th May 2020
Accepted on 14th July 2020
doi: 10.1049/trit.2020.0031
www.ietdl.org

Mohammed Fadhel Aljunid ✉, *Manjaiah Doddaghatta Huchaiah*

Computer Science, Mangalore University, Mangalore, India
✉ *E-mail: ngm505@yahoo.com*

Abstract: As a result of a huge volume of implicit feedback such as browsing and clicks, many researchers are involving in designing recommender systems (RSs) based on implicit feedback. Though implicit feedback is too challenging, it is highly applicable to use in building recommendation systems. Conventional collaborative filtering techniques such as matrix decomposition, which consider user preferences as a linear combination of user and item latent features, have limited learning capacities, hence suffer from a cold start and data sparsity problems. To tackle these problems, the research direction towards considering the integration of conventional collaborative filtering with deep neural networks to maps user and item features. Conversely, the scalability and the sparsity of the data affect the performance of the methods and limit the worthiness of the results of the recommendations. Therefore, the authors proposed a multi-model deep learning (MMDL) approach by integrating user and item functions to construct a hybrid RS and significant improvement. The MMDL approach combines deep autoencoder with a one-dimensional convolution neural network model that learns user and item features to predict user preferences. From detail experimentation on two real-world datasets, the proposed work exhibits substantial performance when compared to the existing methods.

1 Introduction

Recommender system (RS) is an information filtering algorithm that makes suggestions for items that lead users based on their interest. In today's internet era, the RSs play a vital role in handling the information overload complexities. With the swift growth of internet and commercial enterprises, the size of data loaded to the internet rises alarmingly. The huge volume of data has led to information overload related complexities on the internet. The RS has proven to be a successful and operative way to address the challenges associated with information overload problem over the internet. The RSs effectively deliver users with valuable information to their queries such as movies [1], music [2, 3], books [4], news [5], research articles [6] and products in general.

The RSs have become an integral component in websites such as Google, Amazon, Netflix, YouTube and others [7–11]. RSs used many algorithms such as content-based ones [9, 12], collaborative filtering [13] and trust-based RS [14] algorithm. The collaborative filtering algorithm [9, 15, 16] is most popularly used. It does not need any previous knowledge about users or items, instead, it makes recommendations based on the interaction between them. The primary purpose of a collaborative filtering algorithm is to project users' preferences on items based on their past browsing and reaction history such as labelling ratings, browsing, clicks and so on. Although a collaborative filtering algorithm is efficient and simple, it suffers from numerous problems such as cold start, prediction accuracy [17] and a shortage of capturing complex interactions between the user and the item [18]. Among the various collaborative filtering algorithms, the matrix decomposition (MD) [8, 19, 20] matches users and items into a common latent space using a vector of latent features showing a user or an item. Then, a user's interaction on an item got mapped as the dot product of their corresponding latent vectors.

Popularised by the MovieLen and Netflix Prize, MD becomes the de facto method in the latent factor model-based recommendation. Many research works are going on to enhance the MD in the direction of combining it with neighbour-based models [21], integration of the model with topic-based models of the item's content [22] and extension of the method to MD for generic modelling of features [23]. Despite of the success of MD method in collaborative filtering, it is well known that its performance is highly hindered based on the interaction function of the dot product. The dot product is not effective in capturing the complex structure of user interaction data since it works by combining the multiplication of latent features linearly [24].

Nowadays, the deep neural networks (DNNs) are achieving remarkable performance in various research areas ranging from image and video processing, speech recognition and to text processing [25–28]. In contrast, to the vast amount of literature on MD approaches, there are few research work on employing DNNs in recommendation systems. Recent improvements are applied to DNNs in recommendation tasks and promising results are achieved. Most of the recent works use DNNs to model supplementary information such as the textual description of items, audio features of music and visual content of images [22, 29, 30]. With regard to modelling of recommendation systems, the vital collaborative filtering effect is still resorted and applied in MD by combining the user and item latent features using an inner product. He *et al.* [24] integrated multi-layer perceptrons (MLPs) and MD to form a neural network-based collaborative filtering (NCF) method to map user and item latent vectors that could learn an arbitrary function from data to tackle the complex interaction of the user and the item. The method exhibits notable enhancement over corresponding models. They have implemented replacing the dot product with a neural model that can learn an arbitrary auction from data. According to Low *et al.* [31], the drawback of NCF method is that every perceptron on a given layer is connected to every perceptron in the next layer in MLP. Such full connectivity makes the model less effective in the case of learning new features.

Our work basically formalises a neural network modelling approach for collaborative filtering algorithm. Our focus is on implicit feedback, which indirectly reflects users' preference through behaviours such as purchasing products, watching videos and clicking items. When compared with the explicit feedback (i.e. reviews and ratings), implicit feedback enables to be tracked automatically, hence is much easier to collect for content providers. Nevertheless, it is too challenging to utilise, since user satisfaction is not observed (not rated) and there is a natural shortage of negative feedback. This work addresses the above-mentioned research problems by utilising DNNs to project noisy implicit feedback signals.

In our work, we propose a multi-model deep learning (MMDL) approach that considers the strengths of deep auto-encoder neural network (DeepAEC) and one-dimensional conventional neural network approach (1D-CNN) to effectively enhance the performance of collaborative filtering algorithm. We have carried out extensive experimentation on two real-world datasets to demonstrate the effectiveness of our proposed DeepAEC and 1D-CNN work in collaborative filtering algorithm.

The rest of this work is organised as follows. The related works are presented in Section 2, a methodology is given in Section 3. Experiments and results are given in Section 4. Finally, a conclusion and future work are given in Section 5.

2 Related works

In this section, we explore recent related works and presented them in paragraphs according to their domain.

Several model-based recommendation approaches have been proposed such as Bayesian approaches [32], latent semantic approaches [33], clustering approaches [34], regression-based approaches [15] and matrix factorisation approaches [35] in order to ameliorate the above-mentioned predicaments. Among the several collaborative filtering approaches, the MD is the most popular method. This algorithm projects both users and items to vectors with the same dimension, which represents user and item latent features. The representative works of this algorithm are such as non-singular value decomposition [36], singular value decomposition (SVD) [36], probabilistic matrix factorisation (PMF) [37] and parametric probabilistic principal component analysis [38]. However, there is inefficiency in the latent vectors learned by MD algorithms, more especially in the case of rating matrix is very sparse.

Recently, the application of deep learning is achieving remarkable successes in the area of collaborative filtering [39], which is considered to be the first work to apply deep learning approach. It performs inference by applying the restricted Boltzmann machine approach. Wang *et al.* [22] proposed collaborative deep learning model from two models namely stacked denoising autoencoder and PMF. Xue *et al.* [40] developed a depth MD model. The conventional MD approach is used to decompose the user and items feature matrix. The multilayer feed-forward neural network is used to deeply mine associated features. The inner product of the corresponding low-dimensional features is the predicted rating of the recommendation system. Zhang *et al.* [41] proposed an Auto SVD++ model that applies the video data features learned by shrinking the auto-encoder and the implicit feedback captured by SVD++ to enhance the recommendation accuracy.

Ouyang *et al.* [42] developed auto-encoder based collaborative filtering (ACF). The ACF method splits the user's ratings value of the item into five vectors. The limitations of this method are, it solves the integer scoring prediction problem that rises the sparsity of the scoring matrix that reduces the prediction accuracy of the ACF algorithm. In addition, Sedhain *et al.* [43] developed AutoRec. The core purpose of the AutoRec model is to reconstruct the original input data. Even though the AutoRec model solves the issue of non-integer prediction scoring values, it does not add noise to the input that leads the model to be less robust and highly prone to overfitting. Wu *et al.* [44] developed CDAE that is used to make the ranking prediction. The input of this model is the user's implicit feedback data of the items. More specifically, every perceptron in the input part of the model corresponds to an item, and is able to consider as user's preference of the item's interest and the user's preference for an item is denoted by the values of 0 or 1. Finally, the items associated to the predicted values of the output layer perceptrons in the model are sequentially recommended to the user. According to Yan *et al.* [45], the limitation of the above two models is that there is a cold start problem. Strub *et al.* [46] proposed a CFN model that merges content information and a scoring matrix to display as final prediction results. The recommendation accuracy of the model has enhanced compared with the previous methods. According to Yan *et al.* [45], the disadvantage of this model is that the content information is comparatively easy and the data is very sparse.

Convolutional neural networks (CNNs) are most commonly used in image processing and computer vision. Basically, CNN consists of convolutional layers (CL) followed by pooling layer (PL) and fully connected layers (FL). CNNs have fewer parameters compared with MLP with the same number of the perceptron, which makes it easy to train [47]. The CL extract features from the input and generates n feature maps, where n is the number of filters. The PL is accountable in reducing the dimensionality of features to address the problems associated with high curse of dimensionality with the feature maps. The ConvMF [48] incorporates CNN with PMF to utilise the contextual information of documents in order to address the data sparsity issue and enhance the prediction accuracy. The bag of word approach is not effective in addressing sparcity related issues as it ignores the order of the words so as to deteriorate the contextual meaning of the textual information. To alleviate this problem, CNN model is used to generate the document latent vector and integrate them with the epsilon variable in the PMF model to produce the final prediction report.

CNNs are developed to operate exclusively on 2D data such as images and videos. That is why they are often represented as 2D-CNNs. Recently, a modified version of the 2D CNN model namely 1D-CNNs is attracting the attention of researchers [49]. Several studies and applications have shown that 1D-CNNs model are powerful than their 2D counterparts in dealing with 1D features, due to the forward and back-propagation, the 1D-CNNs require simple array operations [50]. It means that the computational complexities of 1D-CNNs are significantly lower than the 2D-CNNs. Recent studies show that 1D-CNNs with relatively shallow architectures (i.e. a small number of hidden layers and perceptrons) can learn challenging tasks involving 1D features [50]. On the other hand, the 2D-CNNs model requires more profound, more in-depth architecture for training and implementation. The 1D-CNNs are well-suited for real-time and low-cost applications such as mobile and hand-held devices, due to their lower computational requirements.

3 System overview

The aim of this work is to explore a collaborative filtering approach based on implicit feedback, so we have selected userID and movieID (itemID) as feature. To learn user–item interaction, we proposed a MMDL approach that combines deep autoencoder (DeepACE) with 1D-CNN represented in Fig. 1. Both models share the same input. In the subsections, we start by defining the problem of collaborative filtering approach based on implicit data in Section 3.1. We present the DeepACE model in Section 3.2. The 1D-CNN is presented in Section 3.3. Lastly, the proposed MMDL model is described in Section 3.4.

3.1 Problem formulation

In this work, the recommendation task is targeted for implicit feedback of collaborative filtering algorithm.

In the implicit feedback recommendation scenario, it is often referred to the positive feedback and not to the negative feedback, unlike explicit feedback which has both negative and positive feedbacks. In the score ranges 1–5, it presents the degree of tendency from 'dislike' to 'very like' (see Fig. 2*b*). Implicit feedback includes only observed (selected) and unobserved (unselected) events (see Fig. 2*a*). The selected scenario can be regarded as a positive tendency, and the unselected cannot, it is simply regarded as a negative tendency, because unselected items are mixed with items that users are actually not interested in and items that users do not find but are interested. Therefore, lack of

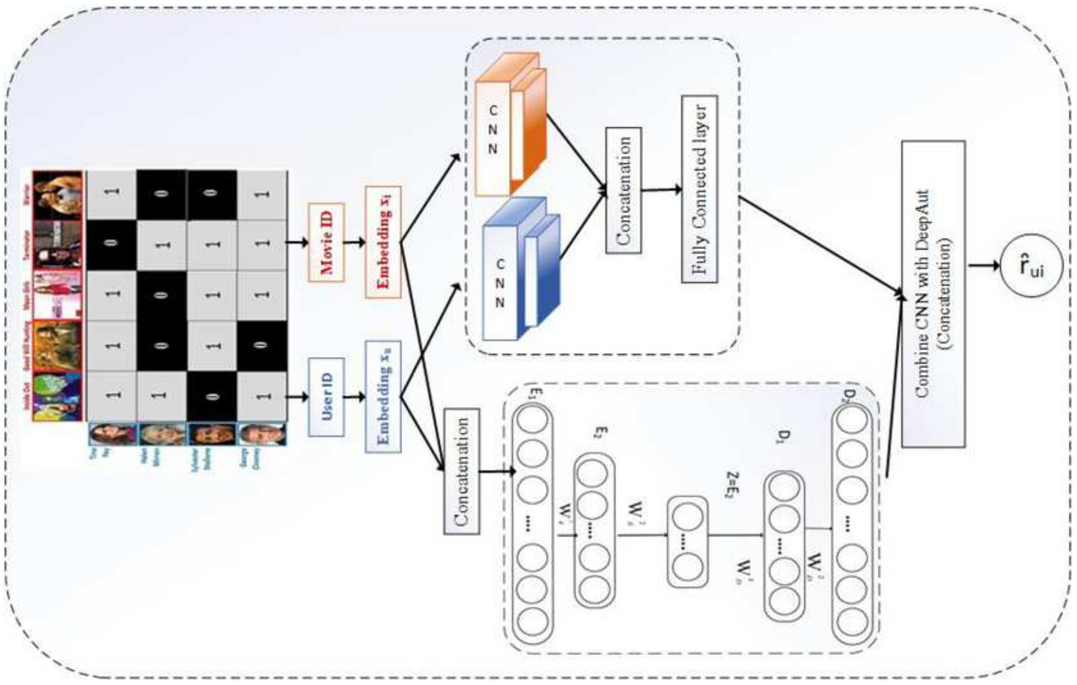


Fig. 1 *Proposed MMDL framework*

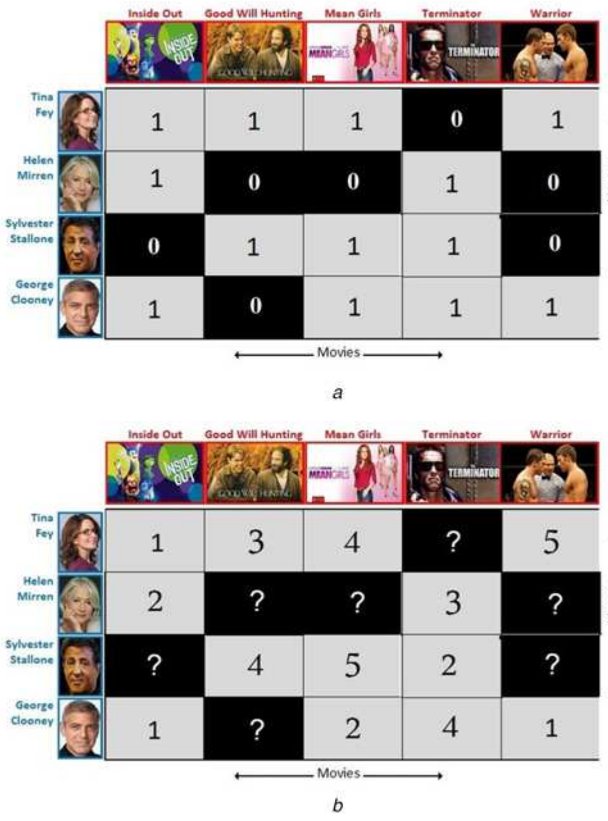


Fig. 2 *Simple example of the observed difference between implicit feedback and explicit feedback rating matrices*

a Rating matrices for implicit feedback
b Rating matrices for explicit feedback

negative settings can cause difficulties in trainings in recommendation systems.

Suppose m and n indicate the set of users and items (movies), respectively (see Fig. 2*a*). We define the user/movies implicit feedback matrix $\boldsymbol{R} \in \mathbb{R}^{m\times n}$ as

$$r_{ui} = \begin{cases} 1, & \text{if interaction (user } u\text{, movie } i\text{) is observed;} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

The entry r_{ui} denotes user's ith rating score of the movie ith. If the value of r_{ui} is 1, then is shows an interaction between user and movie. Otherwise, there is no interaction between the user and the movie. Since these interactions do not specify whether actually the user likes or dislikes the movie, there is a possibility of noise signals. Moreover, we let the user's latent feature matrix be $\boldsymbol{x} \in R^{m\times a}$, the vector $\boldsymbol{x}_u$ of the ith represents the features of ith users and n represents the dimension of features. Similarly, the latent feature of movies is represented as $\boldsymbol{x} \in R^{n\times b}$.

We formulate predicted recommendations from implicit data as predicting the scores of unobserved entries in R, which are used for ranking the movies.

3.2 Deep autoencoder neural network

For the DeepAEC model [51, 52], the initial input layer has got two input vectors namely, $\boldsymbol{x}_u$ and $\boldsymbol{x}_i$ that represent the features of userID u and movieID i, respectively. These are sparse binary vectors with one-hot encoding. These vectors are concatenated as features based on the following equation

$$x = \text{Concatenate}(\boldsymbol{x}_u, \boldsymbol{x}_i) \quad (2)$$

Next to the input layer, the embedding layer comes, which is fed into the DeepACE fully-connected layers that consist of encoder and decoder functions to map the latent vectors to predict the scores. More precisely, the fully-connected layers for the encoding model target to convert the original data with high dimensionality into a low-dimensional space. Likewise, fully-connected decoder layers are considered as the inverse procedure of the encoder network that are used to reconstruct the original data from the code and then encode and decode before finally fused on layer z (see (3)

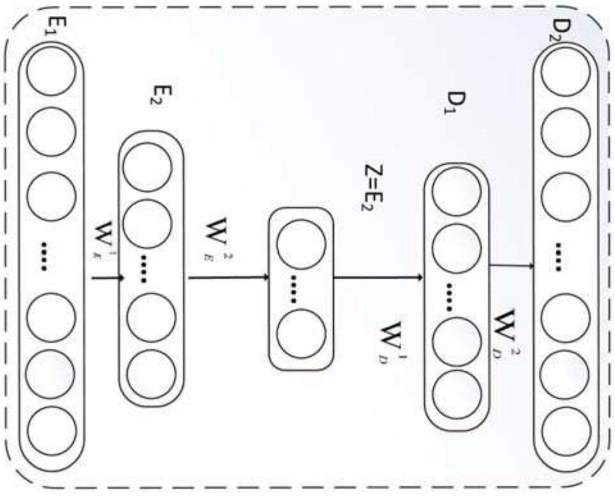


Fig. 3 *Encoder and decoder model*

and Fig. 3)

$$\left.\begin{aligned} E_1 &= \mathrm{Relu}(W_E^1)\cdot x + b_1 \\ E_2 &= \mathrm{Relu}(W_E^2)\cdot E_1 + b_2 \end{aligned}\right\}\text{(Encoder)}$$
$$z = E_2 \tag{3}$$
$$\left.\begin{aligned} D_1 &= \mathrm{Relu}(W_D^1)\cdot z + b_3 \\ D_2 &= \mathrm{Relu}(W_D^2)\cdot D_1 + b_4 \end{aligned}\right\}\text{(Decoder)}$$

Where by $\boldsymbol{W}$ and $\boldsymbol{b}$ denote the weight matrices and biases of each layer, and $E_1, E_2 \ldots E_L$, $D_1, D_2 \ldots D_L$ denote the output of the encoder and decoder layer, respectively, which are activated by the rectified linear unit (ReLU) function. We have used the Relu as the activation function in the hidden layer because it is the most efficient and easier to compute and translate [53]

$$\mathrm{Relu}(e) = \max(e, 0) \tag{4}$$

Finally at the output layer, the features acquired from the previous inner layer got projected into the output layer to produce the estimated score based on the following equation:

$$\hat{r}_{ui} = \delta(w_L^{\mathrm{T}} D_L + b_L) \tag{5}$$

where δ is the sigmoid function, which is given as $\delta(z) = (1/(1+\delta^{-z}))$.

3.3 1D convolution neural network model

The 1D CNN model [50] takes two vectors, namely, userID $\boldsymbol{x}_u$ and movieID $\boldsymbol{x}_i$, as input. In the 1D CNN separate feature extraction models operate on each vector, which summaries latent vectors $\boldsymbol{x}_u$ and $\boldsymbol{x}_i$ into shorter vector by convolving throughout the 1D CNN. More specifically: suppose long vector $\boldsymbol{Z}$ with n elements having weight $\boldsymbol{W}$ convolved to form m elements into short vector $\boldsymbol{Y}$ with $n-m+1$ elements as shown in the following equation:

$$y_i = \sum_{j=m-1}^{0} z_{i+j} * w_j \tag{6}$$

where $i = (1, n-m+1)$

$$\begin{pmatrix} x\colon & z_1 & z_2 & z_3 \\ w\colon & \frac{1}{2} & \frac{1}{2} & \\ w\colon & & \frac{1}{2} & \frac{1}{2} \end{pmatrix} = \left(y\colon \frac{z_1+z_2}{2} \frac{z_2+z_3}{2}\right) \tag{7}$$

So, if we have a vector of length n, and the weight matrix is also length n $w_i = 1/n$, then the convolution will produce a vector of length z equal to the average value of all values in the input matrix. It is a sort of degenerate convolution. If the same weight matrix is one shorter than the input matrix, then we get a moving average in the output of length 2 (see (7)). Based on (6), the input of 1D-CNN model has two long input vectors namely $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$, which represent the features of users and movies, respectively

$$c_{\mathrm{user}} = \sum_{j=m-1}^{0} p_{u+j} * w_j \tag{8}$$

$$c_{\mathrm{item}} = \sum_{j=m-1}^{0} q_{i+j} * w_j \tag{9}$$

The convolution layer uses PLs as shown in the following equations:

$$\boldsymbol{x}_u = \mathrm{pooling}(\mathrm{Conv}(c_{\mathrm{user}})) \tag{10}$$

$$\boldsymbol{x}_i = \mathrm{pooling}(\mathrm{Conv}(c_{\mathrm{item}})) \tag{11}$$

Pooling $(\cdot)$ function has the avgpool $(\cdot)$ and maxpool $(\cdot)$ variants. In our work, we used the MaxPooling that reduces the computational complexity by reducing the number of parameters to learn and provide basic translation that are invariant to the internal representation. Then the results from both vectors are concatenated into one long vector (see (12) and Fig. 4) followed by a FL as shown in (13) at the output layer

$$x_L(\boldsymbol{x}_u, \boldsymbol{x}_i) = \begin{pmatrix} \boldsymbol{x}_u \\ \boldsymbol{x}_i \end{pmatrix} \tag{12}$$

$$x_{ui}^{\mathrm{1Dcnn}} = \delta(w_L^{\mathrm{T}} x_L + b_L) \tag{13}$$

3.4 Integrate the eepACE and 1D-CNN models

In this section, the proposed multi-modal neural network (MMDL) is presented to enhance the model of complex user–item interactions.

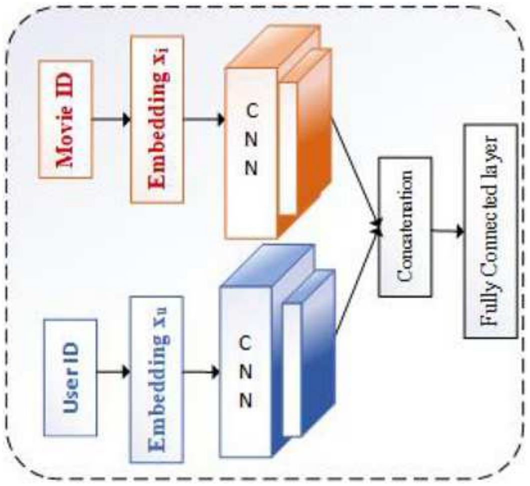


Fig. 4 *Feature of user and movie is processing by 1D-CNN model and the outputs of each of these features are combined to gather followed by a FL*

The easiest method is to fuse the DeepACE and 1D-CNN model to enhance the reinforcement learning due to the combined models to learn the complex user–item interactions from the data. The DeepACE (see (14)) and 1D-CNN (See (15)) models have embedding layers that give a sort of flexibility to fuse the models [24]. We have concatenated the last hidden layers of DeepACE and 1D-CNN to get the enhanced result and predict the i_{th} user's rating score on the i_{th} item as shown in (16) and Fig. 1

$$\phi^{\text{DeepACE}} = \delta(w_L^{\text{T}} D_L + b_L) \tag{14}$$

$$\phi^{\text{1D-CNN}} = \delta(w_L^{\text{T}} x_L + b_L) \tag{15}$$

$$\hat{r}_{ui}^{\text{MMDL}} = \begin{bmatrix} \phi^{\text{1Dcnn}} \\ \phi^{\text{DeepACE}} \end{bmatrix} \tag{16}$$

The proposed model enables to a collaborative filtering based on implicit feedback that does not require any additional information besides from the interaction between users and items and effectively enhanced the recommendation accuracy. First, we used the user–item rating matrix to obtain the features of the users and items. Then, we regarded these features as the input of the DeepACE and 1D-CNN. We trained the DeepACE and 1D-CNN models to learn separate embeddings and combine the two models by concatenating their last hidden layer. The blueprint framework of our work is shown in Fig. 1 and is mathematically expressed in (16). In the output layer, the model obtained the probability values that represent the scores that the user might give. Finally, the score with the highest probability is used as the prediction result of the proposed model.

3.4.1 Model training: There are various objective functions to train and validate models [54]. The most common objective functions to train recommendation systems are of three types namely pair-wise, point-wise and list-wise. The pair-wise objective function deals on users' preferences considering pairs of items that are assumed to be suitable to pick-out the top-N recommendations. The point-wise objective function targets at getting precise ratings that are essential in rating prediction tasks [9]. The list-wise objective function mainly focused on users' preferences towards a list of items that are used in deep learning algorithms. In our proposed model, we used a point-wise objective function and the general calculation is shown in the following equation:

$$L = \sum_{u \in U} \sum_{i \in I} C_{\text{lossfun}}(r_{ui}, \hat{r}_{ui}) + \lambda \Omega(\Theta) \tag{17}$$

where $C(\cdot)$ denotes the loss function, $\Omega(\Theta)$ represents a regularisation expression that regulates the complexity of the model and encodes prior information regarding the sparsity of the data, non-negativity or graph. There are various types of point-wise loss functions [54]: the squared loss and log loss function. The squared loss is suitable for explicit feedback. The log loss function (shown in (18)), which we chose it to train the proposed model, is mainly used for implicit feedback for classification tasks and it performs well with implicit feedback when compared with the squared loss

$$L = \sum_{u \in U} \sum_{i \in I} r_{ui} \log \hat{r}_{ui} + (1 - r_{ui}) \log (1 - \hat{r}_{ui}) \tag{18}$$

3.4.2 Making recommendations: After the training of the proposed model, we were able to utilise it to predict a user's rating score on the movies that have not been observed (rated) by the user. When making recommendations for a certain user, we were able to recommend the movies with the best-predicted score for the user.

4 Experimental study

In this section, we present our experimentation, which includes experimental setup in Section 4.1, dataset description is given in Section 4.2, evaluation metrics are presented in Section 4.3. The settings of the proposed model are given in Section 4.4 and finally experimental results are demonstrated in Section 4.5.

4.1 Experimental setup

We implemented our experimentation of the models on Ubuntu 16.4 operating system running on Intel® Core™ i5-2400 CPU 3.10 GHz 4 processors and a hard disk of 500 GB. We used the python language version 2.7 and the Keras 2.0 with TensorFlow 3.0 as backend.

4.2 Dataset description

Movielens is a movie rating dataset which was collected through the on-going Movielens project. It is distributed by GroupLens Research at the University of Minnesota. It is one of the most commonly used datasets for evaluating collaborative filtering models. There are different kinds of this dataset available in http://www.movielens. MovieLens 100k and MovieLens 1M datasets are used to test the performance of the proposed model and other models used for comparison. The MovieLens 100k dataset consists of 100,000, while MovieLens 1M datasets consist of 1,000,000 (million) movie ratings, each rating is an integer between 1 (the poorest) to 5 (the highest), and each user has given more than 20 ratings. These ratings given by user are explicit, and we have selected this specific dataset explicitly to evaluate the learning of implicit feedback from explicit ratings. We transformed it to implicit data by converting every entry to 1 or 0 which indicates whether user has selected (rated) the item or not. During testing, all test functions take (user, item) pairs as input parameters and return the predicted rating. After reading in the data, the ratings matrix is filled with the scored data, with user as the row and item as the column, forming a matrix of (943×1682), (6040×3952) for MovieLens 100k and MovieLens 1M, respectively. Detail description of datasets is presented in Table 1.

4.3 Evaluation metrics

The root mean square error (RMSE) [55] is used to evaluate the prediction performance of the proposed model. The RMSE is defined as shown in the following equation:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{ui} (r_{ui} - \hat{r}_{ui})^2} \tag{19}$$

where n is the total number of predicted movies, r_{ui} represents the predicted value for user u on movie i and $\hat{r}_{ui}$ is the true rating.

4.4 Modelling setting

The MMDL model is implemented in Python based on the Keras 2 with TensorFlow 3 as back-end. We randomly initialised the MMDL model parameters using a normal distribution with means of 0 and standard deviation of 0.01. We tested the embedding vector size set to 8 and 10 for user and movie embedding vector size, respectively. We experimented with several optimisers, and Adam obtains the best result (see Table 2). In consequence, Adam

Table 1 Description of the MovieLens datasets

	MovieLens 100K	MovieLens 1M
number of users	943	6040
number of items	1682	3952
number of ratings	100,000	1,000,209
number of ratings per user	106.41	165.60
number of ratings per movies	59.451	253.090
ratings sparsity	93.701%	95.810%

Table 2 Experimental results for different optimisers

Optimiser result	Movilens 100K	Movielens 1M
Adadelta	1.032 ± 0.02600	0.953 ± 0.0203
Adagrad	0.941 ± 0.01639	0.951 ± 0.0185
Adam	**0.931 ± 0.00356**	**0.881 ± 0.00189**
Adamax	0.952 ± 0.02668	0.947 ± 0.0142
RMSprop	0.943 ± 0.01768	0.9441 ± 0.0200
SGD	1.132 ± 0.01209	0.952 ± 0.0154

Significance of bold is the algorithm obtained the best value among the values in the table

optimiser is selected with 0.001 learning rate and parameter values as provided in [56].

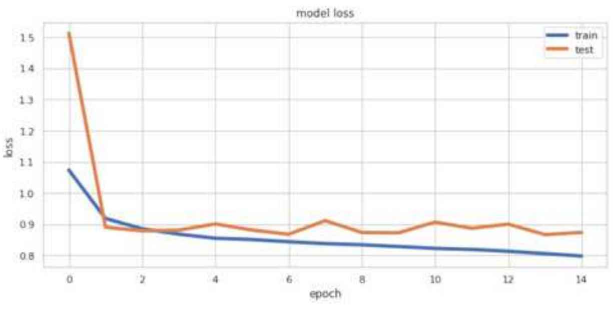
All biases are initialised with zero. The batch size has been set to 64 and 128 for 100k and 1M MovieLen dataset, respectively. We run the models for 15 epochs to each datasets.

4.4.1 DeepAEC model: We have concatenated the userId and moviesId features to a single layer followed by encoder hidden layer with ($256 \rightarrow 128 \rightarrow 64 \rightarrow 50$) and decoder hidden layer with ($50 \rightarrow 64 \rightarrow 128 \rightarrow 256$) perceptrons. The dropout layers are added to the input and hidden layers to prevent over-fitting, which is set 0.2. The output layer has been set in 100 perceptrons.

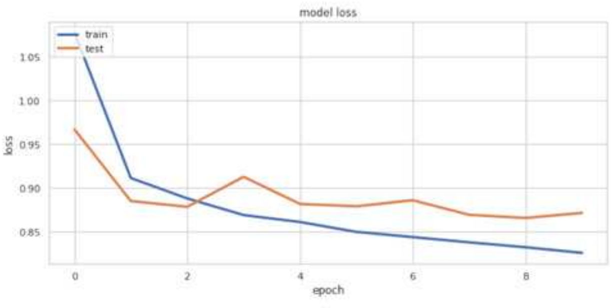
4.4.2 1D-CNN model: First 1D-CNN layer applies 16 filters with kernel size of 2 with padding let to be the same. We have used maxpooling function with 2 size, strides is 2, data format is channels first, and combined into one vector by flattening to form a single vector. The hidden features in this vector are fully connected to a hidden layer of 100 perceptron.

4.5 Experimental results

In this section, experimental results of the proposed approach and corresponding comparison with state-of-the-art methods are presented. Figs. 5*a* and *b* illustrate the performance of RMSE for the train and test data versus the number of training epochs on 100k and 1M MovieLen dataset, respectively. The RMSE is also used as the evaluation metric to measure the performance among the different models. The five-fold cross-validation approach has been used in each dataset. This process is repeated five times and calculated the mean value of RMSE for each model and their mean is shown in Table 3. Especially, the value of RMSE of the proposed model implemented on the Movilens-100k datasets is (0.931 ± 0.00356), and the RMSE value of the proposed model implemented on MovieLens 1M datasets is (0.881 ± 0.00189). Additionally, Figs. 6*a* and *b* depict the clear line of the comparison of the proposed model with the state-of-the-art method for the two datasets. Based on the experimental results, the proposed model has the highest recommendation accuracy compared with the existing approaches.



a



b

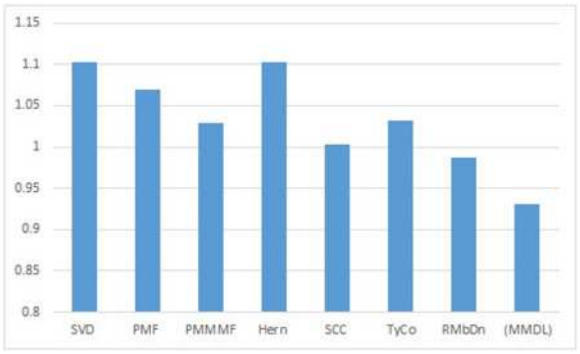
Fig. 5 *RMSE for MovieLens dataset*

a RMSE performance on MovieLens 100k dataset
b RMSE performance MovieLens 1M dataset

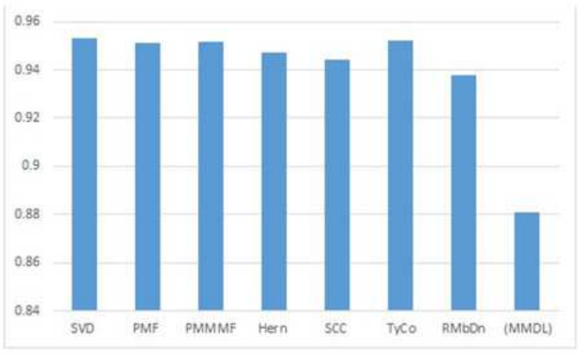
Table 3 Experimental results on Movilens 100K and 1M using RMSE

Recommendation systems	Movilens 100K	Movielens 1M
SVD [36]	1.103 ± 0.0303	0.9533 ± 0.0203
PMF [37]	1.069 ± 0.0284	0.9512 ± 0.0185
PMMMF [57]	1.0291 ± 0.031	0.9515 ± 0.0184
Hern [58]	1.1022 ± 0.030	0.9473 ± 0.0142
SCC [59]	1.0036 ± 0.029	0.9441 ± 0.020
TyCo [60]	1.0316 ± 0.027	0.9522 ± 0.015
RMbDn [53]	0.987 ± 0.0291	0.938 ± 0.0151
proposed model (MMDL)	**0.931 ± 0.00356**	**0.881 ± 0.00189**

Significance of bold is the algorithm obtained the best value among the values in the table



a



b

Fig. 6 *Experimental results on the two real-world datasets with different recommendation models*

a Recommendation models performance comparison on MovieLens 100k dataset
b Recommendation models performance comparison MovieLens 1M dataset

5 Conclusion and future work

Collaboration filtering (CF) techniques play a vital role in designing and developing recommendation systems. CF techniques have a limitation that it suffers the sparsity of the data, which represents matrix ratings, scalability and integrals nature of data. In this work, we proposed a multi-modal deep learning approach for a collaborative RS (MMDL), which combines between DeepACE neural network with a 1D conventional neural network. We have made a comparative study of the proposed model with stat-of-the-art. Our experimental results show that the MMDL depicts the best performance on RMSE measures among the existing approaches. We evaluated the model on two real-world datasets, which is 100k and 1M MovieLens dataset. In our future work, we are planning to work on the explicit feedback as the implicit feedback is not sufficient in providing full-fledged recommendation system.

6 References

[1] Carrer-Neto, W., Hernández-Alcaraz, M.L., Valencia-García, R., *et al.*: 'Social knowledge-based recommender system. Application to the movies domain', *Expert Syst. Appl.*, 2012, **39**, (12), pp. 10990–11000

[2] Bogdanov, D., Haro, M., Fuhrmann, F., *et al.*: 'Semantic audio content-based music recommendation and visualization based on user preference examples', *Inf. Process. Manage.*, 2013, **49**, (1), pp. 13–33

[3] Al-Hassan, M., Lu, H., Lu, J.: 'A semantic enhanced hybrid recommendation approach: a case study of e-government tourism service recommendation system', *Decis. Support Syst.*, 2015, **72**, pp. 97–109

[4] Kim, H.K., Oh, H.Y., Gu, J.C., *et al.*: 'Commenders: a recommendation procedure for online book communities', *Electron. Comm. Res. Applic.*, 2011, **10**, (5), pp. 501–509

[5] Cleger-Tamayo, S., Fernández-Luna, J.M., Huete, J.F.: 'Top-n news recommendations in digital newspapers', *Knowl.-Based Syst.*, 2012, **27**, pp. 180–189

[6] Son, J., Kim, S.B.: 'Academic paper recommender system using multilevel simultaneous citation networks', *Decis. Support Syst.*, 2018, **105**, pp. 24–33

[7] Koutrika, G.: 'Modern recommender systems: from computing matrices to thinking with neurons'. Proc. 2018 Int. Conf. on Management of Data (ACM), Houston TX, USA, May 2018, pp. 1651–1654

[8] Aljunid, M.F., Manjaiah, D.: 'Movie recommender system based on collaborative filtering using apache spark', in Balas, V., Sharma, N., Chakrabarti, A. (Eds.): 'Data management, analytics and innovation', Advances in Intelligent Systems and Computing, vol. 839 (Springer, Singapore, 2018), pp. 283–295

[9] Aljunid, M.F., Manjaiah, D.: 'A survey on recommendation systems for social media using big data analytics', *Int. J. Latest Trends Eng. Technol., Spec. Issue (SACAIM 2017)*, 2017, pp. 48–58

[10] Nassar, N., Jafar, A., Rahhal, Y.: 'A novel deep multi-criteria collaborative filtering model for recommendation system', *Knowl.-Based Syst.*, 2020, **187**, p. 104811

[11] Aljunid, M.F., Manjaiah, D.: 'An improved ALS recommendation model based on apache spark'. Soft Computing Systems. ICSCS 2018. Communications in Computer and Information Science, Kollam, India, 2018, vol. 837, pp. 302–311

[12] Ricci, F., Rokach, L., Shapira, B.: 'Recommender systems: introduction and challenges', in Ricci, F., Rokach, L., Shapira, B. (Eds.): 'Recommender systems handbook' (Springer, Boston, MA, 2015), pp. 1–34

[13] Ekstrand, M.D., Riedl, J.T., Konstan, J.A., *et al.*: 'Collaborative filtering recommender systems', *Found. Trends® Hum.–Comput. Interact.*, 2011, **4**, (2), pp. 81–173

[14] O'Donovan, J., Smyth, B.: 'Trust in recommender systems'. Proc. of the 10th Int. Conf. on Intelligent user Interfaces, San Diego California, USA, October 2005, pp. 167–174

[15] Sarwar, B.M., Karypis, G., Konstan, J.A., *et al.*: 'Item-based collaborative filtering recommendation algorithms'. Proc. of the 10th int. Conf. on World Wide Web, Hong Kong, Hong Kong, 2001, vol. 1, pp. 285–295

[16] Zhang, H., Shen, F., Liu, W., *et al.*: 'Discrete collaborative filtering'. Proc. of the 39th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, Pisa, Italy, July 2016, pp. 325–334

[17] Wasid, M., Ali, R.: 'An improved recommender system based on multi-criteria clustering approach', *Procedia Comput. Sci.*, 2018, **131**, pp. 93–101

[18] Fu, M., Qu, H., Yi, Z., *et al.*: 'A novel deep learning-based collaborative filtering model for recommendation system', *IEEE Trans. Cybern.*, 2018, **49**, (3), pp. 1084–1096

[19] Li, S., Kawale, J., Fu, Y.: 'Deep collaborative filtering via marginalized denoising auto-encoder'. Proc. of the 24th ACM Int. on Conf. on Information and Knowledge Management, Melbourne, Australia, October 2015, pp. 811–820

[20] He, X., Zhang, H., Kan, M.Y., *et al.*: 'Fast matrix factorization for online recommendation with implicit feedback'. Proc. of the 39th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, Pisa, Italy, July 2016, pp. 549–558

[21] Koren, Y.: 'Factorization meets the neighborhood: a multifaceted collaborative filtering model'. Proc. of the 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 2008, pp. 426–434

[22] Wang, H., Wang, N., Yeung, D.Y.: 'Collaborative deep learning for recommender systems'. Proc. of the 21th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Sydney NSW, Australia, August 2015, pp. 1235–1244

[23] Rendle, S.: 'Factorization machines'. 2010 IEEE Int. Conf. on Data Mining, Sydney, Australia, December 2010, pp. 995–1000

[24] He, X., Liao, L., Zhang, H., *et al.*: 'Neural collaborative filtering'. Proc. of the 26th Int. Conf. on World Wide Web, Perth, Australia, April 2017, pp. 173–182

[25] Zhang, H., Yang, Y., Luan, H., *et al.*: 'Start from scratch: towards automatically identifying, modeling, and naming visual attributes'. Proc. of the 22nd ACM Int. Conf. on Multimedia, Orlando Florida, USA, November 2014, pp. 187–196

[26] Hong, R., Hu, Z., Liu, L., *et al.*: 'Understanding blooming human groups in social networks', *IEEE Trans. Multimed.*, 2015, **17**, (11), pp. 1980–1988

[27] He, K., Zhang, X., Ren, S., *et al.*: 'Deep residual learning for image recognition'. Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 2016, pp. 770–778

[28] Collobert, R., Weston, J.: 'A unified architecture for natural language processing: deep neural networks with multitask learning'. Proc. of the 25th Int. Conf. on Machine Learning, Helsinki, Finland, 2008, pp. 160–167

[29] Van den Oord, A., Dieleman, S., Schrauwen, B.: 'Deep content-based music recommendation'. Advances in Neural Information Processing Systems, Lake Tahoe, NV, United States, 2013, pp. 2643–2651

[30] Zhang, F., Yuan, N.J., Lian, D., *et al.*: 'Collaborative knowledge base embedding for recommender systems'. Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 2016, pp. 353–362

[31] Low, Y.H., Yap, W.S., Tee, Y.K.: 'Convolutional neural network-based collaborative filtering for recommendation systems'. Int. Conf. on Robot Intelligence Technology and Applications, Kuala Lumpur, Malaysia, December 2018, pp. 117–131

[32] Miyahara, K., Pazzani, M.J.: 'Collaborative filtering with the simple Bayesian classifier'. PRICAI 2000 Topics in Artificial Intelligence, Australia, 2000 (LNCS, 1886), pp. 679–689

[33] Hofmann, T., Puzicha, J.: 'Latent class models for collaborative filtering'. IJCAI'99: Proc. of the 16th int. joint Conf. on Artificial intelligence, Stockholm, Sweden, 1999, vol. 99, no. 1999

[34] Liu, J., Jiang, Y., Li, Z., *et al.*: 'Domain-sensitive recommendation with user-item subgroup analysis', *IEEE Trans. Knowl. Data Eng.*, 2015, **28**, (4), pp. 939–950

[35] Koren, Y., Bell, R., Volinsky, C.: 'Matrix factorization techniques for recommender systems', *Computer*, 2009, **42**, (8), pp. 30–37

[36] Sarwar, B., Karypis, G., Konstan, J., *et al.*: 'Application of dimensionality reduction in recommender system-a case study', Minnesota Univ. Minneapolis Dept. of Computer Science, 2000

[37] Mnih, A., Salakhutdinov, R.R.: 'Probabilistic matrix factorization'. Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 2008, pp. 1257–1264

[38] Yu, K., Zhu, S., Lafferty, J., *et al.*: 'Fast nonparametric matrix factorization for large-scale collaborative filtering'. Proc. of the 32nd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, Boston, MA, USA, July 2009, pp. 211–218

[39] Salakhutdinov, R., Mnih, A., Hinton, G.: 'Restricted Boltzmann machines for collaborative filtering'. Proc. of the 24th Int. Conf. on Machine Learning, Corvalis, Oregon, USA, June 2007, pp. 791–798

[40] Xue, H.J., Dai, X., Zhang, J., *et al.*: 'Deep matrix factorization models for recommender systems'. Proc. of the Twenty-Sixth Int. Joint Conf. on Artificial Intelligence (IJCAI-17), Melbourne, Australia, 2017, pp. 3203–3209

[41] Zhang, S., Yao, L., Xu, X.: 'AutoSVD++: an efficient hybrid collaborative filtering model via contractive auto-encoders'. Proc. of the 40th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, Shinjuku Tokyo, Japan, August 2017, pp. 957–960

[42] Ouyang, Y., Liu, W., Rong, W., *et al.*: 'Autoencoder-based collaborative filtering'. Int. Conf. on Neural Information Processing, Kuching, Malaysia, November 2014, pp. 284–291

[43] Sedhain, S., Menon, A.K., Sanner, S., *et al.*: 'Autorec: autoencoders meet collaborative filtering'. Proc. 24th Int. Conf. on World Wide Web, Florence, Italy, May 2015, pp. 111–112

[44] Wu, Y., DuBois, C., Zheng, A.X., *et al.*: 'Collaborative denoising auto-encoders for top-n recommender systems'. Proc. of the Ninth ACM Int. Conf. on Web Search and Data Mining, San Francisco, California, USA, February 2016, pp. 153–162

[45] Yan, W., Wang, D., Cao, M., *et al.*: 'Deep auto encoder model with convolutional text networks for video recommendation', *IEEE Access*, 2019, **7**, pp. 40333–40346

[46] Strub, F., Gaudel, R., Mary, J.: 'Hybrid recommender system based on autoencoders'. Proc. of the 1st Workshop on Deep Learning for Recommender Systems, Boston MA, USA, September 2016, pp. 11–16

[47] Alfarhood, M., Cheng, J.: 'DeepHCF: a deep learning based hybrid collaborative filtering approach for recommendation systems'. 17th Int. Conf. on Machine Learning and Applications (ICMLA), Orlando, Florida, December 2018, pp. 89–96

[48] Kim, D., Park, C., Oh, J., *et al.*: 'Convolutional matrix factorization for document context-aware recommendation'. Proc. of the 10th ACM Conf. on Recommender Systems, Boston, Massachusetts, USA, September 2016, pp. 233–240

[49] Kiranyaz, S., Ince, T., Hamila, R., *et al.*: 'Convolutional neural networks for patient-specific ECG classification'. 37th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC), Milan, Italy, August 2015, pp. 2608–2611

[50] Kiranyaz, S., Avci, O., Abdeljaber, O., *et al.*: '1d convolutional neural networks and applications: a survey', *arXiv preprint arXiv:190503554*, 2019

[51] Stuhlsatz, A., Lippel, J., Zielke, T.: 'Feature extraction with deep neural networks by a generalized discriminant analysis', *IEEE Trans. Neural Netw. Learn. Syst.*, 2012, **23**, (4), pp. 596–608

[52] Hinton, G.E., Salakhutdinov, R.R.: 'Reducing the dimensionality of data with neural networks', *Science*, 2006, **313**, (5786), pp. 504–507

[53] Zhang, L., Luo, T., Zhang, F., *et al.*: 'A recommendation model based on deep neural network', *IEEE Access*, 2018, **6**, pp. 9454–9463

[54] Chen, W., Cai, F., Chen, H., *et al.*: 'Joint neural collaborative filtering for recommender systems', *ACM Trans. Inf. Syst.*, 2019, **37**, (4), p. 39

[55] Silveira, T., Zhang, M., Lin, X., *et al.*: 'How good your recommender system is? a survey on evaluations in recommendation', *Int. J. Mach. Learn. Cybern.*, 2019, **10**, (5), pp. 813–831

[56] Kingma, D.P., Ba, J.: 'Adam: a method for stochastic optimization', *arXiv preprint arXiv:14126980*, 2014

[57] Kumar, V., Pujari, A.K., Sahu, S.K., *et al.*: 'Proximal maximum margin matrix factorization for collaborative filtering', *Pattern Recognit. Lett.*, 2017, **86**, pp. 62–67

[58] Hernando, A., Bobadilla, J., Ortega, F.: 'A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model', *Knowl.-Based Syst.*, 2016, **97**, pp. 188–202

[59] Liao, C.L., Lee, S.J.: 'A clustering based approach to improving the efficiency of collaborative filtering recommendation', *Electron. Comm. Res. Applic.*, 2016, **18**, pp. 1–9

[60] Cai, Y., Leung, H.-f., Li, Q., *et al.*: 'Typicality-based collaborative filtering recommendation', *IEEE Trans. Knowl. Data Eng.*, 2013, **26**, (3), pp. 766–779