International Journal of Engineering

Journal Homepage: www.ije.ir

Multi-criteria– Recommendations using Autoencoder and Deep Neural Networks with Weight Optimization using Firefly Algorithm

G. Spoorthy*, S. G. Sanjeevi

Department of CSE, NIT Warangal, Warangal, India

P A P E R I N F O

Paper history:
Received 12 July 2022
Received in revised form 06 October 2022
Accepted 15 October 2022

Keywords:
Multi-Criteria Recommendation Systems
Autoencoder
Firefly Algorithm
Deep Neural Networks
Deep Learning

A B S T R A C T

Demand for personalized recommendation systems elevated recently by e-commerce, news portals etc., to grab the customer interest on the sites. Collaborative filtering proves to be powerful technique but it always suffers from data sparsity, cold-start and robustness issues. These issues have been tackled by some approaches resulting in higher accuracy. Few of them take user profiles, item attributes and rating time as the side information along with ratings to give interpretative personalized recommendations. These type of approaches tries to find which factors mainly impacted the user to rate an item. Another approach extends the single-criteria ratings of collaborative filtering to multi-criteria ratings. Our approach exploits non-linear interpretative recommendations by exploring Multi-criteria ratings by combination of Autoencoders with dropout layer and firefly algorithm optimized weights for deep neural networks. Our approach solves data sparsity, scalability issues and fetch accurate recommendations. Experimental evaluations have been done using Yahoo! Movie and MovieLens datasets. Our approach outperforms in robustness and accuracy with respect to previous research works.

***doi**: 10.5829/ije.2023.36.01a.15*

NOMENCLATURE

RS	Recommendation System	AaRS	Attribute aware Recommendation System
MC	Multi-Criteria	AE	Autoencoder
SC	Single Criteria	DNN	Deep Neural Networks
FA	Firefly Algorithm	U, I, W	Users, Items, Weights
MCAE-FADNN	Multi-Criteria based Recommendations using Autoencoder and Deep Neural Networks with Firefly Algorithms for Weight Optimization		

1. INTRODUCTION

Over the past decade, recommendation systems (RS) have got the popularity around many applications. Applications such as e-commerce, news portals, Instagram actively use the RS tools to drag interest of the and so on customers. RS foresee the unknown user interests by past user preferences, product relevance to recommend products that the user may be interested in near future. The popularity of RS is mainly due to dealing the vast data (number of products in catalogue) and producing highly relevant recommendations which help the customers. The main objective of RS is to fetch interesting items/products to the users/customers from the vast list of products that user maybe interested in near future. It is not surprising RS helps the users to great extent in making decisions and decreases the search time by reducing the search space of a lakh to few tens. RS are also used in software development to learn resources [1]. RS models can be built using content-based filtering (CBF) [2], collaborative filtering (CF) [3-5] and Hybrid (combination) strategies [6]. CF is further divided into user-based CF [3], item-based CF [4] and trust aware CF [7]. The pitfall of traditional RS are data sparsity and cold start issues [8]. Data Sparsity issue is caused due to limited ratings provided by the user and the vast variety of items in catalogue. Cold-start issue is caused due to new users who do not have past history and due to newly uploaded items which does not have a rating yet. To surpass the data sparsity pitfalls, latent vectors are used

*Corresponding Author Institutional Email: gspoorthy6@gmail.com
(G. Spoorthy)

Please cite this article as: G. Spoorthy, S. G. Sanjeevi, Multi-criteria– Recommendations using Autoencoder and Deep Neural Networks with Weight Optimization using Firefly Algorithm, ***International Journal of Engineering, Transactions A: Basics***, Vol. 36, No. 01, (2023), 130-138

which lead to the linear models such as PCA [9], Matrix factorization [10]. They are important techniques used with RS. Later on non-linear models such as AutoRec [11], Variational Autoencoder [12] became prominent. To surpass Cold-start pitfalls attribute-aware RS [13] using the user profiles, product attributes came into existence.

Recent years of research saw major setback with the RS using single rating based observations of user preferences. Single rating cannot be the major part of recommendation as it possesses abstract information regarding the basis by which the rating has been given. Such cases lead to inaccurate recommendations. For example, in movie recommendations the rating can be given by a user may be due to his/her interest in a particular genre, or cast, or age of the user. Predominantly based on distinct product attributes and user profiles a user chooses a product. Extending information of users and items can improve prediction accuracy leading to robust recommendations. These Systems are known as Attribute-aware Recommender systems (AaRS). As for the previous research [14], these AaRS can be classified into four types namely: (i) Discriminate Matrix factorization (ii) Generative Matrix factorization (iii) Factorization Machines and (iv) Heterogeneous graphs. The variation of these categories comes from user, product and attributes interactions. Discriminate matrix factorization models take the attributes as the prior input knowledge to latent representation of users/ items as output. Generative matrix factorization models take attribute distributions and rating distributions to learn. Factorization Machines consider the attributes as user identity representation and build latent representations for predicting ratings. Heterogeneous graphs, as the name suggests represents attributes, users and items as heterogeneous graphs and link prediction is viewed as recommendation. Several topics on AaRS [14] are produced by extending matrix factorization, kernel based models, probabilistic models, and models of deep neural networks. Enhancing AaRS can be done using multi-criteria systems (MCs). These systems try to build the user preferences with respect to two or more criteria ratings to attain robust recommendations. For example, movieLens data [15] have cast attribute ratings from the IMDB *url* attribute by which the recommender system can be trained using actor rating, director rating, etc., These MCs try to include the items quality factor for recommender model. MCs gain popularity because of the quality of recommendations and the robustness of the model. Multi-matrix factorization (MMF) [16] calculates the attribute ratings by computing inner product of user latent vectors and attribute latent vectors using MMF. Later on both user preference ratings and attributes overall performance are integrated to generate recommendations. The performance of the above model depends on the weights chosen for integrating task. MovieANN [17] clusters users and also clusters movies using k-means and x-means respectively. In recommendation phase user and movie clusters are mapped to the target user. A multi layered neural network is used to decide whether to recommend the movie or not. The performance depends on parameters and with linearity of matrix factorization. To surpass this limitation we are using non-linear transformations to enrich beauty of matrix factorization expressiveness. Our model uses a deep neural network and meta-heuristic approach to perform non-linear transformations.

Meta-heuristic algorithms include nature inspired algorithms (such as PSO, FA), evolutionary algorithms (such as GA) tend to get high-level near optimal solutions based on the behaviour of agents (such as particles, firefly and chromosomes). It is said that in literature [18] compared with back propagation (with feed forward network) PSO gives better non-linear function to train neural networks. Meta-heuristic approach gives an optimal solution to many problems and improves the model accuracy. PSO [19], GA [20], FA [21], Projectiles optimization [22], combination of PSO and back-propagation [23] and combination of GA and PSO [24] have been applied in literature to train neural networks.

The deep learning (DL) plays a prominent role in the emerging research domains. DL extracts features that gives more meaning to the data. Neural networks [25] with the non-linear transformations efficiently find the non-linear interactions between users and items. For MCs deep neural networks gives higher quality recommendations. Our models primarily address the following: non-linear transformations for user-item interactions to overcome scalability issue. Quality and robustness of the recommendations are attained by shifting single criteria ratings to MCs. To attain the optimization of weights in the DNN we use Firefly Algorithm. The paper flow goes as follows: Preliminaries are discussed in the section 2. Proposed methodology MCAE-FADNN is described in section 3. Experimental Evaluation is shown in section 4 and finally conclusions described in section 5.

2. PRELIMINARIES

Fundamental concepts of MCs, DNN and Firefly algorithm and its importance in dealing with the issues caused by data sparsity, robustness and quality of the RS are discussed below:

2. 1. Problem Definition Collaborative filtering-based RS attained popularity by computing nearest neighbours of the users who share similar preferences based on the "ratings" given by users for overall product (based on single rating). Technically written as function

of U × I → R where 'U','I', 'R' are set of users, items and ratings respectively. From this data we find patterns that represent the preferences of the users. Single criteria (SC) frameworks try to approximate the utility function.

$$\arg\max_{i \in items} f(u,i) \forall u \in U, \forall i \in I \quad (1)$$

The model selects 'k' items that maximizes Equation (1). But internally, entire model depends on a sparse matrix for the recommendations. For example, in movieLens dataset the utility function is based on the overall rating given by user on a movie i.e., SC. Such ratings may not reflect the user opinion on the movie: like user may like the cast but not the movie (or) user may have liked the movie but not the cast (or) the movie may not be having good visuals etc. To increase the quality of the recommendations and to maximize utility function MC based approaches are being currently used.

2. 2. Multi-criteria based Recommendations MC based Recommendations surpass the limitations of SC by extending utility function from over all item rating score to including all the criteria ratings which effect the opinion of user. Technically written as:

$$f(u,i) = r_1 \times r_2 \times \ldots \times r_c \quad (2)$$

where $r_1, r_2, .., r_c$ are the ratings of the item from 1 to c w.r.t 'c' criteria.

Users rate the items on different criteria. For instance, movie RS can extend their preferences based on four different criteria namely genres, actors, directors and plot. A user may like the genre and plot but strongly dislikes actors and director has rated the overall rating as 4. The ratings for the users preferences according to the above four criteria are (5, 2, 1, 5). That user preference ratings may be partly or entirely different from other users. Two of the other users might have given the overall rating as 4 but their user preferences for the above four criteria could be for example (4, 3, 1, 5) and (2, 1, 2, 5) respectively. The 3 users maybe classified as similar by SC since it considers only overall ratings. Hence, MCs are preferred over SCs. Table 1 summarized user-item multi-criteria matrix. With the extension of SCs to MCs we need to change our models which can adopt all the rating criteria into the model. Model based approaches often learn from the training model and build the prediction model to predict the unknown ratings.

Machine learning models like probabilistic models, SVR, kernel backdrop models are shown to have good prediction accuracy. All these model based techniques have similar aggregate function: the conjunction of all the MC ratings to get overall rating prediction.

2. 3. Autoencoders The objective of Auto-encoder is to attain a 'd' dimensional representation (where d<<m) of a matrix n × m ∀ min (Error(x, decode(encode(x)))). In this paper, we learn non-linear latent vectors for users criteria using auto-encoders. For each auto-encoder we give ratings and criteria matrix. The non-linear latent vectors formed from criteria matrix are termed as criteria latent vectors or attribute latent vectors as termed in literature [16]. While performing the task we place the dropout layers (removing the connections temporarily) so that the preference of each criteria gets more elevated. The mathematical formulae for this phase is given below:

Encoder: It encodes high dimensional matrix $X = \{x_1, x_2, \ldots, x_m\}$ to fewer dimensional matrix called hidden representation $h = \{h_1, h_2, \ldots, h_d\}$ by a function 'f(x)' having activation function 'a_e' at encoder

$$h = a_e(W_x + b) \quad (3)$$

In-order to penalize least important data and avoid getting into over-fitting issues we use the dropout layer. The dropout layer temporarily drops the penalized neuron links so that activation function can't be applied on them.

Decoder: It decodes the hidden representation $h = \{h_1, h_2, \ldots, h_k\}$ back to reconstructed matrix of x: $x' = \{x'_1, x'_2, \ldots, x'_n\}$ by a function g(h) having activation function a_d at decoder

$$x' = a_d(W'_h + b') \quad (4)$$

After reconstructing the network error is calculated as follows:

$$\text{Error}(E) = \sum_{x_o \in X} \|x_o - g(f(x_0))\|_2^2 + \lambda(\|W_1\|_2^2 + \|W_1\|_2^2) \quad (5)$$

TABLE 1. Representation of MCs for user(U), Item(I)

	i_1	i_2	i_3	i_4	i_5
u_1	$4_{(5,2,1,5)}$	$3_{(4,1,2,2)}$			$1_{(2,1,2,2)}$
u_2			$5_{(5,NA,4,4)}$		$3_{(2,3,2,3)}$
u_3	$4_{(4,3,1,5)}$	$5_{(3,5,4,3)}$			$2_{(4,2,1,1)}$
u_4	$1_{(1,2,1,1)}$		$3_{(1,4,1,3)}$	$5_{(5,5,NA,2)}$	
u_5	$4_{(2,1,2,NA)}$	$2_{(1,3,3,1)}$			

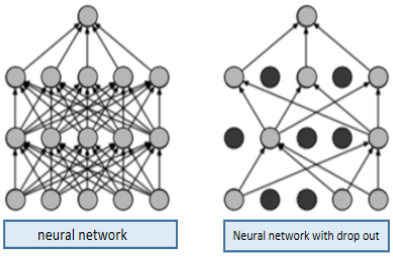

Figure 1. Autoencoder and Autoencoder with dropout layer

2. 4. Deep Neural Networks for multi criteria based recommendation Deep learning (DL) possess a vast range of applications namely image processing, natural language processing and computer vision domains. In Recommendation systems data sparsity issues are cleared using DL models. Along with data sparsity issues DL methods try to capture non-linear interactions between users and items. Neural networks surpass limitations of matrix factorization and enhance the approximations. With the explicit SC feed back, AutoRec [11] attains non-linear user item interactions using auto encoder architecture. Auto encoder learns to reprint the input to output to give low-dimensional representations. AutoRec focuses on reconstructing the output layer in such a way that the network fills the missing entries giving more scope to the increase in prediction accuracy. Deep factorization machines [26] learn pairwise-linear interactions between users and items by using multilayer perceptron and deep network models gives high end non-linear interactions. MovieANN [17] combines content based filtering and collaborative filtering models by mapping user and items clusters formed in the initial phase and fed into multilayer perceptron to attain recommendations. DNN based recommendations using MC ratings with stacked encoders are done [27]. In this approach, link between the overall ratings and individual criteria are attained by rigorous adjustment of loss function by changing the weights of hidden layers and the output layer. Later on in DHARS [26] combination of neural collaborative filtering and stacked denising auto encoder enhance the RS accuracy. Comparison of SC based RS and MC based RS is done with an artificial neural network framework [25] proves the MC Based RS outperforms SC based RS.

2. 5. Firefly Algorithm Firefly algorithm (FA) proposed by Yang [28], is a meta-heuristic which is nonlinear and stochastic in nature. The entire FA depends on two key points: light intensity and degree of attractiveness between fireflies.

$$I_i = I_0 e^{-\gamma r_{ij}} \quad (6)$$

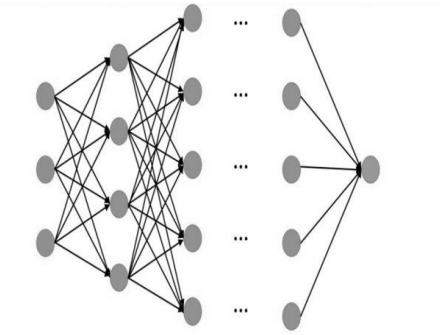

Figure 2. Deep Neural Network

where I (i) is the light intensity at 'i', I (0) is the light intensity at r (ij)=0.

$$\beta_{ij} = \beta_0 e^{-\gamma r_{ij}} \quad (7)$$

where β (ij) is the attraction between 'i' and 'j', β (0) is the initial attraction at r (ij)=0.

$$x_i = x_i + \beta_{ij}(x_j - x_i) + \alpha\varepsilon_{ij} \quad (8)$$

A meta-heuristic algorithm should deal two components namely exploration and exploitation. The exploration component is also known as intensification. This aspect is achieved by the randomness of the FA. Fine tuning of the randomness will make the FA best with respect to local and global search. The exploitation component is also known as diversification. This aspect is achieved by the knowing the local information. The exploitation increases the convergence speed whereas exploration decreases the convergence speed.

3. MCAE-FADNN

This section deals with the proposed algorithm: Multi criteria based recommendations using Autoencoder and deep neural network with weight initialization by Firefly algorithm (MCAE-FADNN). Our model is divided into 3 phases where in the phase 1 deals with predicting missing criteria ratings using AutoEncoder (AE), phase 2 deals with predicting overall ratings for the missing criteria in the training set and phase 3 predicts the recommendations for test set and evaluates the performance of the method.

3. 1. Firefly Algorithm for Weight Optimization

Step 1: The initial population and initialize cluster centers are randomly generated.

Step 2: Repeat 3 to 16 from pseudo code.

Step 3: Pick smallest distance of weights from center as weights of MCAE-FADNN.

Firefly algorithm (FA) for weight optimization is given below:

TABLE 2. FA for weight optimization

Initialization:

Maximum iteration T=200, t=1

N_{pop}, m, n are number of population, dimensions and clusters respectively.

$\alpha = 0.5$

The initial population are randomly generated.

$$W = \begin{bmatrix} W_1 \\ \vdots \\ W_{N_{pop}} \end{bmatrix}, \text{ where}$$

$$W_i = [w_1, w_2, \cdots, w_n], \forall i = 1,2,\ldots, N_{pop},$$

$$C_j = [c_1, c_2, \ldots, c_m] \forall j = 1,2,\ldots, n$$

And $w_1 = w_l + \text{rand}(0) \times (w_u - w_l)$

W_i is the solution, C_j is the j^{th} cluster of i^{th} solution, w_i is the position of *w* at *i*.

Objective Function

$$fitness: d_{W_i,c_j} = \sum_{k=1} (w_{ik} - c_{jk})^2$$

1. Initialize each firefly as a cluster

$$C = \{c_i\} \forall i = 1,2,...,m$$

2. Calculate distance between clusters
3. Randomly select k fireflies
4. Find the initial light intensities using Equation (6)
5. Light intensity at I_i at w_i is determined by f(x)
6. While(t<T)
7. For i=1 to m
8. For j=1to m
9. $if\ (I_i < I_j)$
10. Move firefly i to j using Equation (7)
11. End if
12. End For
13. End For
14. Update the light intensity f(w) using Equation (8)
15. t=t+1
16. End While
17. Pick smallest distance of weights from center as weights of MCAE-FADNN

$$\sum_{w_i \in W} \min_{i,j} (dist(W_i, c_j))$$

3. 2. Multi-criteria Ratings The related work shows the importance of multi- criteria ratings. This section deals with how the criteria ratings are exploited to build fine connections between user preference w.r.t all the criteria. To attain the MC ratings we will append user (U), item (I) and rating (R) matrix to U, I, R, C1,C2,…Ck, where C1,C2,…Ck are 'k' different criteria. We explore user interest as shown in Table 1 and consider the matrix is represented as $R_{k\times n\times m}$. The entire model works in three phases. The three phases are described below:

Phase 1: It starts with taking every criteria as input. In order to increase the credibility of the system and reduce the effect of missing values they are replaced by '0'. Then we build the fewer representation of the training data using autoencoder (AE). Dropout layer is appended to AE to get a generalized model which does not suffer from over-fitting. The autoencoder in the decoder layer reconstructs missing values on which we predict the criteria based rankings in training phase.

Phase 2: Once the criteria based ranking values of missing values are calculated we normalize them. For deep neural network weights are learnt using firefly algorithm. The above step is to reduce the effect of random weights on the system. Firefly algorithm enables optimized weights of the DNN. Using optimal weights and the normalized rating values as input DNN to predicts the overall ratings for missing values.

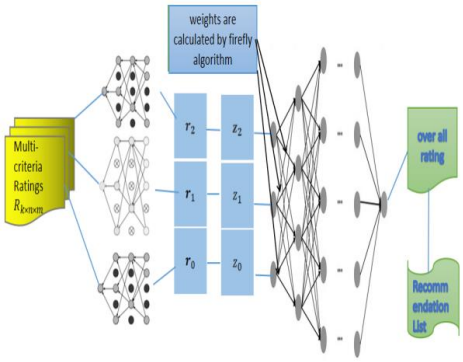

Figure 3. MCAE-FADNN

Phase 3: In this phase, the prediction task is done for the test set without using dropout layer. Once the overall ratings are calculated the recommendations are made.

TABLE 3. MCAE-FADNN (pseudo code)

Input: $R_{k\times n\times m}$
***Parameter initialization*:**
Number of epochs:200
Number of hidden layers: 'h'
Number of neurons in each hidden layer: 'l'
Learning rate:0.001
Dropout:0.1
Loss Function : MSE
Output: $Pred_{n\times m}$
Phase1: Autoencoder AE
1: Decompose $R_{k\times n\times m}$ into 'k' $R_{n\times m}$ matrices
2: For Each 'k' $R_{n\times m}$ criteria based matrix C do
3: Update the C matrix with 0 for all missing values (ex: [7 NAN 4 2 1] -> [7 0 4 2 1])
4: Split dataset to create training and test datasets (80% and 20%)
// constructing fewer representation by building an autoencoder AE
5: For each Epoch do
6: For each user U in training set R is given as input to AE
7: Encode the preferences of users in C by Equation (3). Dropout the neurons with least importance.
8: Decode the resultant in 7 by Equation (4)
9: Find reconstruction error by Equation (5)
10: Update weights and biases of AE
11: End For
12: End For
13: Now the trained network of AE is AE'
14: Predict the criteria based ranking for missing values in C using AE' ($\{r_0, r_1, \ldots, r_k\}$)
15: End For
Phase 2: Train a DNN to predict overall rating
16: for each user in U do
17: Normalize the ratings input obtained in 15 using mean (μ) and standard deviation(σ) in the below formula

$$z_i = \frac{r_i - \mu_i}{\sigma_i}$$

18: The output of normalization of input vector is
$Z = [z_0, z_1, \ldots, z_k]^T$
// Dense ReLU is used as activation function for Z
19: Calculate initial weights using firefly Algorithm
20: For each user u in U do
21: For each z in Z do
22: Compute ReLU(z)=max(z,0)
23: For each hidden 'l' layers in L do
24: $h_l = ReLU(W_l h_{l-1} + b_l)$
25: End For
26: Predict overall rating using:
$r_{ui} = ReLU(W_L h_{L-1} + b_L)$
Phase 3: Recommendation phase
28: for each user u in U and item i in I in test set
29: Calculate criteria ratings $[r_0, r_1, \ldots, r_k]^T$
30: Normalize $[r_0, r_1, \ldots, r_k]^T$ to $[z_0, z_1, \ldots, z_k]^T$
31: Compute overall rating r_{ui}
32: Recommend items using the ratings r_{ui}
33:End For
34: Analyse the Recommendations

4. EXPERIMENTAL EVALUATION

We compare our model with AEMC [29], MovieANN [17], Multi-criteria recommendations using stacked encoder [27]. These models are denoted by their respective reference numbers [17, 27, 29] in the graphs shown below in section 4.2.

4. 1. Dataset Description In this paper, for evaluating our model we have used 2 datasets namely: Yahoo!Movie(YM), MovieLens 1M. Yahoo! Movie MC Dataset contains 1716 users, 965 movies and 34800 MC ratings with 4 criteria. A movie is rated by the user in four categories: Actors, Directors, story and visuals. Along with the four criteria ratings, user gives the overall rating for the movie. Overall we have 34800 ratings. As a pre-processing step we transformed the ratings of each criteria from A^+,A,A^+,..., C^-ranging from 1 to 5. For MovieLens 1M dataset, multi-criteria ratings are extracted from IMDB and they are mapped to our MovieLens dataset. A movie is rated by the user in four categories: Actors, Directors, genre and plot. LDA [21] is used if any criteria vaguely structured to make it fit for our model. The parameters shown at the beginning of the Table 3 are used with the MCAE-FADNN algorithm. For criteria ratings, we have set Adam optimizer with learning rate 0.001 and dropout as 0.001. The dropout can happen at any stage of the autoencoder layers. Weights of the DNN are learned by using firefly algorithm.

4. 2. Classification of Research Issues Many recommendation models focus on certain research issues. The research issues maybe scalability, sparsity, cold-start and accuracy. One can consider only one aspect or all aspects. In our model, we focused on dealing sparsity, scalability and accuracy aspects.

Autoencoder with dropout layer is used due to the larger weights of the neural network becoming more complex, making the model vulnerable to over fitting. Dropout layers with autoencoders temporarily drop certain amount of nodes (dropped nodes are temporarily out of reach). This in turn helps the model compute fast and it serves as a regularization model. Autoencoders with drop out layer makes our model deal with the sparse data. If single criteria ratings suffer from sparsity issues there is a possibility for multi-criteria ratings to also suffer from sparsity issues. This part of our model is specifically included for dealing with sparsity issues.

Another issue we handled is scalability issue. Recommendation systems require large amount of data to train the model. Usually clustering methods are used to solve the scalability issues. Recently for deep neural networks the scalability is the prominent issue to deal with when the accuracy of the recommendation model is at stake. To deal with this issue we have used the normalization technique. The normalization is done for each of the criteria ratings given by the user using step 17 in the pseudo code.

The main objective of the multi-criteria based recommendations is to recommend the more relevant recommendations which are based on the overall ratings given by the user. The overall ratings depend on multiple attributes/criteria of an item. The overall ratings are calculated with respect to the preferences of the user. The preferences of different users may be different. Based on the user preferences the weights for the attributes should vary. To capture the user preference we change weights to optimize objective function d_{W_i,c_j} shown in Firefly algorithm pseudo code. The weight optimization searches optimal weights which reflect the user preferences. This leads to more prominent and effective solution of recommendations as every aspect of user behaviour have been captured. In the previous research works [30, 31] have used Genetic algorithm (GA) and Particle swarm optimization (PSO) for multi criteria based recommendations. The above models failed to address sparsity and scalability issues. We have addressed them in our model. We have used firefly algorithm for weight optimization. The reason for us to use firefly algorithm is computational cost for firefly algorithm is much less compared to GA and PSO. This has been proved by Yang et al. [32]. He et al. [33] proved that FA is the efficient algorithm as FA deals with both exploration and exploitation components of meta heuristic approaches. PSO doesn't have randomization element making it vulnerable in exploration aspect. The average time complexity of GA, PSO and FA are $O(n^{3/2} \log n)$, O(ntm) and O(n log n) respectively, where 'n' is population size, 't' is number of iterations and 'm' is complexity of cost function. The above factors make our model more efficient and accurate.

4. 3. Evaluation We compare M.A.E, RMSE, Precision, recall and F2 obtained using MCAE-FADNN with related previous research works AEMC [29], MovieANN [17] and stacked AE [27]. The formulae for calculating M.A.E, RMSE, Precision and recall are shown below.

$$MAE = \frac{\sum_{u,i}^{N} |r(u,i) - P(u,i)|}{N} \quad (9)$$

where r(u ,i) and P(u ,i) are actual rating and predicted ratings and 'N' is number of items.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{u,i}^{N}(P(u,i) - r(u,i))^2} \quad (10)$$

where r(u ,i) and P(u ,i) are actual rating and predicted ratings and 'N' is number of items.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (11)$$

where TP is True Positives, FP is the False Positives

$$\text{Recall} = \frac{TP}{TP+FN} \quad (12)$$

where TP is True Positives, FN is False Negatives

$$\text{MAP} = \frac{\sum_{n=1}^{K} \text{Precision(n)} \times \text{Relevant(n)}}{\text{MIN}\{K,|\{\text{Relevant Movies}\}|\}} \quad (13)$$

For 'K' recommendations, Precision(n) is precision and Relevant(n) returns 1 if the item is relevant else 0.

$$F2 = \frac{5 \times \text{Precision} \times \text{Recall}}{4 \times \text{Precision} + \text{Recall}} \quad (14)$$

MCAE-FADNN shows that by using AE with dropout layer makes our model deal successfully with over-fitting problem. MCAE-FADNN makes the DNN learn optimized weights. MCAE-FADNN is similar to the approach of AEMC [29] except that it differs by using DNN for aggregation. MCAE-FADNN is similar to MovieANN [17] and stacked AE [27] except that MCAE-FADNN uses weight optimization by Firefly Algorithm (FA) where as MovieANN [17] and stacked AE [27] perform weight optimization using back-propagation learning algorithm. Performance of our model is shown in comparison to [17, 27, 29] using MAE, RMSE, Precision, Recall, MAP and F2.

We varied number of epochs to find how the model is working (Table 4).

TABLE 4. MAE and RMSE variations w.r.t epochs

	#Epochs	MAE	RMSE
Yahoo! Movie	50	0.6635	0.7757
Yahoo! Movie	100	0.6303	0.7324
Yahoo! Movie	200	0.6261	0.7094
MovieLense	50	0.6935	0.7757
MovieLense	100	0.6603	0.7424
MovieLense	200	0.6461	0.7294

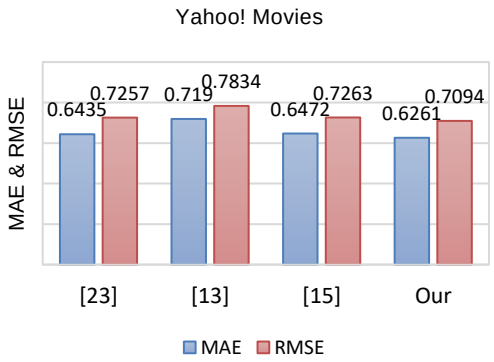

Figure 3. Comparison of performance of MCAE-FADNN w.r.t. related research works [17, 27, 29] for Yahoo! Movie dataset

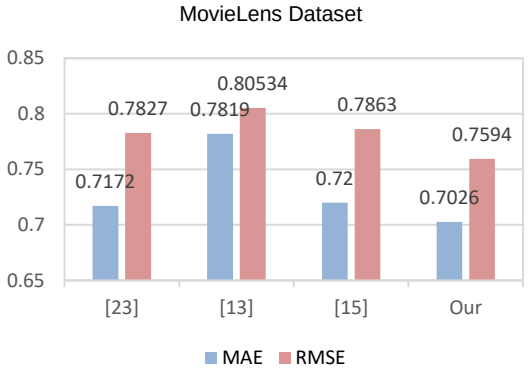

Figure 4. Comparison of performance of MCAE-FADNN w.r.t. previous research works [17, 27, 29] for MovieLens dataset

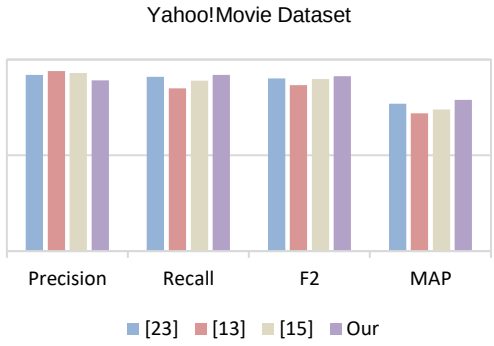

Figure 5. MCAE-FADNN performance w.r.t. Precision, Recall, F2 and MAP compared with previous research works [17, 27, 29] for Yahoo! Movies dataset

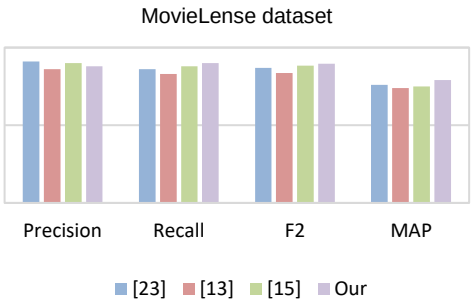

Figure 6. MCAE-FADNN performance w.r.t. Precision, Recall, F2 and MAP compared to compared with previous research works [17, 27, 29] for MovieLens dataset

5. CONCLUSIONS

Focus in the research of multi-criteria based recommendation systems revolves around the accuracy of the recommendations. In this paper, we focus on dealing with data sparsity, scalability and accuracy issues for recommendation systems. We propose MCAE-FADNN which works in three phases: (i) predicts criteria wise ratings using Autoencoder with dropout layer, (ii) builds non-linear interaction between users and items using DNN with optimized weights attained using firefly algorithm. The phase 1 deals with data sparsity issues. In phase 2 we normalized the predicted ratings in phase 1 to deal with the scalability issues. Along with that in phase 2 using weight optimization technique we increased our model accuracy. These models are multi-criteria based recommendation systems using GA and PSO as prime concepts which dealt with accuracy aspect. Our model outperforms with respect to accuracy, efficiency and computational cost by choosing Firefly technique. Finally, we compared MCAE-FADNN with AEMC, Stacked AE and MovieANN with respect to measures of MAE, RMSE, Precision, Recall, F2 and MAP and showed that using MCAE-FADNN gave better results compared to previous research works AEMC, Stacked AE and MovieANN. We would like to suggest improving the accuracy of the algorithm by dealing the cold- start issues for future scope.

6. REFERENCES

1. Tayefeh Mahmoudi, M., Badie, K., Moosaee, M. and Souri, A., "A compositional adaptation-based approach for recommending learning resources in software development", ***International Journal of Engineering, Transactions A: Basics***, Vol. 35, No. 7, (2022), 1317-1329. https://doi.org/10.5829/ije.2022.35.07a.11
2. Lops, P., Gemmis, M.d. and Semeraro, G., "Content-based recommender systems: State of the art and trends", ***Recommender Systems Handbook***, (2011), 73-105. https://doi.org/10.1007/978-0-387-85820-3_3
3. Hofmann, T. and Puzicha, J., "Latent class models for collaborative filtering", in IJCAI. Vol. 99, (1999).
4. Linden, G., Smith, B. and York, J., "Amazon. Com recommendations: Item-to-item collaborative filtering", ***IEEE Internet Computing***, Vol. 7, No. 1, (2003), 76-80. https://doi.org/10.1109/ MIC. 2003.1167344
5. Ferreira, D., Silva, S., Abelha, A. and Machado, J., "Recommendation system using autoencoders", ***Applied Sciences***, Vol. 10, No. 16, (2020), 5510. https://doi.org/10.3390/app10165510
6. Konstan, J.A., Riedl, J., Borchers, A. and Herlocker, J.L., "Recommender systems: A grouplens perspective", in Recommender Systems: Papers from the 1998 Workshop (AAAI Technical Report WS-98-08), AAAI Press Menlo Park., (1998), 60-64.
7. Barzegar Nozari, R., Koohi, H. and Mahmodi, E., "A novel trust computation method based on user ratings to improve the recommendation", ***International Journal of Engineering, Transactions C: Aspects***, Vol. 33, No. 3, (2020), 377-386. https://doi.org/10.5829/ije.2020.33.03c.02
8. Adomavicius, G. and Tuzhilin, A., "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions", ***IEEE Transactions on Knowledge and Data Engineering***, Vol. 17, No. 6, (2005), 734-749. https://doi.org/10.1109/TKDE.2005.99
9. Vozalis, M.G. and Margaritis, K.G., "A recommender system using principal component analysis", in Published in 11th panhellenic conference in informatics, Citeseer. (2007), 271-283.
10. Koren, Y., Bell, R. and Volinsky, C., "Matrix factorization techniques for recommender systems", ***Computer***, Vol. 42, No. 8, (2009), 30-37. https://doi.org/10.1109/MC.2009.263
11. Sedhain, S., Menon, A.K., Sanner, S. and Xie, L., "Autorec: Autoencoders meet collaborative filtering", in Proceedings of the 24th international conference on World Wide Web. (2015), 111-112.
12. Liang, D., Krishnan, R.G., Hoffman, M.D. and Jebara, T., "Variational autoencoders for collaborative filtering", in Proceedings of the 2018 world wide web conference., (2018), 689-698.
13. Vlachos, M., Vassiliadis, V.G., Heckel, R. and Labbi, A., "Toward interpretable predictive models in b2b recommender systems", ***IBM Journal of Research and Development***, Vol. 60, No. 5/6, (2016), https://doi.org/10.1147/JRD.2016.2602097
14. Chen, W.-H., Hsu, C.-C., Lai, Y.-A., Liu, V., Yeh, M.-Y. and Lin, S.-D., "Attribute-aware recommender system based on collaborative filtering: Survey and classification", ***Frontiers in big Data***, Vol. 2, (2020), 49. https://doi.org/10.3389/fdata.2019.00049
15. Harper, F.M. and Konstan, J.A., "The movielens datasets: History and context", ***Acm Transactions on Interactive Intelligent Systems (tiis)***, Vol. 5, No. 4, (2015), 1-19. https://doi.org/10.1145/2827872
16. Su, Y., Erfani, S.M. and Zhang, R., "Mmf: Attribute interpretable collaborative filtering", in 2019 International Joint Conference on Neural Networks (IJCNN), IEEE., (2019), 1-8.
17. Yücebaş, S.C., "Movieann: A hybrid approach to movie recommender systems using multi layer artificial neural networks", Çanakkale Onsekiz Mart Üniversitesi Fen Bilimleri Enstitüsü Dergisi, Vol. 5, No. 2, (2019), 214-232. https://doi.org/10.1007/s00521-022-07012-y
18. Tallapally, D., Sreepada, R.S., Patra, B.K. and Babu, K.S., "User preference learning in multi-criteria recommendations using stacked auto encoders", in Proceedings of the 12th ACM conference on recommender systems., (2018), 475-479.

19. Ujjin, S. and Bentley, P.J., "Particle swarm optimization recommender system", in Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706), IEEE. (2003), 124-131.
20. Kim, K.-j. and Ahn, H., "A recommender system using ga k-means clustering in an online shopping market", ***Expert Systems with Applications***, Vol. 34, No. 2, (2008), 1200-1209. https://doi.org/10.1016/j.eswa.2006.12.025
21. Spoorthy, G., "Hybrid cluster based collaborative filtering using firefly and agglomerative hierarchical clustering", ***International Journal of Computer and Information Technology (2279-0764)***, Vol. 10, No. 6, (2021). https://doi.org/10.24203/ijcit.v10i6.170
22. Kahrizi, M. and Kabudian, S., "Projectiles optimization: A novel metaheuristic algorithm for global optimization", ***International Journal of Engineering, Transactions A: Basics***, Vol. 33, No. 10, (2020), 1924-1938. https://doi.org/10.5829/ije.2020.33.10a.11
23. Zhang, J.-R., Zhang, J., Lok, T.-M. and Lyu, M.R., "A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training", ***Applied Mathematics and Computation***, Vol. 185, No. 2, (2007), 1026-1037. https://doi.org/10.1016/j.amc.2006.07.025
24. Juang, C.-F., "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design", ***IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)***, Vol. 34, No. 2, (2004), 997-1006. https://doi.org/10.1109/TSMCB.2003.818557
25. Hassan, M. and Hamada, M., "A neural networks approach for improving the accuracy of multi-criteria recommender systems", ***Applied Sciences***, Vol. 7, No. 9, (2017), 868. https://doi.org/10.3390/APP7090868
26. Kiran, R., Kumar, P. and Bhasker, B., "Dnnrec: A novel deep learning based hybrid recommender system", ***Expert Systems with Applications***, Vol. 144, (2020), 113054. https://doi.org/10.1016/j.eswa.2019.113054
27. Gudise, V.G. and Venayagamoorthy, G.K., "Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks", in Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706), IEEE., (2003), 110-117.
28. Yang, X.-S., "Firefly algorithms for multimodal optimization", in International symposium on stochastic algorithms, Springer., (2009), 169-178.
29. Shambour, Q., "A deep learning based algorithm for multi-criteria recommender systems", ***Knowledge-Based Systems***, Vol. 211, (2021), 106545. https://doi.org/10.1016/j.knosys.2020.106545
30. Gupta, S. and Kant, V., "An aggregation approach to multi-criteria recommender system using genetic programming", ***Evolving Systems***, Vol. 11, No. 1, (2020), 29-44. https://doi.org/10.1007/s12530-019-09296-3
31. Choudhary, P., Kant, V. and Dwivedi, P., "A particle swarm optimization approach to multi criteria recommender system utilizing effective similarity measures", in Proceedings of the 9th International Conference on Machine Learning and Computing. (2017), 81-85.
32. Yang, X.-S. and He, X., "Firefly algorithm: Recent advances and applications", arXiv preprint arXiv:1308.3898, (2013). https://doi.org/10.1504/IJSI.2013.055801
33. He, X., Liao, L., Zhang, H., Nie, L., Hu, X. and Chua, T.-S., "Neural collaborative filtering", in Proceedings of the 26th international conference on world wide web. (2017), 173-182.

Persian Abstract

چکیده

تقاضا برای سیستم های توصیه شخصی اخیراً توسط تجارت الکترونیک، پورتال های خبری و غیره افزایش یافته است تا علاقه مشتری را در سایت ها جلب کند. ثابت شده است که فیلتر مشارکتی یک تکنیک قدرتمند است، اما همیشه از مشکلات پراکندگی داده، شروع سرد و استحکام رنج می برد. این مسائل با برخی رویکردها حل شده است که منجر به دقت بالاتر می شود. تعداد کمی از آنها نمایه های کاربر، ویژگی های آیتم ها و زمان رتبه بندی را به عنوان اطلاعات جانبی همراه با رتبه بندی ها برای ارائه توصیه های شخصی تفسیری می گیرند. این نوع رویکردها تلاش می‌کنند تا مشخص کنند چه عواملی عمدتاً بر کاربر برای رتبه‌بندی یک آیتم تأثیر گذاشته است. رویکرد دیگر، رتبه‌بندی‌های تک معیاری فیلترینگ مشارکتی را به رتبه‌بندی‌های چند معیاره گسترش می‌دهد. رویکرد ما از توصیه‌های تفسیری غیرخطی با بررسی رتبه‌بندی‌های چند معیاره با ترکیبی از رمزگذارهای خودکار با وزن‌های بهینه‌شده لایه حذفی و الگوریتم کرم شب تاب برای شبکه‌های عصبی عمیق استفاده می‌کند. رویکرد ما پراکندگی داده ها، مشکلات مقیاس پذیری و دریافت توصیه های دقیق را حل می کند. ارزیابی های تجربی با استفاده از Yahoo! مجموعه داده های Movie و MovieLens. رویکرد ما در استحکام و دقت نسبت به کارهای تحقیقاتی قبلی بهتر عمل می کند.