EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Design and Implementation of a Collaborative Filtering Algorithm Based on Deep Learning and Matrix Factorization

Meiyu Fang[1,*,†], Zhe Zhu[2,†] , Leyi Cai[1] and Zelin Chen[3]

[1] School of International Business, Zhejiang International Studies University, Hangzhou 310023, China
[2] School of Economics, Zhejiang University of Technology, Hangzhou 310023, China
[3] Zhejiang Aerospace Hengjia Data Technology Co., Ltd,Jiaxing, Zhejiang, China

Abstract

INTRODUCTION: Collaborative filtering (CF) algorithms based on deep learning and matrix factorization aim to learn deeper latent features of users and items from user ratings. However, such methods often face challenges due to the sparsity of rating data, which limits their recommendation performance.
OBJECTIVES: This study aims to design a hybrid recommendation algorithm named auto-encoder deep learning and matrix factorization (AED-MF), which integrates deep learning and matrix factorization to address the sparsity issue in rating data and improve the extraction of deeper latent features of users and items.
METHODS: The AED-MF algorithm combines an auto-encoder-based deep learning approach with matrix factorization to model both deeper hidden representations and nonlinear relationships between users and items. The methodology includes data downloading, mounting, cleaning, model training, and experimental evaluation, with matrix factorization applied to predict missing ratings in the rating matrix.
RESULTS: The proposed AED-MF algorithm demonstrated strong performance in key recommendation metrics, effectively learning deeper user and item representations and applying them to capture complex nonlinear relationships. It also reduced the impact of rating data sparsity while maintaining high recommendation accuracy.
CONCLUSION: The AED-MF recommendation algorithm successfully alleviates the problem of sparse rating data and enhances recommendation accuracy by leveraging the strengths of both deep learning and matrix factorization, offering an effective solution for improving collaborative filtering-based recommender systems.

Keywords: Collaborative filtering; Deep learning; Matrix factorization; Recommendation algorithm; Data sparsity; Recall

Received on 15 November 2025, accepted on 13 April 2026, published on 20 April 2026

doi: 10.4108/eetsis.10935

1. Introduction

Rapid integration of internet technologies with traditional industries and emerging economies has spurred diverse business models and at the same time, overwhelming user choices necessitate efficient recommender systems [1]. These systems explore user behaviors and item attributes for identifying preferences to streamline decision-making [2]. Available research reports have mainly focused on enhancing recommendation accuracy, efficiency, and data utilization across algorithms.

Current recommender systems have originated from "The Daily Me" concept introduced by Nicolay Negroponte, which envisioned personalized information filtering [3]. Modern analyses have confirmed personalized information filtering's foundational role in

*Correspondence: myfang@zisu.edu.cn
†These authors contributed equally to this work

shaping algorithmic paradigms [4]. Current algorithms could be classified into three categories of content-based, collaborative filtering (CF), and association rule-based approaches. To address their limitation in obtaining dynamic user preferences, a session-based XLNet transformer approach was introduced in this research which achieved up to 136.23% improvement on multi-domain Amazon data [5]. Recent hybrid methods have further demonstrated that data sparsity distortions were significantly decreased by combining ratings with review semantics [6].

Content-based recommendations rely on user profiles and item features to suggest similar items, generally applying information retrieval techniques [7]. In addition to being intuitive and user-independent, they struggle with the accuracy of feature extraction due to semantic gaps in multimodal data and cannot uncover latent user interests without graph-based relational modeling [8-10].

CF, introduced by Goldberg, leverages collective user behavior and can be classified into memory-based (user/item neighborhood models) and model-based methods [11]. Neighborhood-based CF predicts preferences based on similarity metrics on user-item matrices [12], while model-based methods use techniques such as matrix factorization to address sparsity [13-16]. Hybrid models integrating social networks and topic modeling further improve performance [17,18].

Association rule-based systems mine historical purchase patterns (e.g., rule X→Y) for recommendations and are extensively applied in retail but are prone to popularity bias and cold-start issues.

Each method has trade-offs (Table 1): Content-based recommendation algorithms are simple in principle, easy to use, and independent among users. However, they are restricted by content modeling integrity, cannot extract the potential interests of users, and suffer from low recommendation accuracy. CF-based recommendation algorithms only need the association matrices of users and items, have simple structures and do not need prior knowledge on the recommended content. However, these algorithms have cold start problem and are easily affected in the case of sparse data. Association rule-based recommendation has an easy principle; however, popular items can easily be over-recommended and because of the application of user data, there are inevitable cold start and sparsity problems. To address these problems, we proposed AED-MF algorithm by combining autoencoder-based deep learning with matrix factorization to learn latent features from sparse ratings. Its effectiveness in key recommendation metrics was verified by experiments.

Table 1. Advantages and disadvantages of various recommendation algorithms

Recommendation algorithm	Advantages	Disadvantages
Content-based recommendation algorithm	1. Simple principle, easy to implement. 2. High interpretability. 3. Independent users. 4. No cold start issues in items.	1. Difficult item feature extraction [19]. 2. Inability to mine potential interest of users and low recommendation accuracy. 3. Inability to generate recommendations for new users.
CF Recommendation	1. Based on user behavior, no prior knowledge of the recommended content is required. 2. Only needs the correlation matrix of users and items, resulting in a simple structure. 3. Works well with rich user behaviors.	1. Requires a great amount of explicit or implicit user behaviors and has cold start problem. 2. Assumes that the interests of users entirely depend on previous actions, without considering the current context. 3. Needs to be linked through the exact same product and similar products cannot meet the standard. 4. Vulnerable to sparse data.
Association rule-based recommendation	1. Easy to understand principle. 2. Easy to explain that the refined rules are relatively stable. 3. No need for frequent updates.	1. Not truly personalized recommendations. 2. Hot items are easy to be over-recommended. 3. Cold start and sparsity problems.

In this research, AED-MF recommendation algorithm was designed based on the characteristics of deep learning of autoencoder and CF algorithm of matrix factorization combining the two approaches to solve rating data sparsity problem. The algorithm focused on learning deeper latent features of users and items from user ratings. AED-MF recommendation algorithm performed well in key recommendation effect indicators throughout data download, mount, data cleaning, model training and model experiment steps.

2. Materials and Methods

This research designed a CF recommendation algorithm based on AED-MF.

2.1 Problem Description

Traditional recommendation algorithms suffer from cold start and data sparsity problems. While external knowledge infusion (e.g., DBpedia in MF-LOD) has shown promise for cold-start scenarios [20], our AED-MF focused on learning endogenous latent patterns from sparse ratings. Although CF-based recommendation algorithms perform

well in recommendation systems, they is still vulnerable in the case of sparse data. For example, when the number of users in the CF-based recommendation algorithm based on nearest neighbors is too large, finding the nearest neighbors is very difficult, which affects the efficiency of recommendation. Furthermore, probabilistic matrix factorization-based recommendation algorithms cannot achieve good results when the rating matrix is sparse. Therefore, the developed AED-MF algorithm was applied to further investigate rating data sparsity problem.

Trying to solve data sparsity problem, Chu et al. combined autoencoder and clustering to propose a CF-based algorithm, which mainly reduced the search space of the nearest neighbor to enhance the accuracy of the developed recommendation algorithm [21]. Oord et al. (2013) [22] applied deep learning to weighted matrix decomposition, determined potential factors after processing implicit feedback, and mapped audio content to potential factors. However, the above research works had only modified deep learning algorithms on the basis of CF and did not directly couple matrix decomposition and deep learning models. Since the AE is a self-supervised algorithm, it mines supervised information from unsupervised data, and then employs the mined supervised information to train the network. Hence, as long as the input is trained to achieve the specific encoder required, the encoder and decoder of AE could be built and the loss function could be set to measure the magnitude of loss. Here, both encoder and decoder are parametric equations.

Recently, many studies have been performed on the deep learning of autoencoder, but few research works have directly combined it with matrix factorization and applied it in CF. The AED-MF algorithm proposed in this research combined the advantages and disadvantages of CF-based algorithms based on deep learning and matrix factorization and combined these two approaches. The potential rating data was predicted through deep matrix factorization and then, rating data sparsity problem was reduced to a certain extent.

2.2 Design of the AED-MF Algorithm

2.2.1 Matrix Factorization

Matrix factorization is defined as the decomposition of a matrix into the product of two lower-dimensional matrices (e.g., user and item latent factor matrices). In practical applications such as engineering operations and model analyses, matrix factorization is mainly applied as a mathematical tool to solve matrix problems. This tool extracts hidden characteristics in the data. It represents items and users in a way that is a vector of factors inferred from the item scoring pattern and predicts missing ratings by iteratively optimizing the factor matrices to minimize the reconstruction error.

Currently, a variety of matrix factorization models are available for various application requirements. Spectral decomposition is the most commonly applied matrix decomposition method capable of decomposing a normal matrix (any normal matrix can be transformed into a diagonal matrix) into the product of eigenvalues and eigenvectors, simplifying model training and shortening training process. Assuming A to be a square normal matrix, it could be simply calculated as:

$$A = \sum_{i=1}^{n} \lambda_i A_i \quad (1)$$

where λ is an eigenvalue satisfying $A_i^* = A_i \neq 0 (i = 1,2,3\ldots,\mathrm{n})$, $A_i A_j \neq 0 (i \neq j)$.

2.2.2 Autoencoder

MF-based recommendation algorithms predict user preferences by computing the inner product of user and item latent factor vectors. However, the latent factors learned by MF are derived from a linear decomposition of the rating matrix, showing that the user-item relationships obtained by this method are relatively rough. In fact, this relationship is nonlinear, directly leading to the fact that simple matrix factorization is not able to capture too complex relationship, resulting in over-fitting phenomenon.

Recent studies have demonstrated the effectiveness of deep learning in various machine learning tasks, such as speech recognition, image processing, and natural language processing [23], and these techniques have also been successfully applied to recommendation systems. Deep learning models, such as auto-encoding (AE) and denoising autoencoder (SDAE) [24] have shown great potential for learning effective and compact representations and capturing hidden data relationships. Deep learning can also be applied in computer vision and natural language processing, such as in YouTube [25]. In fact, in the application of probability matrix-factorization (PMF), SDAE performs well and may significantly enhance the performance of recommendation.

The above analysis showed that deep nonlinear computation-based AE can better present its latent vectors on relatively sparse data sets. Hence, this research proposed producing the corresponding network based on the characteristic situations obtained by AE, more accurately capturing the latent vectors of users and items. In addition, in order to enhance the efficiency of the given algorithm in big data environment, we applied matrix factorization method for the prediction of the data in the matrix.

Furthermore, AE is a type of neural network that employs a backpropagation algorithm to make the output value equal to the input value, which belongs to unsupervised learning while autoencoders provide an excellent tool to decrease dimensionality. In essence, an autoencoder is a data compression algorithm where the encoder (the part responsible for the compression of the input into latent space representation) and the decoder (the part responsible for reconstruction of the input from the latent space representation) are employed by a neural network. The main value lies in the latent space representation compressed by the encoder.

Recent advances in cross-domain recommendation have demonstrated that transferring knowledge from auxiliary domains can further alleviate data sparsity [26-27]. For instance, Li and Tuzhilin [28] proposed a deep dual transfer model to migrate user preferences across domains.

To address the domain gap issue, Zhao et al. [29] developed a user interest alignment method that learns domain-invariant representations. Similarly, Xu et al. [30] proposed a multi-view graph contrastive learning framework (MGCL) to bridge domain gaps by fusing sequential information (user sequence graphs) and collaborative information (cross-domain global graphs). These studies suggest that integrating cross-domain information with our AED-MF framework could be a promising direction for future work.

2.2.3 Process Design of AED-MF Algorithm

Based on probabilistic matrix factorization method, user-context matrix is generated using user common evaluation factors and auto-encoder deep learning is combined with CF based on matrix factorization to decrease the dimension. The relevant information of users on co-purchase could be fully mined by avoiding direct operation on sparse high-dimensional matrix.

AED-MF algorithm can be considered as a hidden layer neural network that takes $x \in R^m$ as input and maps the input to $z \in R^n$ by a mapping function, that is,

$$z = f(x) = \sigma(W^T x + b) \quad (2)$$

where $W \in R\hat{}(m \times n)$ and b is offset vector. The resulting implicit representation vector is reconstructed back to the vector $x \in R^m$ by the function $\hat{x} = \sigma(W'z + b')$, where $W' = W$. Training error is stated as $\min = \frac{1}{n}\sum_{i=1}^{n} l(x_i, \hat{x}_i)$. This is mainly obtained via learned implicit representation minimization where l is loss function. To further improve generalization, future work might incorporate dual error correction (DTEC) that jointly optimizes user-item training residuals [31].

The flowchart of the developed AED-MF algorithm is presented in Figure 1.

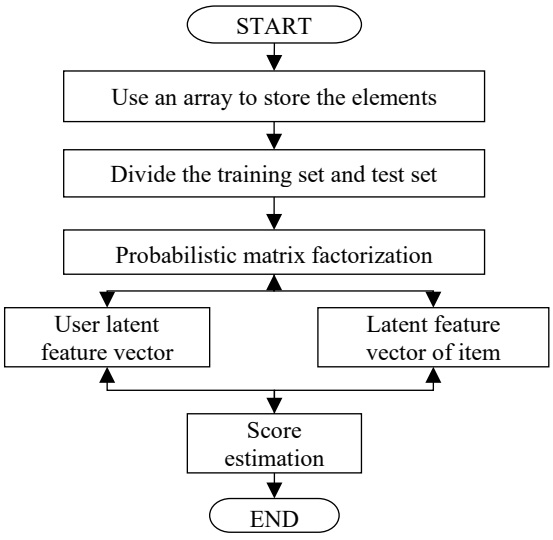

Figure 1. The flowchart of the developed AED-MF algorithm

2.2.4 Design of AED-MF Algorithm

In rating-based CF, this research aimed to design a user-based autoencoder. For example, for movie recommendation, m users, n movies, and a user-movie rating matrix $(m \times n)$ were assumed. Then, rating matrix was factorized into a probabilistic matrix and a user $u \in U$ is represented by his/her rating vector $r_u = r(R_{u1}, R_{u1}, \ldots, R_{un}) \in R^n$. We took the resulting invisible vector $r(u)$ as input and projected it into a low-dimensional latent space. The missing rating part in rating matrix was predicted and filled by reconstruction and then, the user was recommended to achieve recommendation purpose.

Both AE encoder and decoder parts of AED-MF algorithm consisted of feedforward neural networks with classical fully connected layers. Several neurons were connected to construct a neural network, the output of one neuron was fed into the input of another neuron, and a concrete network was constructed through layer by layer propagation calculation. Finally, using the developed neural network, the error between the output and input of the neural network was minimized.

The developed AED-MF algorithm was as follows.

1. Loading the data. The 'load_data()' function returned training and test data as well as the number of users and items.

2. Building the autoencoder and deep learning network and using the training set to train to obtain the autoencoder model.

3. Extracting item feature vectors from the autoencoder. Here, each item needed to be encoded and the resulting vector was then applied for matrix factorization.

4. Building a matrix factorization model. Models were trained using alternating least squares (ALS), SVD++, latent Dirichlet allocation (LDA), etc.

5. Training matrix factorization model using the training set and obtaining matrix factorization vector (often referred to as latent factor vector) for each item.

6. Predicting test set using matrix factorization model and item feature vectors and calculating model performance metrics, including Root Mean Square Error (RMSE) for rating prediction accuracy and Recall@20 for top-N recommendation performance.

The following Table 2 further describes the developed AED-MF algorithm in the form of pseudo-code:

Table 2. The developed AED-MF algorithm in the form of pseudo-code

Algorithm AED-MF
Input:
None // Data is implicitly loaded through load_data()
Output:
metrics: dict // It includes metrics such as RMSE and recall

1: Begin:
// Step 1: Load data
2: train_data, test_data, num_users, num_items = load_data()

```
    // Step 2: Build and train the autoencoder
3:  autoencoder = build_autoencoder()  // Create a network structure
4:  autoencoder.train(train_data)      // Train using the training set
    // Step 3: Extract the feature vector of the item
5:  item_features = []
6:    for each item i in range(num_items):
7:        feature_vector = autoencoder.encode(item_i)
    // Encoded items
8:        item_features.append(feature_vector)
    // Step 4: Construct the matrix factorization model
9:  mf_model = MatrixFactorization(method='ALS')  // Optional ALS/SVD++/LDA
    // Step 5: Train the MF model
10: latent_vectors = mf_model.train(
11:       train_data,
12:       item_features=item_features  // Pass in the autoencoder features
13:   )
    // Step 6: Prediction and Evaluation
14: predictions = []
15:   for each (user, item, _) in test_data:
16:       pred_rating = mf_model.predict(user, item)
17:       predictions.append(pred_rating)
18:   metrics = {
19:       'RMSE': calc_rmse(predictions, test_data.ratings),
20:       'Recall': calc_recall(predictions, test_data)
21:   }
22:   Return metrics
23: End
```

2.3 Technical details and parameter values

The proposed AED-MF algorithm integrated Autoenco der (AE)-based deep learning with Matrix Factorization (MF) to address data sparsity and cold-start issues in CF-ba sed recommendation systems. It combined nonlinear featu re extraction via AE with latent factor modeling through MF to enhance the accuracy of recommendation.

2.3.1 Matrix Factorization (MF)

- **Objective**: Decomposition of user-item rating matrix into latent factor matrices to predict missing ratings.
- **Method**: Spectral decomposition is employed to derive eigenvectors and eigenvalues.

Eq. (1) is used for matrix decomposition:
where λ_i are eigenvalues and A_i are eigenvectors.

- **Implementation:** Singular value decomposition (SVD) is used in code, retaining top-k singular values (k=20) to decrease dimensionality.

2.3.2 Autoencoder (AE)

Parameter values are summarized in Table 3.

- **Architecture**:

Encoder: A feedforward neural network with one hidden layer (32 neurons, ReLU activation).
Decoder: Reconstructs input using a sigmoid-activated output layer.

- **Training Configuration**:

Loss Function: Binary cross-entropy to minimize reconstruction error.
Optimizer: Adam with 50 training epochs and a batch size of 256.
Input/Output: Sparse user-item rating matrix (converted to dense array).

Table 3. Parameter values used in the developed model

Component	Parameter	Value/Description
Autoencoder	Input Layer	shape=(num_items)
	Hidden Layer (Encoder)	32 neurons, ReLU activation
	Output Layer (Decoder)	shape=(num_items), sigmoid
	Loss Function	Binary cross-entropy
	Optimizer	Adam
	Training Epochs	50
	Batch Size	256
Matrix Factorization	Latent Factors (kk)	20
	Decomposition Method	SVD (via np.linalg.svd)

2.3.3 AED-MF Integration

1. **Data Preprocessing**:

A user-item rating matrix $R \in R^{m \times n}$ (filled with zeros for missing ratings) is constructed.
The matrix is normalized and splitted into training/test sets.

2. **AE Training**:

AE is trained to compress the input $x \in R^m$ into a latent vector $z \in R^{32}$, then x is reconstructed.
Latent user vectors user_vecs$\in R^{m \times 32}$ is extracted from the encoder.

3. **MF on AE Output**:

SVD is applied to user_vecs, retaining k=20 latent factors.
User and item latent factor matrices $U_{sk} \in R^{m \times k}$ and $V_{Tk} \in R^{k \times n}$ are derived.

4. **Recommendation**:

Computation of recommendation scores via dot product:
scores=item_vecs·user_vecs[user_index]
Excluding already rated items and returning top-N recommendations.

3. Results

3.1 Dataset

The data used in this research were obtained from the publicly available Movielens 1M dataset. This dataset was

developed by GroupLens research group at the University of Minnesota and is extensively applied for movie recommendations to evaluate the efficiency of collaborative filtering algorithms. The dataset was consisted of 3,900 movies and 100,209 ratings from 6,040 users on a scale of 1-5 where each user rated at least 20 movies. Furthermore, the dataset provided 18 different movie genres with a maximum of three genres per movie.

After cleaning Movielens 1M dataset, user records without ratings were eliminated. The obtained valid dataset contained 9664 items and 534633 ratings. We rounded each rating to a value ranging from 1 (worst) and 5 (best). The final dataset was consisted of {user, movie, rating, timestamp}. Rating matrix was user-movie rating matrix ($m * n$), where m and n are the number of users and movies, respectively. User data had different fields such as user ID, gender, age, occupation, and zip code and rating data had fields such as user ID, movie ID, rating, and timestamp. Among them, rating field was the target of the model learning and timestamp did not need to be processed.

This dataset contains three types of data: movie, user, and rating. The Movie table includes the main fields: {MovieId, MovieName, MovieType};The User table includes the main fields: {UserId, gender, age, JobID, ZipCode};The Rating table includes the main fields: {UserId, MovieID, Rating}.

3.2 Data Preprocessing

In data preprocessing step, we left UserID and MovieID in the original dataset unchanged and replaced F with 0 and M with 1 in gender field. At the same time, one-hot encoding was applied for converting the type field MovieType into a number; that is, only 0 and 1 were applied to represent, **among them, *N* states require *N* bits of register for encoding.**

3.3 Experimental Process

3.3.1 Analysis of the Impact of the Number of Neurons in the Autoencoder Hidden Layer on Model Performance

In the AED-MF algorithm, the number of neurons in the hidden layer of the autoencoder directly affects the model's ability to extract latent features of users and items. Too few hidden neurons may lead to underfitting, failing to fully capture the nonlinear relationships in the data; conversely, too many neurons can easily cause overfitting, reducing the model's generalization capability. To investigate the impact of the number of hidden neurons on model performance, this experiment fixed other hyperparameters (such as learning rate, batch size, regularization coefficient, etc.) and set the number of hidden neurons to 32, 64, 128, 256, and 512 on the MovieLens 1M dataset. The changes in Recall and MSE loss on the test set were observed under different settings.

During the experiments, the autoencoder adopted a single hidden layer structure with ReLU as the activation function and linear activation for the output layer. The number of training epochs was uniformly set to 200, the optimizer was Adam, and the learning rate was set to 0.001. To reduce the impact of randomness, each group of experiments was repeated five times and the average values were taken. The results are shown in Table 4.

Table 4. Performance Comparison under Different Numbers of Hidden Neurons

Number of Hidden Neurons	Recall	MSE loss
32	0.893	1.213
64	0.921	1.102
128	0.947	1.085
256	0.951	1.101
512	0.938	1.154

From Table 4, it can be seen that as the number of hidden neurons increases from 32 to 128, the recall gradually improves and the MSE loss gradually decreases, indicating that the model's expressive power is enhanced and it can more accurately fit user preferences. When the number of neurons reaches 128, the recall reaches 0.947 and the MSE loss is 1.085, achieving the optimal performance. Continuing to increase the neurons to 256 results in a slight improvement in recall (0.951), but the MSE loss rebounds (1.101), indicating early signs of overfitting. When the number of neurons increases to 512, the recall drops to 0.938 and the MSE loss further increases, making the overfitting phenomenon more pronounced.

Based on the above analysis, under the experimental conditions of this paper, setting the number of hidden neurons to 128 achieves the best balance between recall and prediction error. Therefore, the final AED-MF algorithm adopts 128 neurons in the autoencoder hidden layer. This sensitivity analysis validates the rationality of the model structure design and provides a basis for subsequent parameter tuning.

3.3.2 Data Partitioning, Transformation, and Cleaning

After parameter adjustment, the data were divided into training, validation and test sets by the proportions of 80%, 10% and 10%, respectively. Also, the data was consolidated into {user, movie, rating, timestamp} for easy management.

After data partitioning, data format conversion and data cleaning were performed and the ratings of the movies that have not been rated by users in the data set were all set to zero. First, we matched the users with the movies, and if the users has not rated the movies, we set the rating field to “0”. Specifically, we aligned user-movie pairs and assigned a rating of "0" to any unmatched entries.

In this experiment, Python programming language was applied. Since the MovieID of the movie data table in the dataset started from 1 and the default value in python starts

from 0, in order to match the python syntax, all Movieids were subtracted by 1.

Then, we looked for rows with ratings of 0 in the training, validation, and test sets and deleted such rows entirely. The data was loaded into DataLoad interface (essentially an iterable object) in python torch library, batch size was set to 16, and custom Dataset was transformed into Tensor dataset for subsequent application in the experiment.

3.3.3 Model Definition, Loss Function, and Training Function

AE Model Definition: AE was originally employed to learn a data representation (encoding). It could be decomposed into two parts: the encoder, which could decrease data dimension, and the decoder which was applied for converting the encoding back to its original form. Due to the reduction of dimensionality, in order to be able to reconstruct the input, neural networks need to learn a low-dimensional representation of the input (latent space). To employ AE for the prediction of new recommendations, the input and output were click vectors (usually the input and output of AE are the same). Using a large dropout (0.9) after the input layer required the reconstruction of click vector for predicting the recommended value for a given click vector since an element in the input was missing.

Here, we have discussed loss and training functions. If the user only rated part of the movies, to avoid affecting the results when calculating the MSE loss (MSE refers to the Mean Squared Error loss function), only the movies that were rated by many users were considered and those that were not rated too many time were ignored . The total data of our original dataset (ml-1m) contained 3952 movies, 6040 users and 100209 rating data. Also, the total data of the new dataset (ml-latest-small) contained 9742 movies, 610 users and 100836 rating data. For the convenience of description, we used the following simplified data as an example. MSE was calculated as follows (using 2 users rating 5 movies): [[1, 2, 0, 3, 4], [2, 3, 0, 0, 1]]. Then, AE rate data was reconstructed as: [[1.1, 2.3, 0, 3.3, 4.7], [2.1, 3.2, 0, 0, 1.2]]. MSE per user was determined by first taking the squared two-norm of the two ratings (the largest eigenvalue of the product of the transpose conjugate matrix of A and matrix A) and then dividing by the number of movies each user rated. Finally, for each batch (a parameter update that backpropagated the model weights with a fraction of the samples in the training set), we employed torch.mean function to calculate MSE. Furthermore, only movies threated by many users were considered in training process. Therefore, even though we set the ratings of the movies that the user had not rated to 0, we needed to set the ratings of the movies that the model predicted the user would not rate (in the loop) to 0 as well. This was done to avoid adding to the loss and updating the weights. Finally, a plot was drawn to show the variations of MSE loss during training and validation processes.

Finally, we have discussed prediction function. The role of prediction function was to apply the model for predicting the movies that had not been rated by the user to obtain prediction scores and then, recommend the movies that had not been watched by users based on the prediction scores. This was performed as described below: First, we created a temporary array of IDs for later sorting. We inserted indices into the result arrays, transposed the rows and columns (inverted dimensions), and converted the recommendations to a list for further processing.. Then, movie_not_seen was used to store the unwatched movies and put the top 20 into recommend_movie by sorting them in descending order of prediction scores. Finally, the index of the 20 movies in the original movies was extracted, movie names were exported, and relevant recommendation result was reported, which was to recommend 20 movies that a given user might like from the movies that he had not seen before.

3.4 Comparison of Experimental Effects

3.4.1 Model Training

Model Training and Evaluation Parameter Values. AE hidden layer: 32 neurons, ReLU activation.Training epochs: 50, batch size: 256. Latent factors in MF: k=20. Code Implementation Highlights is as follows.

```
# Python
# Autoencoder Architecture
input_layer = Input(shape=(R.shape[1],))
encoded = Dense(32, activation='relu')(input_layer)  #
Latent space: 32 dimensions
decoded = Dense(R.shape[1],
activation='sigmoid')(encoded)
autoencoder = Model(input_layer, decoded)

# Training Configuration
autoencoder.compile(optimizer='adam',
loss='binary_crossentropy')
autoencoder.fit(R.toarray(), R.toarray(), epochs=50,
batch_size=256)

# Matrix Factorization (k=20)
k = 20
U, sigma, V_T = np.linalg.svd(user_vecs,
full_matrices=False)
item_vecs = V_T[:k, :].T  # Item latent factors
```

3.4.2 Evaluation Metrics

The following two evaluation metrics are commonly used in the comparison experiments of recommendation algorithms: Recall and Mean-Squared loss (MSE loss). These two metrics are also applied to evaluate recommendation algorithm running results on the test data set.

Recall is defined as the ratio of accurate prediction and actual test data set recommended to users based on the ranking of predicted ratings, which measures retrieval system recall rate. Recall for a user is calculated as follows:

$$recall(M) = \frac{The\ number\ of\ items\ purchased\ by\ the\ user\ appears\ in\ the\ recommended\ items\ list}{The\ number\ of\ items\ the\ user\ actually\ purchased}$$

MSE loss is applied for the evaluation of the mean the sum of squared errors in predicted and actual scores, with smaller values indicating higher accuracy. The MSE loss was calculated as follows.

$$\text{MSE-loss} = \frac{\Sigma(R_{ij} - \hat{R}_{ij})^2}{num} \quad (4)$$

where num is rating score, R_{ij} is actual score and $\hat{R}ij$ is predicted score. In this experiment, we mainly called the functions in python and torch.nn libraries to calculate MSE loss and visually display it through images.

3.4.3 Experimental Results

In the above data division, the rating data of the dataset was divided into training, validation and test sets with the ratios of 80%, 10% and 10%, with the amount of data in each set being 8000167:100021:10002, respectively. After data cleaning, the data set applied in the final experiment contained 6040 users and 3043 movies with data density of 0.542. After 200 epochs, test set MSE loss was 1.0805.

However, validation set loss was first decreased and then slightly increased, as illustrated in Figure 2, which was considered to be caused by poor generalization ability.

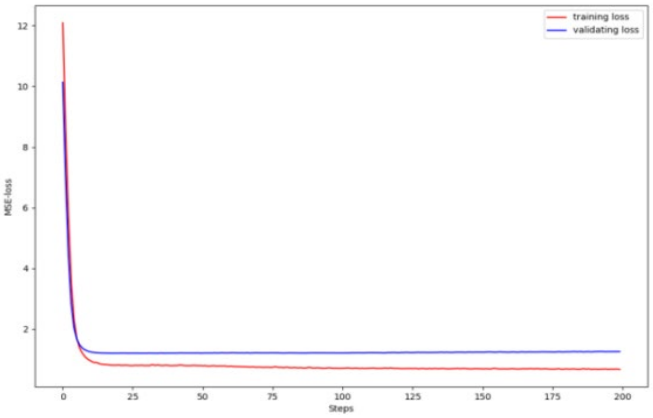

Figure 2. AE loss analysis

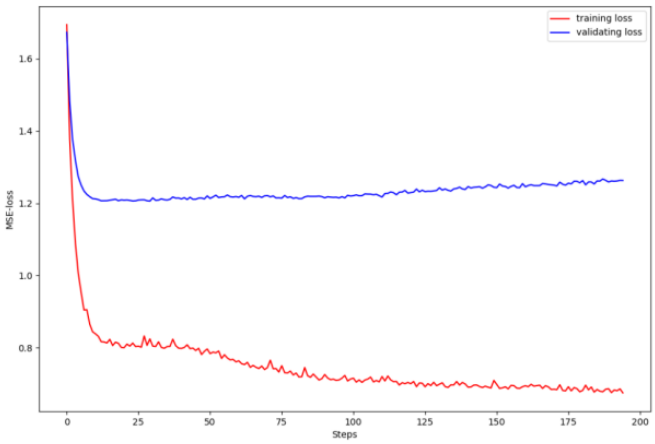

Figure 3. AE optimization results

In order to solve the generalization problem caused by AE, we later tried the corresponding optimization using VAE, as shown in Figure 3. The results show that although the training loss of VAE is larger than that of AE, its validation loss is smaller than that of AE. Although the training loss of VAE is larger than that of AE, VAE achieves better performance on the test and validation sets because it captures more complex patterns in the data. And the loss pairs of the new dataset under the VAE algorithm are shown in the following figure 4 and Figure 5.

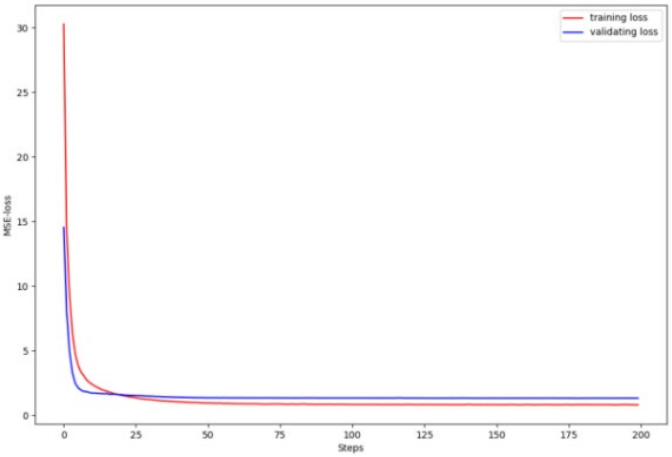

Figure 4. VAE loss analysis

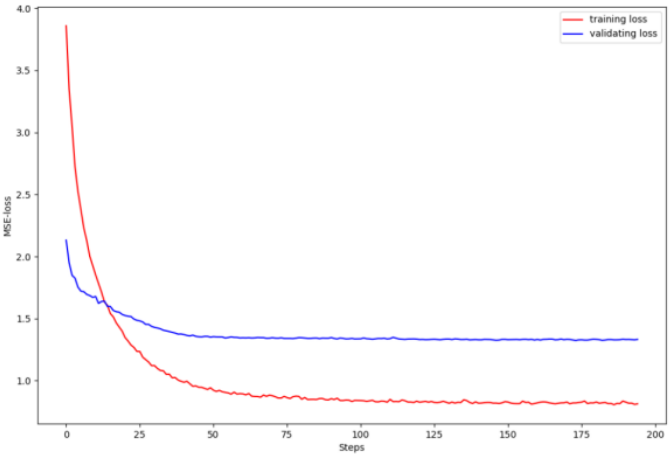

Figure 5. VAE optimization results

The developed algorithm obtained 95% recall. Generally, value of above 90% indicate that the recommendation algorithm was good.

4. Discussion

The aim of this research was to improve the recommendation efficiency of the recommendation system, fully mine information, and achieve better recommendation efficiency. By reviewing the early

recommendation systems, this research surveyed the studies on recommendation algorithms and summarized the limitations of current recommendation algorithms. The recommendation accuracy of content-based recommendation algorithm was low and the recomme0ndation algorithms based on CF and association rules inevitably suffered from sparsity.

Therefore, based on the characteristics of deep learning of autoencoder and CF algorithm of matrix factorization, this research combined the two methods to construct AED-MF recommendation algorithm. This algorithm solved rating data sparsity problem while ensuring high accuracy of recommendation. By constructing an autoencoder and deep learning network, training set was used to train the model. Autoencoder model was thus obtained and item feature vector was extracted from it to establish matrix factorization model. Finally, matrix factorization model and item feature vectors were employed for the prediction of test set. The results showed that the recall rate of AED-MF recommendation algorithm was 95%, which was higher than that of previous CF-based algorithms. In addition, the value of VAE loss was decreased with the increase of data volume and the above comparison experiments verified that the rating prediction accuracy obtained in this research was improved.

5. Conclusions

This research proposed a CF-based algorithm based on deep learning and matrix factorization (AED-MF) by combining AE and matrix factorization to solve data sparsity problem. The trained model effectively extracted deeper representations of hidden vectors in users and items and effectively applied to nonlinear relationships among users and items. In addition, matrix factorization method was applied for predicting missing rating data in the rating matrix. The proposed AED-MF algorithm also made full use of the great advantages of deep learning algorithms in extracting latent feature vectors. Experiments revealed that the recall rate of the developed algorithm was enhanced compared with previous CF-based algorithms and VAE loss value was decreased by increasing data volume, indicating that rating prediction accuracy was improved.

Although the AED-MF recommendation algorithm developed in this research presented a certain improvement in recommendation effect, there are still many problems in the field of recommendation algorithms that need to be investigated and solved. According to the findings of this research, more in-depth analysis and improvement had to be performed: first, a recommendation algorithm more suitable for actual needs could be mined and transformed. Furthermore, with the help of other more complex and effective deep learning models (e.g., deep neural networks such as BERT to integrate with recommendation algorithms), the performance and accuracy of recommendation are continuously improved, while the generalization ability of the algorithm is enhanced by decreasing fitting degree. Finally, AED-MF recommendation algorithm can be applied to more recommendation scenarios and its effectiveness can be explored.

Funding

This research was supported by the High-level Research Institute of New International Relations, Zhejiang International Studies University, under the projects “Strategies for Enhancing Trade Cooperation among BRICS Countries Driven by China’s Cross-Border E-Commerce” (Project No. 2025GYYZD09) and “Analysis of the Business Environment and Policy of Cross-Border E-Commerce in BRICS Countries: A Perspective of Cross-Border E-Commerce Ecosystem Development” (Project No. 2025GYYZD09-1).

References

[1]. Guttentag D. Airbnb: disruptive innovation and the rise of an informal tourism accommodation sector. Current issues in Tourism, 2015, 18(12): 1192-1217.
[2]. Yang Z R, He Z Y, Wang C D, et al. Collaborative Meta-Path Modeling for Explainable Recommendation. IEEE Transactions on Computational Social Systems, 2023, pp. 1–11.
[3]. Negroponte N. Being Digital. Knopf, 1995, pp. 26-33.
[4]. Wang Y, Ma W, Zhang M, et al. A survey on the fairness of recommender systems[J]. ACM Transactions on Information Systems, 2023, 41(3): 1-43.
[5]. Morales-Murillo V G, Pinto D, Perez-Tellez F, et al. A Transformer-Based Multi-Domain Recommender System for E-commerce[J]. International Journal of Combinatorial Optimization Problems and Informatics, 2024, 15(2): 95.
[6]. Kuo R J, Li S S. Applying particle swarm optimization algorithm-based collaborative filtering recommender system considering rating and review[J]. Applied Soft Computing, 2023, 135: 110038.
[7]. Walek B, Fajmon P. A hybrid recommender system for an online store using a fuzzy expert system. Expert Systems with Applications, 2023, 212: 118565, 1-16.
[8]. Lee Y H, Wei C P, Hu P J H, et al. Small clues tell: A collaborative expansion approach for effective content-based recommendations. Journal of Organizational Computing and Electronic Commerce, 2020, 30(2): 111-128.
[9]. Ge S, Wu C, Wu F, et al. Graph enhanced representation learning for news recommendation[C]//Proceedings of the web conference 2020. 2020: 2863-2869.
[10]. Yi J, Zhu Y, Xie J, et al. Cross-modal variational auto-encoder for content-based micro-video background music recommendation[J]. IEEE Transactions on Multimedia, 2021, 25: 515-528.
[11]. Goldberg D, Nichols D, Oki B M, et al. Using collaborative filtering to weave an information tapestry. Communications of the ACM, 1992, 35(12): 61-70.
[12]. Lee G Y, Tseng W P. An enhanced memory-based collaborative filtering approach for context-aware recommendation //Proceedings of the World Congress on Engineering. 2015, 1, pp. 198-202.
[13]. Choi K, Suh Y. A new similarity function for selecting neighbors for each target item in collaborative filtering. Knowledge-Based Systems, 2013, 37: 146-153.

[14]. Ungar L H, Foster D P. Clustering methods for collaborative filtering//AAAI workshop on recommendation systems. 1998, 1: 114-129.
[15]. Chien Y H, George E I. A bayesian model for collaborative filtering//AISTATS. 1999, pp. 1-6.
[16]. Heckerman D, Chickering D M, Meek C, et al. Dependency networks for inference, collaborative filtering, and data visualization. Journal of Machine Learning Research, 2000, 1(Oct): 49-75.
[17]. Jamali M, Ester M. A matrix factorization technique with trust propagation for recommendation in social networks//Proceedings of the fourth ACM conference on Recommender systems. 2010: 135-142.
[18]. Wang C, Blei D M. Collaborative topic modeling for recommending scientific articles//Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. 2011: 448-456.
[19]. Narducci F, Basile P, Musto C, et al. Concept-based item representations for a cross-lingual content-based recommendation process. Information Sciences, 2016, 374: 15-31.
[20]. Natarajan S, Vairavasundaram S, Natarajan S, et al. Resolving data sparsity and cold start problem in collaborative filtering recommender system using linked open data[J]. Expert Systems with Applications, 2020, 149: 113248.
[21]. Chu H, Xing X, Meng Z, et al. Towards a deep learning autoencoder algorithm for collaborative filtering recommendation//2019 34rd Youth Academic Annual Conference of Chinese Association of Automation (YAC). IEEE, 2019: 239-243.
[22]. Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In NIPS, 2013, pages 2643–2651.
[23]. Zhao W, Ma H, Li Z, et al. Improving social and behavior recommendations via network embedding. Information Sciences, 2020, 516: 125-141.
[24]. Majumdar A. Blind denoising autoencoder. IEEE transactions on neural networks and learning systems, 2018, 30(1): 312-317.
[25]. Covington P, Adams J, Sargin E. Deep neural networks for youtube recommendations. Proceedings of the 10th ACM conference on recommender systems. 2016: 191-198.
[26]. Khan M M, Ibrahim R, Ghani I. Cross domain recommender systems: A systematic literature review[J]. ACM Computing Surveys (CSUR), 2017, 50(3): 1-34.
[27]. Zhu F, Wang Y, Chen C, et al. Cross-domain recommendation: challenges, progress, and prospects[J]. arXiv preprint arXiv:2103.01696, 2021.
[28]. Li P, Tuzhilin A. Ddtcdr: Deep dual transfer cross domain recommendation[C]//Proceedings of the 13th international conference on web search and data mining. 2020: 331-339.
[29]. Zhao C, Zhao H, He M, et al. Cross-domain recommendation via user interest alignment[C]//Proceedings of the ACM web conference 2023. 2023: 887-896.
[30]. Xu Z, Chen S, Pan W, et al. A multi-view graph contrastive learning framework for cross-domain sequential recommendation[J]. ACM Transactions on Recommender Systems, 2025, 3(4): 1-28.
[31]. Panagiotakis C, Papadakis H, Papagrigoriou A, et al. Improving recommender systems via a dual training error based correction approach[J]. Expert Systems with Applications, 2021, 183: 115386.