Sādhanā (2022) 47:158
https://doi.org/10.1007/s12046-022-01924-0

GSO-CRS: grid search optimization for collaborative recommendation system

GOPAL BEHERA* and NEETA NAIN

Department of Computer Science and Engineering, Malaviya National Institute of Technology Jaipur, Jaipur, Rajasthan, India
e-mail: 2019rcp9002@mnit.ac.in; nnain.cse@mnit.ac.in

MS received 4 December 2021; revised 7 June 2022; accepted 12 June 2022

Abstract. Many online platforms have adopted a recommender system (RS) to suggest an actual product to the active users according to their preferences. The RS that provides accurate information on users' past preferences is known as collaborative filtering (CF). One of the most common CF methods is matrix factorization (MF). It is important to note that the MF technique contains several tuned parameters, leading to an expensive and complex black-box optimization problem. An objective function quantifies the quality of a prediction by mapping any possible configuration of hyper-parameters to a numerical score. In this article, we show how a gird search optimization (GSO) can efficiently obtain the optimal value of hyper-parameters an MF and improve the prediction of the collaborative recommender system (CRS). Specifically, we designed a 4×4 grid search space, obtained the optimal set of hyper-parameters, and then evaluated the model using these hyper-parameters. Furthermore, we evaluated the model using two benchmark datasets and compared it with the state-of-the-art model. We found that the proposed model significantly improves the prediction accuracy, precision@k, and NDCG@k over the state-of-art-the models and handles the sparsity problem of CF.

Keywords. Recommender system; collaborative filtering; hyper-parameter; grid search; SGD; ALS; random search.

1. Introduction

A Recommender framework is an essential aspect of online platforms, suggesting products (e.g., music, books, movies, etc.) based on users' choices. Recommender frameworks are heavily deployed for suggesting films [1], songs [2], books [3], e-commerce items [4], etc. RS techniques are divided into two categories [5]: collaborative filtering (CF) and content-based (CB). The CB technique requires detailed profiles of users and items for suggesting an item to an active user. On the other hand, the CF technique requires only the user's history to suggest a product. In this work, we only focus on CF. The CF first creates a database of user-item interactions (rating matrix). This matrix is very sparse, with few users interacting with a few items. The goal is to predict all missing values of the interaction matrix. After the Netflix competition, the matrix factorization (MF) method is successful among CF techniques [1, 5–7]. MF predicts the missing interaction through the latent features. In order to reduce the dimensionality of the interaction matrix, latent features are extracted by finding a low-rank approximation of the actual matrix using correlations between columns (or rows). A minimization problem is used to identify this approximation, whose goal is to factorize the actual matrix into two low-rank matrices. The approximation error determined the rank (i.e., K) of decomposed matrix and was used as a hyper-parameter of a model. Similarly, parameters such as regularization terms are used to reduce the over-fitting and non-linearity effect of the model. The learning rate hyper-parameter of stochastic gradient descent (SGD) [5] is deployed to retrieve a low-rank approximation of a matrix. It is very difficult to know the best values of the above hyper-parameters a priori. Grid search optimization (GSO) [8] is a recently used optimization technique in machine learning to find the optimal values of the hyper-parameters.

The outline of our contributions are:

- We designed a 4×4 grid search space and tuned four important hyper-parameters ($n_{factors}$, n_{epochs}, learning rate, and regularization term) that affect the performance of the model-based collaborative recommendation system.
- Validate and compare the results with random search and baseline MF [1, 9] on real-world movielens datasets.

*For correspondence

Published online: 10 August 2022

- Verify the influence of sparsity and latent factors on the performance of the proposed approach and compare the result with the state-of-the-art algorithms.

The rest of the article is organized as: section 2 elaborates on previous recommender systems and hyper-parameter optimization. Section 4 describes our proposed methodology. experiments, evaluation, and results analysis are presented in section 6. Conclusions are in section 8.

2. Related work

This section discusses the previous work from two perspectives: collaborative filtering (CF) and hyperparameter (HP) optimization.

Collaborative filtering (CF): is one of the most widely used recommender techniques among CF techniques [5]. As the CF approach is not domain-specific and generally more accurate due to its ability to uncover hidden patterns, it has gained much attention from research communities [10]. It is used in many commercial applications [11]. CF uses two primary techniques for constructing predictive models based on previous behavioral data: neighborhood algorithms and latent models. Neighborhood algorithms, such as item-based approaches [5], estimate the relationship among products and consumers, then used for recommendation. Latent model, such as Singular Value Decomposition (SVD) [12], calculate feature vectors representing the users and products. It is more accurate to model latent factors than neighborhood models [13].

Goldberg *et al* [14], Behera and Nain [5] discussed collaborative filtering (CF) based RS and found that this RS is helpful for users to find the relevant information from a corpus. RS retrieves relevant information from text documents and is used in various fields, from commercial products to news, audio, video, etc. Deshpande and co [15] discussed CF using kNN and computed the similarity matrices using Pearson correlation. In recent years, matrix factorization (MF) [1] has played a vital role in RS. MF algorithms have become very much popular in real-life applications like Netflix competition. MF decomposes the user rating matrix into two latent factors, which are used to predict missing values in the original matrix. Sarwar *et al* [12] have proposed SVD to learn the feature matrices and found that the MF model learned by SVD is prone to overfitting. Thus Pan *et al* [16] and Hu *et al* [17] have proposed regularized learning techniques using least square optimization to reduce the negative examples. Hofmann [18] proposed a probabilistic approach of latent factor model (PMF) for item recommendation. While Wu Zhipeng *et al* [19] described the optimized MF recommendation algorithm on rating centrality. MF uses stochastic gradient descent (SGD) to decompose the rating matrix into item and user matrices. Therefore, an additional parameter that characterizes SGD (learn rate α) needs to be optimized. Zeng *et al* [20] have recognized the difficulty in parameter setting for MF and proposed MF with scale-invariant parameters that, once set, can be transferred to matrices of different sizes. Which makes it easier for the selection of models; still, finding the optimal parameter values remains a challenge. This paper uses two scalable latent factor modeling algorithms,SGD [1, 13] and ALS-WR [21].

Hyper-parameter optimization(HO):

Almost all CF techniques come with at least one adjustable hyper-parameter, such as regularization, number of latent factors, and learning rate. Choosing the appropriate values for these hyper-parameters play a vital role in the generalizability and accuracy of your predictive model [22]. Hyper-parameter optimization means finding a better hyper-parameter value for a method to increase accuracy or optimize a specific goal of the model. Multiple disciplines have studied this problem, for instance, the application of response surfaces [23] for tuning hyper-parameters in statistics, while in machine learning (ML), advanced techniques such as genetic algorithms have been proposed to optimize hyper-parameters. However, a grid search [8] and random search [24] are two techniques that are widely used in industries. Grid search is an exhaustive search through a subset of hyper-parameter space designed by the user. Whereas random search selects a value according to a probability distribution for every hyper-parameter.

Jones *et al* [25] explained EGO (Efficient Global Optimization) technique to find the expected improvement by combining stochastic Gaussian process (GP) with linear regression. Bayesian optimization [26–28] is the most widely used global optimization strategy for sensor networks and simulation optimization problems. Some of the Bayesian optimization examples are found in [13, 29]. The current study on CF, in general, uses trial and error to set hyper-parameters. If we only need to optimize hyper-parameters once at the start, this approach may be acceptable. However, it is impractical when we need continuous hyper-parameter optimization when data is added repeatedly. In this article, we use grid search technique for optimizing CF algorithms.

3. Preliminaries and problem formulation

In this part, we have discussed preliminaries that are associated with problem formulation as well as described a list of notations that are used in this work. Table 1 shows a list of notations.

3.1 *Problem definition*

Let $U = \{u_1, u_2, \cdots, u_M\}$ and $I = \{i_1, i_2, \cdots, i_N\}$ be consists of m users and n items, respectively. Let $r_{ui} \in \mathcal{R}^{M \times N}$ represents the scoring matrix of u^{th} user on i^{th} item. Where each user rates few items, hence the entries of scoring

Table 1. List of notations.

Notations	Description
U	Set of M users
I	Set of N items
T	Total number elements in the dataset
X, Y	Two optimization algorithm
H_X, H_Y	Errors generated from X and Y
μ	Global bias
b_u	User bias
b_i	Item bias
$\hat{r}_{ui}$	Predicted rating
r_{ui}	Actual rating
p_u	User latent factor
q_i	Item latent factor
K	Latent Dimension
λ_r	Regularization term
η	Learning rate
λ, θ	Set of hyperparameters
$L()$	Loss Function
G_y	Natural Distribution
λ^*, θ^*	Optimized parameter
Λ	Search space
ψ	Hyperparameter response function

matrix are known only for a few index that is $(u, i) \in T$, with $|T| \ll M \times N$. The set T is partitioned into training T_{Train} and testing set T_{Test} such that $T_{Train} \cap T_{Test} = \emptyset$ and $T_{Train} \cup T_{Test} = T$. The goal of the CF is to predict for T_{Test} with the help of T_{Train} in such way that the performance of the CF will be enhanced.

The idea behind MF methods is to factorize the $\mathcal{R}$ into two low-rank matrices such as: $\mathcal{R} \approx p_u \cdot q_i^T$, where $p_u \in P^{M\times K}$ and $q_i \in Q^{K\times N}$ are user and item latent features with K latent dimension. Generally, $(M, N) \gg K$, and both p_u and q_i contain real values, whereas $\mathcal{R}$ consists of integers only. The MF is computed by reducing an error or loss on T_{Train} as a function of the feature vectors (p_u, q_i). Therefore, the optimization problem becomes.

$$L = \underset{p,q}{argmin}\left[\frac{1}{2}\sum_{(u,i)\in T_{Train}}\left[(r_{ui} - \hat{r}_{ui})^2\right] + \lambda_r \sum_{k=1}\left(p_{uk}^2 + q_{ki}^2\right)\right] \tag{1}$$

Where $\hat{r}_{ui} = \sum_{k=1} p_{uk} \cdot q_{ki}^T$ is the predicted score of ith item. $\lambda_r \geq 0$ represents regularization. While the loss function defined in equation 1 is to learn to minimize the error. The computation of error on the T_{Test} signify that how good could be that approximation is close to the actual ratings.

4. Grid search optimization (GSO) for hyper-parameter optimization

4.1 *Hyper-parameter optimization*

Hyper-parameters (HP) are the parameters that are not learned and set as input compared to learnable parameters. However, hyper-parameters significantly affect the system performance; hence to find the best set of HP, we need to use optimization techniques that can enhance the performance of the recommender system (RS). Some of the algorithms are more sensitive to hyper-parameter settings, so these algorithms need human experts. For example, k-means require the setting of parameter k (number of centroids), which affects clustering outcome. Sometimes experts have no idea about good parameter values in advance. Hence, they use hit and trial to find the appropriate parameter settings. An optimization method is deployed to find the optimal set of hyper-parameters instead of searching optimal parameters randomly. It is essential to compare two or more optimization algorithms with different settings of hyper-parameters. Let X and Y be two optimization algorithms with different settings of hyper-parameters, let H_X and H_Y be the error measure obtained after evaluation. Let $H_X < H_Y$. When hyper-parameter tuning is done by manually, it is tough to know whether the error difference is due to X being better than Y or because human experts tuned X better than Y. Hence, it is tough to conclude that method X is reliable than Y, which is not always possible. To marginalize the influence of an expert, objective and fair tuning components are required. HP optimization techniques play a vital role in tuning the objective component. Let X be the learning algorithm, and f be a function that minimizes the expected error $L(y; f)$ over i.i.d. Where y is a sample derived from a natural distribution G_y and $X : Y^{train} \rightarrow f$. So that X produces f through the optimization of training criterion w.r.t, θ. Whereas actual X is obtained after choosing $\lambda(hyper - parameters)$ and is denoted by X_λ, and $f = X_\lambda(Y^{train})$ for a training set of Y^{train}. We need to choose λ to minimize the error $E_{y\sim G_y}[L(y; X_\lambda(Y^{train}))]$. Finding accurate values of λ is the problem of hyper-parameter optimization, defined in equation 2.

$$\lambda^* = \underset{\lambda\in\Lambda}{argmin}\ E_{y\sim G_y}[L(Y; X_\lambda(Y^{train}))] \tag{2}$$

Further, equation 2 is addressed in terms of cross-validation as it is very difficult to evaluate the expectation over G_y, which is defined in equations 3 to 5.

$$\lambda^* \simeq \underset{\lambda\in\Lambda}{argmin}\ \underset{y\in Y^{valid}}{mean}[L(Y; X_\lambda(Y^{train}))] \tag{3}$$

$$\equiv \underset{\lambda\in\Lambda}{argmin}\ \Psi(\lambda) \tag{4}$$

$$\cong \underset{\lambda \in \{\lambda^1, \lambda^2, \cdots \lambda^s\}}{argmin} \quad \Psi(\lambda) \equiv \hat{\lambda} \tag{5}$$

Where Ψ is the response function that optimizes the parameter $\Psi(\lambda)$ over $\lambda \in \Lambda$ and Λ is the search space. We partially adopted Bergstra and Bengio's [24] notations for the formulation of our optimization problem in collaborative recommender system. RS is a tool that can handle the information overload problem. The main goal of RS is to predict relevant items for active users. RS applies to Netflix, commercial products, scientific articles, websites, medications, etc. Like other algorithms (k-means, Support vector classifier), RS is also sensitive to finding the correct set of parameters. Therefore, we use the grid search optimization process to find the best set of hyper-parameters in our work.

4.2 *Grid search optimization (GSO)*

Let X be a target algorithm with k parameters to be tuned, and parameter θ_i be a value within the interval $[x_i, y_i]$ in parameter search space $\Theta = [x_1, y_1] \times \cdots \times [x_k, y_k]$. $H : \Theta \rightarrow R$ be a performance measurement function that maps θ to a numeric score. The error H is computed on cross-validation, and Θ consists of four CF parameters: n_{epochs}, $n_{factors}$, $learning_{rate}$, and λ_r.

Figure 1 represents a visualization of the grid and random search (RS) techniques [8, 24]; grid search tries to evaluate every combination of the hyperparameters and note the accuracy. Once all combinations are evaluated, then the model provides the set of parameters with the best accuracy. Whereas RS is similar to grid search but, it tries to combine the parameters randomly. An example of the grid and random search optimizing with 3×3 sets of parameters is shown in Figure 2, which illustrates how 9 trial points are to be tested using both random and grid search techniques and is found that random search is the best technique compared to grid search for lower dimensional data as the time taken to find the best set of parameters are less with less number of iteration.

Let Λ be a set indexed of K configuration variables. The grid search(GS) needs to obtain the optimal values from a set of values on each variable $(L^1, L^2, \cdots, L^k)$, so that number of trials in a GS is $S = \prod_{k=1} |L^k|$ elements.

5. Proposed method

In this study, we have designed a two-stage collaborative model using grid search optimization to improve RS's accuracy, as shown in figure 3.

In the first stage, we find the best set of hyper-parameters from Θ for a targeted algorithm using grid search optimization and assume that these hyper-parameters can improve the prediction accuracy of an RS. Regular parameters such as user feature vector (p_u) and item feature vector (q_i) characterize the MF model and are estimated by training the model using predefined data, whereas, hyper-parameters such as learning rate (η), regularization (λ), number of epochs, and number of latent factors (K) control how the above feature vectors $(p_u$ and $q_i)$ are estimated. Finding appropriate settings for these hyperparameters is crucial since they strongly influence the prediction performance of a collaborative recommender system (CRS) using MF techniques.

In this work, grid search optimization (GSO) technique, is utilized to find the best hyperparameters which can be expressed as follows.

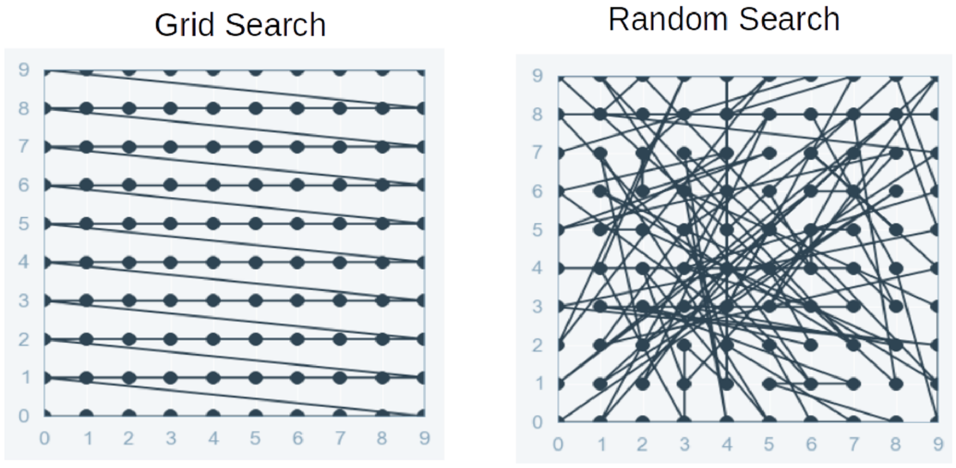


Figure 1. Visualization of grid search and random search.

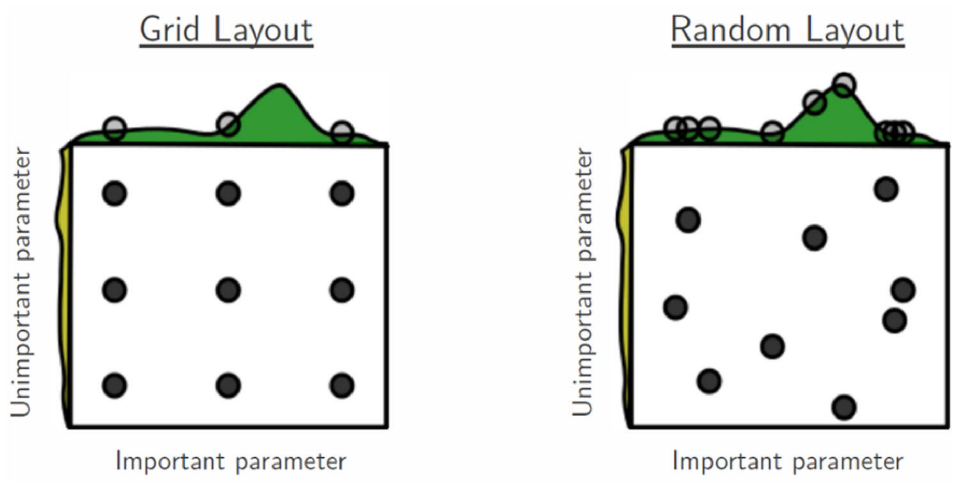


Figure 2. Random and grid search optimizing 9 trials. With grid search there are only three distinct places to test the 9 trials. Where as random search explores all 9 trials.

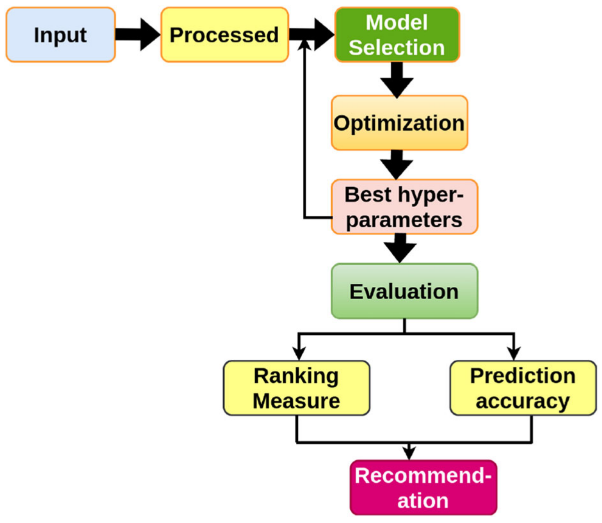


Figure 3. Framework of proposed model: In the first stage, model search the best hyper parameters. In the second stage it evaluates and recommend using the optimal parameters.

$$\theta^* = \underset{\theta}{argmin}\, L(p_u^v, q_i^v, r_{ui}^v, M(p_u^{tr}, q_i^{tr}, r_{ui}^{tr}, \theta)) \tag{6}$$

The performance of CRS on a set of hyperparameters mentioned in table 2 is evaluated by the objective function defined in equation 7.

$$L(p_u^t, q_i^t, r_{ui}^t, M(p_u^{tr+v}, q_i^{tr+v}, r_{ui}^{tr+v}, \theta^*)) \tag{7}$$

Where L and M represent the cost and model function respectively; p_u^{tr} and q_i^{tr} are the user and item latent features of training data respectively. p_u^v and q_i^v are the validation feature vectors of user and item and p_u^t and q_i^t are the feature vectors of test data whereas θ is the set of hyperparameters.

The pseudo-code of grid search is shown in algorithm 5. In the second stage, these optimal sets of hyper-parameters are used for finding the prediction score. We named these optimized techniques GSVD for grid search SVD and GPMF for grid search PMF.

Table 2. Hyper parameters for GSVD and GPMF.

Hyper parameters	Description	Values
$n_{factors}$	# of the latent space	[10,100]
n_{epochs}	# of Iterations	[20,100]
lr_{all}	Learning rate	[0.001,0.005]
$\lambda_{r_{all}}$	Regularization to prevent over-fitting	[0.01,0.05]

Algorithm 1: Evaluation Procedure of Grid search Optimization

```
Input: Dataset (D); Algorithm (A); Θ (hyper parameter
          Space), GS (Grid Search)
Output: Optimized Parameters
res← { };
for i to N do
        θ ← select hyperparameter(GS, A, Θ)
        Model← train(A,θ,D train)
        res ← eval(Model,D test)
end
Θ ← Ad just(Θ, eval(res))
```

A grid of 4×4 search space is designed, in which the grid takes the hyperparameters: learning rate (lr_{all}), number of factors ($n_{factors}$), regularization term ($\lambda_{r_{all}}$), and several epochs (n_{epochs}) for GSVD and GPMF algorithms. Table 2 describes details of hyper-parameters.

In order to minimize the error, the loss function defined in Equation 1 is to learn the feature matrices of both users and items. The Two most popular derivative techniques:

SGD (Stochastic gradient descent) and ALS (Alternating Least Square), are used to handle the optimization problem along with hyper-parameter optimization.

5.1 *Stochastic gradient descent (SGD)*

SGD initializes both item and user feature vectors with random values and computes the cost function of the gradient. It also updates each feature with steps in the negative direction of the gradient [1, 9]. A biased SGD [30] is used in this work so that prediction accuracy is improved. Taking the derivatives w.r.t each variable and adding bias terms for both user and item (because some users might rate all movies high or rate consistently low), the prediction is defined in Equation 8.

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T \cdot q_i \tag{8}$$

Where μ :global bias, $b_i, (b_u)$: item and user bias respectively. The error function defined in Equation 1 is redefined as in Equation 9.

$$\begin{aligned} L = & \sum_{ui}(r_{ui} - \hat{r}_{ui})^2 + \lambda_{r_{ub}} \sum_u \|b_u\|^2 + \lambda_{r_{ib}} \sum_i \|b_i\|^2 \\ & + \lambda_{p_{uk}} \sum_u \|p_u\|^2 + \lambda_{q_{ki}} \sum_i \|q_i\|^2 \end{aligned} \tag{9}$$

Here two more bias regularization terms are added to overcome the over-fitting, then we update each feature. For example, user bias is updated as shown in the Equation 10.

$$b_u^{new} = b_u - \eta \frac{\partial L}{\partial b_u} \tag{10}$$

Where η is the learning rate, after taking the derivative of cost function defined in Equation 9 w.r.t b_u, the updated b_u is shown in Equation 11.

$$b_u^{new} = b_u + \eta(e_{ui} - \lambda_{r_{ub}} b_u) \tag{11}$$

Similarly, the remaining features are updated and are shown in the Equations 12, 13, and 14, respectively.

$$b_i^{new} = b_i + \eta(e_{ui} - \lambda_{r_{ib}} b_i) \tag{12}$$

$$p_u^{new} = p_u + \eta(e_{ui} q_i - \lambda_{r_{uk}} p_u) \tag{13}$$

$$q_i^{new} = q_i + \eta(e_{ui} p_u - \lambda_{r_{ki}} q_i) \tag{14}$$

Where e_{ui} denotes the prediction error. In this technique we need to tune two hyper parameters: η and λ_r, that is learning rate and regularization terms.

5.2 *Alternating least squares(ALS)*

ALS initializes user and item feature vectors with a random values and updates these features using ALS in such way that the quadratic cost/loss function [21], as shown in Equation 1 is minimized. However, an ALS approach's computational cost is more expensive than SGD, whereas ALS requires fewer iterations to obtain similar prediction accuracy as SGD. Furthermore, ALS requires fewer hyper-parameter settings to minimize the loss function. In ALS, we must keep one set of latent vectors constant and compute the derivatives w.r.t other vectors then equate to zero. The user and item feature vectors are represented in Equations 15 and 16, respectively.

$$p_{uk} = r_{ui} q_{ik} (q_{ki} q_{ik}^T + \lambda_{p_{uk}} I_{kk})^{-1} \tag{15}$$

$$q_{ik} = r_{ik} p_{uk} (p_{uk} p_{ku}^T + \lambda_{q_{ki}} I_{kk})^{-1} \tag{16}$$

The only hyper-parameter to be tunned in ALS is λ that is regularization term to prevent overfitting.

6. Experimental evaluation

In this Section, we have discussed the experimental evaluation process and analyze the results over all datasets and make a comparison with the state-of-the-art techniques.

6.1 *Datasets*

We have used two benchmark datasets for evaluation purposes in this work, namely movielens-1M (ML-1M) [1] and 100K (ML-100K) [2]. The ML-100K corpora contains $100K$ ratings given by the 943 users to 1682 movies, and at least each user has interacted with 20 movies. In contrast, the ML-1M corpora contains 1000209 ratings for 3900 movies given by 6040 users. The detailed description of datasets is shown in table 3.

6.2 *Evaluation metrics*

The prediction accuracy of a Recsys indicates the percentages of deviation from the actual prediction. RecSys, predicts user preferences based on predictive models that retrieve the prediction score of unseen user-item pairs. The score lies within the interval [0, 1]. A high score signifies that the user is more likely to buy the product. In our work, we have used evaluation measures, such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) [31]. RMSE is defined in Equation 17. Also, we investigate top-k recommendations by computing ranking metrics like normal discounted cumulative gain (NDCG), mean average precision (MAP), precision@k, and recall@k here; we have considered $k = 10$, that is top-10 recommendations.

$$RMSE = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(r_i^a - r_i^p)^2} \tag{17}$$

Where r_i^a: is the actual rating value and r_i^p is the predicted

[1]http://grouplens.org/movilens/ml-1m.zip.

[2]http://grouplens.org/movielens/ml-100k.zip.

Table 3. Detail description of experimental datasets.

Datasets	# of users	# of items	# of interaction	% of Sparsity
ML-100K	943	1682	100,000	93.70%
ML-1M	6040	3952	1,000,209	95.81%

rating value. The lower the prediction error indicates, the better or accurate the predictive model. Similarly, MAE is defined in Equation 18.

$$MAE = \frac{1}{T}\sum_{i=1}^{T}\left|r_i^a - r_i^p\right| \tag{18}$$

precision@k and recall@k are defined in Equations 19 and 20, respectively, where precision@k indicates the percentage of items recommended for top-k recommendations. precision@k and recall@k are defined in Equations 19 and 20 respectively. Where precision@k indicates the percentage of items recommended for top-k recommendations.

$$Precision@k = \frac{\#Relavant\ items\ at\ top\ k}{\#Retrived\ item} \tag{19}$$

$$Recall@k = \frac{\#Relevant\ items\ at\ top\ k}{\#Relevant\ item} \tag{20}$$

NDCG and MAP [32] are defined in Equations 21 and 23 respectively.

$$NDCG = \frac{Actual\ DCG}{Ideal\ DCG} \tag{21}$$

DCG (Discounted cumulative gain) for a ranking document is the total accumulated gain at a particular rank p and defined in Equation 22.

$$DCG_p = rel_1 + \sum_{i=2}^{p}\frac{rel_i}{\log(i)} \tag{22}$$

Where rel_1 is the first relevant item, $\frac{1}{\log(i)}$ is the discount at rank i and p represents p^{th} rank.

$$MAP = \frac{1}{P}\sum_{r=1}^{p} AveragePrecision_r \tag{23}$$

7. Result analysis

In this section, we explored the results. We split the datasets randomly into 80% and 20% as train-test set respectively and evaluate the experiment with the optimal set of hyper-parameters.

7.1 *Impact of latent factor*

In order to examine the proposed approach and compare with the baseline methods such as SVD [33], PMF [34], and ALSWR (weighted alternating least squares) [21] along with random search techniques [24] of (RSVD, RPMF) in-depth under different K (latent factors) varying from 20 to 100. Table 4 shows the percentage of deviation of loss in terms of RMSE of the proposed method, i.e., GSVD and GPMF for ML-100K dataset, and found that our method has reduced loss at least by 0.002 from the state-of-the-art method. Similarly, compared to other methods, our method reduced loss by 1.4%, 2.8%, and 4.6% than SVD, PMF, and ALS, respectively. Prediction error in terms of MAE, our approach produces a lower prediction error than the baseline method. That is, loss or prediction error is reduced at least by 0.002 value than RSVD and reduced the prediction error of 1%, 2%, and 3% than SVD, PMF, ALS, respectively.

Table 5 represents the comparison of top-k recommendations of the proposed method with baseline algorithms and found that the proposed technique improves the precision@k, at least 3.8% from SVD, 2.1% from PMF, 6.4% from ALS, and 4.8% from RSVD. Furthermore, the NDCG of the proposed method significantly improves over the benchmark methods at least by 2% on the ML-100K dataset.

Table 6 shows the percentage of deviation of loss or prediction error in terms of RMSE of the proposed approach and other benchmark methods on the ML-1M dataset. It is found that the proposed technique has obtained a lower prediction error than benchmark methods. The prediction error is reduced at least by 0.002, 0.023, and 0.033 values from RSVD, SVD, and PMF, respectively, on ML-1M. Similarly, GSVD and GPMF have produced a lower prediction error in MAE than the baseline method, at least by 0.003 to the RSVD. In contrast, prediction error of the proposed method is reduced by 0.025, 0.026, and 0.045 values than SVD, PMF, and ALS, respectively.

Further Table 7 shows a comparison of the top-k recommendation of the proposed method and benchmark algorithms and found that the proposed technique improves the precision@k at least by 0.019 from SVD, 0.009 from PMF. Similarly, the NDCG of our model is significantly improved over the benchmark methods, at least by 0.009 value from the benchmark methods on the ML-1M dataset.

Figure 4a shows prediction error in terms of RMSE of the proposed method decline with an increase of latent

Table 4. Comparison of accuracy for 100*K*.

Metric	SVD	PMF	NMF	ALSWR	RSVD	RPMF	GSVD	GPMF
RMSE	0.9353	0.9491	0.9755	0.9671	0.9214	0.9351	**0.9209**	**0.9287**
MAE	0.7364	0.7423	0.7671	0.7583	0.7245	0.7402	**0.7224**	**0.7350**

Table 5. Comparison of top *k* recommendation for 100*K*.

Metric	SVD	PMF	ALSWR	RSVD	RPMF	GSVD	GPMF
Precision@k	0.0781	0.0945	0.0519	0.0758	0.0986	**0.0828**	**0.1161**
Recall@k	0.0313	0.0453	0.0175	0.0334	0.0471	**0.0361**	**0.0557**
NDCG@k	0.0852	0.1063	0.0474	0.0845	0.1102	**0.0908**	**0.1319**
MAP@k	0.0126	0.0193	0.0057	0.0134	0.0197	**0.0146**	**0.0253**

Table 6. Comparison of accuracy on $ML-1M$ dataset.

Metric	SVD	PMF	NMF	ALSWR	RSVD	RPMF	GSVD	GPMF
RMSE	0.8760	0.8855	0.9201	0.8866	0.8554	0.8568	**0.8525**	0.8557
MAE	0.6964	0.6974	0.7267	0.7167	0.6743	0.6791	**0.6711**	0.6755

Table 7. Comparison of top *k* recommendation on $ML-1M$ dataset

Metric	SVD	PMF	ALSWR	RSVD	RPMF	GSVD	GPMF
Precision@k	0.0816	0.0921	0.0136	0.0830	0.1018	0.0846	0.1011
Recall@k	0.0271	0.0326	0.0058	0.0298	0.0376	0.0302	0.0380
NDCG@k	0.0916	0.1030	0.0155	0.0925	0.1133	0.0935	0.1124
MAP@k	0.0114	0.0142	0.0024	0.0125	0.0163	0.0126	0.0165

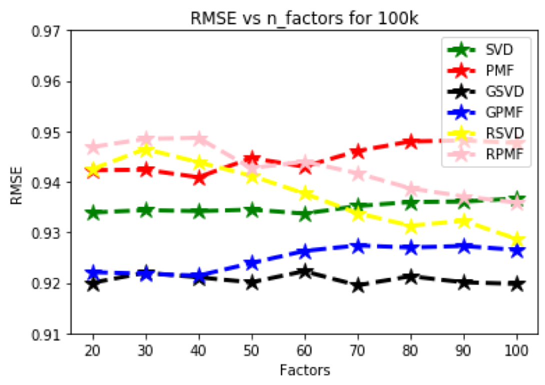
factor and decrease by 0.002 than second the best method (RSVD). Figure 4b shows that MAE of GSVD and GPMF are declined with the increase of *K* on the ML-100K dataset. Figure 4a shows prediction error in terms of RMSE of the proposed method decline with an increase of latent factor and decrease by 0.002 than second the best method (RSVD). Figure 4b shows that MAE of GSVD and GPMF is declined with the increase of *K* for ML-100K.

Figure 5a shows a loss in terms of RMSE of different methods on the ML-1M dataset and found that the proposed method decreases prediction error at least by 0.0029 than the second-best method. Figure 5b shows that loss in terms of MAE of GSVD and GPMF declined with the increase of *K* that is reduced at least 0.003 MAE value than a second-best method, which indicates our approach is better than the benchmark. Figures 4 and 5 show the performance of 100K and 1M datasets under different *K* and found that with an increase of *K*, the proposed method declined and maintained stability with lower MAE and RMSE than benchmark methods on both datasets.

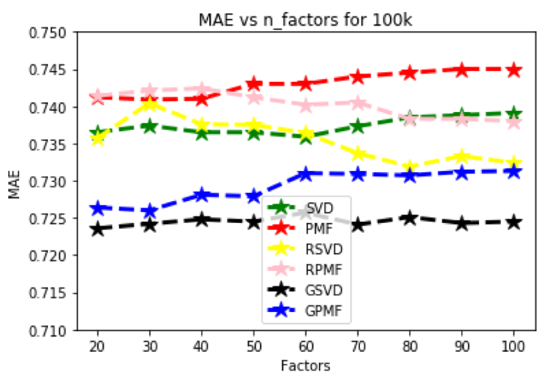
7.2 *Impact of sparsity*

The performance of recommender systems is strongly influenced by lack of sparsity [20, 35–38]. We conduct several experiments by varying the training ratio (α) from 50% to 80% and seeing the effect of sparsity on the performance of recommender models.

Tables 8 and 9 represent the experimental outcome on ML-100K and ML-1M corpora, respectively. It can be noted that the performance of the recommender model is greatly affected due to the sparseness of datasets. Table 8 depicts the results of the ML-100K dataset. It is found that the RMSE of the proposed model is smaller as compared with other models when α varies from 50% to 80%. Even though the proposed approach offers a small improvement

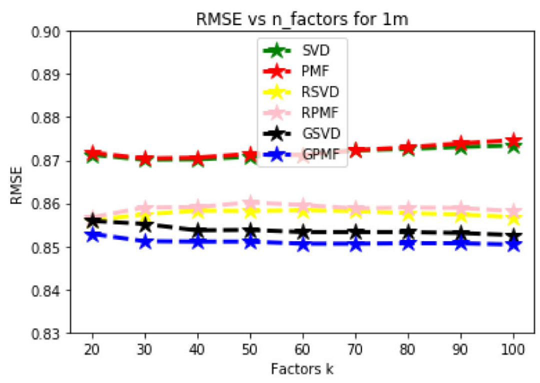


(a) RMSE vs n_ factors on 100K dataset

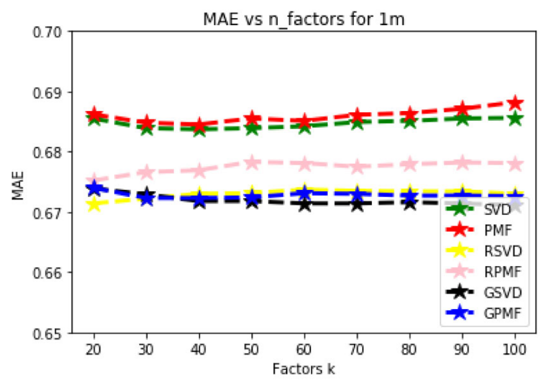


(b) MAE vs n_ factors on 100K dataset

Figure 4. Performance comparison of various models under different latent factors (K) on $ML - 100K$ dataset.



(a) RMSE vs n_ factors on 1M dataset



(b) MAE vs n_ factors on 1M dataset

Figure 5. Performance comparison of various models under different latent factors (K) on $ML - 1M$ dataset.

Table 8. Performance of different model on ML-100K dataset under different sparsity ratio.

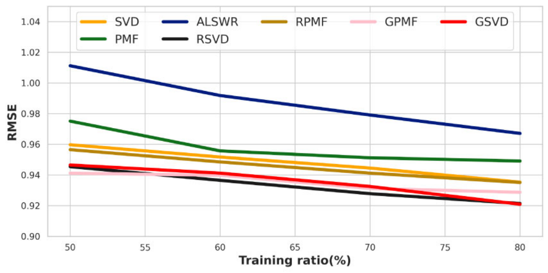
Metric	α	SVD	PMF	ALSWR	RSVD	RPMF	GSVD	GPMF
RMSE	50	0.9597	0.9751	1.0112	0.9454	0.9565	0.9465	0.9412
	60	0.9517	0.9557	0.9918	0.9365	0.9485	0.9412	0.9398
	70	0.9445	0.9512	0.9791	0.9278	0.9412	0.9325	0.9312
	80	0.9353	0.9491	0.9671	0.9214	0.9351	0.9209	0.9287
MAE	50	0.7642	0.7811	0.7709	0.7624	0.7551	0.7536	0.7542
	60	0.7501	0.7612	0.7725	0.7425	0.7565	0.7422	0.7545
	70	0.7412	0.7521	0.7635	0.7344	0.7498	0.7331	0.7426
	80	0.7364	0.7423	0.7583	0.7245	0.7402	0.7224	0.7350

of performance, it is still substantially better than benchmark models. Table 9, it is seen that our model outperforms overall methods under different sparsity levels. When $\alpha = 50\%$, the RMSE of the proposed model is 0.008 smaller than ALSWR and 0.0221 smaller than that of PMF. Similarly, the MAE of the proposed model is improved significantly with the increase of α overall methods.

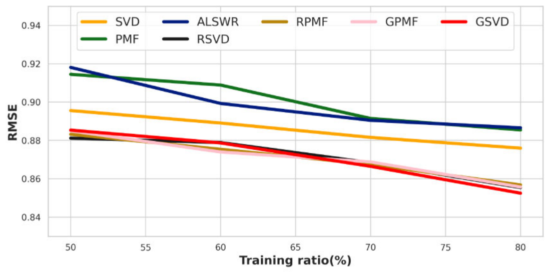
Figure 6a and b describes the influence of the different sparsity ratio on the performance of the models on both datasets. We can see that with an increase in training ratio, the performance of the model is improved.

Table 9. Performance of different model on ML-1M dataset under different sparsity ratio.

Metric	α	SVD	PMF	ALSWR	RSVD	RPMF	GSVD	GPMF
RMSE	50	0.8956	0.9145	0.9181	0.8812	0.8832	0.8854	0.8925
	60	0.8891	0.9089	0.8993	0.8789	0.8754	0.8787	0.8898
	70	0.8816	0.8905	0.8915	0.8684	0.8754	0.8665	0.8678
	80	0.8760	0.8855	0.8866	0.8554	0.8568	0.8525	0.8557
MAE	50	0.7334	0.7394	0.7287	0.7085	0.7098	0.7002	0.7058
	60	0.7212	0.7201	0.7298	0.6935	0.6991	0.6945	0.6912
	70	0.7012	0.7056	0.7235	0.6854	0.6881	0.6821	0.6811
	80	0.6964	0.6974	0.7164	0.6743	0.6791	0.6711	0.6755



(a) RMSE vs training ratio on 100K dataset



(b) RMSE vs training ratio on 1M dataset

Figure 6. Performance of the models under different training ratio on both datasets.

8. Conclusions and future work

In our work, we designed a grid search optimization procedure for a collaborative recommender system to improve the performance of CF. Specifically, we obtained the optimal set of hyper-parameters through GSO and then used these hyper-parameters in the next stage of the model to improve prediction accuracy. We found that the optimal values are $lr_{all} = 0.005$, $\lambda_{all} = 0.002$, $n_{factors} = 100$, and $n_{epochs} = 100$, respectively. The experimental outcomes indicate that the proposed approach (GSVD and GPMF) has lower prediction errors than benchmark methods (e.g., SVD, PMF, ALS, RSVD, and RPMF). Further, we investigated top-k recommendations and found that precision and NDCG are significantly improved over benchmark methods. As latent factors control the dimension of latent space, that is, the size of user vector p_u and item vector q_i, which affects prediction quality. Hence, we investigated the impact of latent factors on prediction error. We found that the proposed method's prediction error (RMSE and MAE) declines gradually with the increase of K from 20 to 100 on both datasets and obtains less prediction error than benchmark methods, hence improving recommendation accuracy. In the future, we will extend the work with the help of advanced optimization procedures to improve the accuracy of the recommender system.

References

[1] Yehuda Koren, Robert Bell, and Chris Volinsky 2009 Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37

[2] Seok Kee Lee, Yoon Ho Cho and Soung Hie Kim 2010 Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations. *Information Sciences*, 180(11): 2142–2155

[3] Rubén González Crespo, Oscar Sanjuán Martínez, Juan Manuel Cueva Lovelle, B Cristina Pelayo García-Bustelo, José Emilio Labra Gayo and Patricia Ordoñez De Pablos 2011 Recommendation system based on user interaction data applied to intelligent electronic books. *Computers in Human Behavior*, 27(4): 1445–1449

[4] Kevin McNally, Michael P O'Mahony, Maurice Coyle, Peter Briggs and Barry Smyth 2011 A case study of collaboration and reputation in social web search. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(1): 1–29

[5] Gopal Behera and Neeta Nain 2021 Collaborative recommender system (crs) using optimized sgd-als. In: *International Conference on Advances in Computing and Data Sciences*, pages 627–637. Springer

[6] Fidel Cacheda, Víctor Carneiro, Diego Fernández and Vreixo Formoso 2011 Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web (TWEB)*, 5(1): 1–33

[7] Gopal Behera and Neeta Nain 2022 Trade-off between memory and model-based collaborative filtering recommender system. In: *Proceedings of the International Conference on Paradigms of Communication, Computing and Data Sciences*, pages 137–146. Springer

[8] Gopal Behera and Neeta Nain 2019 Grid search optimization (gso) based future sales prediction for big mart. In: *2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pages 172–178. IEEE

[9] Gábor Takács, István Pilászy, Bottyán Németh and Domonkos Tikk 2009 Scalable collaborative filtering approaches for large recommender systems. *The Journal of Machine Learning Research*, 10: 623–656

[10] Dorin Militaru and Costin Zaharia 2010 A survey of collaborative filtering-based systems for online recommendation. In: *Proceedings of the 12th International Conference on Electronic Commerce: Roadmap for the Future of Electronic Business*, pages 43–47

[11] Greg Linden, Brent Smith and Jeremy York 2003 Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1): 76–80

[12] Badrul Sarwar, George Karypis, Joseph Konstan and John Riedl 2002 Incremental singular value decomposition algorithms for highly scalable recommender systems. In: *Fifth international conference on computer and information science*, volume 1, pages 27–8. Citeseer

[13] Hastagiri P Vanchinathan, Isidor Nikolic, Fabio De Bona and Andreas Krause 2014 Explore-exploit in top-n recommender systems via gaussian processes. In: *Proceedings of the 8th ACM Conference on Recommender systems*, pages 225–232

[14] David Goldberg, David Nichols, Brian M Oki and Douglas Terry 1992 Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12): 61–70

[15] Mukund Deshpande and George Karypis 2004 Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1): 143–177

[16] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz and Qiang Yang 2008 One-class collaborative filtering. In: *2008 Eighth IEEE International Conference on Data Mining*, pages 502–511. IEEE

[17] Yifan Hu, Yehuda Koren and Chris Volinsky 2008 Collaborative filtering for implicit feedback datasets. In: *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272

[18] Thomas Hofmann 2004 Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1): 89–115

[19] Zhipeng Wu, Hui Tian, Xuzhen Zhu and Shuo Wang 2018 Optimization matrix factorization recommendation algorithm based on rating centrality. In: *International Conference on Data Mining and Big Data*, pages 114–125. Springer

[20] Guangxiang Zeng, Hengshu Zhu, Qi Liu, Ping Luo, Enhong Chen and Tong Zhang 2015 Matrix factorization with scale-invariant parameters. In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*

[21] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber and Rong Pan 2008 Large-scale parallel collaborative filtering for the netflix prize. In: *International conference on algorithmic applications in management*, pages 337–348. Springer

[22] Peter M Rasmussen, Lars K Hansen, Kristoffer H Madsen, Nathan W Churchill and Stephen C Strother 2012 Model sparsity and brain pattern interpretation of classification models in neuroimaging. *Pattern Recognition*, 45(6): 2085–2100

[23] Claus Weihs, Karsten Luebke and Irina Czogiel 2006 Response surface methodology for optimizing hyper parameters. Technical Report

[24] James Bergstra and Yoshua Bengio 2012 Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1): 281–305

[25] Donald R Jones, Matthias Schonlau and William J Welch 1998 Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4): 455–492

[26] Peter I Frazier 2018 A tutorial on bayesian optimization. *arXiv preprint*arXiv:1807.02811

[27] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams and Nando De Freitas 2015 Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175

[28] Bruno Giovanni Galuzzi, Ilaria Giordani, Antonio Candelieri, Riccardo Perego and Francesco Archetti 2019 Bayesian optimization for recommender system. In: *World Congress on Global Optimization*, pages 751–760. Springer

[29] I Dewancker, M McCourt and S Clark 2016 Bayesian optimization for machine learning: a practical guidebook, *arXiv preprint*arXiv:1612.04858

[30] Yehuda Koren 2008 Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434

[31] Gopal Behera and Neeta Nain 2019 A comparative study of big mart sales prediction. In: *International Conference on Computer Vision and Image Processing*, pages 421–432. Springer

[32] Guy Shani and Asela Gunawardana 2011 Evaluating recommendation systems. In: *Recommender systems handbook*, pages 257–297. Springer

[33] Shuai Zhang, Lina Yao and Xiwei Xu 2017 Autosvd++ an efficient hybrid collaborative filtering model via contractive auto-encoders. In: *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 957–960

[34] Andriy Mnih and Russ R Salakhutdinov 2008 Probabilistic matrix factorization. In: *Advances in neural information processing systems*, pages 1257–1264

[35] C Selvi and E Sivasankar 2018 A novel similarity measure towards effective recommendation using matusita coefficient for collaborative filtering in a sparse dataset. *Sādhanā*, 43(12):1–13, 2018

[36] Miha Grčar, Dunja Mladenič, Blaž Fortuna and Marko Grobelnik 2005 Data sparsity issues in the collaborative filtering framework. In: *International workshop on knowledge discovery on the web*, pages 58–76. Springer

[37] Gopal Behera and Neeta Nain 2022 Deepnnmf: deep nonlinear non-negative matrix factorization to address sparsity problem of collaborative recommender system. *International Journal of Information Technology*, pages 1–9

[38] Gopal Behera and Neeta Nain 2022 Handling data sparsity via item metadata embedding into deep collaborative recommender system. *Journal of King Saud University-Computer and Information Sciences*