Knowledge and Information Systems (2022) 64:1–34
https://doi.org/10.1007/s10115-021-01629-6

SURVEY PAPER

A review on matrix completion for recommender systems

Zhaoliang Chen[1,2] · Shiping Wang[1,2]

Received: 1 February 2021 / Revised: 9 November 2021 / Accepted: 12 November 2021 /
Published online: 16 January 2022

Abstract
Recommender systems that predict the preference of users have attracted more and more attention in decades. One of the most popular methods in this field is collaborative filtering, which employs explicit or implicit feedback to model the user–item connections. Most methods of collaborative filtering are based on matrix completion techniques which recover the missing values of user–item interaction matrices. The low-rank assumption is a critical premise for matrix completion in recommender systems, which speculates that most information in interaction matrices is redundant. Based on this assumption, a large number of methods have been developed, including matrix factorization models, rank optimization models, and frameworks based on neural networks. In this paper, we first provide a brief description of recommender systems based on matrix completion. Next, several classical and state-of-the-art algorithms related to matrix completion for collaborative filtering are introduced, most of which are based on the assumption of low-rank property. Moreover, the performance of these algorithms is evaluated and discussed by conducting substantial experiments on different real-world datasets. Finally, we provide open research issues for future exploration of matrix completion on recommender systems.

Keywords Matrix completion · Recommender systems · Collaborative filtering · Low-rank learning · Matrix factorization

1 Introduction

The primary target of recommender systems is making recommendations for users to meet their needs or tastes based on their past behavior, thereby significantly saving the time for users to find useful information [1–3]. Rating is a typical user explicit feedback that visually reflects how much a user likes a related item. A multitude of ratings that users left on the

✉ Shiping Wang
shipingwangphd@163.com

Zhaoliang Chen
chenzl23@outlook.com

1 College of Computer and Data Science, Fuzhou University, Fuzhou 350116, China

2 Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fuzhou 350116, China

Internet come into being a vast matrix, which is employed by recommender systems to make predictive recommendations. Since the number of items is huge, users tend to rate only a small part of the items, which leads to the sparsity of matrices [4–6]. This problem brings great difficulty in profiling users and items and has been a hot research issue for decades. Huang et al. [7] dealt with this problem by applying an associative retrieval framework and related spreading activation algorithms to explore transitive associations among users through their past feedback. Moshfeghi et al. [8] presented a framework taking item-related emotions and semantic data into concern to handle the sparsity problem in recommender systems. Li et al. [9] proposed a cross-domain framework that transferred user–item rating patterns from a dense auxiliary rating matrix in other domains to a sparse rating matrix in the target domain, so that two datasets in different domains worked together to solve the problem of sparsity. Implicit feedback is another useful information for recommender systems [10–12]. Although this type of feedback is not as clear as the explicit feedback, it is more common on the Internet. For example, click records, shopping carts of shopping sites, and favorites or forwardings of the articles are part of the implicit feedback [13]. Such feedback does not necessarily reflect the preference of users; however, it promotes the modeling of users or items and reduces the sparsity of matrix [14–16].

In general, there are correlations between different items or users, which can be measured by user–item interaction matrices generated from explicit or implicit feedback [17]. Collaborative filtering (CF) [18–21] discovers these correlations in a data-driven manner to make accurate recommendations and solve the sparsity issues. Matrix completion is extensively applied in CF [22–25], which attempts to recover missing values in interaction matrices. There are mainly two types of methods for CF with matrix completion: neighborhood-based models (NBMs) and latent factor models (LFMs). NBMs compute similarities between users to recommend items for a specific user according to ratings from other similar users. It is also possible to recommend items similar to a known favorite item for the user by computing the similarity between items. LFMs focus on profiling features of users and items and then project them into low-dimensional vectors, which are also called latent factors. It is a common method to reap feature matrices by matrix factorization or singular value decomposition (SVD) [26]. In some state-of-the-art methods, latent factors are obtained via neural networks [27]. For example, Cheng et al. [28] proposed wide and deep learning which jointly trained wide linear models and deep neural networks to improve recommender systems. Covington et al. [29] applied deep neural network (DNN) on Youtube website recommendations. He et al. [30] presented the neural network-based collaborative filtering (NCF) to express and generalize matrix factorization.

Low-rank matrix is an essential assumption for matrix completion with LFMs, which considers that most information in interaction matrices is redundant and can be compressed. As shown in Fig. 1, only top 20% singular values record the most information of original matrices in recommender systems. This motivates us to enforce low-rank property on interaction matrices. Existing methods have also proved the effectiveness of low-rank property in solving sparsity issues [31–33]. Many approaches have been investigated for low-rank matrix completion, including low-rank matrix factorization [34–37], nuclear norm heuristic, singular value thresholding (SVT) [38] and robust principal component analysis (Robust PCA) [39,40], etc. In this paper, we provide a review on recommender systems with matrix completion techniques, most of which follow the low-rank assumption. It is noted that although these methods are developed with the low-rank assumption, they do not necessarily minimize the rank of the matrix explicitly. Therefore, the review is divided into several aspects, i.e., matrix factorization models, neural network models and rank minimization models. Only

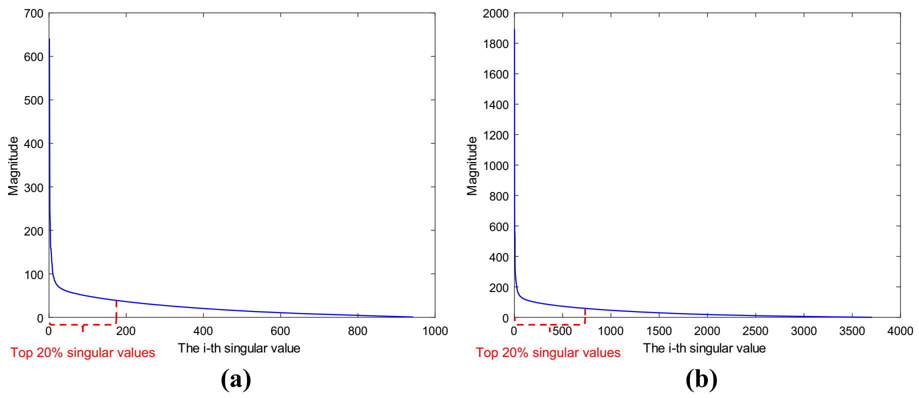

Fig. 1 Singular values of rating matrices generated from two widely used datasets for recommender systems: **a** MovieLens-100K and **b** MovieLens-1M, where top 20% singular values account for 51.10% and 55.97% of the sum of all singular values, respectively

rank minimization methods optimize the rank function explicitly. The main contributions of this paper are listed as follows:

– We look into the matrix completion approaches used in recommender systems, including matrix factorization models, neural network models and rank minimization models, which cover primary techniques of matrix completion-based recommender systems. The advantages and drawbacks of these models are also analyzed.
– Substantial experiments on rating and top-k recommendation prediction that are typical applications of recommender systems are conducted for the selected well-known and state-of-the-art algorithms.
– According to the experimental observation, we analyze and summarize existing models of recommender systems with matrix completion, and discuss the challenges and potential research directions, which may bring insights for readers.

For the rest of this paper, we introduce algorithms for matrix completion methods used in recommender systems in Sect. 2, including matrix factorization models, neural network models and rank minimization models. In Sect. 3, we conduct substantial experiments with these different types of models to compare the performance of various algorithms. Future directions of recommender systems based on matrix completion and insights for researchers are summarized and discussed in Sect. 4.

2 Matrix completion methods

2.1 Problem formulation

In general, models of recommendation systems in the real world can be divided into two types: rating prediction models and top-k recommendation models. Because recommender systems are feedback-driven and most user feedback can be transformed into one or multiple incomplete matrices, optimization methods on matrix completion can address recommender system problems effectively. Consequently, most approaches preprocess the feedback data at the beginning of algorithms so that matrix completion methods can be applied to recommender systems. First of all, we discuss rating tasks in recommender systems which

Table 1 Explanations for commonly used mathematical symbols

Symbols	Explanations
M	The observed incomplete matrix
Ω	The set containing observed user–item pairs
P, Q	Latent factor matrices obtained from matrix factorization of M
p_u	Latent feature vector of user u (the uth row of P)
q_i	Latent feature vector of item i (the ith column of Q)
s_{uv}, s_{ij}	User and item similarities
y_{ui}	Rating value (or other forms of feedback) of user u on item i
$\mathcal{L}$	Loss or objective function
$rank(X)$	The rank of matrix X
$Y = U\Sigma V$	SVD of matrix Y
σ_i	The ith singular value of the matrix

predict missing values of a rating matrix. Given a set of users $u \in \{1, \ldots, m\}$ and items $i \in \{1, \ldots, n\}$, the rating of user u on item i is denoted by y_{ui}. Massive ratings are transformed into a matrix $M \in \mathbb{R}^{m\times n}$ which corresponds to ratings of items rated by users. If all observed user–item pairs are stored in the set $\Omega = \{(u, i)|y_{ui}\ is\ observed\}$, the interaction matrix M is defined by

$$M_{ui} = \begin{cases} y_{ui}, & (u, i) \in \Omega, \\ null, & (u, i) \notin \Omega, \end{cases} \tag{1}$$

or we can replace the unknown ratings with 0, that is,

$$M_{ui} = \begin{cases} y_{ui}, & (u, i) \in \Omega, \\ 0, & (u, i) \notin \Omega. \end{cases} \tag{2}$$

In some top-k recommendation tasks, y_{ui} may be other values ranging in $[0, 1]$ representing affinities of users. Matrix completion methods fill the missing values in a user–item interaction matrix and provide top-k recommendation list according to the predicting values in the matrix. Usually, most entries of the interaction matrix M are unknown because most users only rated a tiny part of items. In this section, we look into some traditional and popular methods for matrix completion in recommender systems. In the beginning of this section, we first provide a table explaining commonly used mathematical symbols for better readability. Other method-specific mathematical symbols, e.g., different regularization coefficients, are explained when they first appear.

2.2 Matrix factorization models

2.2.1 NMF

Matrix factorization is a popular and classical technique of CF for rating problems. It aims to decompose the user–item rating matrix M into the user latent factors and item latent factors that profile users and items accurately. In this subsection, we start from the well-known nonnegative matrix factorization (NMF) algorithm which has proved to be effective for learning a partial representation of the data [34,35]. Given a rating matrix $M \in \mathbb{R}^{m\times n}$,

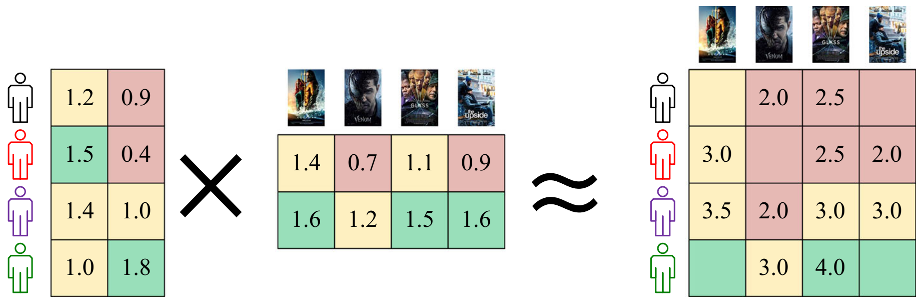

Fig. 2 A simple example for NMF. The incomplete user–item interaction matrix is approximated by the multiplication of two nonnegative matrices, one profiles the user latent features while the other one profiles the item latent features

it considers the problem by finding nonnegative matrices $P \in \mathbb{R}^{m\times r}$ and $Q \in \mathbb{R}^{r\times n}$ that follow $M \approx PQ$, as illustrated in Fig. 2. Usually, the parameter r is set much smaller than $\min(m, n)$ to promise low-rank property, so that the model learns compressed representations from the original matrix.

To measure the quality of the approximation, loss functions are defined to compute the distance between two arbitrary nonnegative matrices A and B. One widely used loss function is

$$\mathcal{L}(A, B) = \sum_{ij} \left(A_{ij} - B_{ij}\right)^2, \tag{3}$$

which computes the square of the Euclidean distance [41]. The gradient descent-based algorithm is applied to update two learnable matrices P and Q [42]. The multiplicative updating rules to minimize this loss function are

$$Q_{\alpha\mu} \leftarrow Q_{\alpha\mu}\frac{(P^T M)_{\alpha\mu}}{(P^T P Q)_{\alpha\mu}}, \quad P_{i\alpha} \leftarrow P_{i\alpha}\frac{(M Q^T)_{i\alpha}}{(P Q Q^T)_{i\alpha}}. \tag{4}$$

Because matrices P and Q are nonnegative, they are explained as user latent factors and item latent factors, respectively. This constraint enhances the interpretability for low-rank matrix factorization, and values of predictive ratings are directly computed by multiplying nonnegative matrices P and Q.

2.2.2 SVD++

Before the introduction to SVD++, we first review some methods for matrix completion which are based on NBMs. An appropriate similarity measure for the item-oriented NBMs is described by

$$s_{ij} = \frac{n_{ij}}{n_{ij} + \lambda_1}\rho_{ij}, \tag{5}$$

where variable n_{ij} represents the number of users who have rated both items i and j. The variable ρ_{ij} is the Pearson correlation coefficient [43] which measures the chances if the user will rate items i and j similarly. Equation (5) can be regarded as a shrunk correlation coefficient controlled by the hyperparameter λ_1. The typical value of λ_1 is 100. Theoretically, if the number of users that have co-rated both items i and j is higher, the value of s_{ij} should be more close to ρ_{ij}. Namely, similarity values computed with more existing ratings are far more convincing. On the contrary, similarity values computed with few ratings should be shrunk considerably. Using this similarity measure, the prediction for an unknown rating is

$$\hat{y}_{ui} = b_{ui} + \frac{\sum_{j \in S^k(i;u)} s_{ij}(y_{uj} - b_{uj})}{\sum_{j \in S^k(i;u)} s_{ij}}, \tag{6}$$

where $b_{ui} = \mu + b_u + b_i$ is the baseline estimation for the unknown rating y_{ui} and explains for the user and item effect (denoted by b_u and b_i). In Eq. (6), $S^k(i; u)$ denotes a set consisting of k items that are most similar to the item i, all of which are rated by the user u. Koren [44] improved the basic neighborhood model by exploiting implicit feedback. If the item set rated by user u is denoted by $R(u)$ and the set containing all items which the user u has provided implicit feedback is denoted by $N(u)$, the improved model is computed with

$$\hat{y}_{ui} = b_{ui} + \left|R^k(i; u)\right|^{-\frac{1}{2}} \sum_{j \in R^k(i;u)} (y_{uj} - b_{uj})w_{ij} + \left|N^k(i; u)\right|^{-\frac{1}{2}} \sum_{j \in N^k(i;u)} c_{ij}, \tag{7}$$

where $R^k(i; u) \overset{def}{=} R(u) \cap S^k(i)$ and $N^k(i; u) \overset{def}{=} N(u) \cap S^k(i)$. $R^k(i; u)$ and $N^k(i; u)$ are sets containing k items most similar to the item i, and all of these items are rated by the user u. The model assumes that users who have provided more ratings should generate more significant deviations from baseline estimations.

The LFM proposed by Koren [44] is named SVD++, which considers the user implicit feedback, as shown in Eq. (8). Similar examples of such model have been proposed in other literature [45,46].

$$\hat{y}_{ui} = b_{ui} + q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right). \tag{8}$$

In Eq. (8), a user u is modeled as $p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j$. The sum $|N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j$ denotes the perspective of implicit feedback. Learnable variables of the model are learned by minimizing the squared error function through the gradient descent-based method.

Koren finally integrated the SVD++ model with the neighborhood model by adding the results of Eqs. (7) and (8) directly:

$$\begin{aligned} \hat{y}_{ui} = \mu + b_u + b_i + q_i^T &\left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) \\ &+ \left|R^k(i; u)\right|^{-\frac{1}{2}} \sum_{j \in R^k(i;u)} (y_{uj} - b_{uj})w_{ij} + \left|N^k(i; u)\right|^{-\frac{1}{2}} \sum_{j \in N^k(i;u)} c_{ij}. \end{aligned} \tag{9}$$

The equation above is a three-tier model for matrix completion of CF. In the first tier, $\mu + b_u + b_i$ is a baseline estimation for y_{ui}, without taking any other interactions into account. In the second tier, term $q_i^T(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j)$ predicts the interactions between the user profile and the item profile. The final tier works as a 'neighborhood tier' which explains the influence of implicit feedback. This three-tier framework significantly improves the accuracy of the matrix completion model by considering both neighbor information and latent embeddings of users or items.

2.2.3 SLIM and FISM

Inspired by the item-based k-nearest neighbor (Item-KNN) method, which is another important model of NBMs, Ning and Karypis [47] proposed a sparse linear model (SLIM) that learns an item–item similarity matrix $S \in \mathbb{R}^{n \times n}$ by solving the optimization problem

$$\arg\min_{S} \frac{1}{2} \|M - MS\|_F^2 + \frac{\beta}{2} \|S\|_F^2 + \lambda \|S\|_1 , \quad s.t. \;\; S \geq 0, \; diag(S) = 0, \tag{10}$$

where matrix S is constrained to be sparse because generally $k \ll n$. Especially, $M \in \mathbb{R}^{m \times n}$ is the binary matrix for implicit feedback. The value of y_{ui} equals one if the user u has provided feedback (such as likes and click history) on item i and 0 otherwise. Though SLIM learns the latent features through S from the existing data, it only discovers relationships between items that have been co-rated, which leads to missing the transitive nature of such relations.

To solve this problem and overcome the sparsity inherent in SLIM, Kabbur et al. [48] provided an improved algorithm dubbed factored item similarity model (FISM) which takes matrix factorization model into concern:

$$\hat{y}_{ui} = b_u + b_i + (n_u^+)^{-\alpha} \sum_{j \in R(u)} p_j q_i^T , \tag{11}$$

where $p_j q_i^T$ computes the similarity between items i and j. Variables p_j and q_i are the corresponding vectors from latent matrices P and Q. The set of items rated by user u is denoted by $R(u)$. Constant n_u^+ is the number of items rated by user u, and α ranges in [0, 1]. Consequently, the term $(n_u^+)^{-\alpha}$ represents the degree of agreement between items rated by the user u according to their similarities.

Kabbur et al. [48] introduced two variations for FISM which are different in optimization targets. For the model computed by the root mean squared error (RMSE), the model aims to address the optimization problem

$$\arg\min_{P,Q} \frac{1}{2} \sum_{u,i \in \Omega} \left\| y_{ui} - \hat{y}_{ui} \right\|_F^2 + \frac{\beta}{2} \left(\|P\|_F^2 + \|Q\|_F^2 \right) + \frac{\lambda}{2} \|b_u\|_2^2 + \frac{\gamma}{2} \|b_i\|_2^2 , \tag{12}$$

where P and Q are updated via stochastic gradient descent (SGD) [49]. For the prediction of the item i, the estimated rating y_{ui} is computed by excluding the current item i. Namely,

$$\hat{y}_{ui} = b_u + b_i + (n_u^+ - 1)^{-\alpha} \sum_{j \in R(u) \setminus \{i\}} p_j q_i^T . \tag{13}$$

FISM takes advantage of SVD-based models to project feedback into latent spaces, and then uses the product of low-rank latent matrices to learn item similarities. This promotes the model to learn transitive relationships implicitly in top-k recommendation tasks. However, the improvement also complicates the computation and requires more time.

2.2.4 LLORMA

For most matrix factorization models, there is an assumption that the observed matrix should be low rank. Because most rating records in the user–item matrix are similar and related in rows or columns, the information within the matrix is redundant. With such an assumption, hidden information is mined by decomposing a matrix into low-rank matrices, as we have discussed before. However, instead of considering the entire matrix M to be low rank, Lee et al. [50] presented a new model with the assumption that M should be a low-rank matrix in the vicinity of certain row-column combinations. The algorithm makes a smooth convex combination of local low-rank matrices, each of which approximates the original matrix M in its local region.

Lee et al. [50] first developed local extensions of incomplete SVD model in the vicinity of $(a, b) \in [m] \times [n]$. The local incomplete SVD model is

$$\hat{\mathcal{T}}(a, b) = \arg\min_{X} \| K_h(a, b) \odot M - X \|_F^2 \quad s.t. \ \ rank(X) = r, \tag{14}$$

where $\mathcal{T}(\cdot, \cdot)$ denotes the operators of form $\mathcal{T} : [m] \times [n] \to \mathbb{R}^{m \times n}$. The rank of optimal X is subjected to r. Here $K_h(a, b)$ is a smoothing kernel parameterized by a bandwidth parameter $h > 0$. Lee et al. [50] applied Epanechnikov kernel in their experiments, as shown below:

$$K_h(s_1, s_2) \propto \left(1 - d(s_1, s_2)^2\right) \mathbf{1}[d(s_1, s_2) < h], \tag{15}$$

where the item similarity or user similarity is applied to measure the distance $d(s_1, s_2)$.

In order to obtain an efficient estimation $\hat{\hat{\mathcal{T}}}(s)$ for all $s \in [m] \times [n]$ with t local models $\hat{\mathcal{T}}(s_1), \cdots, \hat{\mathcal{T}}(s_t)$, $s_t \in [m] \times [n]$, Nadaraya–Watson regression [50,51] is applied to obtain a more precise global approximation by calculating

$$\hat{\hat{\mathcal{T}}}(s) = \sum_{i=1}^{t} \frac{K_h(s_i, s)}{\sum_{j=1}^{t} K_h(s_j, s)} \hat{\mathcal{T}}(s_i), \tag{16}$$

where the sum of weights equals one. Namely, it is a weighted average of $\hat{\mathcal{T}}(s_1), \ldots, \hat{\mathcal{T}}(s_t)$. The weights indicate that $\hat{\mathcal{T}}$ close to s at indices is more important than that further away from s. The final matrix approximation is conducted by $\hat{\hat{Y}}_{ab} = \hat{\hat{\mathcal{T}}}_{ab}(a, b), \quad (a, b) \in [m] \times [n]$.

As for the choosing of $s_1, \ldots, s_t$, anchor points are sampled from the whole set $[m] \times [n]$ or training set evenly. With the increase in the number of local models t and the degree of continuity of $\hat{\mathcal{T}}$, the accuracy of $\hat{\hat{\mathcal{T}}}$ improves. Accordingly, the accuracy of the local models $\hat{\mathcal{T}}(s_1), \ldots, \hat{\mathcal{T}}(s_t)$ directly controls the accuracy of global estimation $\hat{\hat{\mathcal{T}}}_{a,b}$. If $\hat{\mathcal{T}}(s_1), \ldots, \hat{\mathcal{T}}(s_t)$ are precise enough and t is large enough, the prediction error between $\hat{\hat{\mathcal{T}}}$ and $\mathcal{T}$ should be small [41]. This algorithm is termed as the local low-rank matrix approximation (LLORMA) model.

In addition, because the t iterations of this algorithm are independent from each other, they can be computed parallelly and work with high efficiency. The idea of solving local matrix with lower dimensions also helps speed up the computation. As a result, the time cost of LLORMA is t times solving a single regularized SVD problem. Due to the time-consuming issue of conducting SVD over the whole matrix, it is a common solution to divide the matrix completion problem into several subproblems. Multi-Schatten-p norm surrogate (MSS) also handles rank optimization problems with several subproblems like LLROMA. Both of them accelerate the computation speed, and we will discuss MSS later in Sect. 2.4.3.

2.2.5 Cofactor

Word embedding models [52–54] have been widely investigated in natural language processing and obtained great success. Word2vec [55] developed by Google is one of the well-known models. These algorithms project words or phrases from the real world into low-dimensional vectors which are applied as the inputs of other models. Inspired by word embedding models, Liang et al. [56] proposed the Cofactor to improve the quality of matrix factorization models.

In this model, apart from the user–item matrix, an item co-occurrence matrix across all users is built to learn an item embedding, with the assumption that the pairs of items preferred by different users should be similar. This is very similar to the word embedding models which transform the documents into a set of co-occuring words.

The whole framework is made of two parts: the matrix factorization model and the item embedding model. In the matrix factorization part, the model attempts to factorize the given implicit feedback matrix $M \in \mathbb{R}^{m\times n}$ into the user latent vector $p_u \in \mathbb{R}^r$ $(u = 1, \ldots, m)$ and the item latent vector $q_i \in \mathbb{R}^r$ $(i = 1, \ldots, n)$ with low-rank assumption. The objective function for this model is defined as

$$\mathcal{L}(\hat{y}_{ui}, p_u, q_i) = \sum_{u,i\in\Omega} c_{ui}\left(\hat{y}_{ui} - p_u^T q_i\right)^2 + \lambda_p \sum_{u=1}^{m} \|p_u\|_2^2 + \lambda_q \sum_{i=1}^{n} \|q_i\|_2^2, \tag{17}$$

where c_{ui} is a hyperparameter that is usually set to be $c_{y=1} > c_{y=0}$. This scaling parameter is applied to balance the missing ratings $(y = 0)$ which are far more than the existing ratings $(y = 1)$ in most click-based data. The optimization of $\mathcal{L}$ is considered as maximizing a posteriori estimate of the probabilistic Gaussian matrix factorization model [45,56].

As for the item embedding model, because sequences of items are similar to sequences of words, the idea of word embedding models is analogously applied in the item embedding. For a text document, the context words of word i are surrounding words within a fixed window. Point-wise mutual information (PMI) [57] matrix between a word i and its context word j is defined by

$$PMI(i, j) = log\frac{\#(i, j)\cdot D}{\#(i)\#(j)}, \tag{18}$$

where $\#(i, j)$ represents the frequency that word j appears in the context of word i and D denotes the sum of word-context pairs. Levy and Goldberg [58] have proved the equivalence between skip-gram word2vec trained with negative sampling value of k and implicit decomposing the PMI matrix shifted by $log\ k$. They also recommended implementing the word embedding by spectral dimensionality reduction on the (sparse) shifted positive PMI (SPPMI) matrix defined as follows:

$$SSPMI(i, j) = max\{PMI(i, j) - log\ k, 0\}, \tag{19}$$

where k becomes the hyperparameter controlling the sparsity of SSPMI matrix. In the item embedding model, if matrix H is the co-occurrence SPPMI matrix, the item embedding can be obtained by decomposing H. Given a rated item i from a specific user, its context j is represented as all other items in the click history. The click history refers to items that the user has rated or consumed. The value of h_{ij} is obtained by the empirical estimates of $PMI(i, j)$ defined in Eq. (18), where particularly $\#(i, j)$ is the total number of users that rated both items i and j. The combination of the matrix factorization model and item embedding model is

$$\begin{aligned}\mathcal{L}(\hat{y}_{ui}, p_u, q_i, h_{ij}) = &\sum_{u,i\in\Omega} c_{ui}\left(y_{ui} - p_u^T q_i\right)^2 \\ &+ \sum_{h_{ij}\neq 0}\left(h_{ij} - q_i^T \gamma_j - w_i - c_j\right)^2 \\ &+\lambda_p \sum_{u} \|p_u\|_2^2 + \lambda_q \sum_{i} \|q_i\|_2^2 + \lambda_\gamma \sum_{j} \|\gamma_j\|_2^2.\end{aligned} \tag{20}$$

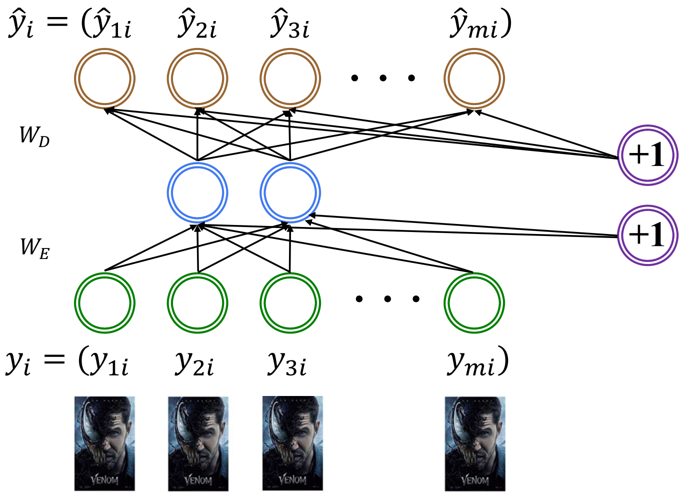
$\hat{y}_i = (\hat{y}_{1i}\ \hat{y}_{2i}\ \hat{y}_{3i} \cdots \hat{y}_{mi})$
W_D
W_E
+1
+1
$y_i = (y_{1i}\ y_{2i}\ y_{3i} \cdots y_{mi})$
VENOM

Fig. 3 The structure of an item-based AutoRec model

In this objective function, the first line is the matrix factorization model and the second line is the item embedding model, the regularization term that avoids overfitting is defined in the third line. Because the matrix factorization model encodes item vectors to represent the latent features of items, while the item embedding has to explain item co-occurrence patterns, matrix factorization part and item embedding part share the same item latent factor q_i in Eq. (20). As a result, q_i explains both user–item interactions and item–item co-occurrence. Besides, γ is introduced as an additional model parameter for PMI matrix factorization. Here, c_{ui} is not only a scaling parameter to balance the observed and unobserved information in the click matrix, but also works on balancing the matrix factorization part and item embedding part of the model.

2.3 Neural network models

2.3.1 AutoRec

The autoencoder is a neural network which aims to learn a representation for a set of data by projecting original data into low-dimensional space and rebuilding it, which achieves desired performance in many fields like natural language processing [59–61] and computer vision [62–64]. Because matrices in recommender systems are generally low-rank, low-dimensional vectors in hidden layers also contain compressed information as latent factors. Therefore, Sedhain et al. [65] proposed a new CF model based on the autoencoder framework, dubbed AutoRec, which works effectively in finding compressed representation for user–item rating matrices.

In this model, each user $u \in \{1, \ldots, m\}$ is represented by an observed vector $y_u = [y_{u1}, \ldots, y_{un}] \in \mathbb{R}^n$. In the same way, each item is represented by an observed vector $y_i = [y_{1i}, \ldots, y_{mi}] \in \mathbb{R}^m$. The AutoRec builds an item-based or user-based autoencoder model which takes each observed y_i or y_u as input, and then compresses the inputs onto low-dimension vectors. Finally, the model reconstructs y_i or y_u in the output layer to predict unknown ratings of the user–item rating matrix. The model is illustrated in Fig. 3.

Given a set of rating vectors $S \in \mathbb{R}^d$, the autoencoder aims to solve the optimization problem

$$\arg\min_{\theta} \sum_{y \in S} \|y - h(y;\theta)\|_2^2, \tag{21}$$

where $h(y;\theta)$ rebuilds the input $y \in \mathbb{R}^d$ from hidden layers with

$$h(y;\theta) = f\left(W_D g(W_E y + \mu) + b\right). \tag{22}$$

Functions $f(\cdot)$ and $g(\cdot)$ are arbitrary activation functions. The parameter set $\theta = \{W_E, W_D, \mu, b\}$ is updated with gradient descent and back propagation. The optimization for item-based AutoRec is defined as

$$\arg\min_{\theta} \sum_{i=1}^{n} \|y_i - h(y_i;\theta)\|_F^2 + \frac{\lambda}{2}\left(\|W_E\|_F^2 + \|W_D\|_F^2\right), \tag{23}$$

where only existing ratings in the training set are considered due to partial observation on ratings. This optimization target corresponds to a neural network with a single hidden layer which has k hidden units. Term $\frac{\lambda}{2}(\|W_E\|_F^2 + \|W_D\|_F^2)$ is the regularization term that avoids overfitting, and $\lambda > 0$ controls the regularization strength. After training, the prediction for y_{ui} is computed by $\hat{y}_{ui} = (h(y_i;\theta))_u$. A main shortcoming of AutoRec is that it only projects data with linear layers. This makes the projection become an identity function, which may lead to inadequate feature learning.

2.3.2 CDAE

AutoRec is a simple but efficient application for matrix completion via autoencoders. On the basis of it, Wu et al. [66] further presented a model for top-k recommender dubbed collaborative denoising autoencoder (CDAE) with denoising autoencoder (DAE) framework. DAE is widely used in many fields [67–69], which learns latent representations from the original and corrupted features of the training set and then trains the model to rebuild the original values.

CDAE model is a neural network with one hidden layer, as shown in Fig. 4. In the input layer, there are n item input nodes and a specific user input node. The user node is a k-dimensional vector that is learned during training. The item nodes $y_u = \{y_{u_1}, \ldots, y_{un}\}$ denote the n-dimensional implicit feedback vector of user u on all items, where $y_{ui} = 1$ if the item i is preferred by the user u in the training set. DAE generates corrupted $\tilde{y}_{ui}$ and then attempts to reconstruct it into the original y_{ui}, for the purpose of making the hidden layer discover more robust features and preventing from merely learning the identity function [70]. Wu et al. [66] applied multiplicative mask-out/drop-out noise to reconstruct each dimension of y_{ui} with 0 via a probability of q, formulated as

$$\begin{aligned} P(\tilde{y}_{ui} = \delta y_{ui}) &= 1 - q, \\ P(\tilde{y}_{ui} = 0) &= q. \end{aligned} \tag{24}$$

In order to make the corruption unbiased, uncorrupted values are recomputed via multiplying original values by $\delta = \frac{1}{(1-q)}$.

In the hidden layer, there are k nodes fully connecting to the nodes of the input layer, as well as an additional node representing the bias effect. $W_E \in \mathbb{R}^{n \times k}$ is a weight matrix between item input nodes and hidden layer. Notice that $V_u \in \mathbb{R}^k$ is a user-specific node in

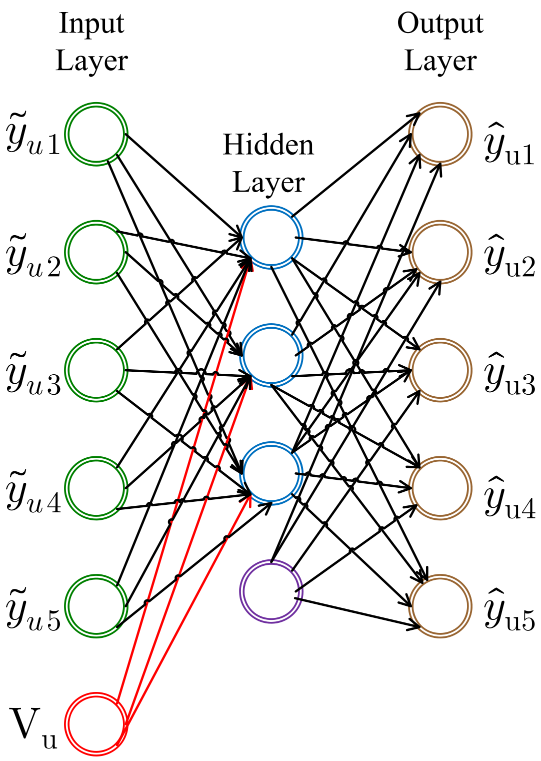

Fig. 4 The structure of CDAE model for a specific user u

the input layer, and each user has their own user vector. In the hidden layer, the model first projects the input into a latent vector z_u by

$$z_u = h\left(W_E^T \tilde{y}_u + V_u + b_E\right), \tag{25}$$

where $h(\cdot)$ is the activation function, e.g., the sigmoid function. The output layer rebuilds the input vector via

$$\hat{y}_u = f\left(W_D^T z_u + b_D\right), \tag{26}$$

where $W_D \in R^{n\times k}$ and b_D are the weight matrix and bias vector between the hidden layer and the output layer. $f(\cdot)$ is also an activation function. Finally, parameters are learned by minimizing the average reconstruction error over all m users:

$$\underset{W_E, W_D, V, b_E, b_D}{\arg\min} \frac{1}{m}\sum_{u=1}^{m} \mathbb{E}_{p(\tilde{y}_u|y_u)}[l(\tilde{y}_u, \hat{y}_u)] + T\left(W_E, W_D, V, b_E, b_D\right), \tag{27}$$

where $T(\cdot)$ is the L_2 regularization that avoids overfitting, as shown below:

$$T(\cdot) = \frac{\lambda}{2}\left(\|W_E\|_2^2 + \|W_D\|_2^2 + \|V\|_2^2 + \|b_E\|_2^2 + \|b_D\|_2^2\right). \tag{28}$$

All trainable parameters are learned by SGD. In order to accelerate the speed of this model, the framework only computes parameters with part of the rating set. If set $R(u)$ is the collection of items in the training set rated by user u and $\bar{R}(u)$ is a set of unrated items for user u, the model samples a subset of negative items $S(u)$ from $\bar{R}(u)$ randomly for parameter updating.

Meanwhile, the model employs AdaGrad [71] to automatically adapt the step size during the training procedure:

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\eta g_\theta^{(t)}}{\sqrt{\beta + \sum_{s=1}^{t} g_\theta^{(s)}}}, \tag{29}$$

where $\theta^{(t)}$ and $g_\theta^{(t)}$ are values of θ and gradient at the tth SGD step, respectively. When the model makes predictions, the first k items with the largest values in the output layer are recommended to the specific user.

2.3.3 DMF

CDAE model mentioned above only illustrates the preference of users by addressing on the implicit feedback. Xue et al. [72] presented a neural network-based model dubbed deep matrix factorization (DMF) for top-k recommendations with both explicit ratings and implicit feedback. DMF is also a matrix factorization model, but it is implemented with deep neural networks. As shown in Fig. 5, there are two parallel multilayer networks to transform the representation of user u and item i, respectively. Both user u and item i are projected onto low-dimensional vectors by

$$\begin{aligned} p_u &= f_{\theta_N^U}\left(\cdots f_{\theta_3^U}\left(W_{U2} f_{\theta_2^U}\left(y_u W_{U1}\right)\right)\cdots\right), \\ q_i &= f_{\theta_N^I}\left(\cdots f_{\theta_3^I}\left(W_{I2} f_{\theta_2^I}\left(y_i W_{I1}\right)\right)\cdots\right), \end{aligned} \tag{30}$$

where W_{Uk} and W_{Ik} are the kth weight matrices for extracting hidden information of users and items. The similarity between the user u and the item i is measured by

$$\hat{y}_{ui} = F^{DMF}(u, i|\theta) = cosine(p_u, q_i) = \frac{p_u^T q_i}{\|p_u\| \, \|q_i\|}. \tag{31}$$

Because the predicted value in Eq. (31) may be negative, mapping $\hat{y}_{ui}^o = max(\mu, \hat{y}_{ui})$ is applied to transform the prediction to a nonnegative value.

A new loss function is employed to consider both explicit and implicit information in this model, so that both two types of feedback are used together for optimization. The new loss function is dubbed normalized cross-entropy (NCE) loss, defined by

$$\mathcal{L}(\hat{y}_{ui}, y_{ui}) = -\sum_{(u,i)\in\Omega}\left(\frac{y_{ui}}{max(R)} log\hat{y}_{ui} + \left(1 - \frac{y_{ui}}{max(R)}\right) log\left(1 - \hat{y}_{ui}\right)\right), \tag{32}$$

where $max(R)$ represents the maximum value among all ratings. If it is a classical 5-star rating recommender, $max(R)$ equals 5. As a result, different values of y_{ij} lead to different influence on the loss function.

2.3.4 Graph-based methods

Owing to the powerful ability of integrating connecting patterns between nodes in the non-Euclidean domain, graph-based methods have achieved significant performance in recent years. Spectral graph convolution is a typical method of graph-based approaches, which has

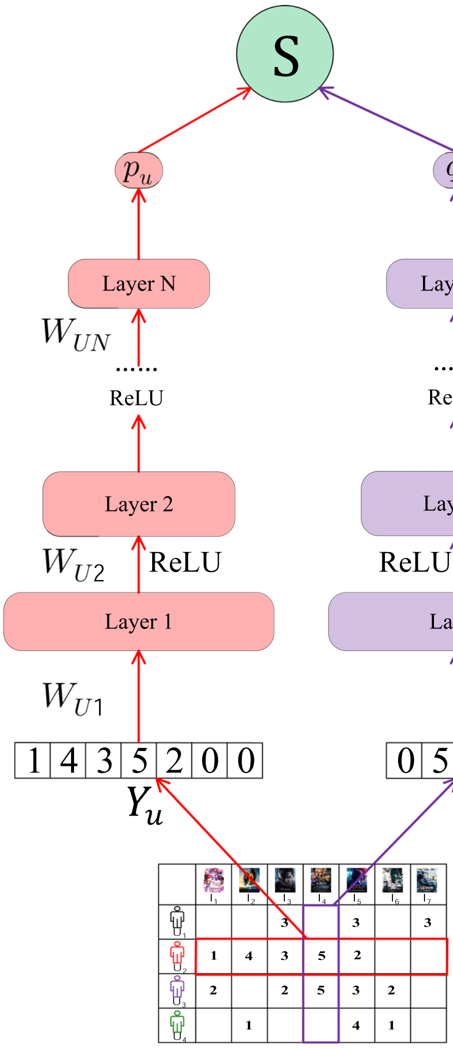

Fig. 5 The basic architecture of the DMF model

been widely applied in recommender systems. As an example, SpectralCF [73] was proposed to address the cold-start problem in recommender systems, exploiting the bipartite user–item relationship graph and a new convolution operation to estimate recommendations in the spectral domain. It can also be regarded as a variant of graph convolutional network (GCN). GCN was developed by Kipf et al. [74] to perform convolution operations on graph-structured data, formulated as

$$H^{(l)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l-1)}W^{(l)}\right), \tag{33}$$

where $\tilde{A} = A + I$ denotes the adjacency matrix considering the self-connections, and $[\tilde{D}]_{ii} = \sum_j [\tilde{A}]_{ij}$. The weight matrix in the lth layer is denoted by $W^{(l)}$. It is a first-order approximation of truncated Chebyshev polynomial, which is deduced from spectral convolutions on graphs, that is,

$$g_\theta \star x = U g_\theta U^\top x, \tag{34}$$

where U denotes the eigenvalue matrix of the normalized graph Laplacian matrix, $x \in \mathbb{R}^m$ is the input feature and $g_\theta = diag(\theta)$ is the parameterized filter. GCN aims to learn the embedding $H^{(l)}$ from network typology, because of which it is extensively applied in recommender systems to recover missing values of an interaction matrix via exploring the latent relationship between users and items. Numerous methods based on GCN have been developed recently. For example, Berg et al. [75] employed graph autoencoders derived from GCN to retrieval the missing values in an incomplete matrix, where the matrix completion task was converted into the link prediction problem on graphs. Monti et al. [76] combined GCN with recurrent neural networks (RNN) to explore the underlying graph-structured patterns between users. Wang et al. [77] presented a neural graph collaborative filtering (NGCF) framework which integrates user–item interactions into the GCN framework and explicitly leverages the collaborative signal.

2.4 Rank minimization models

2.4.1 IRNN

In this section, we take a look into rank minimization models for low-rank optimization problems. Different from the low-rank matrix factorization described before, rank optimization methods focus on minimizing the rank function with

$$\arg\min_M h(M) = f(M) + rank(M), \tag{35}$$

where $f(\cdot)$ is a differentiable loss function and $rank(\cdot)$ is the rank function on the matrix M. For matrix completion problems in recommender systems, $f(\cdot)$ is generally defined as

$$f(M) = \|M_\Omega - M^*_\Omega\|_F^2, \tag{36}$$

where M^*_Ω is the reconstructed matrix. Because $rank(\cdot)$ is exactly the sum of nonzero singular values of the input matrix, it is nondifferentiable. Sometimes, it becomes an NP-hard problem which is difficult to solve. To tackle this problem, we can relax $rank(\cdot)$ to some other surrogate functions $g(\cdot)$ like ℓ_p norm, nuclear norm or Schatten-p norm. Accordingly, Eq. (35) is transformed into

$$\arg\min_M h(M) = f(M) + \lambda g(M), \tag{37}$$

where λ is the regularization coefficient. Generally, the surrogate function $g(\cdot)$ and loss function $f(\cdot)$ should satisfy the following assumptions.

Assumption 1 $g(\cdot)$ is continuous, nonconvex and monotonically increasing on $[0, \infty)$. It is possibly nonsmooth.

Assumption 2 $f(\cdot)$ is a differentiable and smooth function whose gradient is Lipschitz continuous as

$$\|\nabla f(A) - \nabla f(B)\|_F \leq L(f)\, \|A - B\|_F\,, \tag{38}$$

for any $A, B \in \mathbb{R}^{m\times n}$. Here $L(f) > 0$ is the Lipschitz constant. Notice that $f(\cdot)$ is possibly nonconvex.

Assumption 3 $h(M) \to \infty$ if and only if $\|M\|_F \to \infty$, which guarantees the convergence.

Table 2 Several specified nonconvex definitions of functions $g(\theta)$

Norm	Definition $g(\theta)$, $\theta \geq 0$ with $\lambda \geq 0$
ℓ_p-norm [78]	$g(\theta) = \lambda\theta^p, 0 < p < 1$
Logarithm [79]	$g(\theta) = \frac{\lambda}{\log(\gamma+1)} \log(\gamma\theta + 1)$
Geman [80]	$g(\theta) = \frac{\lambda\theta}{\theta+\gamma}$
Laplace [81]	$g(\theta) = \lambda(1 - \exp(-\frac{\theta}{\gamma}))$
ETP [82]	$g(\theta) = \lambda\frac{1-\exp(-\gamma\theta)}{1-\exp(-\gamma)}$

Table 2 shows some specified nonconvex surrogates of $g(\cdot)$. Applying theory of supergradient [83], Lu et al. [84] proposed iteratively reweighted nuclear norm (IRNN) to optimize rank minimization problems with surrogates of $g(\cdot)$. IRNN updates M at the kth iteration by solving the minimization problem

$$M^{k+1} = \arg\min_M f(M) + \sum_{i=1}^{r} w_i^k \sigma_i, \tag{39}$$

where σ_i is the ith singular value of M and w_i^k is the supergradient computed via

$$w_i^k \in \partial g_\lambda(\sigma_i). \tag{40}$$

Because of the antimonotone property of surpergradient, a significant singular value has a smaller weight w_i. The iterative updating rule of the proximal gradient method for solving Equation (39) is derived from

$$\begin{aligned} M^{(k+1)} = & \arg\min_M f(M^{(k)}) + \langle\nabla f(M^{(k)}), M - M^{(k)}\rangle \\ & + \frac{L}{2}\left\|M - M^{(k)}\right\|_F^2 + g(M) \\ = & \arg\min_M \frac{L}{2}\left\|M - M^{(k)} + \frac{1}{L}\nabla f(M^{(k)})\right\|_F^2 + \sum_{i=1}^{r} w_i^k \sigma_i, \end{aligned} \tag{41}$$

where L is the Lipschitz constant. Equation (41) has a closed-form solution given by weighted singular value thresholding

$$M^{k+1} = U\eta_{\lambda\mathbf{w}}(\Sigma)V^T, \tag{42}$$

where $Y = M^{(k)} - \frac{1}{L}\nabla f(M^{(k)})$, $U\Sigma V^T = Y$ is the SVD of Y, and $\eta_{\lambda\mathbf{w}} = diag\{(\Sigma_{ii} - \lambda w_i)_+\}$. Σ is the diagonal singular value matrix, which means that Σ_{ii} denotes the ith singular value of Y. Each iteration of IRNN is a two-step learning scheme that updates w_i^k with Eq. (40) and M^{k+1} with Eq. (41), respectively.

2.4.2 DNNR

Inspired by IRNN and properties of supergradient, Zhang et al. [85] improved IRNN with the weighted singular value function (WSVF), which was formulated as

$$\rho_{\mathbf{w}}(\sigma(M)) = \sum_{i=1}^{r} w_i \rho(\sigma_i), \tag{43}$$

where $\rho(\cdot)$ is a nonconvex and lower semicontinuous function on $[0, \infty)$. The same as IRNN, a lower weight indicates a more significant singular value. When $\rho(\sigma_i)$ is the identity function, Eq. (43) is degraded to $g(\cdot)$ in IRNN. With aforementioned notations, the rank minimization problem can be solved by

$$\arg\min_{M} h(M) = f(M) + \lambda\rho_{\mathbf{w}}(\sigma(M)). \tag{44}$$

Because the optimization target can be derived from

$$\arg\min_{M} h(M) = f(M) + \lambda\sum_{i=1}^{r}\rho_1(\rho(\sigma_i)) \tag{45}$$

with concepts of supergradient and reweighted strategies, where $\rho_1(\cdot) = \rho(\cdot)$, it is also dubbed double nonconvex nonsmooth rank (DNNR) minimization problem. By linearizing Equation (44), the optimal solution is achieved by

$$\arg\min_{M} \frac{1}{2}\|M^k - Y\|_F^2 + \lambda\rho_{\mathbf{w}}(\sigma(M^k)), \tag{46}$$

which has a closed form solution $M^{k+1} = Udiag(\delta^*(Y))V$, termed as WSVF thresholding operator. The ith operator solves the problem with

$$\delta_i^* \in prox_{\rho}(\sigma_i) = \arg\min_{\delta_i \geq 0} \lambda w_i \rho(\delta_i) + \frac{1}{2}(\delta_i - \sigma_i)^2. \tag{47}$$

Existing works [86–88] have derived the closed-form solutions when $\rho(\cdot)$ is the ℓ_p-norm with $p = \frac{1}{2}$ or $p = \frac{2}{3}$. For simplicity, we denote λw_i as ξ, σ_i as σ and δ_i as δ, respectively. When $p = \frac{1}{2}$, we have

$$\delta^* = \begin{cases} \frac{2}{3}\sigma(1 + cos(\frac{2\pi}{3} - \frac{2\phi(\sigma)}{3})), & \sigma > \varphi(\xi), \\ 0, & otherwise, \end{cases} \tag{48}$$

where $\phi(\sigma) = arccos(\xi/4(\sigma/3)^{-3/2})$ and $\varphi(\xi) = 3\sqrt[3]{2}/4(2\xi)^{2/3}$. Similarly, when $p = \frac{2}{3}$, the optimal solution is computed by

$$\delta^* = \begin{cases} ((\varpi + \sqrt{2\sigma/\varpi - \varpi^2})/2)^3, & \sigma > \varphi(\xi), \\ 0, & otherwise, \end{cases} \tag{49}$$

where $\varpi = 2/3^{1/2}(2\xi)^{1/4}cosh(arccosh(27\sigma^2/16(2\xi)^{-3/2})/3)^{1/2}$ and $\varphi(\xi) = 2/3$ $(3(2\xi)^3)^{1/4}$. Analogous to IRNN, the updating rules for DNNR include 2 steps at each iteration. The model first computes the weight w_i^k with supergradient

$$w_i^k \in \partial\rho(\rho(\sigma_i(M^k))), \tag{50}$$

and then updates M^{k+1} by solving optimization problem (46). Distinct from the aforementioned IRNN, DNNR method is more general than IRNN because of double nonconvex constraint functions on singular values. Due to this reason, the updating rules of DNNR for X^{k+1} and w^k are based on the singular value function $\rho(\sigma_i(\cdot))$ instead of depending on $\sigma_i(\cdot)$ directly. However, there is a critical problem that IRNN and DNNR face the time-consuming issue because of conducting SVD on the whole interaction matrix M, which brings difficulties for applying them to large-scale datasets.

2.4.3 MSS

To avoid conducting SVD of the entire matrix and reduce time consumption, Xu et al. [89] proposed a unified convex surrogate for the Schatten-p norm minimization problem, where the optimization problem was decomposed into several subproblems so that SVD could be conducted over a matrix with lower dimensions.

Inspired by low-rank matrix factorization, the low-rank minimization problem defined by Eq. (37) can be rewritten as

$$\arg\min_{U,V} h(U,V) = f(U,V) + \lambda(g(U)+g(V)), \tag{51}$$

where $U \in \mathbb{R}^{m\times d}$ and $V \in \mathbb{R}^{n\times d}$ are the unknown latent factor matrices that $M = UV^T$ holds. Recent related works have attempted to find surrogates for specific p values when $g(X)$ is denoted by the Schatten-p norm

$$\|X\|_{S_p} = \left(\sum_{i=1}^{\min\{m,n\}} \sigma_i(X)^p\right)^{\frac{1}{p}} = \left(Tr((X^TX)^{\frac{p}{2}})\right)^{\frac{1}{p}} \tag{52}$$

for any factors X. It is a well-known unitarily invariant norm. When $p = 1$, Schatten-p norm becomes the widely used nuclear norm or trace norm. Srebro et al. [90] has investigated the bi-Frobenius norm surrogate for the nuclear norm as

$$\|M\|_* = \arg\min_{U,V:M=UV^T} \frac{1}{2}\|U\|_F^2 + \frac{1}{2}\|V\|_F^2. \tag{53}$$

Moreover, Shang et al. [91,92] proved the following equalities when $p = \frac{1}{2}$ and $p = \frac{2}{3}$:

$$\begin{aligned} 2\|M\|_{S_{1/2}}^{1/2} &= \arg\min_{U,V:M=UV^T} \|U\|_* + \|V\|_*, \\ \frac{3}{2}\|M\|_{S_{2/3}}^{2/3} &= \arg\min_{U,V:M=UV^T} \|U\|_* + \frac{1}{2}\|V\|_F^2. \end{aligned} \tag{54}$$

Summarized from these existing work, Xu et al. [89] speculated and proved that the bilinear surrogate for Schatten-p norm could be extended to a more general problem, dubbed multi-Schatten-p norm surrogate (MSS) optimization problem:

$$\frac{1}{p}\|M\|_{S_p}^p = \arg\min_{X_i} \sum_{i=1}^{I} \frac{1}{p_i}\|X_i\|_{S_{p_i}}^{p_i}, \tag{55}$$

where any $p_i > 0$ satisfies $\frac{1}{p} = \sum_{i=1}^{I}\frac{1}{p_i}$, $M = \prod_{i=1}^{I} X_i$ with $X_1 \in \mathbb{R}^{m\times d_1}$, $X_i \in \mathbb{R}^{d_i\times d_i}$, $i = 2,\dots,I-1$ and $X_I \in \mathbb{R}^{d_I\times n}$. The optimization problem for MSS can be solved by block coordinate descent (BCD) [93] which minimizes each X_i at a single iteration by fixing the remaining blocks. Each subproblem is solved by the proximal alternating linearized minimization (PALM) algorithm. Specifically, the proximal gradient method for each factor X_i at each iteration is computed by

$$\begin{aligned} X_i^{(k+1)} &= \arg\min_{X_i} f(X_i^{(k)}) + \langle \nabla f(X_i^{(k)}), X_i - X^{(k)} \rangle \\ &\quad + \frac{L_i^{(k-1)}}{2} \left\| X_i - X_i^{(k)} \right\|_F^2 + \frac{1}{p_i} \|X_i\|_{S_{p_i}}^{p_i} \\ &= \arg\min_{X_i} \frac{L_i^{(k-1)}}{2} \|X_i - Y\|_F^2 + \frac{1}{p_i} \|X_i\|_{S_{p_i}}^{p_i}, \end{aligned} \tag{56}$$

which can be solved by closed-form solutions for a specific p_i value. Furthermore, the acceleration technique [94] is adopted, where $\hat{X}_i^{(k)}$ is updated via

$$\hat{X}_i^{(k)} = X_i^{(k)} + w_i^k (X_i^k - X_i^{k-1}), \tag{57}$$

and w_i^k is computed by

$$w_i^k = \min \left\{ \frac{t_k - 1}{t_k}, 0.9999 \sqrt{\frac{L_i^{k-1}}{L_i^k}} \right\} \tag{58}$$

with $t_1 = 1$ and $t_{k+1} = (1 + \sqrt{1 + 4t_k^2})/2$.

Generally, the learning scheme for MSS of each subproblem is similar to IRNN and DNNR. However, compared with IRNN and DNNR, MSS solves the time-consuming issue by transforming the original problem into subproblems that are easier to solve. This avoids conducting SVD of the whole matrix and further speeds up the computation on large-scale datasets. In addition, rather than only specific to some p values of DNNR ($p = \frac{1}{2}$ and $p = \frac{2}{3}$), the unified model can solve more p values flexibly by considering different combinations of subproblems.

2.4.4 ISVTA

Distinct from the convex and nonconvex rank relaxations we have introduced before, Zhang et al. [95] presented a modified Schatten-p norm as a surrogate of the rank function, denoted as

$$\min_X \left\{ \mathcal{H}_\lambda(X) = \frac{1}{2} \|\mathcal{A}(X) - b\|_F^2 + \lambda \|X\|_{S_{p,\epsilon}}^p \right\}, \tag{59}$$

where the objective $\mathcal{H}_\lambda(X)$ is nonconvex and cannot be optimized directly. However, it can be solved by the linearized strategy or adding more variables (e.g., alternating direction method of multipliers), which may guarantee that each subproblem has the closed-form solution. In order to gain the closed-form solution directly, Zhang et al. optimized the surrogate function of $\mathcal{H}_\lambda(X)$ by adding several quadratic terms, as shown below:

$$\begin{aligned} \min_{X,Y} \Bigg\{ & \mathcal{H}_{\lambda,\mu}(X, Y) \\ &= \mu \left[\frac{1}{2} \|\mathcal{A}(X) - b\|_F^2 + \lambda \sum_i \frac{\sigma_i(X)}{(\sigma_i(Y) + \epsilon_i)^{1-p}} \right] \\ &\quad - \frac{\mu}{2} \|\mathcal{A}(X) - \mathcal{A}(Y)\|_F^2 + \frac{1}{2} \|X - Y\|_F^2 \Bigg\}, \end{aligned} \tag{60}$$

Table 3 Statistics of the real-world datasets in our experiments

	FilmTrust	ML-100K	ML-1M	Netflix	Epinions	Jester
Number of users	1508	943	6040	1500	3586	15,000
Number of items	2071	1682	3706	2000	12,000	150
Number of ratings	35,497	100,000	1,000,209	137,962	81,513	390,772
Rating scale	[0.5, 4.0]	[1.0, 5.0]	[1.0, 5.0]	[1.0, 5.0]	[1.0, 5.0]	[−10.0, 10.0]
Density	0.01137	0.06305	0.04468	0.04599	0.00190	0.17368

which can be optimized more efficiently. Furthermore, Zhang et al. devised the iterative singular value thresholding algorithm (ISVTA) to solve Eq. (60). Each iteration of ISVTA can be summarized as follows:

$$\mathcal{B}_\mu\left(X^k\right) = X^k - \mu\mathcal{A}^*\left(\mathcal{A}\left(X^k\right) - b\right), \tag{61}$$

$$\lambda^{k+1} = \kappa^k\lambda_0 \leq \lambda_t, \quad 0 < \kappa < 1, \tag{62}$$

$$X^{k+1} = \mathcal{G}_{\lambda,\mu}\left(\mathcal{B}_\mu\left(X^k\right)\right) = U^k\mathcal{S}_{\tau w}\left(\Sigma^k_{\mathcal{B}_\mu}\right)\left(V^k\right)^T, \tag{63}$$

where $\Sigma^k_{\mathcal{B}_\mu}$ is the singular values of $\mathcal{B}_\mu\left(X^k\right)$ and the soft thresholding operator is denoted as $\mathcal{S}_{\tau\mathbf{w}}\left(\Sigma^k_{\mathcal{B}_\mu}\right) = \mathrm{Diag}\left\{\left(\Sigma^k_{\mathcal{B}_\mu,ii} - \tau w_i\right)_+\right\}$ for $i = 1, 2, \ldots, r$. In particular, we can set $\tau = \lambda\mu$ and $w_i = \frac{1}{(\sigma_i(X^*)+\epsilon_i)^{1-p}}$ with $0 < p < 1$. In theory, this method can reduce the number of iterations to improve the computational consumption and provide a better global convergence guarantee compared to other methods introduced before [95].

3 Experiments

In this section, we conduct substantial experiments on the models mentioned in Sect. 2 with different real-world datasets. Because some methods are designed specially for top-k recommendations, and some methods focusing on rating prediction are not suitable for top-k tasks, experiments are divided into rating (NMF [34], SVD++ [44], LLORMA [50], AutoRec [65], GCMC [75], sRGCNN [76], IRNN [84], DNNR [85], MSS [89], ISVTA [95]) and ranking (Item-KNN, SLIM [47], FISM [48], CDAE [66], Cofactor [56], DMF [72], SpectralCF [73], NGCF [77]) tasks for a fair comparison. Different evaluation measurements are utilized to compare the performance of different algorithms, as well as the time cost for predicting.

3.1 Datasets description

In this paper, comparing experiments are conducted over several real-world datasets, including movie recommendation datasets, joke rating datasets and shopping recommendation datasets, etc. The textual descriptions of these datasets are listed below:

FilmTrust[1] dataset is a small movie recommender dataset crawled from the FilmTrust website in 2011, which contains 35,497 rating records over 1508 users and 2071 items.

[1] https://www.librec.net/datasets.html.

Movielens[2] is provided by GroupLens Research from the MovieLens website and has many versions of datasets collected at different times. In this paper, MovieLens-100K (ML-100K) and MovieLens-1M (ML-1M) are selected to conduct experiments.

Netflix[3] is a popular online movies and TVs website. Its dataset contains about 100 million ratings and is used in the Netflix Prize competition. We only extract part of the data with 1,500 users and 2,000 items.

Epinions[4] was collected from the Epinions website where people have provided ratings for different types of products. The dataset in our experiments contains 81,513 ratings over 3586 users and 12,000 items.

Jester[5] is a benchmark dataset for joke recommender systems which contains substantial ratings of users on different jokes. Different from other datasets, its ratings range in $[-10.0, 10.0]$.

The detailed statistics of these datasets are shown in Table 3, including dimensions, numbers of ratings, rating scales, and density.

3.2 Performance evaluation

For rating problems, we use root mean squared error (RMSE) and mean absolute error (MAE) [96] to evaluate the performance of recommendation models for rating. Both RMSE and MAE measure the deviation between the observed data and the real data, and smaller values of these two metrics indicate better performance of models. Given t testing entries with their real values $y_1, \ldots, y_t$ and predictive values $\hat{y}_1, \ldots, \hat{y}_t$, Eqs. (64) and (65) are used to compute RMSE and MAE:

$$\text{RMSE} = \sqrt{\frac{1}{t}\sum_{i=1}^{t}(y_i - \hat{y}_i)}, \tag{64}$$

$$\text{MAE} = \frac{1}{t}\sum_{i=1}^{t}\|y_i - \hat{y}_i\|. \tag{65}$$

As for top-k problems, we adopt two metrics designed for ranking known as normalized discounted cumulative gain (NDCG) [97] and RECALL [98] to evaluate the list of k recommended items. NDCG emphasizes the ranks of the estimating results and takes the variation between real ranks and predicted ranks into concern. When an item of high preference for a specific user appears in the high ranking, the value of NDCG would be higher. To explain the definition of NDCG, we need to introduce the discounted cumulative gain (DCG)

$$\text{DCG} = \sum_{i=1}^{k}\frac{2^{rel_i} - 1}{log_2(i+1)}, \tag{66}$$

where rel_i is the relationship that measures the importance of item i. In our experiments, we use the real score of item i to represent rel_i, because a higher rating value indicates a higher preference of a user. Ideal DCG (iDCG) is the ideal value of DCG, which is also computed by Eq. (66). With DCG and iDCG computed, NDCG is computed with $\text{NDCG} = \frac{\text{DCG}}{\text{IDCG}}$.

[2] https://grouplens.org/datasets/movielens/.

[3] https://www.kaggle.com/netflix-inc/netflix-prize-data.

[4] http://www.trustlet.org/wiki/Epinions_dataset.

[5] https://goldberg.berkeley.edu/jester-data/.

RECALL is different from NDCG, which holds the view that all items in the recommended list are equivalent without considering predicted ranking. It is defined by

$$\mathrm{RECALL} = \frac{\#TP}{\#TP + \#FN}, \tag{67}$$

where $\#TP$ is the number of true positive items and $\#FN$ is the number of false negative items.

3.3 Performance comparison

For all algorithms, we follow the settings in original papers or codes if feasible. Some parameter settings that we selected via our substantial experiments are clarified in advance to gain more credible results. Learning rates of all algorithms are selected in $\{0.1, 0.001, 0.005, 0.0001\}$. The dimensions of latent factors for matrix factorization models range in [50, 300] with step size 50. Other specific parameters of some algorithms are listed as follows: **SVD++**: fix regularization coefficients λ_p, λ_q and $\lambda_b = 0.01$; **LLORMA**: fix number of anchor points $t = 55$, set local $\lambda_P, \lambda_Q = 0.01$, global $\lambda_P, \lambda_Q = 0.1$, and apply Epanechnikov kernel with $h_1 = h_2 = 0.8$; **AutoRec**: fix $\lambda = 0.001$; **Item KNN and SLIM**: adopt numbers of KNN neighbors ranging in $[10, 20, \ldots, 100]$. **FISM**: set α and β ranging in [0.1, 1.0] with step size 0.1, and fix $\lambda = \gamma = 0.1$; **CDAE**: set sigmoid as the activation function and set $\lambda = 0.01$; **DMF**: set ReLU as the activation function, and fix $\lambda = 0.001$. The number of hidden layers is set $N = 8$; **Cofactor**: set $\lambda_U = \lambda_V = 0.000001$, the ratio $c_{y=0} = 0.1$ and $c_{y=1} = 1$; **IRNN, DNNR and ISVTA**: initialize $\lambda_0 = \alpha \| P_\Omega(M) \|_\infty$, where α ranges in $[1, 100, 200, \ldots, 1000]$; **MSS**: fix $\eta = 0.1$ and $\lambda = 200$, and the factor number is set 4 or 5 with $p_i = 1, i = 1, \ldots, I$, that is, $p = 0.25$ or 0.2. For top-k recommendation tasks, we set $k = 100$ to generate top 100 recommendation lists.

In order to discover the effect of varying numbers of latent factors, we run comparison experiments for all algorithms based on learning user or item embedding, including AutoRec, CDAE, and all matrix factorization models. We keep the other parameters as constants and then consider the number of factors as a variable. Experiments are conducted over ML-100K and Jester datasets. For AutoRec and CDAE, we define the dimension of the hidden layer as the number of latent factors. Figure 6 records the results of experiments. On the whole, the performance of most algorithms increases as the factors increase, while may decline when the number of factors is too large. From the figures, we find that the performance of FISM is stable, so better recommendations can be obtained with a small computational cost. The performance of AutoRec and CDAE fluctuates greatly in all tested datasets; CDAE especially demands for more factors to rebuild the corrupted data for obtaining a higher value of NDCG. Contrary to other algorithms, Cofactor reaches its best performance when the number of factors is small compared with other models, and performs poorly with too many factors. For rank minimization methods, we compare the performance across different p values, as shown in Fig. 7. Generally, smaller p values lead to lower RMSE for IRNN. On the contrary, MSS gains better performance when p is close to 0. Although DNNR is developed from IRNN, it only works when $p = \frac{1}{2}$ and $\frac{2}{3}$. The best p value for DNNR method is uncertain.

Furthermore, we make comparison over all tested datasets for all selected algorithms. Fivefold cross-validation is applied to all experiments, and we record the average as well as the standard deviation of all metrics. The performance of rating and ranking tasks is listed in Tables 4 to 7. From these tables, we have the following observations: First of all,

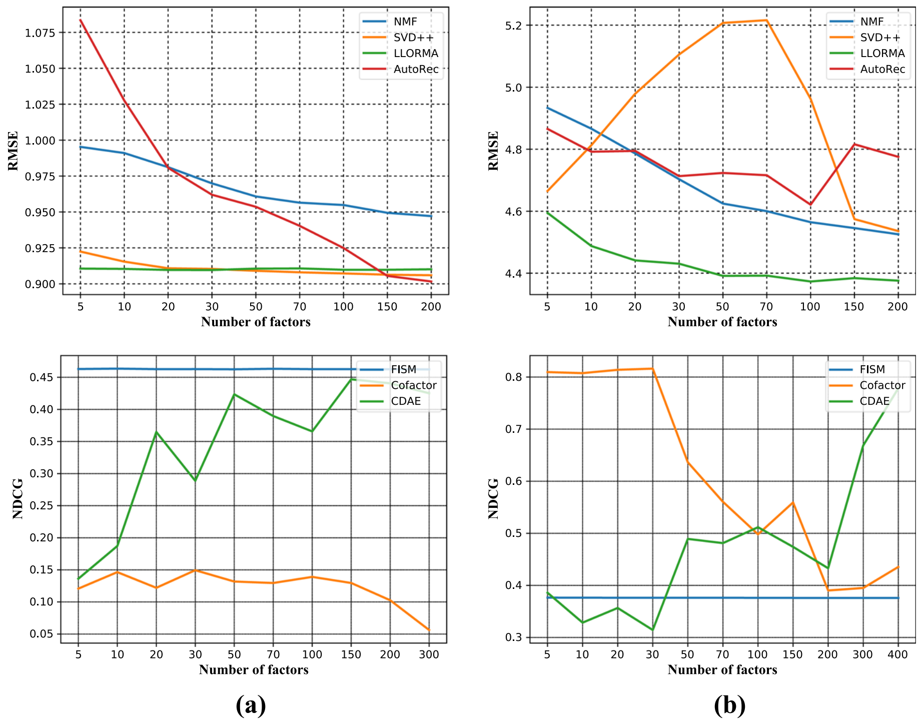

Fig. 6 Effect of factor number on **a** ML-100K and **b** Jester

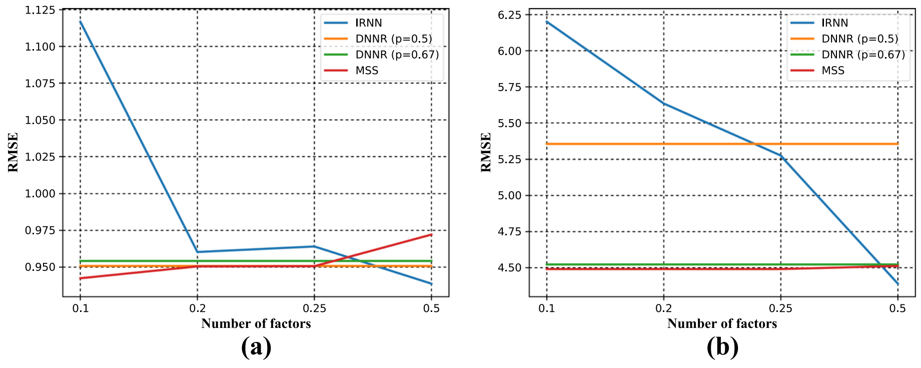

Fig. 7 Effect of varying p values on **a** ML-100K and **b** Jester datasets. Because DNNR only allows $p = \frac{1}{2}$ and $\frac{2}{3}$, we plot it with two lines across different p values

it can be found from the experimental results that SVD++ has excellent performance on most datasets among all tested methods except graph-based models. This is because that SVD++ makes full use of implicit feedback and explicit feedback, and considers both matrix factorization methods and neighborhood methods. Graph-based models (i.e., GCMC and sRGCNN) achieve pleasurable performance in rating prediction tasks, because generated graph-structured features can better depict the relationships between users and items, and graphs are able to propagate information more effectively. Secondly, on the whole, rank

Table 4 The performance (MAE±std%) comparison for rating tasks on all tested data. The lower the better

	FilmTrust	ML-100K	ML-1M	Netflix	Epinions	Jester
NMF [34]	0.643±0.7	0.751±0.3	0.727±0.1	0.749±0.2	0.805±0.7	3.437±0.7
SVD++ [44]	0.616±0.7	0.714±0.2	**0.661±0.1**	0.703±0.2	0.777±0.5	3.395±1.1
LLORMA [50]	0.634±0.8	0.715±0.2	0.676±0.1	0.698±0.3	0.876±0.6	**3.242±1.1**
AutoRec [65]	0.647±1.0	0.707±0.3	0.682±0.7	**0.695±0.1**	0.894±0.9	3.522±2.6
GCMC [75]	0.621±0.7	0.709±0.5	0.674±0.7	0.698±0.3	**0.764±0.7**	3.402±2.8
sRGCNN [76]	**0.603±0.8**	**0.702±0.1**	0.690±0.9	0.696±0.1	0.878±1.2	3.312±2.2
IRNN [84]	0.740±0.7	0.721±0.4	0.730±0.5	0.752±0.2	1.401±0.8	4.248±1.8
DNNR [85]	0.736±0.8	0.743±0.3	0.749±0.6	0.748±0.2	1.489±0.4	3.998±1.5
MSS [89]	0.620±0.9	0.741±0.2	0.688±0.7	0.731±0.3	0.812±0.8	3.391±1.4
ISVTA [95]	0.904±0.8	0.784±0.3	0.671±0.4	0.762±0.4	1.324±1.2	4.352±1.8

Table 5 The performance (RMSE±std%) comparison for rating tasks on all tested data. The lower the better

	FilmTrust	ML-100K	ML-1M	Netflix	Epinions	Jester
NMF [34]	0.860±0.9	0.954±0.5	0.921±0.2	0.961±0.1	1.081±0.9	4.524±0.9
SVD++ [44]	0.800±0.8	0.909±0.4	**0.847±0.2**	0.907±0.3	1.016±0.7	4.514±1.5
LLORMA [50]	0.855±1.0	0.908±0.2	0.864±0.2	0.917±0.3	1.182±1.0	**4.379±1.1**
AutoRec [65]	0.845±1.1	0.904±0.4	0.870±1.1	0.918±0.2	1.153±0.7	4.645±1.6
GCMC [75]	0.803±0.5	**0.902±0.4**	0.853±0.6	0.903±0.2	**0.992±0.7**	4.432±3.1
sRGCNN [76]	**0.796±1.1**	0.927±0.3	0.881±0.4	**0.891±0.3**	1.192±1.3	4.602±2.5
IRNN [84]	1.001±0.8	0.938±0.3	0.930±0.6	0.979±0.2	1.883±0.9	5.634±1.9
DNNR [85]	0.998±0.7	0.951±0.2	0.969±0.5	0.963±0.2	1.915±1.2	5.354±1.7
MSS [89]	0.810±0.7	0.942±0.4	0.874±0.6	0.949±0.4	1.047±0.7	4.490±1.5
ISVTA [95]	1.210±0.9	1.018±0.2	0.935±0.4	1.012±0.5	1.765±1.3	5.893±1.9

minimization methods perform worse than other types of models, which may be attributed to the fact that these methods concentrate more on recovering existing values of the matrix instead of predicting missing values in test sets. As to the top-*k* experiments, methods based on KNN (Item-KNN and SLIM) achieve superior performance, which indicates that ideas of KNN are still effective. It is obvious that SLIM and CDAE gain extremely high NDCG values in Jester dataset. This is probably because that both of them learn meaningful user or item embedding when the number of items is further smaller than the number of users. Besides, FISM and DMF both gain acceptable performance, while CDAE performs poorly on some sparse datasets. The experimental results figure out that neural network methods are not always better than traditional machine learning models, and even sometimes achieve worse performance. This motivates us to continue developing traditional optimization methods. Last but not the least, it is clear that all algorithms perform poorly with sparse datasets, which shows that an essential challenge for recommender systems is sparsity. Because the scale of ratings in Jester is wide, the results of RMSE and MAE are extraordinarily larger than other datasets.

Table 8 records runtime comparison of various algorithms on different tested datasets. It is evident that neural network-based models such as AutoRec, CDAE, DMF, and all graph-

Table 6 The performance (NDCG±std%) comparison for ranking tasks on all tested data. The higher the better

	FilmTrust	ML-100K	ML-1M	Netflix	Epinions	Jester
Item-KNN	0.607±0.7	0.569±0.7	0.455±1.9	0.507±1.2	**0.208±0.8**	0.375±0.4
SLIM [47]	**0.666±0.8**	**0.587±0.9**	**0.542±1.5**	**0.567±1.0**	0.187±0.7	0.585±0.5
FISM [48]	0.624±0.3	0.463±0.4	0.430±1.2	0.480±0.5	0.142±0.5	0.377±0.1
Cofactor [56]	0.202±0.7	0.161±1.2	0.189±1.7	0.104±0.3	0.092±1.8	**0.822±0.7**
CDAE [66]	0.305±1.3	0.439±4.1	0.260±4.5	0.447±8.3	0.019±0.2	0.819±0.1
DMF [72]	0.436±1.8	0.430±3.0	0.434±3.8	0.413±6.2	0.146±2.1	0.500±1.2
SpectralCF [73]	0.621±1.3	0.523±2.7	0.482±4.1	0.492±5.8	0.153±1.4	0.643±1.1
NGCF [77]	0.597±1.1	0.548±0.9	0.532±2.3	0.539±4.4	0.198±1.1	0.674±0.9

Table 7 The performance (RECALL±std%) comparison for ranking tasks on all tested data. The higher the better

	FilmTrust	ML-100K	ML-1M	Netflix	Epinions	Jester
Item-KNN	**0.894±0.5**	**0.667±0.8**	0.351±1.2	0.608±0.8	**0.288±0.5**	0.739±0.5
SLIM [47]	0.884±0.5	0.601±0.7	0.353±1.3	**0.630±0.9**	0.219±0.3	0.956±0.6
FISM [48]	0.887±0.2	0.416±0.1	0.290±1.6	0.556±0.3	0.175±0.4	0.709±0.2
Cofactor [56]	0.392±0.4	0.287±0.8	0.268±1.3	0.258±0.1	0.213±0.7	0.346±0.8
CDAE [66]	0.298±0.7	0.294±0.2	0.204±0.2	0.310±0.5	0.024±0.1	**0.970±0.1**
DMF [72]	0.867±1.2	0.561±0.9	**0.405±1.4**	0.570±0.7	0.210±0.4	0.943±0.9
SpectralCF [73]	0.872±1.1	0.574±1.1	0.335±1.8	0.623±0.8	0.193±1.1	0.791±0.8
NGCF [77]	0.864±1.0	0.623±0.8	0.298±1.1	0.365±1.2	0.209±0.9	0.785±0.8

based methods, cost more time for estimations than other models. Because item-KNN has no iterative updating procedure, it is the fastest method among all tested models. Apart from item-KNN, LLORMA and MSS work swiftly in most datasets, followed by SVD++. This is because that both LLORMA and MSS consider localized subproblems with lower dimensions. The runtime of IRNN and DNNR is extremely high among machine learning methods, due to the inefficient SVD of the entire matrix at each iteration. It is noted that the computational cost of some gradient-based methods is not related to the dimension of the feedback matrix, because they may converge slowly on some datasets.

4 Insights and discussions

4.1 Advantages of matrix completion-based methods

In this paper, we start the survey from the concept of low-rank matrix completion problems. Although the real-world applications are diverse, most techniques of recommender systems can be formulated as matrix completion problems that attempt to recover the missing values in incomplete feedback matrices. This is because that the large amount of user feedback data naturally come into being various huge incomplete matrices. It is beneficial for researchers to look into the recommender systems by starting from a well-defined optimization problem,

Table 8 Average runtime (seconds) of matrix completion algorithms on all tested datasets

	FilmTrust	ML-100K	ML-1M	Netflix	Epinions	Jester
NMF [34]	47	158	1184	125	193	464
SVD++ [44]	174	298	1419	369	204	579
LLORMA [50]	56	73	412	145	88	50
AutoRec [65]	7085	16,111	700,344	36,546	39,813	239,740
GCMC [75]	1033	2311	4031	4268	6621	5534
sRGCNN [76]	895	1570	4293	3321	5439	6543
IRNN [84]	391	155	4530	470	42125	20
DNNR [85]	739	108	5030	474	30579	7
MSS [89]	377	56	8024	279	2338	4
ISVTA [95]	1980	172	7165	691	39533	548
ItemKNN	4	3	23	3	28	3
SLIM [47]	452	2626	43187	2603	956	1210
FISM [48]	2775	2880	39937	5697	22478	1344
Cofactor [56]	157	442	7967	73	4758	306
CDAE [66]	1031	697	4854	1287	8864	720
DMF [72]	4113	17258	52861	35578	70985	1913
SpectralCF [73]	2342	3513	5412	4298	7734	4523
NGCF [77]	1239	4321	6753	4659	6985	5576

so that researchers that are new in this field can easily follow previous works. Consequently, in real-world applications, we first need to consider how to transform the problems of recommender systems into matrix completion optimization. Besides, as the rank optimization problems we have discussed in Sect. 2.4, a method developed via the concept of matrix completion tends to have better interpretability and can be solved by traditional iterative optimization methods, not limited to the deep learning methods that may lack theoretical explanation. Despite the fact that deep learning has achieved promising improvements in recommender systems, we encourage researchers to explore related algorithms from the concept of traditional matrix completion algorithms, because it is still a vital learning problem and may inspire us to develop new neural network structures with better interpretability. We will discuss these in the next subsection.

Besides, in real-world applications, although some recommender systems do not need to predict the missing values in user–item interaction matrices, techniques of matrix completion still play an essential role in many scenarios. For example, matrix factorization-based methods which decompose the observed user–item interaction matrices into latent factors of users and items for extracting underlying features, are widely utilized in context-aware [99–101] and sequential recommender systems [102–104]. Therefore, a more in-depth study of matrix completion can help the development of other related techniques for recommender systems.

4.2 Challenges and potential future directions

Traditional Machine Learning Methods First of all, we conclude the methods tested in our experiments. Table 9 shows the brief comparison for all experimented models. As we have analyzed in detail before, traditional convex optimization models like matrix factorization

Table 9 The brief comparison for all algorithms in our experiments

Algorithms	Rating/Top-k	Types of model	Characteristic
NMF [34]	Rating	Matrix factorization model	Latent factors are subject to be nonnegative
SVD++ [44]	Rating	Matrix factorization model	Apply both explicit feedback and implicit feedback
LLORMA [50]	Rating	Matrix factorization model	Assume that matrix should be low-rank locally
FISM [48]	Top-k	Matrix factorization model	Learn item representations for estimation
Cofactor [56]	Top-k	Matrix factorization model	Introduce into word embedding model
AutoRec [65]	Rating	Neural network model	Apply autoencoder framework
CDAE [66]	Top-k	Neural network model	Apply DAE framework
DMF [72]	Top-k	Neural network model	Apply a parallel DNN framework for LFM
GCMC [75]	Rating	Neural network model	Apply graph autoencoder framework
sRGCNN [76]	Rating	Neural network model	Combine GCN and RNN
SpectralCF [73]	Top-k	Neural network model	Discover deep connections between users and items in the spectral domain
NGCF [77]	Top-k	Neural network model	Exploit the user–item graph by propagating embeddings on it
Item-KNN	Top-k	KNN-based model	Predict ranking from neighbors
SLIM [47]	Top-k	KNN-based model	Learn sparse item-based similarity
IRNN [84]	Rating	Rank minimization model	Learn weighted singular value constraint
DNNR [85]	Rating	Rank minimization model	Learn double nonconvex nonsmooth constraint on singular values
MSS [89]	Rating	Rank minimization model	Learn low-rank property with decomposed factors
ISVTA [95]	Rating	Rank minimization model	Optimize modified Schatten-p norm with iterative SVT algorithm

models and rank minimization models still play important roles in recommender systems, and achieve performance that is competitive with or even superior to the deep learning-based models. For rating prediction, SVD++ which exploits both latent factor information and neighborhood coefficients has higher accuracy on most datasets. KNN-based models such as SLIM and Item-KNN gain excellent performance in ranking tasks with smaller time cost. This points out that KNN-based matrix completion methods generally gain pleasurable accuracy in top-k recommendation tasks, owing to the fact that KNN methods are virtually designed for recommending top-k items. These experimental results indicate that traditional ideas of machine learning are still effective in recommender systems. The observation also reveals that conducting matrix completion via exploring neighborhood relationships is profitable to both rating and ranking tasks. We may discover more interpretable and effective models based on machine learning techniques, such as kernel learning, Bayesian learning and clustering. However, the time complexity for some of these methods is high, especially for rank optimization methods. Most rank minimization methods require conducting SVD ($O(\min(m,n)mn)$) of the original matrix at each iteration, and runtime for most similarity measures is $O(n^2)$ and $O(m^2)$ or higher. The time complexity of most popular neural networks is at least $O(\max(n,m)^3)$. As a result, these methods are not suitable for large-scale recommender system datasets. In light of this, how to reduce the computational complexity of recommendation algorithms requires further study. As a matter of fact, LLORMA and MSS that we discussed have attempted to reduce the runtime by avoiding conducting computation on the entire matrix, and decomposing the original optimization problem into several subproblems. The idea of the divide and conquer algorithm can be considered to accelerate the speed of models.

Deep learning methods Due to the rapid development of deep learning, many methods implemented with deep learning have appeared [105]. Most neural networks are applied for extracting features or generating user/item profiles. Not limited to rating matrices, networks like CNN and RNN are widely used for feature engineering on image, audio or text inputs. The extracted features are either used in content-based CF methods or applied as side information. However, most effective deep learning methods seem to need profound understandings of neural networks and substantial trials of experiments, which are tough for researchers to develop a model with theoretical guarantees. Recent study also points out that recommender systems built via neural networks may not perform well compared with traditional iterative machine learning algorithms [106], and we have also found this phenomenon in our experiments.

Consequently, how to integrate traditional machine learning methods into state-of-the-art techniques like deep learning becomes an interesting direction. Some deep learning models are associated with traditional matrix factorization models. For example, DMF that we have discussed conducts matrix factorization with deep neural networks and computes the preference of users with cosine similarity measure. Hence, it is a valuable research direction that we may transform traditional models into deep learning frameworks, because deep learning frameworks inspired by traditional iterative optimization problems generally have better theoretical guarantees. However, most matrix completion optimization problems are based on nonconvex and nondifferentiable optimization objectives, and deep learning methods have difficulties in dealing with these problems. l_1 norm which promotes sparse solutions and nuclear norm which generates low-rank solutions in rank minimization problems are typical examples. The nonsmooth and nondifferentiable properties of these constraints make the gradient descent and backpropagation algorithms not applicable. Although some works have investigated on transforming traditional optimization algorithms into deep learning frameworks [107–110], to our knowledge, there is limited study on handling rank constraints with neural networks. Therefore, how to construct a deep learning framework following the spirits of traditional iterative optimization methods is also a potential direction.

Cold start issues: The performance of most compared models declines due to the sparsity of the user–item feedback matrix, as we have analyzed in experiments. This phenomenon can be regarded as the cold start issue, which has become the primary problem since recommender systems appeared. Generally, it is impossible for users to provide feedback on most items in the database. In real-world applications, it is common to find that a user only has rated a few (even one or two) of millions of items. Thus, the cold start challenges in practical applications are far more severe than experiments on benchmark datasets. Inspired by the excellent performance of SVD++, we may consider exploring more models adopting implicit feedback or side information for matrix completion of recommender systems, which are useful for generating user or item profiles to address the cold start issues. Side information can also be obtained from social relationships, geographic locations, user shopping history, and even time sequences. More embedding methods for these side information are also important for building a more interpretable model, so that unknown values in an incomplete matrix can be predicted more accurately.

5 Conclusion

In this paper, we looked into different types of matrix completion algorithms, including matrix factorization models, neural network models, and rank minimization models. We investigated these methods and discussed the characteristics and improvements. Finally, we introduced different evaluation measurements for recommender systems and used them to evaluate the performance of different algorithms on varying datasets. Some shared hyperparameters were experimented and discussed for investigation. Inspired by experiments and existing research, we further provided insights and potential directions of matrix completion on recommender systems for readers. We believe that a combination of traditional optimization problems in machine learning and popular neural networks will further improve the accuracy of matrix completion. Nowadays, recommender systems are playing more and more critical roles in data mining to discover useful messages and provide suggestions for people. We will explore more efficient algorithms for recommender systems with regard to matrix completion in the future.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant no. U1705262) and Natural Science Foundation of Fujian Province (Grant nos. 2020J01130193 and 2018J07005).

References

1. Batmaz Z, Yurekli A, Bilge A, Kaleli C (2019) A review on deep learning for recommender systems: challenges and remedies. Artif Intell Rev 52(1):1–37
2. Sun Y, Guo G, Chen X, Zhang P, Wang X (2020) Exploiting review embedding and user attention for item recommendation. Knowl Inform Syst 1–24
3. Hashemi SM, Rahmati M (2020) Cross-domain recommender system using generalized canonical correlation analysis. Knowl Inf Syst 62(12):4625–4651
4. Papagelis M, Plexousakis D, Kutsuras T (2005) Alleviating the sparsity problem of collaborative filtering using trust inferences. Trust Manag 224–239

5. Martinez L, Rodriguez RM, Espinilla M, Reja (2009) A georeferenced hybrid recommender system for restaurants, in: Proceedings of the 2009 IEEE/WIC/ACM international joint conference on web intelligence and intelligent agent technology, pp 187–190
6. Singh M (2020) Scalability and sparsity issues in recommender datasets: a survey. Knowl Inf Syst 62(1):1–43
7. Huang Z, Chen H, Zeng D (2004) Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. ACM Trans Inf Syst 22(1):116–142
8. Moshfeghi Y, Piwowarski B, Jose JM (2011) Handling data sparsity in collaborative filtering using emotion and semantic based features. In: Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval, pp 625–634
9. Li B, Yang Q, Xue X (2009) Can movies and books collaborate? Cross-domain collaborative filtering for sparsity reduction. In: Proceedings of the 21th international joint conference on artificial intelligence, vol 9, pp 2052–2057
10. Raza S, Ding C (2019) Progress in context-aware recommender systems–an overview. Comput Sci Rev 31:84–97
11. Palomares I, Browne F, Davis P (2018) Multi-view fuzzy information fusion in collaborative filtering recommender systems: application to the urban resilience domain. Data Knowl Eng 113:64–80
12. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37
13. Miller BN, Konstan JA, Riedl J (2004) Pocketlens: toward a personal recommender system. ACM Trans Inf Syst 22(3):437–476
14. Champiri ZD, Asemi A, Binti SSS (2019) Meta-analysis of evaluation methods and metrics used in context-aware scholarly recommender systems. Knowl Inf Syst 61(2):1147–1178
15. Kulkarni S, Rodd SF (2020) Context aware recommendation systems: a review of the state of the art techniques. Comput Sci Rev 37:100255
16. Shokeen J, Rana C (2020) A study on features of social recommender systems. Artif Intell Rev 53(2):965–988
17. Khan ZY, Niu Z, Sandiwarno S, Prince R (2020) Deep learning techniques for rating prediction: a survey of the state-of-the-art. Artif Intell Rev 1–41
18. Elahi M, Ricci F, Rubens N (2016) A survey of active learning in collaborative filtering recommender systems. Comput Sci Rev 20:29–50
19. Coba L, Symeonidis P, Zanker M (2019) Personalised novel and explainable matrix factorisation. Data Knowl Eng 122:142–158
20. Si M, Li Q (2020) Shilling attacks against collaborative recommender systems: a review. Artif Intell Rev 53(1):291–319
21. Ali Z, Qi G, Kefalas P, Abro WA, Ali B (2020) A graph-based taxonomy of citation recommendation models. Artif Intell Rev 1–44
22. Dax A (2014) Imputing missing entries of a data matrix: a review. J Adv Comput 3(3):98–222
23. Gurini DF, Gasparetti F, Micarelli A, Sansonetti G (2018) Temporal people-to-people recommendation on social networks with sentiment-based matrix factorization. Futur Gener Comput Syst 78:430–439
24. Nguyen LT, Kim J, Shim B (2019) Low-rank matrix completion: a contemporary survey. IEEE Access 7:94215–94237
25. Chen Z, Zhao W, Wang S (2021) Kernel meets recommender systems: a multi-kernel interpolation for matrix completion. Expert Syst Appl 168:114436
26. Wang Y, Zhang Y (2013) Nonnegative matrix factorization: a comprehensive review. IEEE Trans Knowl Data Eng 25(6):1336–1353
27. Da'u A, Salim N (2019) Recommendation system based on deep learning methods: a systematic review and new directions. Artif Intell Rev 1–40
28. Cheng H-T, Koc L, Harmsen J, Shaked T, Chandra T, Aradhye H, Anderson G, Corrado G, Chai W, Ispir M, et al. (2016) Wide & deep learning for recommender systems. In: Proceedings of the 1st workshop on deep learning for recommender systems, pp 7–10
29. Covington P, Adams J, Sargin E (2016) Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM conference on recommender systems, pp 191–198
30. He X, Liao L, Zhang H, Nie L, Hu X, Chua T-S (2017) Neural collaborative filtering. In: Proceedings of the 26th international conference on World Wide Web, pp 173–182
31. Gu Q, Trzasko JD, Banerjee A (2019) Scalable algorithms for locally low-rank matrix modeling. Knowl Inf Syst 61(3):1457–1484
32. Nie F, Huang H, Ding CHQ (2012) Low-rank matrix recovery via efficient schatten p-norm minimization. In: Proceedings of the 26th AAAI conference on artificial intelligence

33. Shijila B, Tom AJ, George SN (2019) Simultaneous denoising and moving object detection using low rank approximation. Futur Gener Comput Syst 90:198–210
34. Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401(6755):788
35. Lee DD, Seung HS (1997) Unsupervised learning by convex and conic coding. In: Advances in neural information processing systems, pp 515–521
36. Fu L, Chen Z, Huang S, Huang S, Wang S (2021) Multi-view learning via low-rank tensor optimization. In: Proceedings of the 2021 IEEE international conference on multimedia and expo, pp 1–6
37. Chen J, Fang J, Liu W, Tang T, Yang C (2020) clmf: A fine-grained and portable alternating least squares algorithm for parallel matrix factorization. Futur Gener Comput Syst 108:1192–1205
38. Cai J-F, Candès EJ, Shen Z (2010) A singular value thresholding algorithm for matrix completion. SIAM J Optim 20(4):1956–1982
39. Wright J, Ganesh A, Rao SR, Peng Y, Ma Y (2009) Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In: Proceedings of the 23rd annual conference on neural information processing systems 2009, pp 2080–2088
40. Shang F, Cheng J, Liu Y, Luo Z, Lin Z (2018) Bilinear factor matrix norm minimization for robust PCA: algorithms and applications. IEEE Trans Pattern Anal Mach Intell 40(9):2066–2080
41. Paatero P (1997) Least squares formulation of robust non-negative factor analysis. Chemom Intell Lab Syst 37(1):23–35
42. Kivinen J, Warmuth MK (1995) Additive versus exponentiated gradient updates for linear prediction. In: ACM Press the 27th annual ACM symposium, pp 209–218
43. Benesty J, Chen J, Huang Y, Cohen I (2009) Pearson correlation coefficient. In: Noise reduction in speech processing, pp 1–4
44. Koren Y (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, pp 426–434
45. Mnih A, Salakhutdinov RR (2008) Probabilistic matrix factorization. In: Advances in neural information processing systems, pp 1257–1264
46. Bell RM, Koren Y (2007) Lessons from the netflix prize challenge. ACM SIGKDD Explorations Newsl 9(2):75–79
47. Ning X, Karypis G (2011) Slim: Sparse linear methods for top-n recommender systems. In: 2011 11th IEEE international conference on data mining, pp 497–506
48. Kabbur S, Ning X, Karypis G (2013) Fism: Factored item similarity models for top-n recommender systems. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining, pp 659–667
49. Saad D (1998) Online algorithms and stochastic approximations, Online. Learning 5:3–6
50. Lee J, Kim S, Lebanon G, Singer Y (2013) Local low-rank matrix approximation. In: Proceedings of the 30th international conference on machine learning, vol 28, pp 82–90
51. Wand MP, Jones MC (1995) Kernel smoothing
52. Bian J, Gao B, Liu T-Y (2014) Knowledge-powered deep learning for word embedding. In: Proceedings of the joint European conference on machine learning and knowledge discovery in databases, pp 132–148
53. Shin B, Yang H, Choi JD (2019) The pupil has become the master: Teacher-student model-based word embedding distillation with ensemble learning. In: Proceedings of the 28th international joint conference on artificial intelligence, pp 3439–3445
54. Zhou T, Sedoc J, Rodu J (2019) Getting in shape: Word embedding subspaces. In: Proceedings of the 28th international joint conference on artificial intelligence, pp 5478–5484
55. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119
56. Liang D, Altosaar J, Charlin L, Blei DM (2016) Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In: Proceedings of the 10th ACM conference on recommender systems, pp 59–66
57. Church KW, Hanks P (1990) Word association norms, mutual information, and lexicography. Comput Linguist 16(1):22–29
58. Levy O, Goldberg Y (2014) Neural word embedding as implicit matrix factorization. In: Advances in neural information processing systems, pp 2177–2185
59. Liou C, Cheng W, Liou J, Liou D (2014) Autoencoder for words. Neurocomputing 139:84–96
60. Ap SC, Lauly S, Larochelle H, Khapra M, Ravindran B, Raykar VC, Saha A (2014) An autoencoder approach to learning bilingual word representations. In: Advances in neural information processing systems, pp 1853–1861

61. Socher R, Huang EH, Pennin J, Manning CD, Ng AY (2011) Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In: Advances in neural information processing systems, pp 801–809
62. Zhang J, Shan S, Kan M, Chen X (2014) Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment, in. Eur Conf Comput Vis 8690:1–16
63. Walker J, Doersch C, Gupta A, Hebert M (2016) An uncertain future: forecasting from static images using variational autoencoders, in. Eur Conf Comput Vis 9911:835–851
64. Tewari A, Zollhofer M, Kim H, Garrido P, Bernard F, Perez P, Theobalt C (2017) Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In: Proceedings of the IEEE international conference on computer vision, pp 3715–3724
65. Sedhain S, Menon AK, Sanner S, Xie L (2015) Autorec: autoencoders meet collaborative filtering. In: Proceedings of the 24th international conference on World Wide Web, pp 111–112
66. Wu Y, DuBois C, Zheng AX, Ester M (2016) Collaborative denoising auto-encoders for top-n recommender systems, in: Proceedings of the 9th ACM international conference on Web Search and Data Mining, pp 153–162
67. Pan Y, He F, Yu H (2020) A correlative denoising autoencoder to model social influence for top-n recommender system. Front Comp Sci 14(3):143301
68. Wang H, Shi X, Yeung D-Y (2015) Relational stacked denoising autoencoder for tag recommendation. In: Proceedings of the 29th AAAI conference on artificial intelligence, pp 3052–3058
69. Ishii T, Komiyama H, Shinozaki T, Horiuchi Y, Kuroiwa S (2013) Reverberant speech recognition based on denoising autoencoder. In: Interspeech, pp 3512–3516
70. Vincent P, Larochelle H, Bengio Y, Manzagol P-A (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on machine learning, pp 1096–1103
71. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 12(Jul):2121–2159
72. Xue H, Dai X, Zhang J, Huang S, Chen J (2017) Deep matrix factorization models for recommender systems. In: Proceedings of the 26th international joint conference on artificial intelligence, pp 3203–3209
73. Zheng L, Lu C-T, Jiang F, Zhang J, Yu PS (2018) Spectral collaborative filtering. In: Proceedings of the 12th ACM conference on recommender systems, pp 311–319
74. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th international conference on learning representations
75. Berg Rvd, Kipf TN, Welling M (2017) Graph convolutional matrix completion, arXiv preprint arXiv:1706.02263
76. Monti F, Bronstein MM, Bresson X (2017) Geometric matrix completion with recurrent multi-graph neural networks. In: Advances in neural information processing systems, pp 3697–3707
77. Wang X, He X, Wang M, Feng F, Chua T-S (2019) Neural graph collaborative filtering. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, pp 165–174
78. Frank LE, Friedman JH (1993) A statistical view of some chemometrics regression tools. Technometrics 35(2):109–135
79. Friedman JH (2012) Fast sparse regression and classification. Int J Forecast 28(3):722–738
80. Geman D, Yang C (1995) Nonlinear image recovery with half-quadratic regularization. IEEE Trans Image Process 4(7):932–946
81. Trzasko JD, Manduca A (2009) Highly undersampled magnetic resonance image reconstruction via homotopic ℓ_0 -minimization. IEEE Trans Med Imaging 28(1):106–121
82. Gao C, Wang N, Yu QR, Zhang Z (2011) A feasible nonconvex relaxation approach to feature selection. In: Proceedings of the 25th AAAI conference on artificial intelligence, pp 356–361
83. Border K (2001) The supergradient of a concave function, http://www.hss.caltech.edu/-kcb/Notes/Supergrad.pdf
84. Lu C, Tang J, Yan S, Lin Z (2014) Generalized nonconvex nonsmooth low-rank minimization. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4130–4137
85. Zhang H, Gong C, Qian J, Zhang B, Xu C, Yang J (2019) Efficient recovery of low-rank matrix via double nonconvex nonsmooth rank minimization. IEEE Trans Neural Netw Learn Syst 30(10):2916–2925
86. Zhang H, Yang J, Shang F, Gong C, Zhang Z (2018) Lrr for subspace segmentation via tractable schatten-p norm minimization and factorization. IEEE Trans Cybern 49(5):1722–1734
87. Cao W, Sun J, Xu Z (2013) Fast image deconvolution using closed-form thresholding formulas of lq (q = 1/2, 2/3) regularization. J Vis Commun Image Represent 24(1):31–41

88. Luo L, Yang J, Qian J, Tai Y, Lu G-F (2016) Robust image regression based on the extended matrix variate power exponential distribution of dependent noise. IEEE Trans Neural Netw Learn Syst 28(9):2168–2182
89. Xu C, Lin Z, Zha H (2017) A unified convex surrogate for the schatten-*p* norm. In: Proceedings of the thirty-First AAAI conference on artificial intelligence, pp 926–932
90. Srebro N, Rennie JDM, Jaakkola TS (2004) Maximum-margin matrix factorization. Adv Neural Inf Process Syst 17:1329–1336
91. Shang F, Liu Y, Cheng J (2016) Scalable algorithms for tractable schatten quasi-norm minimization. In: Proceedings of the 30th AAAI conference on artificial intelligence, pp 2016–2022
92. Shang F, Liu Y, Cheng J (2016) Tractable and scalable schatten quasi-norm approximations for rank minimization. In: Proceedings of the 19th international conference on artificial intelligence and statistics, vol 51, pp 620–629
93. Bolte J, Sabach S, Teboulle M (2014) Proximal alternating linearized minimization for nonconvex and nonsmooth problems. Math Program 146(1–2):459–494
94. Xu Y, Yin W (2013) A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. SIAM J Imag Sci 6(3):1758–1789
95. Zhang H, Qian J, Zhang B, Yang J, Gong C, Wei Y (2020) Low-rank matrix recovery via modified schatten-*p* norm minimization with convergence guarantees. IEEE Trans Image Process 29:3132–3142
96. Willmott CJ, Matsuura K (2005) Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. Climate Res 30(1):79–82
97. Järvelin K, Kekäläinen J (2002) Cumulated gain-based evaluation of ir techniques. ACM Trans Inf Syst 20(4):422–446
98. Powers DM (2011) Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. J Mach Learn Technol 2(1):37–63
99. Baltrunas L, Ludwig B, Ricci F (2011) Matrix factorization techniques for context aware recommendation. In: Proceedings of the ACM conference on recommender systems, pp 301–304
100. Hidasi B, Tikk D (2016) General factorization framework for context-aware recommendations. Data Min Knowl Disc 30(2):342–371
101. Unger M, Bar A, Shapira B, Rokach L (2016) Towards latent context-aware recommendation systems. Knowl Based Syst 104:165–178
102. Rendle S, Freudenthaler C, Schmidt-Thieme L (2010) Factorizing personalized Markov chains for next-basket recommendation. In: Proceedings of the 19th international conference on World Wide Web, pp 811–820
103. Yu H, Hsieh C, Si S, Dhillon IS (2014) Parallel matrix factorization for recommender systems. Knowl Inf Syst 41(3):793–819
104. Wu H, Zhang Z, Yue K, Zhang B, He J, Sun L (2018) Dual-regularized matrix factorization with deep neural networks for recommender systems. Knowl Based Syst 145:46–58
105. Zhang S, Yao L, Sun A, Tay Y (2019) Deep learning based recommender system: a survey and new perspectives. ACM Comput Surv 52(1):5:1–5:38
106. Dacrema MF, Cremonesi P, Jannach D (2019) Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In: Proceedings of the 13th ACM conference on recommender systems, pp 101–109
107. Wang S, Chen Z, Du S, Lin Z (2021) Learning deep sparse regularizers with applications to multi-view clustering and semi-supervised classification. IEEE Trans Pattern Anal Mach Intell. https://doi.org/10.1109/TPAMI.2021.3082632
108. Xie X, Wu J, Liu G, Zhong Z, Lin Z (2019) Differentiable linearized ADMM. In: Proceedings of the 26th international conference on machine learning, pp 6902–6911
109. Yang Y, Sun J, Li H, Z. (2016) Xu, Deep admm-net for compressive sensing MRI. In: Advances in neural information processing systems, pp 10–18
110. Gregor K, LeCun Y (2010) Learning fast approximations of sparse coding. In: Proceedings of the twenty-seventh international conference on machine learning, pp. 399–406

Zhaoliang Chen received his B.S. degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China in 2019. He is currently pursuing the Ph.D degree with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. His current research interests include machine learning, deep learning and recommender systems.

Shiping Wang received his Ph.D. degree from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China in 2014. He worked as a research fellow in Nanyang Technological University from August 2015 to August 2016. He is currently a Full Professor and Qishan Scholar with the College of Mathematics and Computer Science, Fuzhou University. His research interests include machine learning, computer vision and granular computing.