Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Matrix factorization with a sigmoid-like loss control

Yuan-Yuan Xu [a,b], Hui Xiao [a,b], Heng-Ru Zhang [a,b], Wei-Zhi Wu [c,d], Fan Min [a,b,*]

[a] School of Computer Science and Software Engineering, Southwest Petroleum University, Chengdu 610500, China
[b] Lab of Machine Learning, Southwest Petroleum University, Chengdu 610500, China
[c] School of Information Enginering, Zhejiang Ocean University, Zhoushan 316022, China
[d] Key Laboratory of Oceanographic Big Data Mining and Application of Zhejiang Province, Zhejiang Ocean University, Zhoushan 316022, China

ARTICLE INFO

Communicated by W. Pan

Keywords:
Loss function
Matrix factorization
Over-fitting
Recommender system

ABSTRACT

Matrix factorization is one of the fundamental approaches of recommender systems. With the popular L_2 loss, learning models tend to overfit significantly deviated predictions. However, predicting the actual rating of 5 as 1 or 2 makes no essential difference in the application. In this paper, we design a sigmoid-like function to control the loss of each individual prediction, which has two advantages. First, it reduces the loss corresponding to significantly deviated predictions. Therefore, the impact of these predictions, some of which may be caused by outliers, is also reduced. Second, it is independent of two classical over-fitting control techniques using regular terms and validation data, respectively. Hence, it can be combined with them to form a more powerful method. Experiments are undertaken on six benchmark datasets in comparison with different losses. Results show that the proposed loss function has good performance in terms of MAE, RMSE, and NDCG, however not so good in terms of HR and MAP.

1. Introduction

Recommender systems (RSs) aim to provide suggestions of items to users based on historical data. Memory-based approaches [1,2] locate users or items' neighbors according to similarity measures. The idea is that users who have similar tastes in certain items might also have similar interests in other items. Content-based approaches [3,4] use user demographic and item content information to create prediction models. The idea is that user preferences are strongly influenced by factors such as age, occupation, country of origin, and so on. Social network-based approaches [5,6] discover user tastes according to their family members or friends. The idea is that a person frequently shares products that they have purchased with others. Sentiment-based approaches [7,8] sift user reviews to discover their true feelings. The idea is that comments are more reliable than rating data. Popular research has been expanded to include more complex scenarios such as conversational recommendation [9], three-way recommendation [10,11], and more advanced techniques such as neural collaborative filtering [12].

Currently, matrix factorization (MF) [13,14] is still one of the mainstream techniques in RSs for the following three reasons. First, some latent variables are thought to be involved in determining the user–item relationship. As a result, it is reasonable to decompose the rating matrix into the product of two subspaces. Second, rating matrices in the real world are usually sparse. Movielens100K dataset, for example, has a sparsity of 93.7%. Therefore, low-rank subspaces have good approximating ability. Third, MF can be combined with other approaches to improve performance. Sentiment-based MF [15,16] provides more reliable recommendations with the reviews. Neural MF [17] introduces an efficient approach without sampling to overcome the shortcomings of many parameters, expensive computations, and the poor robustness of negative sampling. Deep MF [18] improves training efficiency by combining collaborative filtering and deep learning at the scale of model parameters.

MF approaches often employ L_2 loss as the loss function [19]. Its advantages include differentiability and consistency with mean squared error. The disadvantage is that subspaces are affected by significantly deviated predictions. For example, if the actual rating is 5, it is preferable to change the prediction from 1 to 2 or higher. In this way, the loss is significantly reduced. However, there is no essential difference between the predictions of 1 and 2, because both follow the same behavior: not recommended. The consequence is that such change will sacrifice the precision of more important predictions. In other words, using L_2 loss will lead to over-fitting. Outlier removal is a simple solution to this issue [20]. However, it is difficult to judge which ratings are outliers, and removing data will result in information loss.

In this paper, we design a sigmoid-like function to control the loss of each individual prediction, which has two advantages. First,

* Corresponding author at: School of Computer Science and Software Engineering, Southwest Petroleum University, Chengdu 610500, China.
E-mail addresses: xyy@swpu.edu.cn (Y.-Y. Xu), 202222000525@stu.swpu.edu.cn (H. Xiao), zhanghr@swpu.edu.cn (H.-R. Zhang), wuwz@zjou.edu.cn (W.-Z. Wu), minfan@swpu.edu.cn (F. Min).

https://doi.org/10.1016/j.neucom.2024.127338
Received 1 June 2022; Received in revised form 2 November 2023; Accepted 26 January 2024
Available online 30 January 2024

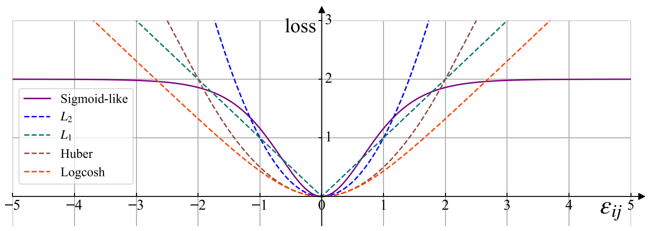


Fig. 1. Comparison of our sigmoid-like loss function with others. The abscissa is the deviation $\varepsilon_{ij} \in [-5,+5]$. ε_{ij} is the deviation between the original rating x_{ij} and its predicted rating $\hat{x}_{ij}$. With the increase of ε_{ij}, L_1 loss grows linearly, and L_2 loss grows quadratically. Huber and Logcosh losses grow much slower than L_2. Our sigmoid-like loss increases at first, then slows as ε_{ij} approaches 2, until the loss reaches 2.

it reduces the loss corresponding to significantly deviated predictions. Fig. 1 compares our loss function with different losses. In many RSs, the original rating x_{ij} takes discrete values in $\{1,2,3,4,5\}$, while the predicted rating $\hat{x}_{ij}$ takes real values in $[1,5]$. Hence the deviation $\varepsilon_{ij} = x_{ij} - \hat{x}_{ij}$ takes real values in $[-5,5]$. L_1 loss (given by $|\varepsilon_{ij}|$) shows linear growth. L_2 loss (given by ε_{ij}^2) shows squared growth. Huber [21] and Logcosh [22] losses grow much slower than L_2 as ε_{ij} increases.

Our loss is greater than L_2, Huber, and Logcosh losses when $\varepsilon_{ij} \le 2$. However, it is much smaller than theirs when $\varepsilon_{ij} > 2$. Additionally, the growth rate of our loss is much smoother than others when $\varepsilon_{ij} > 2$. As a result, our loss not only punishes prediction deviations but also suppresses the impact of outliers. Furthermore, our loss function has an upper bound, ensuring that the loss is under control.

Second, it is independent of two classical overfitting control techniques. *Regular terms* [23] are used to control the coefficients in subspaces. In fact, they are widely used in machine learning to avoid overfitting. Popular regular terms include L_1 [24], L_2 [25], and implicit regularization [26,27]. The *validation data* is used to adjust parameters. In MF, it can be used to control the training process. That is, once the prediction loss of the validation data increases, the training process should be terminated. These three techniques are combined to create a more powerful method.

Experiments are undertaken on six benchmark datasets, i.e., Movielens100K, Amazon, Yelp, FilmTrust, Hetrect, and EachMovie. The prediction quality measures include mean absolute error (MAE) and root mean absolute error (RMSE). The recommendation quality is evaluated by hit ratio (HR), mean average precision (MAP), and normalized discounted cumulative gain (NDCG). Since this is a fundamental work, we only compare six losses using different combinations of the modified sigmoid-like loss function, regular terms, and validation data. The source and data are available at https://gitee.com/xuyuanyuanswpu/mfll. Results show that the proposed loss function has good performance in terms of MAE, RMSE, and NDCG. However, not so good in terms of HR and MAP.

The rest of the paper is organized as follows: Section 3 introduces the preliminaries. Section 4 elaborates on our work. Section 5 demonstrates a running example. Section 6 shows extensive experiments to validate the effectiveness of our algorithm. Finally, Section 7 concludes the paper.

2. Related works

Our work is not purely a new recommendation algorithm. In fact, the focus is the new loss function. To demonstrate the advantages of the new loss function, we apply it to the matrix-factorization-based recommendation. Therefore, we first present the theories and existing applications of other losses. Then, we introduce the matrix factorization based on the losses.

Table 1 lists notations used throughout the paper.

Table 1
Notations.

Notation	Meaning
Ω_{tr}	The training set
Ω_{te}	The testing set
Ω_{va}	The validation set
$\mathbf{X}$	The rating matrix
$\mathbf{U}$	The set of users
m	The number of users
n	The number of items
$\mathbf{x}_i$	The ith row of $\mathbf{X}$
x_{ij}	The rating for u_i to t_j in $\mathbf{X}$
$\hat{\mathbf{X}}$	The prediction matrix
$\hat{x}_{ij}$	The predicted rating for u_i to t_j
$\mathbf{E}$	The deviation matrix
ε_{ij}	The deviation between x_{ij} and $\hat{x}_{ij}$
k	The rank of low-rank matrices
λ	The regularization coefficient

2.1. *Popular loss functions*

This section introduces mainstream loss functions for regression.

2.1.1. *The L_p loss*

The most popular L_p loss are L_1 and L_2 losses. The formulas are

$$L_1 = \sum_{(i,j)\in\Omega_{tr}} f_1(\varepsilon_{ij}), \tag{1}$$

where

$$f_1(\varepsilon_{ij}) = |\varepsilon_{ij}| \tag{2}$$

and

$$L_2 = \sum_{(i,j)\in\Omega_{tr}} f_2(\varepsilon_{ij}), \tag{3}$$

where

$$f_2(\varepsilon_{ij}) = \varepsilon_{ij}^2, \tag{4}$$

respectively.

L_1 loss has the same gradient except at $\varepsilon_{ij} = 0$, which means there is no difference between the large and the small deviations. This is not conducive to the convergence of the function and the learning of the model. Hence, the stable gradient leads to a relatively robust solution.

L_2 loss, as a widely used loss function, can be derived everywhere. We can obtain the optimal solution by the gradient descent algorithm. Since as ε_{ij} decreases, so does the gradient, which is good for convergence. L_2 gives higher weights to the outliers if there are outliers in the dataset, which would sacrifice the prediction effect of other normal data.

2.1.2. *The Huber loss*

Huber loss overcomes the flaws of absolute loss and square loss while combining their benefits [21,28]. It is given by

$$L_{\mathrm{H}} = \sum_{(i,j)\in\Omega_{tr}} f_{\mathrm{H}}(\varepsilon_{ij}), \tag{5}$$

where

$$f_{\mathrm{H}}(\varepsilon_{ij}) = \begin{cases} \frac{1}{2}\varepsilon_{ij}^2, & \text{if } |\varepsilon_{ij}| \le \delta; \\ \delta|\varepsilon_{ij}| - \frac{1}{2}\delta^2, & \text{otherwise.} \end{cases} \tag{6}$$

The threshold δ is greater than zero and set to determine whether to use square or absolute value loss. Compared with the absolute loss, it is differentiable. So it can be applied to the gradient descent method. Compared with the square loss, it clearly limits the loss value of outliers. So it has a low sensitivity to outliers [29].

Huber loss gains achievements in the deep Q-network (DQN), the pre-training model for the RS, and the quantized matrix completion

(QMC). In the DQN algorithm, Huber loss trains the value network to reduce the fluctuation of the click-through rate [30]. In the pre-training model, Huber loss is minimized to predict the values of users' numerical attributes [31]. It can effectively integrate the heterogeneous user information and generate a more expressive user representation. In the QMC algorithm, Huber loss is used for punishing quantization bounds that have been violated [29]. It performs well in learning accuracy and computational complexity.

2.1.3. *The logcosh loss*

The logcosh loss is the logarithm of the hyperbolic cosine of ε_{ij} [22, 32]. Here, we approximate it as

$$L_{\mathrm{LC}} = \sum_{(i,j)\in\Omega_{tr}} f_{\mathrm{LC}}(\varepsilon_{ij}), \tag{7}$$

where

$$f_{\mathrm{LC}}(\varepsilon_{ij}) = \begin{cases} \frac{1}{2}\varepsilon_{ij}^2, & \text{if } |\varepsilon_{ij}| \le \delta; \\ |\varepsilon_{ij}| - \ln 2, & \text{otherwise.} \end{cases} \tag{8}$$

When ε_{ij} is less than δ, it is the same as the Huber loss. It still possesses all the advantages of the Huber loss. Moreover, Logcosh loss can be quadratically differentiable anywhere.

Logcosh loss achievements in the spatiotemporal attention network with a neural algorithm logic unit (STANet-NALU) and Logcosh conditional variational autoencoder (LCVAE). In the STANet-NALU structure, Logcosh loss is used to converge the kernel density estimator [33]. It can effectively represent and integrate spatiotemporal features. In the LCVAE model, Logcosh loss generates positive optimization of the latent space [34]. It can alleviate the class imbalance and improve intrusion detection capabilities.

2.1.4. *The mean squared logarithmic error loss*

The mean squared logarithmic error (MSLE) loss is the relative error under the logarithm [35,36], which is different from L_1 loss. It is given by

$$L_{\mathrm{MSLE}} = \sum_{(i,j)\in\Omega_{tr}} f_{\mathrm{MSLE}}(x_{ij}, \hat{x}_{ij}), \tag{9}$$

where

$$f_{\mathrm{MSLE}}(x_{ij}, \hat{x}_{ij}) = \left(\ln \frac{1 + x_{ij}}{1 + \hat{x}_{ij}}\right)^2. \tag{10}$$

MSLE loss gains achievements in adaptive beamforming by deep learning (ABLE) and the predicting approach to the computational cost of deep learning models. In the ABLE method, MSLE loss balances deviation from desirable image properties [37]. It can produce high-contrast and high-resolution imaging and significantly suppress noise and clutter, while also greatly reducing reconstruction time and computational burden. In the predicting approach, MSLE loss trains the deep learning prediction network to accurately predict the execution time required of various commonly used neural network components [38].

2.2. *Matrix factorization (MF)*

MF-based recommendation maps users and items to a latent subspace [13,39]. Suppose the dataset has m users and n items. Given a rating matrix $\mathbf{X} = (x_{ij})_{m\times n}$, MF employs the product of two low-rank matrices $\mathbf{H}$ and $\mathbf{V}$ to approximate $\mathbf{X}$,

$$\hat{\mathbf{X}} = \mathbf{H}\mathbf{V}^{\mathrm{T}}, \tag{11}$$

where $\mathbf{H} \in \mathbb{R}^{m\times k}$, $\mathbf{V} \in \mathbb{R}^{n\times k}$; k is the rank, and $k \ll m, n$. The hypothesis of low rank leads to its low computational complexity $O(mnk)$.

To make the predicted ratings closer to the original ones, the loss function L is designed to find $\mathbf{H}$ and $\mathbf{V}$ that minimize the loss. We adopt L_2 loss as the loss function here to briefly show the process of MF,

$$\arg\min_{\mathbf{H},\mathbf{V}} L_2 = \sum_{(i,j)\in\Omega_{tr}} (x_{ij} - \hat{x}_{ij})^2, \tag{12}$$

where $\hat{x}_{ij} = \mathbf{h}_i\mathbf{v}_j^{\mathrm{T}}$. So, we update $\mathbf{h}_i$ and $\mathbf{v}_j$

$$\mathbf{h}_i' = \mathbf{h}_i + \beta\left(-\varepsilon_{ij}\mathbf{v}_j^{\mathrm{T}}\right), \tag{13}$$

$$\mathbf{v}_j' = \mathbf{v}_j + \beta\left(-\varepsilon_{ij}\mathbf{h}_i^{\mathrm{T}}\right), \tag{14}$$

where $\varepsilon_{ij} = x_{ij} - \hat{x}_{ij}$ is the deviation, and β is the learning rate. Eq. (12) can fit well on the training set. However, it will lead to poor generalization ability.

In order to improve generalization, regular terms are added to the optimization objective. The mainstream algorithm is to add L_1/L_2 regularization items [40–42]. L_1 is the 1-norm of the vectors in the two subspaces, while L_2 is the 2-norm of them. L_1 regularization is the second term of

$$\arg\min_{\mathbf{H},\mathbf{V}} L_2 = \sum_{(i,j)\in\Omega_{tr}} {\varepsilon_{ij}}^2 + \lambda\left(\sum_{i=0}^{m-1}\|\mathbf{h}_i\|_1 + \sum_{j=0}^{n-1}\|\mathbf{v}_j\|_1\right), \tag{15}$$

where λ is the regularization coefficient. So, we update $\mathbf{h}_i$ and $\mathbf{v}_j$

$$\mathbf{h}_i' = \mathbf{h}_i + \beta\left(-\varepsilon_{ij}\mathbf{v}_j^{\mathrm{T}} + \lambda\right), \tag{16}$$

$$\mathbf{v}_j' = \mathbf{v}_j + \beta\left(-\varepsilon_{ij}\mathbf{h}_i^{\mathrm{T}} + \lambda\right). \tag{17}$$

L_2 regularization is the second term of

$$\arg\min_{\mathbf{H},\mathbf{V}} L_2 = \sum_{(i,j)\in\Omega_{tr}} \varepsilon_{ij}^2 + \lambda\left(\sum_{i=0}^{m-1}\|\mathbf{h}_i\|_2^2 + \sum_{j=0}^{n-1}\|\mathbf{v}_j\|_2^2\right). \tag{18}$$

So, we update $\mathbf{h}_i$ and $\mathbf{v}_j$

$$\mathbf{h}_i' = \mathbf{h}_i + \beta\left(-\varepsilon_{ij}\mathbf{v}_j^{\mathrm{T}} + \lambda\mathbf{h}_i\right), \tag{19}$$

$$\mathbf{v}_j' = \mathbf{v}_j + \beta\left(-\varepsilon_{ij}\mathbf{h}_i^{\mathrm{T}} + \lambda\mathbf{v}_j\right). \tag{20}$$

Eqs. (13), (14), (19) and (20) indicate that ε_{ij} has a linear relationship with the loss.

L_1 and L_2 losses have been utilized in MF [43–46]. To the best of our knowledge, Huber, Logcosh, and MSLE losses have not.

3. Preliminaries

In this section, we revisit preliminary knowledge of RSs, including the rating system and evaluation measures. The rating system is the fundamental data model of RSs. Each evaluation measure identifies the algorithm's performance in a specific aspect.

3.1. *Rating system*

We first revisit the definition of the rating system [10,11,47]. It usually contains user IDs, item IDs, and ratings. For a rating matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$, m and n denote the number of users and items, respectively. An example of a rating matrix is given by

$$\mathbf{X} = \begin{array}{cc} & \begin{array}{ccccccccccccccc} t_0 & & & & & & & & & & & & & & t_{14} \end{array} \\ \begin{array}{c} u_0 \\ \\ \\ \\ \\ \\ \\ \\ \\ u_9 \end{array} & \begin{pmatrix} 2 & 4 & 5 & - & 2 & - & 5 & - & 5 & - & - & - & - & 1 & 5 \\ - & 3 & 4 & 5 & - & 5 & - & - & - & - & 2 & - & 5 & - & - \\ - & 3 & 5 & 3 & - & 5 & - & 1 & 5 & 3 & - & - & 5 & - & - \\ - & - & 1 & 1 & - & - & - & - & 3 & - & - & 3 & - & 5 & - \\ 3 & 5 & - & 3 & - & 1 & 2 & - & 4 & 5 & - & 5 & 3 & 4 & - \\ - & - & - & 4 & - & 1 & 3 & - & 5 & 4 & 3 & - & - & 4 & - \\ 5 & - & 5 & 2 & 4 & 4 & 5 & - & - & 3 & 2 & 5 & - & - & 1 \\ - & 5 & 1 & - & 3 & 4 & 5 & - & - & 3 & - & 1 & 1 & 3 & 3 \\ - & - & 2 & 4 & - & 4 & - & 5 & - & 5 & - & 5 & 5 & - & 5 \\ 4 & - & 3 & - & - & 1 & - & 5 & 5 & 2 & - & - & 1 & - & 2 \end{pmatrix} \end{array}, \tag{21}$$

where $m = 10$, $n = 15$, $r_{ij} \in \{1, 2, 3, 4, 5\}$, and '–' means that the user has not rated respective item.

3.2. Evaluation measures

There are two types of evaluation measures: the error measure and the ranking measure [48,49]. The former evaluates the predictive accuracy, and the latter evaluates the quality of the recommendation list. We define ε_{ij} as the deviation between the original rating x_{ij} and the predicted rating $\hat{x}_{ij}$.

MAE and RMSE are two common error measures [50,51]. They figure out the average deviation of the testing set Ω_{te}, i.e.,

$$\text{MAE} = \frac{\sum_{(i,j)\in\Omega_{te}} |\varepsilon_{ij}|}{car(\Omega_{te})}, \tag{22}$$

and

$$\text{RMSE} = \sqrt{\frac{\sum_{(i,j)\in\Omega_{te}} \varepsilon_{ij}^2}{car(\Omega_{te})}}, \tag{23}$$

where $|\cdot|$ is the absolute value, and $car(\cdot)$ is the cardinality of the set.

HR, MAP, and NDCG are three measures used to evaluate the recommendation list [52]. Let Γ_i be the recommendation list for the user u_i and Π_i be u_i's test items. Let Δ_i^+ be the intersection of Set(Γ_i) and Set(Π_i), where Set($\cdot$) converts the vector to a set. K is the length of Γ_i.

HR evaluates the ratio of the number of users whose Δ_i^+ is not empty [53]. Let m' be the number of hit users. HR is given by

$$\text{HR}@K = \frac{m'}{m}. \tag{24}$$

MAP evaluates the degree of correlation between the item's ranking in Γ_i and Π_i [54], expressed as

$$\text{MAP}@K = \frac{1}{m}\sum_{i\in\text{U}}\frac{1}{car(\Delta_i^+)}\sum_{j=1}^{K}\frac{car(\{1\le p\le j|\Gamma_{ip}\in\Delta_i^+\})}{j}. \tag{25}$$

NDCG reflects the weight that assigns higher scores to the items with a high ranking in Γ_i [55], given by

$$\text{NDCG}@K = \frac{1}{m}\sum_{i\in\text{U}}\frac{\text{DCG}_i@K}{\text{IDCG}@K}. \tag{26}$$

DCG_i is the discounted cumulative gain for u_i,

$$\text{DCG}_i@K = \sum_{j=1}^{K}\frac{2^{rel(\Gamma_{ij})}-1}{\log_2(j+1)}, \tag{27}$$

where $rel(\cdot)$ represents whether Γ_{ij} is in the recommendation list. DCG_i indicates that if the top-ranked items in Γ_i have higher ratings, the RS performs better. IDCG_i is the ideal discounted cumulative gain for u_i. The difference between DCG_i and IDCG_i is the ranking j in the denominator.

4. The proposed approach

In this section, we introduce a novel MF with a sigmoid-like loss.

4.1. Sigmoid-like loss function

The loss function is essential to machine learning. Most works enhance the generalization ability of recommendation models by designing new regularization terms. Nevertheless, our work design a new loss function to improve the ability by suppressing the impact of outliers. The designed sigmoid-like loss function is given by

$$L_{\text{S}} = \sum_{(i,j)\in\Omega_{tr}} f_{\text{S}}(\varepsilon_{ij}), \tag{28}$$

where

$$f_{\text{S}}(x) = \left(\frac{2}{1+e^{-x/C}}-1\right)^2. \tag{29}$$

The subscript S means "sigmoid-like", C is used to control the range, and exponent 2 is used to assure non-negative loss.

$f_{\text{S}}(x)$ expresses our core idea. For the smaller $|x|$, it retains penalties for deviations. For the larger $|x|$, it suppresses the impact of outliers on the loss. Not only should the $f_{\text{S}}(x)$ of outliers be small, but the change in $f_{\text{S}}(x)$ tends to 0 as x increases. In order to analyze the mathematical properties of $f_{\text{S}}(x)$ in more detail, we provide the following lemmas and proofs.

Lemma 1. *$f_{\text{S}}(x)$ is a symmetric function, i.e.,*

$$f_{\text{S}}(-x) = f_{\text{S}}(x).$$

Proof.

$$\begin{aligned} f_{\text{S}}(-x) &= \left(\tfrac{2}{1+e^{x/C}}-1\right)^2 = \left(\tfrac{1-e^{x/C}}{1+e^{x/C}}\right)^2 \\ &= \left(\tfrac{e^{-x/C}-1}{e^{-x/C}+1}\right)^2 = \left(\tfrac{1-e^{-x/C}}{e^{-x/C}+1}\right)^2 \\ &= \left(\tfrac{2}{1+e^{-x/C}}-1\right)^2 = f_{\text{S}}(x). \end{aligned}$$

This completes the proof. □

Lemma 2. *$f_{\text{S}}(x)$ is strictly increasing for $x \ge 0$ and strictly decreasing for $x < 0$, with the minima only for $x = 0$.*

Proof. The first order derivative of $f_{\text{S}}(a)$ is

$$f_{\text{S}}'(x) = \frac{4\left(1-e^{-x/C}\right)e^{-x/C}}{C\left(1+e^{-x/C}\right)^3}.$$

We observe that $\left(1-e^{-x/C}\right) \ge 0$ due to $x \ge 0$ and $C > 0$. Hence, $f_{\text{S}}'(x) \ge 0$. So, $f_{\text{S}}(x)$ is strictly ascending when $x \ge 0$. According to Lemma 1, $f_{\text{S}}(x)$ is strictly decreasing when $x < 0$. So, there is only one minima at $x = 0$. This completes the proof. □

Lemma 3. *The change in $f_{\text{S}}(x)$ tends to 0 as x increases, i.e.,*

$$\lim_{x\to\infty} f_{\text{S}}'(x) = 0.$$

Proof.

$$\lim_{x\to+\infty} f_{\text{S}}'(x) = \frac{4\left(1-e^{-x/C}\right)e^{-x/C}}{C\left(1+e^{-x/C}\right)^3} = 0$$

and

$$\begin{aligned} \lim_{x\to-\infty} f_{\text{S}}'(x) &= \frac{4\left(1-e^{-x/C}\right)e^{-x/C}}{C\left(1+e^{-x/C}\right)^3} \\ &= \frac{4\left(e^{x/C}-1\right)e^{x/C}}{C\left(1+e^{x/C}\right)^3} = 0. \end{aligned}$$

This completes the proof. □

Lemma 4. *$f_{\text{S}}(x)$ is convex in $[C\ln(2-\sqrt{3}), C\ln(2+\sqrt{3})]$, and concave otherwise.*

Proof. The second order derivative of $f_{\text{S}}(x)$ is

$$f_{\text{S}}''(x) = \frac{4}{C^2}\frac{-e^{-3x/C}+4e^{-2x/C}-e^{-x/C}}{\left(1+e^{-x/C}\right)^4}.$$

We obtain

$$f_{\text{S}}''(x)\begin{cases} \ge 0 & \text{if } x\in[C\ln(2-\sqrt{3}), C\ln(2+\sqrt{3})]; \\ < 0 & \text{otherwise.} \end{cases}$$

When $x\in[C\ln(2-\sqrt{3}), C\ln(2+\sqrt{3})]$, $f_{\text{S}}(x)$ is convex. Otherwise, $f_{\text{S}}(x)$ is concave. This completes the proof. □

Let $x = \varepsilon_{ij} = x_{ij} - \mathbf{h}_i\mathbf{v}_j^{\mathrm{T}}$, and

$$p_{ij} = \frac{4\left(1-e^{-\varepsilon_{ij}/C}\right)e^{-\varepsilon_{ij}/C}}{C\left(1+e^{-\varepsilon_{ij}/C}\right)^3}, \tag{30}$$

Table 2
The characteristics of popular loss functions and sigmoid-like loss function for recommender systems → denotes the change in the relationship between the loss and the deviation.

Loss	Differentiability	Prediction deviation	Robustness to outliers	Convexity
L_1 loss	No	retain	Robust	convex
L_2 loss	Yes	suppress → enlarge	Sensitive	convex
Huber loss	Yes	suppress → enlarge → retain	Robust	convex
Logcosh loss	Yes	suppress → enlarge → retain	Robust	convex
Sigmoid-like loss	Yes	enlarge → suppress	Robust	convex → concave

we have

$$\frac{\partial L_S}{\partial \mathbf{h}_i} = \begin{cases} \sum_j p_{ij}\mathbf{v}_j^{\mathrm{T}}, & \text{if } \varepsilon_{ij} < 0; \\ \sum_j -p_{ij}\mathbf{v}_j^{\mathrm{T}}, & \text{otherwise,} \end{cases} \tag{31}$$

and

$$\frac{\partial L_S}{\partial \mathbf{v}_j} = \begin{cases} \sum_i p_{ij}\mathbf{h}_i^{\mathrm{T}}, & \text{if } \varepsilon_{ij} < 0; \\ \sum_i -p_{ij}\mathbf{h}_i^{\mathrm{T}}, & \text{otherwise.} \end{cases} \tag{32}$$

Here is the process:

$$\begin{aligned}\frac{\partial L_S}{\partial \mathbf{h}_i} &= \sum_j 2\left(\frac{2}{1+e^{-\varepsilon_{ij}/C}} - 1\right) \times \left(-\frac{2}{\left(1+e^{-\varepsilon_{ij}/C}\right)^2}\right) \times e^{-\varepsilon_{ij}/C} \times \left(-\frac{1}{C}\right) \times \left(-\mathbf{v}_j^{\mathrm{T}}\right) \\ &= \sum_j -\frac{4\left(1-e^{-\varepsilon_{ij}/C}\right)e^{-\varepsilon_{ij}/C}}{C\left(1+e^{-\varepsilon_{ij}/C}\right)^3}\mathbf{v}_j^{\mathrm{T}}.\end{aligned} \tag{33}$$

Similarly,

$$\frac{\partial L_S}{\partial \mathbf{v}_j} = \sum_i -\frac{4\left(1-e^{-\varepsilon_{ij}/C}\right)e^{-\varepsilon_{ij}/C}\varepsilon_{ij}}{C\left(1+e^{-\varepsilon_{ij}/C}\right)^3}\mathbf{h}_i^{\mathrm{T}}. \tag{34}$$

We update

$$\mathbf{h}_i' = \mathbf{h}_i - \beta\sum_j |p_{ij}|\mathbf{v}_j^{\mathrm{T}}, \tag{35}$$

and

$$\mathbf{v}_j' = \mathbf{v}_j - \beta\sum_i |p_{ij}|\mathbf{h}_i^{\mathrm{T}}. \tag{36}$$

For Eq. (28), we consider adding L_2 regularization term to prevent over-fitting,

$$L_S = \sum_{(i,j)\in\Omega_{tr}}\left(\frac{2}{1+e^{-\left(x_{ij}-\mathbf{h}_i\mathbf{v}_j^{\mathrm{T}}\right)/C}} - 1\right)^2 + \lambda\left(\sum_{i=0}^{m-1}\|\mathbf{h}_i\|_2^2 + \sum_{j=0}^{n-1}\|\mathbf{v}_j\|_2^2\right). \tag{37}$$

So, we update

$$\mathbf{h}_i' = \mathbf{h}_i - \beta(\sum_j |p_{ij}|\mathbf{v}_j^{\mathrm{T}} + \lambda\mathbf{h}_i), \tag{38}$$

and

$$\mathbf{v}_j' = \mathbf{v}_j - \beta(\sum_i |p_{ij}|\mathbf{h}_i^{\mathrm{T}} + \lambda\mathbf{v}_j). \tag{39}$$

4.2. Comparison of characteristics

Table 2 compares our loss with popular losses from several aspects. Since MSLE loss is the relative error, it is not analyzed here with other losses. Here, ε_{ij} represents the deviation, and δ represents the threshold for some losses.

(1) Differentiability. L_2, Huber, Logcosh, and sigmoid-like losses are differentiable. However, L_1 loss is not differentiable when $\varepsilon_{ij} = 0$.
(2) Prediction deviation. L_1 loss retains ε_{ij}s due to their linear relationship. L_2 loss suppresses the loss when $\varepsilon_{ij} \le 1$. It enlarges the impact of outliers when $\varepsilon_{ij} > 1$. Huber and Logcosh losses are piece-wise functions. They suppress the loss when $|\varepsilon_{ij}| < \delta$ and $\delta \le 1$. If $\delta > 1$, they first suppress and then enlarge the loss. They retain the deviation when $\varepsilon_{ij} > \delta$. Sigmoid-like loss enlarges the loss value when $\varepsilon_{ij} \le 2$, while suppressing the loss when $\varepsilon_{ij} > 2$.
(3) Robustness to outliers. L_1 is constantly robust to outliers. Huber and Logcosh are robust to outliers since their partial derivatives are constant when $\varepsilon_{ij} > \delta$. L_2 is sensitive to outliers because the partial derivative is linear with the deviation. Sigmoid-like loss is robust to outliers since its partial derivation tends to zero for large deviations.
(4) Convexity. L_1, L_2, Huber, and Logcosh losses are convex for all deviations since their second-order derivatives are no less than zero. Sigmoid-like loss is convex in $[C\ln(2-\sqrt{3}), C\ln(2+\sqrt{3})]$ and concave otherwise. This is proved in Lemma 4.

4.3. The application of sigmoid-like loss

Algorithm 1 shows the main process of applying our loss function to MF-based RSs. The inputs include the training rating matrix $\mathbf{X}$, the validation set Ω_{va}, the rank k, and the tolerance σ. The outputs are the subspaces $\mathbf{H}$ and $\mathbf{V}$.

Line 1 initializes $\mathbf{H}$ and $\mathbf{V}$ with random numbers. We centralize the data. For example, suppose that the ratings of a dataset are {1, 2, 3, 4, 5} and the average rating is 3.5. After centralization, the rating range becomes $[-2.5, 1.5]$. So, we initialize $\mathbf{H}$ and $\mathbf{V}$ with positive and negative numbers randomly.

Line 2 initializes ξ by the difference between the maximum value $\max x_{ij}$ and the minimum value $\min x_{ij}$ of the dataset.

Line 3 initializes δ.

Line 4 controls the iteration.

Lines 5–8 calculate the predictions and the deviations.

Line 9 updates $\mathbf{H}$ and $\mathbf{V}$ by Eqs. (35) and (36).

Line 10 computes the new ξ on Ω_{tr}.

Line 11 recomputes δ.

Line 12 updates ξ.

Line 14 outputs $\mathbf{H}$ and $\mathbf{V}$.

Algorithm 1 MF with a sigmoid-like loss

Input: The rating matrix $\mathbf{X}$, the training set Ω_{tr}, the validation set Ω_{va}, the rank k of $\mathbf{H}$ and $\mathbf{V}$, the tolerance σ for convergence determination;
Output: $\mathbf{H}$ and $\mathbf{V}$;

```
 1: initialize $\mathbf{H}$ and $\mathbf{V}$ randomly;
 2: $\xi = \max x_{ij} - \min x_{ij}$; // initialize $\xi$ by the maximum and minimum
    values of the dataset.
 3: $\delta = \xi$;
 4: while $\delta > \sigma$ do
 5:     for $(i,j) \in \Omega_{tr}$ do
 6:         $\hat{x}_{ij} = \mathbf{h}_i\mathbf{v}_j^{\mathrm{T}}$;
 7:         $\varepsilon_{ij} = x_{ij} - \hat{x}_{ij}$;
 8:     end for
 9:     update $\mathbf{H}$ and $\mathbf{V}$ by Eqs. (35) and (36);
10:     $\xi' = \frac{\sum_{(i,j)\in\Omega_{va}}|x_{ij}-\hat{x}_{ij}|}{car(\Omega_{va})}$;
11:     $\delta = \xi' - \xi$;
12:     $\xi' = \xi$;
13: end while
14: return $\mathbf{H}$ and $\mathbf{V}$;
```

4.4. Time complexity analysis

We will elaborate the time complexity of Algorithm 1 as follows.

Proposition 1. *The time complexity of Algorithm 1 is $O(Tmnk)$.*

Proof. Suppose our algorithm iterates T times. Line 1 initializes $\mathbf{H}$ and $\mathbf{V}$ with $O(mk+nk)$ of time. Line 2 initializes ξ with $O(1)$ of time. Line 3

Table 3
The time complexity of Algorithm 1.

Lines	Complexity
Line 1	$O(mk + nk)$
Line 2	$O(1)$
Line 3	$O(1)$
Line 4	$O(T)$
Lines 5–8	$O(mn(k + 1))$
Line 9	$O(mk + nk)$
Line 10	$O(mn)$
Line 11	$O(1)$
Line 12	$O(1)$
Total	$O(Tmnk)$

initializes δ with $O(1)$ of time. Line 4 enters the iteration and takes $O(T)$ of time. Line 5 through 8 obtains the predictions and the deviations with $O(mn(k+1))$ of time. Line 9 updates $\mathbf{H}$ and $\mathbf{V}$ and takes $O(mk+nk)$ of time. Line 10 computes the new MAE ξ on Ω_{tr} with $O(mn)$ of time. Line 11 recomputes δ with $O(1)$ of time. Line 12 updates ξ with $O(1)$ of time. Hence, the total time complexity is $O(mk+nk)+O(1)+O(1)+O(T) * (O(mn(k+1))+O(mk+nk)+O(mn)+O(1)+O(1)) = O(Tmnk)$. This completes the proof. $\square$

Table 3 describes the complexity of Algorithm 1 briefly. Our algorithm has the same complexity of time as L_1 and L_2 losses.

5. A running example

In this section, we use $\mathbf{X}$ to compare MF based on sigmoid-like and L_2 losses. Denote the matrices recovered from these two approaches by $\hat{\mathbf{X}}^{L_2}$ and $\hat{\mathbf{X}}^{S}$, respectively. Let further $\mathbf{E}^{L_2} = \mathbf{X} - \hat{\mathbf{X}}^{L_2}$ and $\mathbf{E}^{S} = \mathbf{X} - \hat{\mathbf{X}}^{S}$, i.e., they express the deviations of the recovered value from the original ones. With the original rating matrix given by Eq. (21), we have

$$\mathbf{E}^{L_2} = \begin{pmatrix}
\mathbf{1.16} & 0.07 & 0.90 & – & -0.34 & – & 0.16 & – & 0.29 & – & – & – & – & 0.24 & \mathbf{1.31} \\
– & 0.57 & 0.12 & 0.93 & – & 0.35 & – & – & – & – & 0.33 & – & 0.00 & – & – \\
– & 0.33 & 0.11 & 0.03 & – & 0.18 & – & 0.39 & 0.00 & 0.11 & – & – & 0.14 & – & – \\
– & – & 0.60 & \mathbf{1.05} & – & – & – & – & 0.17 & – & – & 0.10 & – & 0.84 & – \\
0.09 & 0.31 & – & 0.17 & – & 0.07 & 0.14 & – & 0.17 & 0.36 & – & 0.62 & 0.20 & 0.50 & – \\
– & – & – & 0.79 & – & 0.83 & 0.03 & – & 0.31 & 0.35 & 0.50 & – & – & 0.21 & – \\
0.34 & – & 0.00 & 0.59 & 0.21 & 0.40 & 0.05 & – & – & 0.18 & 0.01 & 0.51 & – & – & -0.57 \\
– & 0.11 & 0.76 & – & 0.17 & \mathbf{1.23} & 0.21 & – & – & 0.18 & – & \mathbf{-2.02} & 0.19 & 0.09 & 0.06 \\
– & – & \mathbf{-1.27} & 0.28 & – & 0.33 & – & 0.06 & – & 0.18 & – & 0.00 & 0.40 & – & 0.24 \\
\mathbf{1.08} & – & 0.06 & – & – & \mathbf{1.09} & – & 0.46 & 0.28 & 0.05 & – & – & 0.03 & – & 0.85
\end{pmatrix} \tag{40}$$

and

$$\mathbf{E}^{S} = \begin{pmatrix}
0.51 & 0.23 & 0.39 & – & -0.33 & – & 0.20 & – & 0.00 & – & – & – & – & -0.45 & 0.65 \\
– & 0.25 & 0.00 & 0.63 & – & 0.29 & – & – & – & – & 0.14 & – & 0.00 & – & – \\
– & 0.22 & 0.24 & 0.37 & – & 0.00 & – & 0.60 & 0.18 & 0.19 & – & – & 0.19 & – & – \\
– & – & 0.12 & 0.19 & – & – & – & – & 0.50 & – & – & 0.10 & – & \mathbf{1.47} & – \\
0.10 & 0.18 & – & 0.35 & – & 0.02 & 0.39 & – & 0.25 & 0.61 & – & 0.67 & 0.01 & 0.26 & – \\
– & – & – & 0.40 & – & 0.62 & 0.28 & – & 0.36 & 0.46 & 0.26 & – & – & 0.10 & – \\
0.52 & – & 0.42 & 0.02 & 0.26 & 0.10 & 0.00 & – & – & 0.03 & 0.04 & 0.38 & – & – & \mathbf{-2.50} \\
– & 0.00 & 0.69 & – & 0.15 & \mathbf{1.14} & 0.35 & – & – & 0.41 & – & \mathbf{-2.31} & 0.17 & 0.10 & 0.13 \\
– & – & \mathbf{-1.02} & 0.12 & – & 0.23 & – & 0.36 & – & 0.00 & – & 0.00 & 0.46 & – & 0.30 \\
0.07 & – & 0.51 & – & – & 0.51 & – & 0.16 & 0.31 & 0.37 & – & – & 0.07 & – & 0.08
\end{pmatrix}, \tag{41}$$

where unknown ratings are represented by '–'.

Let us consider the absolute values of these deviations. We have the following observations:

(1) Suppose that a deviation is large if its absolute value is not less than 1.0, and small otherwise. $\mathbf{E}^{L_2}$ has eight large deviations, located at positions $(0,0)$, $(0,14)$, $(3,4)$, $(7,5)$, $(7,11)$, $(8,2)$, $(9,0)$, and $(9,5)$. $\mathbf{E}^{S}$ has five large deviations, located at positions $(3,13)$, $(6,14)$, $(7,5)$, $(7,11)$, and $(8,2)$. In other words, MF based on sigmoid-like loss results in fewer data with large deviations.
(2) The average value of small deviations in $\mathbf{E}^{L_2}$ is 0.29. For $\mathbf{E}^{S}$, this value is only 0.26. In other words, MF based on sigmoid-like loss is more accurate for predictions with small deviations.
(3) The average value of large deviations in $\mathbf{E}^{L_2}$ is 1.28. For $\mathbf{E}^{S}$, this value is 1.69. In other words, MF based on sigmoid-like loss is less accurate for predictions with large deviations.
(4) The average value of the top-5 largest deviations in $\mathbf{E}^{L_2}$ is 1.40. For $\mathbf{E}^{S}$, this value is 1.69. After removing ratings corresponding to the top-5 largest deviations, the average deviation of $\mathbf{E}^{L_2}$ is 0.32. For $\mathbf{E}^{S}$, this value is 0.26. In other words, MF based on sigmoid-like loss is less accurate for top-5 worst predictions. If we take these top-5 data points as outliers, our algorithm is more accurate.

To sum up, observations (1), (2), and (4) show the advantages of our algorithm. Meanwhile, observations (3) and (4) show what we have paid for these advantages. This phenomenon fits with the "no free lunch" philosophy of machine learning. Note that the above analysis is based on the training set instead of the testing set. The predicted performance will be analyzed in Section 6.5.

6. Experiments

In this section, we present the experiment results to answer the following questions:

(1) Are there some hyper-parameter settings suitable for different datasets?
(2) Is the runtime consistent with the time complexity analysis?
(3) Are the prediction results on the testing set consistent with the observations on the running example?
(4) Can the sigmoid-like loss function be effectively combined with regularization terms and the validation mechanism?
(5) Is the proposed approach effective in enhancing performance?

Experiments are implemented in Java and run on the same machine (Intel i5-4460 CPU and 4 GB RAM).

6.1. Datasets

Experiments are undertaken on Movielens100K,[1] Amazon,[2] Yelp,[3] FilmTrust,[4] Hetrect[5] and EachMovie[6] datasets, which are widely used to test RSs [42,55,56].
Table 4 summarizes the characteristics of datasets in our experiments. The rating range of Amazon is [1.0..5.0] with a sparsity of 98.14%. All datasets are divided into three parts: 80% training, 10% testing, and 10% validation.

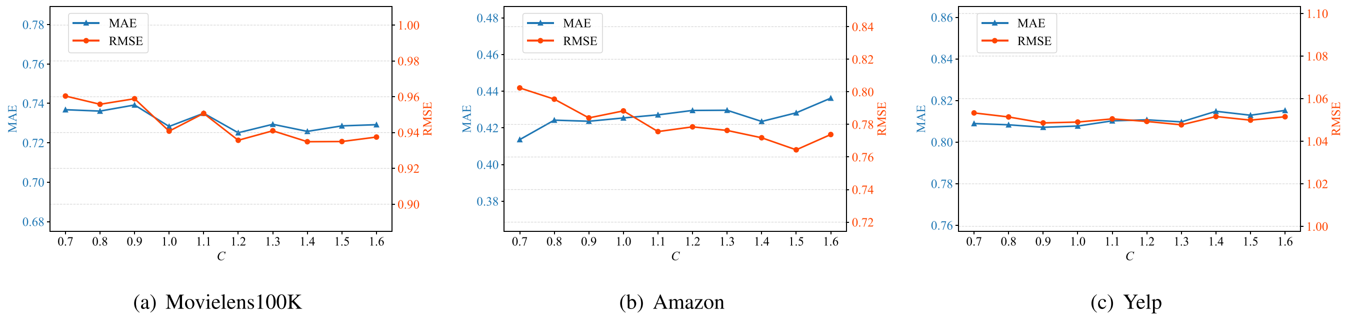
6.2. Parameter settings

A significant parameter in the sigmoid-like loss is C. The effect of C on MAE and RMSE on six datasets is depicted in Fig. 2. The left and right axes represent MAE and RMSE, respectively, with different values of C. Fig. 2(a) shows that for Movielens100K, the minimums of MAE and RMSE are achieved when $C = 1.4$; Similarly, Figs. 2(d) through 2(e) show that the optimal C for Amazon, Yelp, FilmTrust, Hetrect,

[1] https://grouplens.org/datasets/movielens/
[2] http://snap.stanford.edu/data/web-Amazon-links.html
[3] https://www.yelp.com/dataset
[4] https://guoguibing.github.io/librec/datasets.html
[5] https://grouplens.org/datasets/hetrec-2011/
[6] https://grouplens.org/datasets/eachmovie/

Table 4
Characteristics of the datasets used.

Datasets	Users	Items	Ratings	Rating range	Sparsity
Movielens100K	943	1,682	100,000	[1.0, 5.0]	93.70%
Amazon	1,094	1,673	34,120	[1.0, 5.0]	98.14%
Yelp	11,916	3,815	133,985	[1.0, 5.0]	99.71%
FilmTrust	1,508	2,071	35,497	[0.5, 4.5]	98.86%
Hetrect	1,058	7,657	388,293	[0.5, 5.0]	95.21%
EachMovie	72,916	1,628	2,811,983	[1.0, 6.0]	97.63%



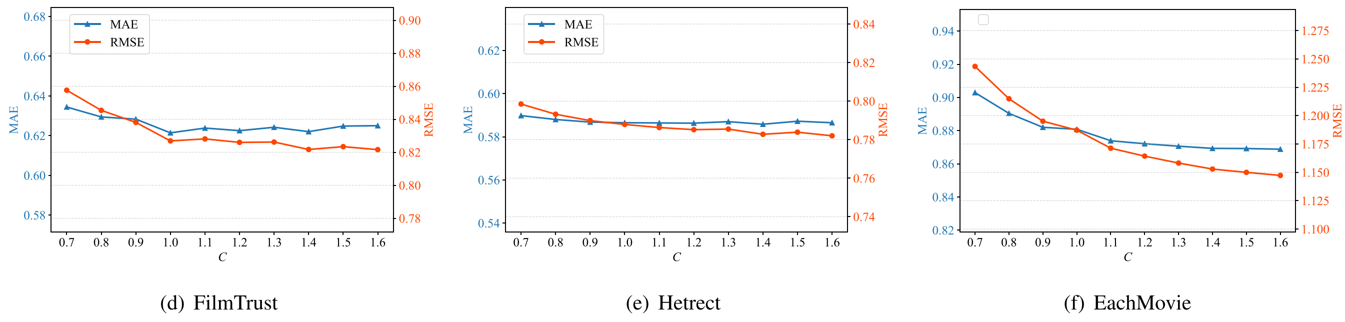


Fig. 2. The impact of C on MAE and RMSE.

and EachMovie is 1.5, 1.3, 1.4, 1.4, and 1.5, respectively. Fortunately, the results are smooth around the optimal settings. To avoid dataset dependencies, we set $C = 1.4$ in the remaining experiments.

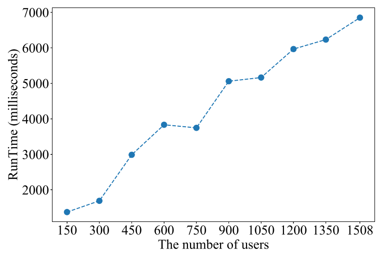
6.3. *The efficiency of our algorithm*

Fig. 3 demonstrates the runtime for varying proportions of users and items on FilmTrust. The runtime increases linearly wrt. the number of users or items. It verifies the overall complexity of $O(Tmnk)$. We choose $T = 1$ because varying numbers of users or items will result in different T. Fig. 3(a) shows that when the number of users increases from 150 to 1,508, the runtime increases from 1,373 to 6,851 s. Overhead processes such as data loading and space allocation are not included. We also see that as the number of users grows by 150, the average runtime grows by 608 s. Fig. 3(b) shows that when the number of items increases from 200 to 2,071, the runtime increases from 414 to 7,449 s. The average runtime increases by 781 s when the number of items increases by 200.

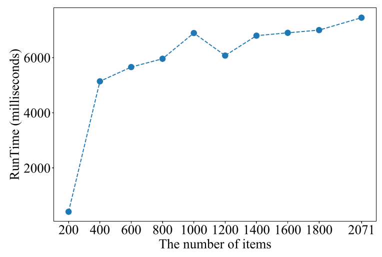
6.4. *Detailed comparison*

We now elaborate on the prediction results of the testing set, which are consistent with the observations on the running example presented in Section 5.

Table 5 lists the deviation comparison to cope with that of the running example. L_1RV, L_2RV, BaisRV, HuberRV, LogcoshRV, and MSLERV represent six MF algorithms based on different losses. In these algorithms, the individual losses are abs(ε_{ij}), ε_{ij}^2, $(\varepsilon_{ij} + \mu + b_i + b_j)^2$, L_{H} (Eq. (5)), L_{LC} (Eq. (7)), L_{MSLE} (Eq. (9)), and the sigmoid-like loss (Eq. (29)), respectively. Meanwhile, both L_2 regularization and the



(a) Runtime for different numbers of users. The number of items is fixed to 1,682.



(b) Runtime for different numbers of items. The number of users is fixed to 942.

Fig. 3. Runtime for different numbers of users and items.

Table 5
Deviations comparison. L_1RV, L_2RV, BaisRV, HuberRV, LogcoshRV, and MSLERV represent six MF algorithms based on different losses. L_1RV represents an MF algorithm based on L_1 loss, which considers both the regularization term and the validation set. The meanings of other algorithm abbreviations are the same as L_1RV. $\mathbf{E}_{te}$ is the deviation matrix of the testing set. $Z_>$ is the set of large deviations in $\mathbf{E}_{te}$, and $Z_<$ is the set of small deviations in $\mathbf{E}_{te}$. Φ is the set of outliers. $Z\backslash\Phi$ represents the set of deviations after removing the outliers. ↓ means that the smaller, the better. ↑ means that the bigger, the better. $car(\cdot)$ is the cardinality of a set. $\zeta(\cdot)$ is the average of all elements in a set.

Dataset	Algorithm	$car(Z_<)$ ↑	$car(Z_>)$ ↓	$\zeta(Z_<)$ ↓	$\zeta(Z_>)$ ↓	$\zeta(\Phi)$	$\zeta(Z\backslash\Phi)$ ↓
Movielens100K	L_1RV	6,031	3,836	0.4962	1.6603	2.5547	0.9051
	L_2RV	7,195	2,672	0.4474	1.5253	2.6490	0.6874
	BiasRV	7,130	2,737	0.4425	1.5295	2.7017	0.6908
	HuberRV	6,654	3,213	0.4629	1.6525	2.7912	0.7851
	LogcoshRV	6,643	3,224	0.4587	1.6072	2.6190	0.7737
	MSLERV	6,105	3,762	0.4963	1.6581	2.5362	0.8760
	SRV	7,254	2,613	0.4455	1.5224	2.6499	0.6785
Amazon	L_1RV	2,530	714	0.5847	1.9269	3.7435	0.8453
	L_2RV	2,809	435	0.2558	1.6932	3.4756	0.4117
	BiasRV	2,796	448	0.2415	1.7395	3.5421	0.4107
	HuberRV	2,702	698	0.5896	1.8368	3.2959	0.7916
	LogcoshRV	2,713	687	0.5918	1.8301	3.2718	0.7925
	MSLERV	2,750	650	0.5956	1.8299	3.2716	0.7955
	SRV	2,847	397	0.2509	1.6343	3.3672	0.3843
Yelp	L_1RV	6,962	5,020	0.3885	1.4573	2.8345	0.7628
	L_2RV	7,635	4,347	0.4332	1.5548	2.8992	0.7644
	BiasRV	7,715	4,267	0.4382	1.5621	2.9129	0.7622
	HuberRV	7,005	5,155	0.3799	1.4520	2.7385	0.7490
	LogcoshRV	7,019	5,141	0.3801	1.4522	2.7381	0.7493
	MSLERV	7,020	5,140	0.3801	1.4522	2.7378	0.7493
	SRV	7,731	4,251	0.4227	1.5176	2.8647	0.7357
FilmTrust	L_1RV	2,792	685	0.5220	1.5248	2.5089	0.6855
	L_2RV	2,759	718	0.4186	1.5387	2.7446	0.6100
	BiasRV	2,746	731	0.4142	1.5312	2.7102	0.6098
	HuberRV	2,807	670	0.4483	1.6934	3.1766	0.6395
	LogcoshRV	2,796	681	0.4811	1.5034	2.4997	0.6453
	MSLERV	2,764	713	0.5014	1.5157	2.5066	0.6720
	SRV	2,824	653	0.4240	1.5208	2.6298	0.5919
Hetrect	L_1RV	26,468	12,293	0.4522	1.5899	3.2399	0.7725
	L_2RV	32,166	6,595	0.4025	1.4696	2.5372	0.5514
	BiasRV	32,018	6,743	0.4040	1.4804	2.5952	0.5578
	HuberRV	30,554	8,207	0.4078	1.6860	3.3339	0.6210
	LogcoshRV	31,628	7,133	0.4101	1.4972	2.6779	0.5714
	MSLERV	26,269	12,492	0.4405	1.5598	2.8337	0.7336
	SRV	32,387	6,374	0.3986	1.4716	2.5712	0.5417
EachMovie	L_1RV	143,079	132,280	0.5186	2.0154	3.0410	1.1466
	L_2RV	181,315	94,044	0.4550	1.7057	3.2120	0.7998
	BiasRV	180,241	95,118	0.4540	1.7298	3.3391	0.8073
	HuberRV	146,134	129,225	0.4852	2.1376	3.9597	1.1278
	LogcoshRV	184,212	91,147	0.4207	1.7668	3.5194	0.7754
	MSLERV	139,829	135,530	0.5083	1.9970	3.2312	1.1380
	SRV	185,223	90,136	0.4514	1.7244	3.3288	0.7848

validation set are considered. In BaisRV, μ is the mean rating of the training data, b_i is the bias of u_i, and b_j is the bias of t_j. $\mathbf{E}_{te}$ is the deviation matrix of the testing set. We consider the absolute values of all deviations. Suppose that a deviation is large if its absolute value is not less than 1.0 and small otherwise. $Z_>$ is the set of large deviations in $\mathbf{E}_{te}$, and $Z_<$ is the set of small deviations in $\mathbf{E}_{te}$. We take the ratings with the top 10% deviations as the outliers, and Φ is the set of the outliers. $Z\backslash\Phi$ represents the set of deviations after removing the outliers. $\zeta(\cdot)$ is the average of all elements in a set. $car(\cdot)$ is the cardinality of a set. $\zeta(\cdot)$ is the average of all elements in a set. ↓ means that the smaller, the better. ↑ means that the bigger, the better.

The results indicate that:

(1) According to $car(Z_>)$ and $car(Z_<)$, SRV always has the least number of large deviations and, equivalently, the largest number of small deviations. In contrast, L_1RV and MSLERV perform the worst on all datasets, in the average ranking.

(2) According to $\zeta(Z_<)$ (in ascending order), the rankings of SRV on all datasets are 2, 2, 5, 3, 1, and 2, respectively. The average ranking is 2.50. Similarly, for L_1RV, L_2RV, BiasRV, HuberRV, LogcooshRV, and MSLERV, the average rankings are 5.83, 3.33, 2.66, 4.00, 3.83, and 5.67, respectively. So, SRV obtains the smallest $\zeta(Z_<)$. Note that the trend is the same as that of the running example.

(3) According to $\zeta(Z_>)$ (in ascending order), the rankings of $\zeta(Z_>)$ of SRV on all datasets are 1, 1, 5, 3, 2, and 2, respectively. The average ranking is 2.33. Similarly, the average rankings of L_1RV, L_2RV, BiasRV, HuberRV, LogcooshRV, and MSLERV are 5.67, 3.00, 4.00, 5.50, 3.33, and 4.00. So, SRV obtains the smallest $\zeta(Z_>)$. The trend is not the same as that of the running example. This shows that our SRV is more accurate for predictions with large deviations on the testing set.

(4) According to $\zeta(\Phi)$ (in ascending order), the ranking of $\zeta(\Phi)$ of SRV on all datasets are 5, 4, 5, 4, 2, and 4. The average ranking

Table 6
Performance comparison of S, SV, and SRV. S means that an MF algorithm on our sigmoid-like loss. SV means that an MF algorithm on our sigmoid-like loss, which considers the validation set is considered. SRV means that an MF algorithm on our sigmoid-like loss, which considers the regularization and the validation set. ↓ means that the smaller, the better. ↑ means that the bigger, the better. We truncate the ranked list at ten for the last three measures. Average values are shown alongside their standard deviation after ten runs. The bold values are the best results.

Dataset	Algorithm	MAE↓	RMSE↓	HR↑	MAP↑	NDCG↑
Movielens100K	S	$0.7473_{\pm 0.0066}$	$0.9759_{\pm 0.0116}$	$0.1248_{\pm 0.0173}$	$0.0322_{\pm 0.0049}$	$0.4374_{\pm 0.0142}$
	SV	$0.7359_{\pm 0.0079}$	$0.9536_{\pm 0.0132}$	$0.2957_{\pm 0.0478}$	$0.1088_{\pm 0.0225}$	$0.5419_{\pm 0.0296}$
	SRV	$\mathbf{0.7292_{\pm 0.0047}}$	$\mathbf{0.9383_{\pm 0.0045}}$	$\mathbf{0.3049_{\pm 0.0319}}$	$\mathbf{0.1155_{\pm 0.0158}}$	$\mathbf{0.5651_{\pm 0.0186}}$
Amazon	S	$0.5209_{\pm 0.0253}$	$0.9446_{\pm 0.0951}$	$0.0281_{\pm 0.0063}$	$0.0088_{\pm 0.0026}$	$0.6112_{\pm 0.0833}$
	SV	$0.5088_{\pm 0.0192}$	$0.8816_{\pm 0.0716}$	$0.0342_{\pm 0.0120}$	$\mathbf{0.0154_{\pm 0.0065}}$	$0.7381_{\pm 0.0506}$
	SRV	$\mathbf{0.4644_{\pm 0.0104}}$	$\mathbf{0.7498_{\pm 0.0217}}$	$\mathbf{0.0362_{\pm 0.0095}}$	$0.0127_{\pm 0.0037}$	$\mathbf{0.7957_{\pm 0.0375}}$
Yelp	S	$1.0718_{\pm 0.0107}$	$1.4614_{\pm 0.0190}$	$0.0099_{\pm 0.0015}$	$0.0030_{\pm 0.0005}$	$0.4395_{\pm 0.0585}$
	SV	$0.8156_{\pm 0.0080}$	$1.0554_{\pm 0.0082}$	$\mathbf{0.0669_{\pm 0.0027}}$	$\mathbf{0.0193_{\pm 0.0015}}$	$0.4644_{\pm 0.0185}$
	SRV	$\mathbf{0.8032_{\pm 0.0041}}$	$\mathbf{1.0399_{\pm 0.0051}}$	$0.0621_{\pm 0.0047}$	$0.0152_{\pm 0.0015}$	$\mathbf{0.4688_{\pm 0.0271}}$
FilmTrust	S	$0.6819_{\pm 0.0063}$	$0.9243_{\pm 0.0114}$	$0.0436_{\pm 0.0114}$	$0.0110_{\pm 0.0039}$	$0.4090_{\pm 0.0330}$
	SV	$0.6257_{\pm 0.0056}$	$0.8228_{\pm 0.0072}$	$\mathbf{0.4836_{\pm 0.0239}}$	$0.3126_{\pm 0.0159}$	$\mathbf{0.8074_{\pm 0.0124}}$
	SRV	$\mathbf{0.6174_{\pm 0.0078}}$	$\mathbf{0.8104_{\pm 0.0078}}$	$0.4610_{\pm 0.0196}$	$\mathbf{0.3291_{\pm 0.0159}}$	$0.8020_{\pm 0.0149}$
Hetrect	S	$0.5930_{\pm 0.0029}$	$0.7946_{\pm 0.0048}$	$0.0251_{\pm 0.0107}$	$0.0056_{\pm 0.0023}$	$0.3831_{\pm 0.0466}$
	SV	$0.5878_{\pm 0.0020}$	$0.7839_{\pm 0.0026}$	$\mathbf{0.4470_{\pm 0.0721}}$	$\mathbf{0.2041_{\pm 0.0475}}$	$0.6156_{\pm 0.0507}$
	SRV	$\mathbf{0.5800_{\pm 0.0025}}$	$\mathbf{0.7720_{\pm 0.0037}}$	$0.4138_{\pm 0.0396}$	$0.1949_{\pm 0.0210}$	$\mathbf{0.6546_{\pm 0.0342}}$
EachMovie	S	$0.9096_{\pm 0.0087}$	$1.2108_{\pm 0.0128}$	$\mathbf{0.2788_{\pm 0.0104}}$	$\mathbf{0.1184_{\pm 0.0029}}$	$\mathbf{0.5923_{\pm 0.0125}}$
	SV	$0.8861_{\pm 6.3864}$	$1.1835_{\pm 0.0016}$	$0.2141_{\pm 0.0101}$	$0.0872_{\pm 0.0046}$	$0.5729_{\pm 0.0044}$
	SRV	$\mathbf{0.8689_{\pm 0.0014}}$	$\mathbf{1.1498_{\pm 0.0025}}$	$0.2013_{\pm 0.0035}$	$0.0812_{\pm 0.0025}$	$0.5711_{\pm 0.0068}$

is 4.00. Similarly, the average rankings of L_1RV, L_2RV, BiasRV, HuberRV, LogcooshRV, and MSLERV are 3.83, 4.00, 5.33, 5.67, 3.00, and 2.16. So, $\zeta(\Phi)$ of SRV is only larger than that of L_1RV LogcoshRV and MSLERV. This shows that our SRV is accurate for top-10% worst predictions. The trend is not the same as that of running example.

(5) According to $\zeta(Z\backslash\Phi)$ (in ascending order), the ranking of $\zeta(Z\backslash\Phi)$ of SRV on all datasets are 1, 1, 1, 1, 1, and 2. The average ranking of SRV is 1.17. Similarly, the average rankings of L_1RV, L_2RV, BiasRV, HuberRV, LogcooshRV, and MSLERV are 6.83, 3.33, 3.16, 4.17, 3.66, and 5.50, respectively. This shows that our SRV outperforms the other six algorithms. The trend is the same as that of running example.

According to (2)) and (4)), our experimental results maintain good consistency with the theoretical analysis in Section 4.1.

6.5. Performance comparison

In this section, we introduce the ablation experiment, the comparison with other functions, and the Friedman test.

6.5.1. Ablation experiment

Table 6 demonstrates the performance comparison of our sigmoid-like loss with the regularization term and the validations set. S means that an MF algorithm on our sigmoid-like loss. SV means that an MF algorithm on our sigmoid-like loss, which considers the validation set is considered. SRV means that an MF algorithm on our sigmoid-like loss, which considers the regularization and the validation set. ↓ means that the smaller, the better. ↑ means that the bigger, the better. We truncate the ranked list at ten for the last three measures.

On all datasets, SRV has the best MAE and RMSE. SV and S come in second and third, respectively. It can be seen that validation and regulation improve the performance of rating predictions. Except for Amazon, Movielens100K, and EachMovie, SV has the best HR on three of the six datasets. Second and third place, respectively, go to SRV and S. It can be seen that the regulation may not help to improve HR.

For the average ranking of MAP, SV gets first. Second and third place, respectively, go to SRV and S. It can be seen that the validation set aids in the improvement of HR as well as the MAP. In terms of NDCG, S, SV, and SRV have average rankings of 2.66, 1.83, and 1.50, respectively. As a result, SRV comes first. For example, on Movielens100K, SRV and SV take first and second place for MAE, RMSE, and NDCG, respectively. MAE and RMSE of SRV are 0.91%, 1.60% lower, respectively, than those obtained by SV. NDCG of SRV is 1.84% higher than that obtained by SV. In conclusion, SRV outperforms S and SV.

6.5.2. Comparison with other loss functions

Table 7 compares our sigmoid-like loss function with the other ones by five measures: MAE, RMSE, HR, MAP, and NDCG. The meanings of L_1RV, L_2RV, BaisRV, HuberRV, LogcoshRV, and MSLERV are the same as Table 5. ↓ means that the smaller, the better. ↑ means that the bigger, the better. We truncate the ranked list at ten for the last three measures. Average values are shown alongside their standard deviation after ten runs. The bold values are the best results.

L_1RV does not perform well compared with the other six ones on each measure. On Movielens100K, for example, L_1RV has the second largest MAE and RMSE. It has the smallest HR and MAP and the third-smallest NDCG. The reason for this is that a small ε_{ij} has the same gradient as a larger ε_{ij} for L_1 loss. This is not conducive to the convergence of the function or the learning of the model.

L_2RV performs worse than SRV and LogcoshRV in terms of MAE, and it performs worse than SRV in terms of RMSE, HR, MAP, and NDCG. It only performs better than L_1RV in terms of HR, MAP, and NDCG. On Amazon, for example, L_2RV has the smallest HR and MAP. The reason for this is that L_2 loss has a linear relationship with ε_{ij} after partial derivation, and it is sensitive to outliers.

BiasRV performs worse than SRV, L_2RV, and LogcoshRV in terms of MAE, and it performs worse than SRV in terms of RMSE and NDCG. On Hetrect, for example, BiasRV has the smallest MAP and NDCG. Compared with L_2 loss, the bias loss has a larger partial derivation. So, it is more sensitive to outliers.

HuberRV achieves the largest HR and MAP on most datasets. In the case of Amazon, HuberRV increases MAP by 175.70%, 186.40%, 178.30%, 41.14%, 82.09%, and 132.28%, respectively, when compared to the other six ones. The reason for this is that a large loss of the Huber loss has only a small partial derivation. Although there can be some punishment for outliers, this is not enough.

LogcoshRV achieves the largest HR, MAP, and NDCG on Hetrect. However, it obtains the smallest NDCG on Movielens100K, and the

Table 7
Performance comparison with other loss functions. L_1RV, L_2RV, BaisRV, HuberRV, LogcoshRV, and MSLERV represent six MF algorithms based on different losses. L_1RV represents an MF algorithm based on L_1 loss, which considers both the regularization term and the validation set. The meanings of other algorithm abbreviations are the same as L_1RV. ↓ means that the smaller the values of MAE and RMSE, the better. ↑ means that the bigger the values of HR, MAP, and NDCG, the better. We truncate the ranked list at ten for the last three measures. Average values are shown alongside their standard deviation after ten runs. The bold values are the best results.

Dataset	Algorithm	MAE↓	RMSE↓	HR↑	MAP↑	NDCG↑
Movielens100K	L_1RV	$0.9442_{\pm0.0047}$	$1.1236_{\pm0.0051}$	$0.0869_{\pm0.0425}$	$0.0267_{\pm0.0164}$	$0.4599_{\pm0.0573}$
	L_2RV	$0.7406_{\pm0.0034}$	$0.9521_{\pm0.0054}$	$0.2498_{\pm0.0253}$	$0.0877_{\pm0.0123}$	$0.5254_{\pm0.0215}$
	BiasRV	$0.7382_{\pm0.0065}$	$0.9539_{\pm0.0114}$	$0.2583_{\pm0.0187}$	$0.0901_{\pm0.0093}$	$0.5238_{\pm0.0199}$
	HuberRV	$0.8485_{\pm0.0016}$	$1.0950_{\pm0.0013}$	$\mathbf{0.4053_{\pm0.0072}}$	$\mathbf{0.1757_{\pm0.0038}}$	$\mathbf{0.6033_{\pm0.0114}}$
	LogcoshRV	$0.7438_{\pm0.0111}$	$0.9784_{\pm0.0238}$	$0.1366_{\pm0.0981}$	$0.0431_{\pm0.0341}$	$0.4170_{\pm0.1048}$
	MSLERV	$0.9596_{\pm0.0870}$	$1.1575_{\pm0.1267}$	$0.1626_{\pm0.0916}$	$0.0413_{\pm0.0240}$	$0.4559_{\pm0.0317}$
	SRV	$\mathbf{0.7292_{\pm0.0047}}$	$\mathbf{0.9383_{\pm0.0045}}$	$0.3049_{\pm0.0319}$	$0.1155_{\pm0.0158}$	$0.5651_{\pm0.0186}$
Amazon	L_1RV	$0.8371_{\pm0.0089}$	$1.0384_{\pm0.0156}$	$0.0310_{\pm0.0047}$	$0.0107_{\pm0.0022}$	$0.6548_{\pm0.0420}$
	L_2RV	$0.4983_{\pm0.0143}$	$0.7952_{\pm0.0276}$	$0.0279_{\pm0.0074}$	$0.0103_{\pm0.0023}$	$0.7066_{\pm0.0748}$
	BiasRV	$0.5043_{\pm0.0103}$	$0.8235_{\pm0.0257}$	$0.0291_{\pm0.0056}$	$0.0106_{\pm0.0023}$	$0.6837_{\pm0.0456}$
	HuberRV	$0.7209_{\pm0.0065}$	$1.1176_{\pm0.0073}$	$\mathbf{0.0604_{\pm0.0063}}$	$\mathbf{0.0295_{\pm0.0030}}$	$0.7164_{\pm0.0304}$
	LogcoshRV	$0.4820_{\pm0.0114}$	$0.8724_{\pm0.0424}$	$0.0576_{\pm0.0131}$	$0.0209_{\pm0.0039}$	$0.6327_{\pm0.0472}$
	MSLERV	$0.8443_{\pm9.2675}$	$1.0528_{\pm0.0011}$	$0.0520_{\pm0.0089}$	$0.0162_{\pm0.0039}$	$0.6171_{\pm0.0442}$
	SRV	$\mathbf{0.4644_{\pm0.0104}}$	$\mathbf{0.7498_{\pm0.0217}}$	$0.0362_{\pm0.0095}$	$0.0127_{\pm0.0037}$	$\mathbf{0.7957_{\pm0.0375}}$
Yelp	L_1RV	$0.8263_{\pm0.0072}$	$1.0585_{\pm0.0091}$	$0.0104_{\pm0.0039}$	$0.0029_{\pm0.0012}$	$0.4573_{\pm0.0471}$
	L_2RV	$0.8512_{\pm0.0095}$	$1.0968_{\pm0.0102}$	$0.0346_{\pm0.0024}$	$0.0106_{\pm0.0011}$	$0.4522_{\pm0.0407}$
	BiasRV	$0.8618_{\pm0.0094}$	$1.1119_{\pm0.0102}$	$0.0380_{\pm0.0025}$	$0.0125_{\pm0.0009}$	$0.4667_{\pm0.0492}$
	HuberRV	$0.8281_{\pm0.0023}$	$1.0670_{\pm0.0012}$	$0.0606_{\pm0.0019}$	$\mathbf{0.0209_{\pm0.0010}}$	$0.4631_{\pm0.0187}$
	LogcoshRV	$0.8259_{\pm0.0045}$	$1.0640_{\pm0.0039}$	$0.0577_{\pm0.0041}$	$0.0196_{\pm0.0014}$	$0.4671_{\pm0.0338}$
	MSLERV	$0.8340_{\pm0.0022}$	$1.0686_{\pm0.0033}$	$0.0382_{\pm0.0031}$	$0.0125_{\pm0.0021}$	$\mathbf{0.4812_{\pm0.0470}}$
	SRV	$\mathbf{0.8032_{\pm0.0041}}$	$\mathbf{1.0399_{\pm0.0051}}$	$\mathbf{0.0621_{\pm0.0047}}$	$0.0152_{\pm0.0015}$	$0.4688_{\pm0.0271}$
FilmTrust	L_1RV	$0.7192_{\pm0.0080}$	$0.9220_{\pm0.0123}$	$0.0316_{\pm0.0262}$	$0.0159_{\pm0.0167}$	$0.5466_{\pm0.1030}$
	L_2RV	$0.6412_{\pm0.0085}$	$0.8480_{\pm0.0096}$	$0.4298_{\pm0.0081}$	$0.2447_{\pm0.0106}$	$0.7339_{\pm0.0168}$
	BiasRV	$0.6527_{\pm0.0118}$	$0.8656_{\pm0.0161}$	$0.4575_{\pm0.0143}$	$0.2785_{\pm0.0100}$	$0.7709_{\pm0.0161}$
	HuberRV	$0.6883_{\pm0.0012}$	$0.9511_{\pm0.0022}$	$\mathbf{0.5317_{\pm0.0190}}$	$\mathbf{0.3324_{\pm0.0114}}$	$\mathbf{0.8140_{\pm0.0111}}$
	LogcoshRV	$0.6217_{\pm0.0036}$	$0.8200_{\pm0.0044}$	$0.4434_{\pm0.0218}$	$0.2836_{\pm0.0120}$	$0.7959_{\pm0.0078}$
	MSLERV	$0.6989_{\pm0.0076}$	$0.8994_{\pm0.0129}$	$0.2832_{\pm0.0346}$	$0.1648_{\pm0.0208}$	$0.7871_{\pm0.0085}$
	SRV	$\mathbf{0.6174_{\pm0.0078}}$	$\mathbf{0.8104_{\pm0.0078}}$	$0.4610_{\pm0.0196}$	$0.3291_{\pm0.0159}$	$0.8020_{\pm0.0149}$
Hetrect	L_1RV	$0.7937_{\pm0.0033}$	$1.0057_{\pm0.0042}$	$0.0811_{\pm0.0497}$	$0.0279_{\pm0.0212}$	$0.4759_{\pm0.0486}$
	L_2RV	$0.5906_{\pm0.0035}$	$0.7829_{\pm0.0053}$	$0.1579_{\pm0.0270}$	$0.0501_{\pm0.0130}$	$0.4816_{\pm0.0309}$
	BiasRV	$0.5942_{\pm0.0043}$	$0.7887_{\pm0.0056}$	$0.0924_{\pm0.0142}$	$0.0236_{\pm0.0043}$	$0.4258_{\pm0.0174}$
	HuberRV	$0.6763_{\pm0.0024}$	$0.9457_{\pm0.0034}$	$0.0938_{\pm0.0088}$	$0.0339_{\pm0.0029}$	$0.5052_{\pm0.0505}$
	LogcoshRV	$0.6086_{\pm0.0012}$	$0.8083_{\pm0.0012}$	$\mathbf{0.5523_{\pm0.0188}}$	$\mathbf{0.2852_{\pm0.0134}}$	$\mathbf{0.6999_{\pm0.0113}}$
	MSLERV	$0.8199_{\pm0.1227}$	$1.0486_{\pm0.1691}$	$0.3197_{\pm0.1809}$	$0.0792_{\pm0.0486}$	$0.4554_{\pm0.0314}$
	SRV	$\mathbf{0.5800_{\pm0.0025}}$	$\mathbf{0.7720_{\pm0.0037}}$	$0.4138_{\pm0.0396}$	$0.1949_{\pm0.0210}$	$0.6546_{\pm0.0342}$
EachMovie	L_1RV	$1.2374_{\pm0.0008}$	$1.5640_{\pm0.0006}$	$0.1017_{\pm0.0080}$	$0.0358_{\pm0.0044}$	$0.5142_{\pm0.0183}$
	L_2RV	$0.8833_{\pm0.0009}$	$1.1552_{\pm0.0019}$	$0.1778_{\pm0.0030}$	$0.0692_{\pm0.0015}$	$0.5548_{\pm0.0049}$
	BiasRV	$0.8966_{\pm0.0019}$	$1.1823_{\pm0.0038}$	$0.1917_{\pm0.0034}$	$0.0762_{\pm0.0013}$	$0.5637_{\pm0.0030}$
	HuberRV	$1.2499_{\pm0.0094}$	$1.6264_{\pm0.0249}$	$\mathbf{0.4939_{\pm0.0036}}$	$\mathbf{0.2270_{\pm0.0024}}$	$\mathbf{0.6447_{\pm0.0071}}$
	LogcoshRV	$0.8690_{\pm0.0016}$	$1.1843_{\pm0.0017}$	$0.1997_{\pm0.0054}$	$0.0785_{\pm0.0032}$	$0.5601_{\pm0.0044}$
	MSLERV	$1.8522_{\pm0.5292}$	$2.2725_{\pm0.5968}$	$0.1188_{\pm0.0869}$	$0.0322_{\pm0.0244}$	$0.4569_{\pm0.0212}$
	SRV	$\mathbf{0.8689_{\pm0.0014}}$	$\mathbf{1.1498_{\pm0.0025}}$	$0.2013_{\pm0.0035}$	$0.0812_{\pm0.0025}$	$0.5711_{\pm0.0068}$

second-smallest HR and NDCG on Movielens100K and Amazon, respectively. In the case of Movielens100K, LogcoshRV decreases NDCG by 9.33%, 20.63%, 20.39%, 30.88%, 8.53%, and 26.20%, respectively, when compared to the other six ones. The reason for this is that the larger ϵ_{ij}s have the same gradient for Logcosh loss. This is not conducive to the convergence of the function or the learning of the model.

MSLERV achieves the largest NDCG on Yelp. However, it obtains the largest MAE on Movielens100K, Amazon, Hetrect and EachMovie, and the second largest MAE on FilmTrust. In addition, it also obtains the largest RMSE on Movielens100K, Hetrect, and EachMovie, and the second largest RMSE on Amazon. In the case of Hetrect, MSLERV increases MAE by 3.30%, 38.82%, 37.98%, 21.23%, 34.72%, and 41.36%, respectively, when compared to the other six ones. MSLERV increases RMSE by 4.26%, 33.94%, 32.95%, 10.88%, 29.73%, and 35.83%, respectively.

Our SRV achieves the smallest MAE and RMSE on all datasets. In the case of Movielens100K, SRV decreases MAE by 22.77%, 1.54%, 1.22%, 14.06%, 1.96%, and 24.01%, respectively, when compared to the other six losses. SRV decreases RMSE by 16.49%, 1.45%, 1.63%, 14.31%, 4.10%, and 18.94%, respectively. We believe that the sigmoid-like loss function reduces the impact of outliers on all losses to some extent, improving fitting and prediction accuracy. As a result, SRV has a significant advantage in rating predictions.

Our SRV achieves the largest HR on Yelp, the second largest on Movielens100K, FilmTrust, Hetrect, and EachMovie, and comes in fourth largest on Amazon. In the case of Yelp, SRV improves HR by over five times, 79.48%, 63.42%, 2.47%, 7.62%, and 62.56% respectively, when compared to the other six losses. Our SRV has the second-largest MAP on Movielens100K, FilmTrust, Hetrect, and EachMovie, the third largest on Yelp, and comes in fourth largest on Amazon. Our SRV achieves the largest NDCG on Amazon and the second largest on Movielens100K, Yelp, FilmTrust, Hetrect, and EachMovie. For Amazon, our NDCG increases by 21.51%, 12.61%, 16.38%, 11.07%, 25.76%, and 28.94%, respectively, when compared to the other six losses. As a result, SRV also has competitive advantages in ranking predictions.

Table 8
Average rankings of algorithms.

Algorithm	MAE	RMSE	HR	MAP	NDCG
L_1RV	5.50	5.00	6.67	6.33	5.67
L_2RV	3.33	2.83	5.17	5.17	4.67
BiasRV	4.00	3.83	4.50	4.67	4.50
HuberRV	5.00	5.67	**1.83**	**1.67**	2.16
LogcoshRV	2.66	3.50	3.16	2.66	4.00
MSLERV	6.50	6.17	4.50	4.83	5.17
SRV	**1.00**	**1.00**	2.16	2.50	**1.83**

Table 9
The Friedman statistic τ_Fs ($z = 7$, $N = 6$) on different measures.

Evaluation measure	τ_F
MAE	14.0852
RMSE	10.8113
HR	8.6158
MAP	6.7155
NDCG	4.3030

Table 8 shows the average rankings of different algorithms. L_1RV achieves average rankings of 5.50, 5.00, 6.67, 6.33, and 5.67 on all five measures, respectively. L_2RV achieves average rankings of 3.33, 2.83, 5.17, 5.17, and 4.67, respectively. BiasRV achieves average rankings of 4.00, 3.83, 4.50, 4.67, and 4.50, respectively. HuberRV achieves average rankings of 5.00, 5.67, 1.83, 1.67, and 2.16, respectively. LogcoshRV achieves average rankings of 2.66, 3.50, 3.16, 2.66, and 4.00, respectively. MSLERV achieves average rankings of 6.50, 6.17, 4.50, 4.83, and 5.17, respectively. SRV achieves average rankings of 1.00, 1.00, 2.16, 2.50, and 1.83, respectively.

In summary, SRV ranks first on MAE, RMSE, and NDCG, and outperforms the other six ones. SRV ranks second on HR and MAP and outperforms L_1RV, L_2RV, BiasRV, LogcoshRV, and MSLERV.

6.5.3. Friedman test

In this paper, we use the Friedman test to judge whether the performance of all algorithms is the same [57]. To achieve this goal, it is necessary to judge whether the Friedman statistic τ_F of an algorithm is less than the critical value. The Friedman statistic τ_F is

$$\tau_F = \frac{(N-1)\chi^2}{N(z-1)-\chi^2}, \tag{42}$$

where

$$\chi^2 = \frac{12N}{z(z+1)}\left(\sum_{i=1}^{z} r_i^2 - \frac{z(z+1)^2}{4}\right), \tag{43}$$

z and N are the numbers of comparison algorithms and datasets, respectively, and r_i denotes the average ranking of the ith algorithm under one evaluation measure. It follows the F-distribution with degrees of freedom $z-1$ and $(z-1)(N-1)$.

Table 9 shows the Friedman statistic τ_Fs ($z = 7$, $N = 6$) on different measures. When the significance level α is 0.05, the critical value of the Friedman test is 2.4205. Obviously, the Friedman statistic of measure is larger than the critical value. So, the hypothesis that all algorithms have the same performance is rejected. To distinguish the algorithms better, we need to use the post-hoc test.

In this paper, we use the Bonferroni–Dunn test as the post-hoc test to distinguish whether SRV is significantly different from other algorithms on each measure [58]. The critical difference (CD) of average ranking is

$$CD = q_\alpha\sqrt{\frac{z(z+1)}{6N}}. \tag{44}$$

Fig. 4 demonstrates the CD diagram on each evaluation measure. Here, the higher the average ranking, the farther to the right. For the Bonferroni–Dunn test, when α is 0.05, q_α is 2.638, and CD is 3.2901. In each subfigure of Fig. 4, a thick line connects an algorithm with SRV if the difference in average ranking between them is less than CD under one evaluation measure. Any algorithm that has no connection with SRV is thought to have significantly different performance from SRV.

Fig. 4(a) demonstrates that our SRV significantly outperforms MSLERV, L_1RV, and HuberRV on MAE. SRV shows comparable performance to BiasRV, L_2RV, and LogcoshRV. Fig. 4(b) demonstrates that our SRV significantly outperforms MSLERV, HuberRV, and L_1RV on RMSE. SRV shows comparable performance to BiasRV, LogcoshRV, and L_2RV. Fig. 4(c) demonstrates that our SRV significantly outperforms L_1RV and L_2RV on HR. SRV shows comparable performance to BiasRV, MSLERV, LogcoshRV, and HuberRV. Fig. 4(d) demonstrates that our SRV significantly outperforms L_1RV and L_2RV on MAP. SRV shows comparable performance to BiasRV, MSLERV, LogcoshRV, and HuberRV. Fig. 4(e) demonstrates that our SRV significantly outperforms L_1RV and MSLERV on NDCG. SRV shows comparable performance to BiasRV, L_2RV, LogcoshRV, and HuberRV. In summary, SRV has competitiveness on each measure.

6.6. Discussions

Now we will answer the questions in the section.

(1) The best settings for the value of C are nearly identical in different datasets. $C = 1.4$ is an appropriate value for all datasets.
(2) The runtime is consistent with the time complexity analysis. It is linear in terms of the number of users and items.
(3) The prediction results on the testing set of SRV is the same as that of the running example in terms of $car(Z_>)$, $car(Z_<)$, $\zeta(Z_<)$ and $\zeta(Z\backslash\Phi)$. It is not the same as that of the running example in terms of $\zeta(Z_>)$ and $\zeta(\Phi)$. This shows that our SRV is more accurate for predictions with large deviations on the testing set, and our SRV is accurate for top-10% worst predictions.
(4) The sigmoid-like loss function can be effectively combined with regularization terms and the validation mechanism. The performance of SRV outperforms S and SV.
(5) Our algorithm is effective. According to the average ranking, SRV ranks first on MAE, RMSE, and NDCG and outperforms the other six. SRV ranks second on HR and MAP and outperforms L_1RV, L_2RV, BiasRV, LogcoshRV, and MSLERV.

The experimental results reflect that our loss can better reveal the characteristics of each evaluation measure. Both MAE and RMSE calculate the difference between the predicted ratings and the original ones. The sigmoid-like loss can be used to suppress the impact of outliers. The greater the difference between the predicted ratings and the original ones, the stronger the suppression. Therefore, it performs well on MAE and RMSE, which is in line with the theoretical analysis in Section 4.1. NDCG, HR and MAP are measures for evaluating the quality of recommendation lists. Suppressing outliers can affect the ranking of recommendations, but it may have a positive or negative impact. In terms of evaluation granularity, NDCG can more accurately reflect the quality of recommendation ranking than HR and MAP. However, these three measures cannot be analyzed theoretically to determine which one is better for our loss compared to other losses. Through experimental observation, we ranked first in terms of NDCG and second in terms of HR and MAP, which indirectly proves our superiority.

7. Conclusions and future work

The main contribution of the paper is the new loss function instead of a recommendation algorithm. We have proposed it to control the loss of individual prediction. It can control the impact of outliers, which

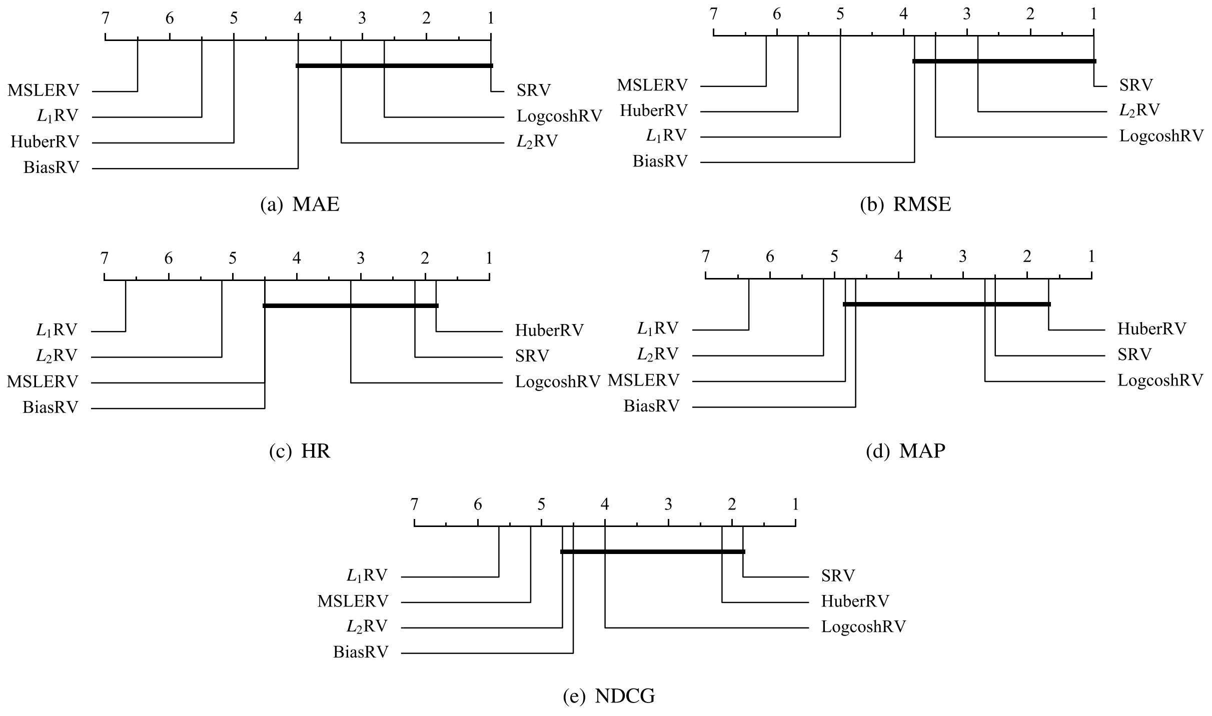


Fig. 4. Performance comparison of loss functions in Bonferroni–Dunn test (the significance level $\alpha = 0.05$). A thick line indicates that the difference in the average ranking between an algorithm and SVR is less than CD.

have a greater impact on the popular loss functions. Meanwhile, it considers the regularization and the validation data to overcome the overfitting. Experimental results show that recommendation performance benefits from our loss function.

In the future, we will apply our loss function to some advanced approaches, such as cross-domain recommendation (CDR) [59,60]. CDR learns user tastes from the source domain (e.g., movie) and migrates to the target domain (e.g., book). We hope that our loss function also improves the recommendation quality of these approaches.

CRediT authorship contribution statement

Yuan-Yuan Xu: Methodology, Software, Writing – original draft. **Hui Xiao:** Software, Validation, Writing – review & editing. **Heng-Ru Zhang:** Investigation, Resources. **Wei-Zhi Wu:** Supervision, Funding acquisition. **Fan Min:** Conceptualization, Supervision, Writing – review & editing.

Declaration of competing interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work. There is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (12371466), the Central Government Funds of Guiding Local Scientific and Technological Development, China (2021ZYD0003), the Open Project of Key Laboratory of Oceanographic Big Data Mining and Application of Zhejiang Province, China (OBMA202005) and the Nanchong Municipal Government-Universities Scientific Cooperation Project, China (23XNSYSX0062, SXHZ051).

References

[1] J. Gemmell, T. Schimoler, M. Ramezani, B. Mobasher, Adapting k-nearest neighbor for tag recommendation in folksonomies, in: ITWP, 2009.
[2] D. Adeniyi, Z.-Q. Wei, Y.-Q. Yang, Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method, Appl. Comput. Inf. 12 (1) (2016) 90–108.
[3] Y.-D. Xia, G. Di Fabbrizio, S. Vaibhav, A. Datta, A content-based recommender system for E-commerce offers and coupons, in: SIGIR Workshop on ECommerce, 2017.
[4] Y.-P. Gu, B. Zhao, D. Hardtke, Y.-Z. Sun, Learning global term weights for content-based recommender systems, in: WWW, 2016, pp. 391–400.
[5] X.-P. Li, J. She, Collaborative variational autoencoder for recommender systems, in: KDD, 2017, pp. 305–314.
[6] Y.-J. Liu, C.-Y. Liang, F. Chiclana, J. Wu, A knowledge coverage-based trust propagation for recommendation mechanism in social network group decision making, Appl. Soft Comput. 101 (2021) 107005.
[7] L. Zheng, N.-C. Guo, W.-H. Chen, J. Yu, D.-Z. Jiang, Sentiment-guided sequential recommendation, in: SIGIR, 2020, pp. 1957–1960.
[8] R.L. Rosa, G.M. Schwartz, W.V. Ruggiero, D.Z. Rodríguez, A knowledge-based recommendation system that includes sentiment analysis and deep learning, IEEE Trans. Ind. Inform. 15 (4) (2019) 2124–2135.
[9] W.-Q. Lei, G.-Y. Zhang, X.-N. He, Y.-S. Miao, X. Wang, L. Chen, T.-S. Chua, Interactive path reasoning on graph for conversational recommendation, in: KDD, 2020, pp. 2073–2083.
[10] H.-R. Zhang, F. Min, Three-way recommender systems based on random forests, Knowl.-Based Syst. 91 (2016) 275–286.
[11] H.-R. Zhang, F. Min, B. Shi, Regression-based three-way recommendation, Inform. Sci. 378 (2017) 444–461.
[12] P.-J. Ren, Z.-M. Chen, J. Li, Z.-C. Ren, J. Ma, M. De Rijke, Repeatnet: A repeat aware neural recommendation machine for session-based recommendation, in: AAAI, vol. 33, (no. 01) 2019, pp. 4806–4813.
[13] X. He, H. Zhang, M.-Y. Kan, T.-S. Chua, Fast matrix factorization for online recommendation with implicit feedback, in: SIGIR, 2016, pp. 549–558.

[14] X.-P. Huang, L. Wu, E.-H. Chen, H.-S. Zhu, Q. Liu, Y.-J. Wang, Incremental matrix factorization: A linear feature transformation perspective, in: IJCAI, 2017, pp. 1901–1908.

[15] R.-P. Shen, H.-R. Zhang, H. Yu, F. Min, Sentiment based matrix factorization with reliability for recommendation, Expert Syst. Appl. 135 (2019) 249–258.

[16] D. Feltoni Gurini, F. Gasparetti, A. Micarelli, G. Sansonetti, Temporal people-to-people recommendation on social networks with sentiment-based matrix factorization, Future Gener. Comput. Syst. 78 (2018) 430–439.

[17] C. Chen, M. Zhang, Y.-F. Zhang, Y.-Q. Liu, S.-P. Ma, Efficient neural matrix factorization without sampling for recommendation, ACM Trans. Inf. Syst. 38 (2) (2020) 1–28.

[18] B.-L. Yi, X.-X. Shen, H. Liu, Z.-L. Zhang, W. Zhang, S. Liu, N.-X. Xiong, Deep matrix factorization with implicit feedback embedding for recommendation system, IEEE Trans. Ind. Inform. 15 (8) (2019) 4591–4601.

[19] Q. Wang, X. He, X. Jiang, X.-L. Li, Robust bi-stochastic graph regularized matrix factorization for data clustering, IEEE Trans. Pattern Anal. Mach. Intell. (1) (2020) 390–403.

[20] M.P. O'Mahony, N.J. Hurley, G. Silvestre, Detecting noise in recommender system databases, in: Proceedings of the 11th International Conference on Intelligent User Interfaces, 2006, pp. 109–115.

[21] P.J. Huber, Robust Estimation of a Location Parameter, Springer New York, 1992.

[22] B. Gecer, J. Deng, S. Zafeiriou, Ostec: One-shot texture completion, in: CVPR, 2021, pp. 7628–7638.

[23] P. Shah, U.K. Khankhoje, M. Moghaddam, Inverse scattering using a joint L1–L2 norm-based regularization, IEEE Trans. Antennas and Propagation 64 (4) (2016) 1373–1384.

[24] P. Mianjy, R. Arora, Stochastic PCA with l2 and l1 regularization, in: ICML, 2018, pp. 3531–3539.

[25] S. Rendle, W. Krichene, L. Zhang, J. Anderson, Neural collaborative filtering vs. matrix factorization revisited, in: RecSys, 2020, pp. 240–248.

[26] S. Arora, N. Cohen, W. Hu, Y.-P. Luo, Implicit regularization in deep matrix factorization, in: NIPS, 2019, pp. 7413–7424.

[27] S. Gunasekar, B. Woodworth, S. Bhojanapalli, B. Neyshabur, N. Srebro, Implicit regularization in matrix factorization, in: ITA, 2018, pp. 1–10.

[28] M.-M. Chen, B. Chang, C. Xu, E.H. Chi, User response models to improve a reinforce recommender system, in: WSDM, 2021, pp. 121–129.

[29] A. Esmaeili, F. Marvasti, A novel approach to quantized matrix completion using huber loss measure, IEEE Signal Process. Lett. 26 (2) (2019) 337–341.

[30] M. Vaali Esfahaani, Y.-B. Xue, P. Setoodeh, Deep reinforcement learning-based product recommender for online advertising, 2021, arXiv e-prints, arXiv–2102.

[31] C.-J. Xiao, R.-B. Xie, Y. Yao, Z.-Y. Liu, M.-S. Sun, X. Zhang, L.-Y. Lin, UPRec: User-aware pre-training for recommender systems, 14, (2) 2021, arXiv preprint arXiv:2102.10989.

[32] P. Xu, H.-L. Gan, H. Fu, Z.-B. Zhang, STEAMCODER: Spatial and temporal adaptive dynamic convolution autoencoder for anomaly detection, Knowl.-Based Syst. 279 (2023) 110929.

[33] Z. Xiao, H. Fang, H.-B. Jiang, J. Bai, V. Havyarimana, H.-Y. Chen, L.-C. Jiao, Understanding private car aggregation effect via spatio-temporal analysis of trajectory data, IEEE Trans. Cybern. (2021) 1–12.

[34] X. Xu, J. Li, Y. Yang, F.-M. Shen, Toward effective intrusion detection using log-cosh conditional variational autoencoder, IEEE Internet Things J. 8 (8) (2020) 6187–6196.

[35] J. Yun, W. Kwak, J. Kim, Multi datasource LTV user representation (MDLUR), in: SIGKDD, 2023, pp. 5500–5508.

[36] L. Yu, X. Xu, G. Trajcevski, F. Zhou, Transformer-enhanced hawkes process with decoupling training for information cascade prediction, Knowl.-Based Syst. 255 (2022) 109740.

[37] B. Luijten, R. Cohen, F.J. de Bruijn, H.A. Schmeitz, M. Mischi, Y.C. Eldar, R.J. van Sloun, Adaptive ultrasound beamforming using deep learning, IEEE Trans. Med. Imaging 39 (12) (2020) 3967–3978.

[38] D. Justus, J. Brennan, S. Bonner, A.S. McGough, Predicting the computational cost of deep learning models, in: ICBD, IEEE, 2018, pp. 3873–3882.

[39] J.-W. Chen, C. Wang, S. Zhou, Q.-H. Shi, J.-B. Chen, Y. Feng, C. Chen, Fast adaptively weighted matrix factorization for recommendation with implicit feedback, in: AAAI, vol. 34, (no. 04) 2020, pp. 3470–3477.

[40] S.-L. Liao, J. Li, Y. Liu, Q.-X. Gao, X.-B. Gao, Robust formulation for PCA: Avoiding mean calculation with $L_{2,p}$-norm maximization, in: AAAI, 2018, pp. 3604–3610.

[41] S.-N. Zeng, J.-P. Gou, L.-M. Deng, An antinoise sparse representation method for robust face recognition via joint L1 and L2 regularization, Expert Syst. Appl. 82 (2017) 1–9.

[42] R. Kiran, P. Kumar, B. Bhasker, DNNRec: A novel deep learning based hybrid recommender system: A novel deep learning based hybrid recommender system, Expert Syst. Appl. 144 (2020) 113054.

[43] R. Devooght, N. Kourtellis, A. Mantrach, Dynamic matrix factorization with priors on unknown values, in: SIGKDD, 2015, pp. 189–198.

[44] H. Xiong, D.-G. Kong, Elastic nonnegative matrix factorization, Pattern Recognit. 90 (2019) 464–475.

[45] X.-N. He, J.-H. Tang, X.-Y. Du, R.-C. Hong, T.-W. Ren, T.-S. Chua, Fast matrix factorization with nonuniform weights on missing data, IEEE Trans. Neural Netw. Learn. Syst. 31 (8) (2019) 2791–2804.

[46] H. Shin, S. Kim, J. Shin, X.-K. Xiao, Privacy enhanced matrix factorization for recommendation with local differential privacy, IEEE Trans. Knowl. Data Eng. 30 (9) (2018) 1770–1782.

[47] H.-R. Zhang, F. Min, Z.-H. Zhang, S. Wang, Efficient collaborative filtering recommendations with multi-channel feature vectors, Int. J. Mach. Learn. Cybern. (2018) 1165–1172.

[48] T. Silveira, M. Zhang, X. Lin, Y.-Q. Liu, S.-P. Ma, How good your recommender system is? A survey on evaluations in recommendation, Int. J. Mach. Learn. Cybern. 10 (5) (2019) 813–831.

[49] Y.-F. Lu, Y. Fang, C. Shi, Meta-learning on heterogeneous information networks for cold-start recommendation, in: KDD, 2020, pp. 1563–1573.

[50] R. Anand, J. Beel, Auto-surprise: An automated recommender-system (AutoRecSys) library with Tree of Parzens Estimator (TPE) optimization, in: RecSys, 2020, pp. 585–587.

[51] R. Barzegar Nozari, H. Koohi, A novel group recommender system based on members' influence and leader impact, Knowl.-Based Syst. 205 (2020) 106296.

[52] R.J. Ziarani, R. Ravanmehr, Deep neural network approach for a serendipity-oriented recommendation system, Expert Syst. Appl. (2021) 115660.

[53] C. Shi, X.-T. Han, L. Song, X. Wang, S.-Z. Wang, J.-P. Du, P.S. Yu, Deep collaborative filtering with multi-aspect information in heterogeneous networks, IEEE Trans. Knowl. Data Eng. 31 (4) (2019) 1413–1425.

[54] Z. Zhao, B. Gao, V.W. Zheng, D. Cai, X.-F. He, Y.-T. Zhuang, Link prediction via ranking metric dual-level attention network learning, in: IJCAI, 2017, pp. 3525–3531.

[55] X. Xin, X.-N. He, Y.-F. Zhang, Y.-D. Zhang, J. Jose, Relational collaborative filtering: Modeling multiple item relations for recommendation, in: SIGIR, 2019, pp. 125–134.

[56] G.-B. Guo, J. Zhang, N. Yorke-Smith, A novel Bayesian similarity measure for recommender systems, in: IJCAI, 2013, pp. 2619–2625.

[57] Z.-S. Jiang, H.-Z. Liu, B. Fu, Z.-H. Wu, T. Zhang, Recommendation in heterogeneous information networks based on generalized random walk model and bayesian personalized ranking, in: WSDM, 2018, pp. 288–296.

[58] A. Zafra, C. Romero, S. Ventura, E. Herrera-Viedma, Multi-instance genetic programming for web index recommendation, Expert Syst. Appl. 36 (9) (2009) 11470–11479.

[59] L. Huang, Z.-L. Zhao, C.-D. Wang, D. Huang, H.-Y. Chao, Lscd: Low-rank and sparse cross-domain recommendation, Neurocomputing 366 (2019) 86–96.

[60] S. Kang, J. Hwang, D. Lee, H. Yu, Semi-supervised learning for cross-domain recommendation to cold-start users, in: CIKM, 2019, pp. 1563–1572.

Yuan-Yuan Xu received the M.S. degree from the School of Optoelectronic Information, University of Electronic Science and Technology of China, Chengdu, China, in 2007. She is currently an associate professor with Southwest Petroleum University, Chengdu, China. Her current research interests is the recommender system.

Hui Xiao is a graduate student with School of Computer Sciences, Southwest Petroleum University, Chengdu, China. His current research interest is the recommender system.

Heng-Ru Zhang received the M.S. degree from the School of Mechanical and Electrical Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2002, and the Ph.D. degree from the School of Sciences, Southwest Petroleum University, Chengdu, in 2019. He is currently a professor with Southwest Petroleum University, Chengdu. He has published more than 40 refereed papers in various journals and conferences, including Applied Soft Computing, Information Sciences, and Knowledge-Based Systems. His current research interests include recommender systems, label distribution learning and granular computing.

Wei-Zhi Wu received the B.S. degree in mathematics from Zhejiang Normal University, Jinhua, China, in 1986, the M.S. degree in mathematics from East China Normal University, Shanghai, China, in 1992, and the Ph.D. degree in applied mathematics from Xian Jiaotong University, Xian, China, in 2002. He is currently a Professor with the School of Mathematics, Physics, and Information Science, Zhejiang Ocean University, Zhejiang, China. His current research interests include approximate reasoning, rough sets, random sets, formal concept analysis, and granular computing.

Fan Min received the M.S. and Ph.D. degrees from the School of Computer Science and Engineering, University of Electronics Science and Technology of China, Chengdu, China, in 2000 and 2003, respectively. He visited the University of Vermont, Burlington, Vermont, from 2008 to 2009. He is currently a professor with Southwest Petroleum University, Chengdu. He has published more than 100 refereed papers in various journals and conferences, including IEEE Transactions on Systems, IEEE Transactions on Geoscience and Remote Sensing, IEEE Transactions on Circuits and Systems for Video Technology, Pattern Recognition, Neurocomputing, and AAAI. His current research interests include recommender systems, active learning and full-waveform inversion. His current research interests include data mining, recommender systems, active learning and granular computing.