Knowledge-Based Systems 205 (2020) 106228

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Adaptive and efficient high-order rating distance optimization model with slack variable

Yifan He, Haitao Zou *, Hualong Yu, Shang Zheng, Shang Gao

School of Computer, Jiangsu University of Science and Technology, Zhen Jiang, China

ARTICLE INFO

Article history:
Received 3 February 2020
Received in revised form 30 April 2020
Accepted 6 July 2020
Available online 7 July 2020

Keywords:
Recommender systems
Slack variable
High-order rating distance
Newton method

ABSTRACT

The classical high-order rating distance model which aims to minimize not only (i) the difference between the estimated and real ratings of the same user–item pair (i.e., the first-order rating distance), but also (ii) the difference between the estimated and real rating difference of the same user across different items (i.e., the second-order rating distance), and use stochastic gradient descent to solve this convex optimization problem in recommender systems, has good performance in prediction accuracy. However, when the manually set parameter for the second-order rating difference is fixed, this model will not converge as the size of dataset increasing, and its performance on efficiency is slow compared with the matrix factorization method. Aiming at improving such model's adaptability and efficiency, we propose an improved high-order rating distance model with omitting rules based on slack variable, in which the static parameter used to balance the first-order rating distance and the second-order rating distance is replaced by a data-scale sensitive function. We choose Newton method to solve the convex recommendation optimization problem defined in this paper instead of stochastic gradient descent. Our model not only achieves the adaptability by eliminating several static parameters for module balancing, reduces the computation complexity, but also accelerates the optimization function convergence speed. We provide solid theoretical support and conduct comprehensive experiments on four real-world datasets. Experimental results show the proposed model has good performance in terms of prediction accuracy and efficiency.

1. Introduction

Recommender systems have been popularly used in online business. E-commerce companies, such as *Amazon* and *eBay*, have greatly extended their businesses by automatically recommending items or products based on personalized information. Among all of the recommendation algorithms, the latent factor model which uses the stochastic gradient descent [1,2] to deal with the loss function, and automatically fills in the blanks for user ratings, performs well in terms of accuracy and efficiency. For example, Zhou et al. [3] convert the user latent vectors and item latent vectors into binary codes, and recommend items by computing and sorting the user similarity based on such code, which improves the efficiency by 100 times compared with the original method. However, user similarity is not equivalent to item preference and the accuracy loss for this method is severe. To solve the above problem, Zhang et al. [4] design a mapping function that may transform the binary code to item preference. Although the recommendation precision is improved, the accuracy loss still exists. Bhargava et al. [5] use tensor factorization on sparse user-generated data and formulates an objective function which simultaneously factorizes coupled tensors and matrices constructed from heterogeneous data sources. Cheng et al. [6] apply a proposed aspect-aware topic model on the review text to model user preferences and item features from different aspects, and estimate the aspect importance of a user towards an item.

Nevertheless, the above mentioned models only consider the difference between the estimated and real ratings of the same user–item pair (i.e., the first-order rating distance), and ignore the estimated and real rating difference of the same user across different items (i.e., the second-order rating distance). Hence, Xu et al. [7] propose a high-order rating distance model called HoORaYs with good accuracy in terms of item ranking and predictive ratings which takes these two kinds of distances into account and uses a static parameter to balance their proportions. Unfortunately, this model still has some flaws in adaptability and efficiency due to its manually-set parameters, its non-convergence, and its solution for the pre-defined optimization function.

In this paper, we propose an adaptive and efficient high-order rating distance model named AE-HoORaYs inspired by HoORaYs [7]. Our contributions are listed as follows:

* Corresponding author.
E-mail addresses: yifanhe1996@outlook.com (Y. He), nkroben@outlook.com (H. Zou), yuhualong@just.edu.cn (H. Yu), szheng@just.edu.cn (S. Zheng), gao_shang@sohu.com (S. Gao).

https://doi.org/10.1016/j.knosys.2020.106228
0950-7051/

- We propose a normalization function varied about data sizes to balance the proportion between the first-order rating distance and the second-order rating distance, so that the algorithm can always have good prediction accuracy in any scale datasets.
- We design rules for omitting tuples from high-order rating distance model by introducing a slack variable, and reduce the computation complexity to improve its efficiency.
- We utilize Newton method to solve the convex optimization problem defined in our model, which makes our model converge efficiently.
- We provide solid theoretical support and discussions to illustrate the rationality of our model.
- We conduct comprehensive experiments on four real-world datasets of different scales. All the results support our claims that the proposed method outperforms the state of the art.

In the remainder of this paper, we describe related works in Section 2, illustrate the problem statement in Section 3 and introduce our AE-HoORaYs model in Section 4. The theoretical supports are presented in Section 5, followed by the experiments in Section 6, and finally, the conclusion is shown in Section 7.

2. Related works

Latent factor model which factorizes the rating matrix into two lower matrices to minimize the regularized squared error on the set of known ratings has been widely used in recommender systems [8,9]. For example, Cheng et al. [10] propose a group latent factor model, which attempts to learn a factorization of latent factor space into subspaces and captures the correlation and heterogeneity of multiple behaviors in order to solve the data sparsity problem. Zhu et al. [11] implement a cross-network collaborative matrix factorization recommendation framework which can integrate multi-source information and alleviate the sparse information problem in each individual network. However, the accuracy of the above mentioned models are completely dependent on the characteristics of the dataset, when it comes to cold-start situations, these methods may not generate satisfying results.

Recently, some researchers turn to consider the ranking quality of recommended item list instead of the predictive rating accuracy modifying the optimization function. Kabbur et al. [12] propose an AUC-like optimization function which is the ranking error based on Bayesian Personalized Ranking, and can generate high-quality recommendations even on sparse datasets. Park et al. [13] propose a collaborative ranking model which takes pairwise-loss as optimization function, and performs well in all ranking measures. Wu et al. [14] use Newton method to optimize the pairwise-loss optimization function, to improve the speed of algorithm. Xu et al. [7] propose a novel Latent factor model by importing the concept of high-order rating distance and can reduce the variance and improves the recommendation accuracy. Nonetheless, these models usually use stochastic gradient descent to solve the optimization function, and may need a manually setting parameter to balance the relevant modules.

The influence of deep learning is pervasive and the field of deep learning in recommender system is flourishing [15]. For instance, Sedhain et al. [16] exploit a novel auto-encoder framework for collaborative filtering and has representational and computational advantages over existing neural approaches. He et al. [17] leverage a multi-layer perception to learn the non-linear interaction between user and item. Although deep learning methods extract more complex user–item interactions and improves recommendation quality, those approaches need to adjust many hyper-parameters which limits their adaptability for datsets with different scales.

In this paper, we propose an adaptive latent factor model with slack variable and use Newton method for optimization. Our method can adapt datasets of any size, and can provide good performance in prediction accuracy and efficiency.

3. Problem statement

In this paper, we consider a recommender system, which consists of m users and n items. We denote the user latent vectors as $U = \{u_i\}_{i=1}^{m}$, and the item latent vectors as $V = \{v_j\}_{j=1}^{n}$. The number of dimensions for these latent vectors is d. The observed ratings are denoted as $R = \{r_{ij} | r_{ij} \in [1, r_{max}]\}$, where r_{ij} represents the rating value from user i on item j, and the value of r_{max} is the maximum rating scale in the target recommender system. The recommendation techniques we want to study in this paper is to predict those blank entries in R by directly obtain the ratings as long as we have learned the latent vectors U and V. We introduce two methods that are relevant to our work in the following subsections.

3.1. Latent factor model: Matrix factorization

Matrix factorization based on collaborative filtering [18,19] becomes one of the most popular methods to solve the recommendation problem. It predicts the rating value $\hat{r}_{ij} = u_i^T v_j$ by minimizing the first-order rating distance shown as follows:

$$\arg\min_{U,V} \sum_{(i,j,r_{ij}) \in D_T} (r_{ij} - u_i^T v_j)^2 \tag{1}$$

where D_T is the training set, in which each data is denoted as (i, j, r_{ij}).

In order to avoid over-fitting, the square of the eigenvector norm is usually added to Function (1) as a regularization term, then we have:

$$\arg\min_{U,V} \sum_{(i,j,r_{ij}) \in D_T} (r_{ij} - u_i^T v_j)^2 + \lambda(\|u_i\|^2 + \|v_j\|^2) \tag{2}$$

where λ is a constant derived from experiments. This function can be achieved by updating U and V using stochastic gradient descent.

3.2. High-order rating distance optimization model: HoORaYs

Unlike the matrix factorization model, HoORaYs model [7] which introduces the high-order distance principle aims at minimizing not only the first-order rating distance but also the second-order rating distance. Suppose X_i denote the item set purchased by user i, Y_j denote the user set who have purchased item j. The second-order rating distance for user i is computed as:

$$\sum_{k \in X_i, k \neq j} (\sigma(u_i^T v_j, r_{ik}) - \sigma(r_{ij}, r_{ik}))^2 \tag{3}$$

and the second-order rating distance for item j is computed as:

$$\sum_{k \in Y_j, k \neq i} (\sigma(u_i^T v_j, r_{kj}) - \sigma(r_{ij}, r_{kj}))^2 \tag{4}$$

where normalization function $\sigma(x, y) = 1/(1 + e^{-(x-y)})$ is used to smooth the results. Generally, most of the rating scale is small for recommender systems (e.g., 1–5 stars), and the contributions of different ratings from other users/items are equal to each other if they share the same rating value. Thus, the second-order rating distance for user i and item j can be merged as:

$$\sum_{(i,j,r_{ij}) \in D_T} \sum_{r=1}^{r_{max}} |\Omega_{r_{ij}}, r|(\sigma(u_i^T v_j, r) - \sigma(r_{ij}, r))^2 \tag{5}$$

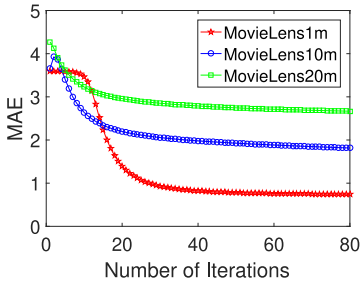


Fig. 1. MAE variation of HoORaYs about the number of iterations on MovieLens datasets when λ_d is fixed. The datasets are chosen from three stable Movielens datasets which contains 1 million ratings from 6040 users on 3706 movies (labeled as MovieLens1m); 10 million ratings applied to 65,133 movies by 71,561 users (labeled as MovieLens10m); and 20 million ratings applied to 131,262 movies by 138,493 users (labeled as MovieLens20m), respectively.

where $|\Omega_{r_{ij}}, r|$ is the total number of ratings that user i has specified on other items with value r or item j has received from other users with value r. And the optimization function for HoORaYs is shown as:

$$\begin{aligned}\arg\min_{U,V} &\sum_{(i,j,r_{ij})\in D_T} (r_{ij} - u_i^T v_j)^2 + \lambda_u \sum_{i=1}^{m} \|u_i\|^2 + \lambda_v \sum_{j=1}^{n} \|v_j\|^2 \\ &+ \lambda_d \sum_{(i,j,r_{ij})\in D_T} \sum_{r=1}^{r_{max}} |\Omega_{r_{ij}}, r|(\sigma(u_i^T v_j, r) - \sigma(r_{ij}, r))^2\end{aligned} \tag{6}$$

where λ_d is used to tune the affection from the second-order rating distance module.

3.3. Weaknesses of HoORaYs model

Although HoORaYs model has better performance in prediction accuracy compared with matrix factorization, our previous work leads us to find its three weaknesses. The first one is shown as the following claim:

Claim 1. *HoORaYs model is not convergent with fixed λ_d.*

Proof. Since $(\sigma(u_i^T v_j, r) - \sigma(r_{ij}, r))^2$ in Function (6) is the difference between the estimated and real rating difference of the same user across different items, it is generally a constant. $|\Omega_{r_{ij}}, r|$ is growing as the size of the system dataset increases. Let $k = (\sigma(u_i^T v_j, r) - \sigma(r_{ij}, r))^2$, $x = |\Omega_{r_{ij}}, r|$. $f(x) = |\Omega_{r_{ij}}, r|(\sigma(u_i^T v_j, r) - \sigma(r_{ij}, r))^2 = kx$, $\lim_{x\to+\infty} f(x) = +\infty$. Hence, when λ_d is fixed, as the size of dataset increases, HoORaYs is not convergent.

According to our experimental results (see Fig. 1), when λ_d is fixed, the precision of HoORaYS becomes worse as the size of datasets increases. When comparing the contribution of the first-order distance module and the second-order distance module to Function (6) in our previous experiments, we compute the average first-order rating distance and its corresponding second-order rating distance for each user/item on three MovieLens datasets (see Fig. 2). The results show that, as the size of the dataset becomes bigger, the magnitude of the value for the second-order rating distance module is much bigger than the first-order rating distance module, and takes the dominant position to determine the value of Function (6). It is a conflict with the premise of "first-order rating distance module is more important than the second-order rating distance module". As a result, to ensure the prediction accuracy of HoORaYs, Xu et al. [7] choose a compromising method to decrease the magnitude of λ_d as the size of dataset increases.

Secondly, the computation complexity for HoORaYs is $O(|R|)$, and it is very large in real-word datasets. Therefore, the performance of efficiency for HoORaYs is low.

Thirdly, HoORaYs uses stochastic gradient descent to update U and V, in which a learning rate parameter α is introduced. Therefore, an inappropriate setting of α may lead to misconvergence or affect the convergence speed.

3.4. Problem definition

In this paper, the problem we want to handle comes from two folds:

(i) How to design a data-sensitive function Λ, so that the second-order rating distance module in HoORaYs model does not increase infinitely as the data scale increases, and can still assist in refining the first-order rating distance.

(ii) How to implement acceleration mechanisms to improve the efficiency of HoORaYs model, which is constructed by two subproblems:

1. How to reduce the computation complexity for HoORaYs model.
2. How to find a proper method with excellent convergence speed used to replace the stochastic gradient descent approach when solving the convex optimization problem defined in HoORaYs model, so as to avoid the affection from the learning rate α, and to maintain a fairly good prediction accuracy.

4. Adaptive HoORaYs model with acceleration mechanisms

4.1. Normalization design for HoORaYs

To achieve the design about the data-sensitive function, it is intuitive to construct a function that involves $|\Omega_{r_{ij}}, r|$ and a normalization process.

According to our previous discussions, one may notice that $|\Omega_{r_{ij}}, r|$ is the parameter caused HoORaYs model not convergent, and λ_d has to be decreased as $|\Omega_{r_{ij}}, r|$ increases. Thus, let $\Lambda = \lambda_d \cdot |\Omega_{r_{ij}}, r|$. If Λ is stable, HoORaYs is convergent. Since $|\Omega_{r_{ij}}, r|$ is user-profile relevant, and its domain is not constrained, we need to normalize it and define a new computation about Λ, formally we define it as:

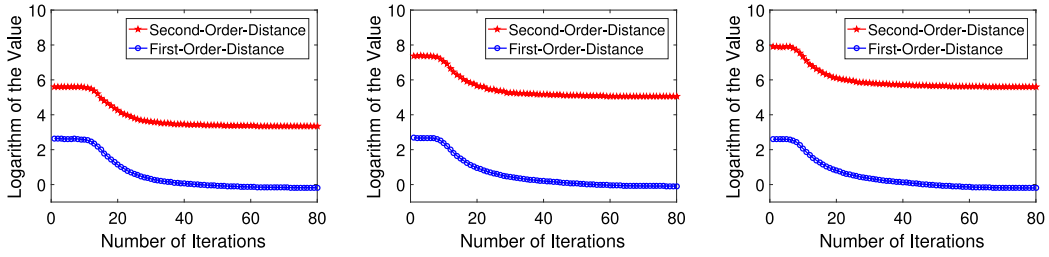
$$\Lambda = \frac{|\Omega_{r_{ij}}, r|}{max(|\Omega_{r_{ij}}, r|)} \tag{7}$$

where $max(|\Omega_{r_{ij}}, r|)$ is the maximum value of $|\Omega_{r_{ij}}, r|$ appeared in statistics.

Then, the problem we want to solve in this paper is to minimize the following function:

$$\begin{aligned}\mathcal{L}(U,V) = &\sum_{(i,j,r_{ij})\in D_T} ((r_{ij} - u_i^T v_j)^2 + \lambda_u \sum_{i=1}^{m} \|u_i\|^2 + \lambda_v \sum_{j=1}^{n} \|v_j\|^2 \\ &+ \sum_{r=1}^{r_{max}} \frac{|\Omega_{r_{ij}}, r|}{max(|\Omega_{r_{ij}}, r|)} (\sigma(u_i^T v_j, r) - \sigma(r_{ij}, r))^2)\end{aligned} \tag{8}$$

where $u_i \in U$ and $v_j \in V$ are user i's latent vector and item j's latent vector defined in previous section. If Λ is stable, the improved model not only maintains the refinement effect from the original one, but also has convergence feature.



(a) *MovieLens1m* (b) *MovieLens10m* (c) *MovieLens20m*

Fig. 2. Numeric Comparison between the first-order distance and the second-order rating distance on MovieLens datasets. We compute the logarithm of the corresponding values for clarity consideration.

4.2. Acceleration design using slack variable

HoORaYs has great improvement in prediction accuracy, but its performance in efficiency is low. Inspiring by the soft margin proposed in Support Vector Machine [20], we introduce a slack variable ϵ to the normalized HoORaYs (defined in Function (8)), and omit those tuples whose first-order distance is less than ϵ from training set for acceleration. We first propose our claim:

Claim 2. *$\forall \epsilon \in (0, +\infty)$, if the first-order distance satisfies:*

$$(r_{ij} - u_i^T v_j)^2 \leq \epsilon^2 \tag{9}$$

then its corresponding second-order distance satisfies:

$$\sum_{r=1}^{r_{max}} \frac{|\Omega_{r_{ij}}, r|}{max(|\Omega_{r_{ij}}, r|)} (\sigma(u_i^T v_j, r) - \sigma(r_{ij}, r))^2 \leq \epsilon^2 \tag{10}$$

Proof. For the simplicity of description, suppose $f(t) = \frac{1}{1+e^t}$, the second-order rating distance can be transformed as:

$$\begin{aligned} &\sum_{r=1}^{r_{max}} \frac{|\Omega_{r_{ij}}, r|}{max(|\Omega_{r_{ij}}, r|)} (\sigma(u_i^T v_j, r) - \sigma(r_{ij}, r))^2 \\ &= \sum_{r=1}^{r_{max}} \frac{|\Omega_{r_{ij}}, r|}{max(|\Omega_{r_{ij}}, r|)} (f(r - u_i^T v_j) - f(r - r_{ij}))^2 \end{aligned}$$

According to Lagrange's mean value theorem, there is a ε in the interval of $(r - u_i^T v_j)$ and $(r - r_{ij})$, which satisfies Eq. (11):

$$\frac{f(r - u_i^T v_j) - f(r - r_{ij})}{(r - u_i^T v_j) - (r - r_{ij})} = \frac{f(r - u_i^T v_j) - f(r - r_{ij})}{r_{ij} - u_i^T v_j} = f'(\varepsilon) \tag{11}$$

where $f'^2(\varepsilon) \leq \frac{1}{16}$, $0 \leq \frac{|\Omega_{r_{ij}}, r|}{max(|\Omega_{r_{ij}}, r|)} \leq 1$, and r_{max} is usually equal to 5 or 10 in real world datasets, then the following inequation stands:

$$\begin{aligned} \sum_{r=1}^{r_{max}} \frac{|\Omega_{r_{ij}}, r|}{max(|\Omega_{r_{ij}}, r|)} (f(r - u_i^T v_j) - f(r - r_{ij}))^2 &\leq \frac{r_{max}}{16} (r_{ij} - u_i^T v_j)^2 \\ &\leq \frac{r_{max}}{16} \epsilon^2 \leq \epsilon^2 \end{aligned}$$

Claim 2 gives us a clue that when the first-order rating distance for user i is less than ϵ^2, its corresponding second-order rating distance is also less than ϵ^2. If ϵ is small enough (e.g. $0 < \epsilon < 1$), then such first-order rating distances and their second-order rating distances tend to be 0, and their contribution to minimizing Function (8) can be ignored. Based on this motivation, we propose the rules for omitting tuples from the normalized HoORaYs model, and they described as:

$$I_{ij}^{(1)} = \begin{cases} 1, & |r_{ij} - U_i^T V_j| > \epsilon \\ 0, & otherwise \end{cases} \tag{12}$$

$$I_{ij}^{(2)} = \begin{cases} 1, & r_{ij} - U_i^T V_j > \epsilon \\ -1, & otherwise \end{cases} \tag{13}$$

where $I_{ij}^{(1)}$ is used to determine whether this tuple to be omitted or not. $I_{ij}^{(2)}$ is used to judge the slack direction(positive or negative) for those remained tuples.

Finally, the optimization problem to be solved in this paper is defined as:

$$\begin{aligned} \arg\min_{U,V} &\sum_{(i,j,r_{ij}) \in D_T} I_{ij}^{(1)} ((r_{ij} - I_{ij}^{(2)} \epsilon - u_i^T v_j)^2 + \lambda_u \sum_{i=1}^{m} \|u_i\|^2 \\ &+ \lambda_v \sum_{j=1}^{n} \|v_j\|^2 \\ &+ \sum_{r=1}^{r_{max}} \frac{|\Omega_{r_{ij}}, r|}{max(|\Omega_{r_{ij}}, r|)} (\sigma(u_i v_j^T, r) - \sigma(r_{ij} - I_{ij}^{(2)} \epsilon, r))^2) \end{aligned} \tag{14}$$

Function (14) means, when updating U and V to achieve the minimization, those tuples where $(r_{ij} - u_i^T v_j)^2 \leq \epsilon^2$ are considered good enough and no need to be processed. The remained tuples where $(r_{ij} - u_i^T v_j)^2 > \epsilon^2$ should be optimized until they satisfy $(r_{ij} - u_i^T v_j)^2 \leq \epsilon^2$. For instance, suppose there exists a data tuple(i, j, r_{ij}), in which $r_{ij} = 3$, and we set slack variable $\epsilon = 0.5$. If $2.5 \leq u_i v_j^T \leq 3.5$, according to formula (12), $|r_{ij} - u_i v_j^T| < \epsilon$, then $I_{ij}^{(1)} = 0$, it means this tuple has no contribution to the objective function of formula (14). So we omit it from the training set; if $u_i v_j^T < 2.5$, according to formula (12) and (13), $r_{ij} - u_i v_j^T > \epsilon$, then $I_{ij}^{(1)} = 1$ and $I_{ij}^{(2)} = 1$, we update u_i and v_j to achieve $u_i v_j^T = 2.5$ which make $|r_{ij} - u_i v_j^T|$ close to ϵ; Similarly, if $u_i v_j^T > 3.5$, we get $I_{ij}^{(1)} = 1$ and $I_{ij}^{(2)} = -1$, then we update u_i and v_j to achieve $u_i v_j^T = 3.5$. The detailed updating processes about u_i and v_j are shown in the following subsection.

4.3. Rapid optimization solution with Newton method

In this section, we discuss how to optimize Function (14) with rapid convergence.

The classical method to solve Function (14) is stochastic gradient descent method. Generally, for each training data (i, j, r_{ij}), the user eigenvector u_i and item eigenvector v_j are updated as:

$$\begin{aligned} u_i &\leftarrow u_i - \alpha \nabla u_i \\ v_j &\leftarrow v_j - \alpha \nabla v_j \end{aligned} \tag{15}$$

where α is denoted as the learning rate as we mentioned before, ∇u_i and ∇v_j are the first order partial derivative for u_i and v_j

correspondingly. They can be calculated as Eq. (16).

$$\begin{aligned}\nabla u_i =& 2I_{ij}^{(1)}(-(r_{ij} - I_{ij}^{(2)}\epsilon - u_i^T v_j)v_j + \lambda_u u_i \\ & - \sum_{r=1}^{max} \frac{|\Omega_{ij}, r|}{max(|\Omega_{ij}, r|)}(\sigma(r_{ij} - I_{ij}^{(2)}\epsilon, r) \\ & \sigma(u_i^T v_j, r)(1 - \sigma(u_i^T v_j, r))v_j \\ & - \sigma(u_i^T v_j, r)^2(1 - \sigma(u_i^T v_j, r))v_j)) \\ \nabla v_j =& 2I_{ij}^{(1)}(-(r_{ij} - I_{ij}^{(2)}\epsilon - u_i^T v_j)u_i + \lambda_v v_j \\ & - \sum_{r=1}^{max} \frac{|\Omega_{ij}, r|}{max(|\Omega_{ij}, r|)}(\sigma(r_{ij} - I_{ij}^{(2)}\epsilon, r) \\ & \sigma(u_i^T v_j, r)(1 - \sigma(u_i^T v_j, r))u_i \\ & - \sigma(u_i^T v_j, r)^2(1 - \sigma(u_i^T v_j, r))u_i))\end{aligned} \tag{16}$$

One may notice that, α is often set manually, and it is chosen based on the experimental results or researchers' subjective opinions. And the value of the objective function decreases for the dual problem sometimes does not imply the decrease of primal objective function value, which often results in slow convergence [14]. To overcome the convergence issues, we propose to achieve the minimization through Newton method using the curvature information to derive a better descent path, we use the inverse of Hessian matrix to replace α in Formula (15), the updating process is shown as:

$$\begin{aligned}u_i &\leftarrow u_i - {H_{u_i}}^{-1}\nabla u_i \\ v_j &\leftarrow v_j - {H_{v_j}}^{-1}\nabla v_j\end{aligned} \tag{17}$$

Since H_{u_i} and H_{v_j} are the second order partial derivative matrices of the user vector and item vector under Eq. (9), respectively, they can be derived as:

$$\begin{aligned}H_{u_i} =& 2I_{ij}^{(1)}(\lambda_u I + v_j v_j^T + \sum_{r=1}^{max} \frac{|\Omega_{r_{ij}}, r|}{max(|\Omega_{r_{ij}}, r|)}(\sigma(r_{ij}, r)(1 - 2\sigma(u_i^T v_j, r)) \\ & - 2\sigma(u_i^T v_j, r) + 3\sigma(u_i^T v_j, r)^2)\sigma(u_i^T v_j, r)(1 - \sigma(u_i^T v_j, r))v_j v_j^T) \\ H_{v_j} =& 2I_{ij}^{(1)}(\lambda_v I + u_i u_i^T + \sum_{r=1}^{max} \frac{|\Omega_{r_{ij}}, r|}{max(|\Omega_{r_{ij}}, r|)}(\sigma(r_{ij}, r)(1 - 2\sigma(u_i^T v_j, r)) \\ & - 2\sigma(u_i^T v_j, r) + 3\sigma(u_i^T v_j, r)^2)\sigma(u_i^T v_j, r)(1 - \sigma(u_i^T v_j, r))u_i u_i^T)\end{aligned} \tag{18}$$

The pseudo code for the updating process is shown in Algorithm 1, where H_{u_i} represents user i's Hessian matrix, ∇_{u_i} denotes the first-order partial derivative vector of user i. θ_{u_i} is the eigenvector gradient for user i, and $\mathcal{L}(u_i, V)$ represents the optimal function for u_i. Noting that we set s as the step length used to accelerate the convergence of Newton method when updating u_i (line 13), and its initial value is 1 (line 8). If the value of function is decreased comparing with its updated value (line 9), the updating process is completed. Otherwise, $s \leftarrow \frac{s}{2}$, and redo this updating process. Due to the irreversibility of Hessian matrix, θ_{u_i} cannot be computed directly by $\theta_{u_i} = {H_{u_i}}^{-1} \cdot \nabla_{u_i}$ (line 8 in Algorithm 1). We derive θ_{u_i} by the conjugate gradient method [21].

Since our proposed model can achieve both adaptability and efficiency, and it is a improved model about HoORaYs, in the rest of this paper, we name our model as AE-HoORaYs.

5. Rationality analysis about AE-HoORaYs model

5.1. Rationality about HoORaYs normalization

Through the analysis of HoORaYs, it is obvious that the first-order rating distance plays a leading role in updating U and V,

Algorithm 1 Newton Method for Updating U and V

Input: Training data D_T
Output: U, V
1: Initialize U, V
2: **while** not converged **do**
3: **procedure** STAGE1 ▷ Fix V and update U
4: **for** $i = 1$ to m **do**
5: $\theta_{u_i} = (\sum H_{u_i})^{-1} \cdot \sum \nabla_{u_i}$
6: **end for**
7: **for** $i = 1$ to m **do**
8: Initialize $s = 1$
9: **while** $\mathcal{L}(u_i - s \cdot \theta_{u_i}, V) > \mathcal{L}(u_i, V)$ **do**
10: $s = \frac{1}{2}s$
11: **end while**
12: **end for**
13: $u_i = u_i - s \cdot \theta_{u_i}$
14: **end procedure**
15: **procedure** STAGE2 ▷ Fix U and update V
16: Update V as same as U
17: **end procedure**
18: **end while**
19: **return** U, V

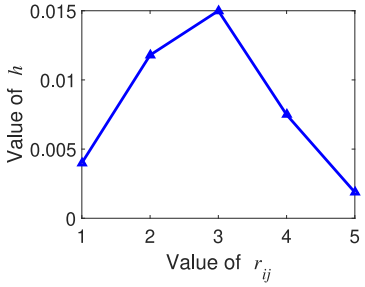


Fig. 3. The variation of the second-order rating distance h about r_{ij} on MovieLens1m. We can see that, h reaches the peak when $r_{ij} = 3$. And the bigger the value of $|r_{ij} - r|$ is, the smaller h becomes, which indicates $|\Omega_{r_{ij}}, r|$ should also reflects the preference from user or item on the rating value of r.

while the second-order rating distance module is the refinement term. Let $h = (\sigma(u_i^T v_j, r) - \sigma(r_{ij}, r))^2$ temporarily, we set $r = 3$ in h, and suppose the difference between the predictive ratings and the real ratings is 0.5, i.e., $u_i^T v_j - r_{ij} = 0.5$, then we draw the variation curve about h on r_{ij}, the result is shown in Fig. 3.

We can see that, replacing $\lambda_d \cdot |\Omega_{r_{ij}}, r|$ with the adaptive value of $\frac{|\Omega_{r_{ij}}, r|}{max(|\Omega_{r_{ij}}, r|)}$ avoids the situation about the second-order rating distance module increasing to infinity, and keeps ratio of user and item's preference on value r.

5.2. Rationality about introducing slack variable ϵ

In this section, we explain the rationality for using slack variable ϵ in AE-HoORaYs model from the perspective of geometry.

Suppose the latent factor space is 2-dimensional, and each user/item has its place, which is presented as the latent vector in the space. Taking user i and item j for example, their latent factor are denoted as $u_i = (x_i, y_i)$, $v_j = (a_j, b_j)$, respectively. Assuming v_j and r_{ij} are given, where v_j is normalized, i.e., $(a_j^2 + b_j^2) = 1$. If we want to find u_i that satisfy $u_i^T v_j = r_{ij}$, the coordinate point (x_i, y_i) for u_i should be on the straight line $a_j x + b_j y = r_{ij}$. When the slack variable ϵ is introduced, it means we want to find the coordinate point (x_i, y_i) for u_i that lie in the zone between $u_i^T v_j + \epsilon = r_{ij}$

and $u_i^T v_j - \epsilon = r_{ij}$, which ignores certain training tuples without compromising accuracy too much.

Based on this interpretation, Fig. 4 shows the geometry calculation about Function (14) where the coordinate point lies in the positive zone $u_i^T v_j + \epsilon = r_{ij}$ (i.e., $I^{(2)} = 1$), and in the negative zone $u_i^T v_j - \epsilon = r_{ij}$ (i.e., $I^{(2)} = -1$). The black dotted line $u_i^T v_j + c = r_{ij}$ in Fig. 4(a), and $u_i^T v_j - c = r_{ij}$ in Fig. 4(b) are where u_i located now (c is a constant, $c \geq 0$), we expect to optimize u_i to the zone between $u_i^T v_j + \epsilon = r_{ij}$ and $u_i^T v_j = r_{ij}$ or between $u_i^T v_j = r_{ij}$ and $u_i^T v_j - \epsilon = r_{ij}$ ($\epsilon < c$), respectively.

In Fig. 4(a), the first-order rating distance is calculated as $(r_{ij} - I_{ij}^{(2)}\epsilon - u_i^T v_j)^2 = (r_{ij} - \epsilon - r_{ij} + c)^2 = (c - \epsilon)^2$, and its objective is to minimize $(c - \epsilon)^2$ to 0, which is $(c - \epsilon)^2 \to 0$. Suppose $u_i^T v_{j'} = r_{ij'}$ is another straight line constructed by user i and item j', the difference between $r_{ij} - c$ and $r_{ij'}$ is the directed line segment $\overrightarrow{L_1} = r_{ij} - c - r_{ij'}$, the difference between the expected rating $r_{ij} - \epsilon$ (i.e., the expectation for predictive rating $u_i^T v_j$) and $r_{ij'}$ is $\overrightarrow{L_2} = r_{ij} - \epsilon - r_{ij'}$. Then, the second-order rating distance is the sum of $|\overrightarrow{L_1} - \overrightarrow{L_2}|^2$ for all the other items that user i has rated, and its objective is to minimize $|\overrightarrow{L_1} - \overrightarrow{L_2}|^2$, which is $[r_{ij} - \epsilon - r_{ij'} - (r_{ij} - c - r_{ij'})]^2 = (c - \epsilon)^2 \to 0$. Since the objective of Function (14) is using the second-order rating distance to compress the solution space of u_i on the basis of the first-order rating distance, from the perspective of geometry, it is making the dot line $u_i^T v_j + c = r_{ij}$ close to the expected zone between $u_i^T v_j + \epsilon = r_{ij}$ and $u_i^T v_j = r_{ij}$. Similar interpretation can also used in Fig. 4(b).

5.3. Rationality about utilizing Newton method

We mainly demonstrate the superiority of utilizing Newton method for solving Function (14). Formally, we have the following claim:

Claim 3. *Newton method converges faster than stochastic gradient descent, when solving the convex optimization problem.*

Proof. When solving the convex optimization problem, the stochastic gradient descent method uses a plane to fit the current local surface, and Newton method uses the quadric surface to fit the current local surface. Hence, Newton method could find a better optimal decent path comparing with the stochastic gradient descent method. Specifically, we first consider the case of the unary function. When using Newton method, the objective function $f(x)$ can be expanded by the second-order Taylor formula as:

$$f(x) = f(x_n + \Delta x) \approx f(x_n) + f'(x_n)\Delta x + f''(x_n)\Delta x^2 \tag{19}$$

where x_n is the current point, $x_n + \Delta x$ is the optimal point we want to find. Let the derivation about function (19) on Δx equal to 0, we have $\Delta x = \frac{-f'(x_n)}{f''(x_n)}$, and $x_{n+1} = x_n + \Delta x = x_n - \frac{f'(x_n)}{f''(x_n)}$. When using the stochastic gradient descent method to solve the unary function, $\Delta x = -\alpha f'(x_n)$, where α is a constant learning rate. Meanwhile, Newton method transforms the learning rate to an adaptive parameter, which is computed as the second derivative of the function, it improves the stability of the iterative process, and speed up the convergence [22].

Next, we consider the multivariate function. The variable in function (19) can be treated as a vector $X = (x_1, \ldots, x_n)^T$. When we use Newton method, $X_{n+1} = X_n - (H(f(X_n)))^{-1}\nabla f(X_n)$, where $\nabla f(X_n) = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})^T$, and $H(f(X_n))$ is the Hessian matrix constructed by the second-order partial derivative of $f(X_n)$, it is computed as:

$$H(f(X_n)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{20}$$

When using the stochastic gradient descent method to solve the multivariate function, $\Delta X = -\alpha \nabla f(X_n)$. Similarly, Newton method still can find a better decent path comparing with the stochastic gradient descent method.

The optimization problem in Function (14) is fixing U and updating V or fixing V and updating U, it is a convex optimization problem. Therefore, using Newton method to solve Function (14) would converge faster than stochastic gradient descent.

5.4. Discussion about the time complexity

Our method contains two steps in the process of optimization: (i) computing the first-order partial derivative vector and Hessian matrix, and (ii) updating U and V. Noting that the number of dimensions for U and V is usually small, and the Hessian matrix calculated by conjugate gradient method is in constant level. Thus, the time complexity for step (i) is $O(|R| + t|R|)$, where t is a constant, $t \ll |R|$. The time complexity for updating U and V in step (ii) is proportional to the number of users and items, i.e. $O(m + n)$, which is smaller than $O(|R|)$. Then, the total time complexity of our method is $O(|R|)$.

6. Experiments

We implement all the experiments on a PC with quad-core CPU (Intel i7, 3.6 GHZ), 32.0 GB main memory and 1T hard disk, running Microsoft Windows 10. We compare the performance about our model with PMF [23], HoORaYs [7] and NeuMF [17] on four real-world datasets, i.e. MovieLens1m,[1] Flixster,[2] Netflix[3] and Eachmovie.[4] Table 1 shows the statistics of the four datasets.

Table 1
Data statistics on 4 datasets.

Data	MovieLens1m	Flixster	Netflix	Eachmovie
Users	6040	147,612	32,703	61,265
Items	3706	48,794	17,770	1623
Ratings	1,000,209	7,334,548	9,615,132	2,911,983
Sparsity	95.53%	99.90%	98.35%	97.08%
Rating scale	[1–5]	[0.5–5]	[1–5]	[0.2–1]

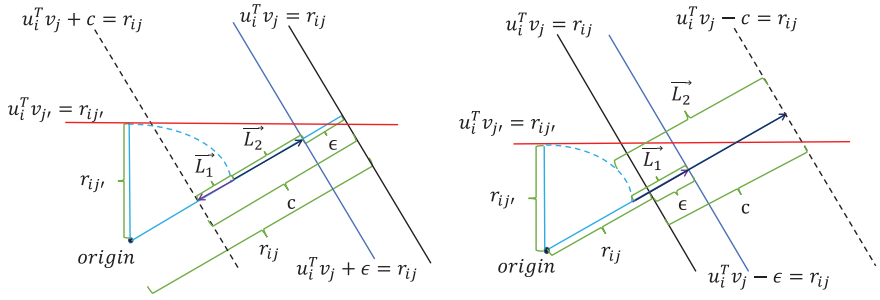
6.1. Methodology and metrics

In this paper, we measure two kinds of prediction accuracy about our method: (i) The accuracy of predictive ratings; and (ii) the accuracy of predictive actions.

Mean Absolute Error (*MAE*) and Root Mean Square Error (*RMSE*) are popularly used as the metrics to measure the accuracy of the predictive ratings, *MAE* and *RMSE* are computed correspondingly as:

$$MAE = \frac{\sum_{r_{ij} \in D_t} |\hat{r}_{ij} - r_{ij}|}{|D_t|} \tag{21}$$

[1] https://grouplens.org/datasets/movielens/.
[2] https://www2.cs.sfu.ca/sja25/personal/datasets/.
[3] http://www.netflixprize.com.
[4] http://www.research.digital.com/SRC/eachmovie/.



(a) $I^{(2)} = 1$: u_i lies in $u_i^T v_j + \epsilon = r_{ij}$ (b) $I^{(2)} = -1$: u_i lies in $u_i^T v_j - \epsilon = r_{ij}$

Fig. 4. Geometry interpretation about introducing slack variable ϵ.

$$RMSE = \sqrt{\frac{\sum_{r_{ij} \in D_t} (\hat{r}_{ij} - r_{ij})^2}{|D_t|}} \tag{22}$$

where D_t is testing set.

The final output of the recommendation algorithms to an end user is not the raw predictive ratings, but the likelihood of items being rated by this user. We choose the normalized Discounted Cumulative Gain (*nDCG*) to measure the goodness of ranked list by considering the item ratings with Discounted Cumulative Gain (*DCG*). Suppose $j_1, \ldots, j_l$ is a ranked list of items recommended to the target user, the *DCG* for user u at rank k and the *nDCG* are computed respectively as:

$$\begin{aligned} nDCG@K &= \frac{DCG@K}{iDCG@K} \\ DCG@K &= \sum_{i=1}^{k} \frac{2^{r(i)} - 1}{log_2(i+1)} \end{aligned} \tag{23}$$

where *iDCG@K* is the maximum possible gain value for user u that is obtained with the optimal re-order of K items in $j_1, \ldots, j_l$.

During the experiments, we compute *nDCG*@10, which requires the users for testing purchased at least 10 items. For the clarity of the experimental results, we eliminate those users whose number of rated items is less than 50, and use 80% of the preserved users as training set, 20% of them as testing set.

6.2. Parameter settings

We choose proper parameters for each comparison method by grid search. We test the regular parameters λ_u and λ_v in {0.1, 0.05, 0.01, 0.005, 0.001} and the learning rate α in {0.1, 0.05, 0.01, 0.005, 0.001}. Then, we choose $\lambda_u = 0.01, \lambda_v = 0.001$ and $\alpha = 0.005$ for PMF.

According to Function (6), HoORaYs needs to set 4 parameters manually: $\lambda_u, \lambda_v, \lambda_d$ and learning rate α. In particular, λ_u and λ_v are data-insensitive parameters, setting them as constants would not affect the algorithms' performance. Therefore, we use the same setting as Xu et al. [7] by setting $\lambda_u = 0.1, \lambda_v = 0.1$. Since λ_d is data-sensitive, we test the λ_d in {0.1, 0.05, 0.01, 0.005, 0.001}, then choose the optimal value for each dataset. We set $\lambda_d = 0.01$ for MovieLens1m, set $\lambda_d = 0.005$ for EachMovie and Flixster, and set $\lambda_d = 0.001$ for Netflix. We also select the proper value for α by setting $\alpha = 0.01$ for MovieLens1m and Eachmovie, setting $\alpha = 0.005$ for Flixster and Netflix.

NeuMF is designed for implicit datasets, it can be applied to explicit by setting loss function as mean squared error and removing the activation function of the output layer. Since NeuMF includes many hyper-parameters, we use the same setting as work [17], except we set batch size as 4096 to alleviate overfitting.

Our AE-HoORaYs model also needs to set λ_u and λ_v. We choose the same value as HoORaYs by setting $\lambda_u = 0.1, \lambda_v = 0.1$, respectively. Besides, our preliminary experiments show that the prediction accuracy will increase as the number of feature dimensions d of user/item latent vectors increase. However, when d is bigger than 45, such variation vanishes. During the experiments, we set the number of dimensions for all the latent vectors to be 45, and set the number of iterations for all the relevant methods to be 80.

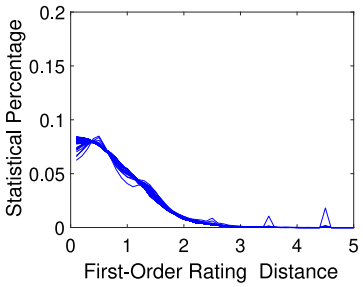
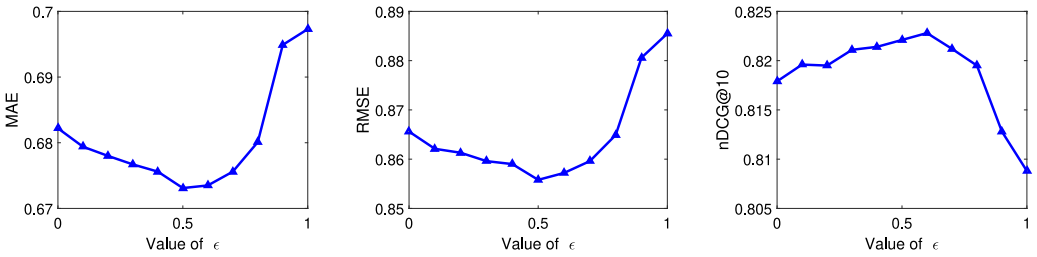


Fig. 5. Value distribution for the first-order rating distance on MovieLens1m.

6.3. Selection of slack variable ϵ

In order to find the reasonable range of ϵ, we first draw the value distribution for the first-order rating distance after 80 iterations, the result is shown in Fig. 5. We can see that, most of those values are in the range of (0, 1], it takes about 70% for MovieLens1m, i.e., if we set $\epsilon = 1$ in our AE-HoORaYs, about 70% of tuples are omitted from the computation of Function (14), which achieves a huge boost in efficiency.

Since the value of ϵ should be small, it is intuitive to focus the analysis for *MAE*, *RMSE* and *nDCG*@10 variation about $\epsilon \in (0, 1]$, the results are shown in Fig. 6. When considering the accuracy of predictive ratings, both *MAE* and *RMSE* get the best performance when $\epsilon = 0.5$ (see Fig. 6(a) and (b)). When considering the accuracy of predictive actions, the variations for *nDCG*@10 reaches the peak when $\epsilon = 0.6$ (see Fig. 6(c)), but there is not much difference comparing with the value when $\epsilon = 0.5$. During the experiments, we set $\epsilon = 0.5$ for MovieLens1m, which causes 40.6% tuples are omitted from the computation, and makes the efficiency doubled. Similarly, we set $\epsilon = r_{max}/10$ for the other three datasets.



(a) *Variation of MAE* (b) *Variation of RMSE* (c) *Variation of nDCG@10*

Fig. 6. Metrics variation about ϵ on MovieLens1m. Since the performance on Flixster, Netflix and Eachmovie have similar variations, we only show the curves on MovieLens1m.

Table 2
Prediction performance on MovieLens1m.

Measures	PMF	HoORaYs	NeuMF	AE-HoORaYs
MAE	0.6922	0.6821	0.6865	**0.6730**
RMSE	0.8689	0.8612	0.8760	**0.8558**
nDCG@10	0.8189	**0.8229**	0.7837	0.8221

Table 3
Prediction performance on Flixster.

Measures	PMF	HoORaYs	NeuMF	AE-HoORaYs
MAE	0.6622	0.6594	0.6465	**0.6441**
RMSE	0.8742	0.8715	0.8767	**0.8700**
nDCG@10	0.7673	0.7735	0.7597	**0.7747**

Table 4
Prediction performance on Netflix.

Measures	PMF	HoORaYs	NeuMF	AE-HoORaYs
MAE	0.6955	0.6822	**0.6665**	0.6707
RMSE	0.8793	0.8664	0.8623	**0.8582**
nDCG@10	0.8137	**0.8223**	0.7979	0.8218

Table 5
Prediction performance on Eachmovie.

Measures	PMF	HoORaYs	NeuMF	AE-HoORaYs
MAE	0.2238	0.2122	0.1916	**0.1865**
RMSE	0.2710	0.2597	0.2367	**0.2358**
nDCG@10	0.8726	0.8800	0.8735	**0.8965**

Table 6
Convergence performance on MovieLens1m.

Measures	PMF	HoORaYs	NeuMF	AE-HoORaYs
Iterations	20	16	42	**10**
Duration (s)	5.6	18.1	**1.3**	16.5
Total time (s)	112	289.6	**54.6**	165

Table 7
Convergence performance on Flixster.

Measures	PMF	HoORaYs	NeuMF	AE-HoORaYs
Iterations	57	43	39	**17**
Duration (s)	**44.5**	194.5	167.2	144
Total time (s)	2536.5	8363.5	6520.8	**2448**

Table 8
Convergence performance on Netflix.

Measures	PMF	HoORaYs	NeuMF	AE-HoORaYs
Iterations	43	52	31	**13**
Duration (s)	**57.1**	219.3	60.9	153.7
Total time (s)	2455.3	11403.6	**1887.9**	1998.1

Table 9
Convergence performance on Eachmovie.

Measures	PMF	HoORaYs	NeuMF	AE-HoORaYs
Iterations	23	17	33	**9**
Duration (s)	**11.6**	39.3	16.7	30.9
Total time (s)	**266.8**	668.1	551.1	278.1

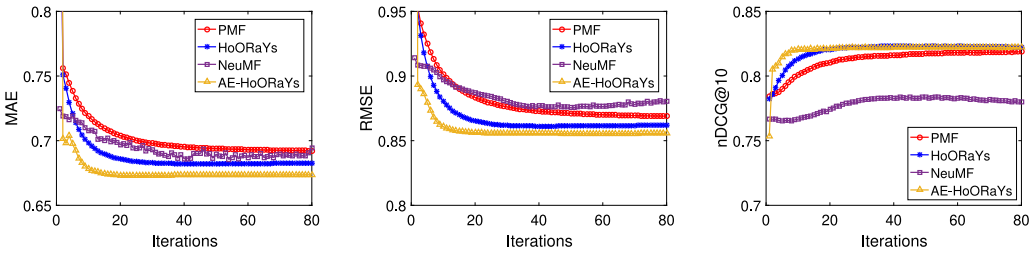
6.4. Evaluation results

We first illustrate the convergence curves for all the methods on four datasets with different scales. The results are shown in Figs. 7–10 on MovieLens1m, Flixster, Netflix, and Eachmovie correspondingly. We can see that, our AE-HoORaYs always converges first in all cases. HoORaYs also has a better convergence performance than PMF and NeuMF. Specifically, we have applied several methods to overcome the overfitting problem in NeuMF (such as changing the regularization method, increasing batch size, replacing activation function etc.), it still has a certain degree of overfitting on all datasets.

Next, we present the prediction accuracy for all methods on four datasets, the results are shown in Tables 2, 3, 4, and 5, respectively. We highlight the optimal values in bold type. We can see that, our method has excellent and stable performance comparing with the other methods on all datasets. Due to the utilization of slack variable, *nDCG*@10 for AE-HoORaYs is a little bit worse than HoORaYs on MovieLens1m and Netflix, but it brings a better performance improvement in terms of *MAE* and *RMSE* than HoORaYs. NeuMF has a better *MAE* on Flixster than AE-HoORaYs, but it has the worst *nDCG*@10, and performs unstable on different datasets.

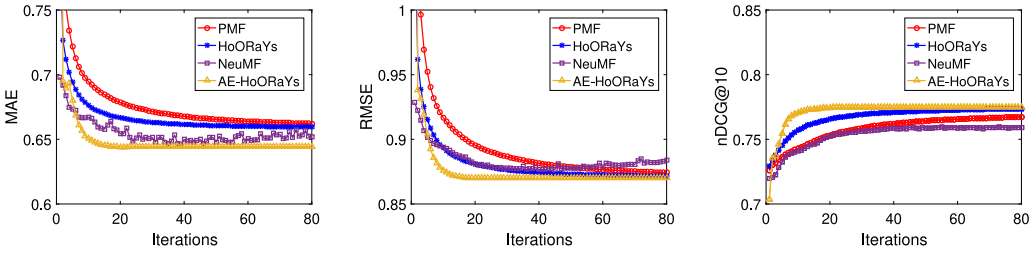
Finally, we compare the number of convergence iterations, the average duration of each iteration, and the total time for each model when it reaches convergence, the results are shown in Tables 6, 7, 8, and 9. We can see that, our AE-HoORaYs needs about 10 iterations to reach convergence on all datasets, while the others need more. Specifically, the time of one iteration for PMF, HoORaYs, and AE-HoORaYs are proportional to the number of training sets, while NeuMF is proportional to the product of the number of training sets and the sum of users and items. Since NeuMF introduces Tensorflow to accelerate the computation, its time for one iteration is short. One may notice that, AE-HoORaYs needs more time for training in one iteration comparing with PMF because of the additional computation for high-order distance. Yet, when comparing with HoORaYs, our AE-HoORaYs reduces the size of training sets, which takes less time for once iteration duration than HoORaYs. When considering the time consumption for total convergence, AE-HoORaYs is the fastest to converge on Flixster, and has a small gap comparing with the champion on the other three datasets.

In summary, the time complexity and space complexity of AE-HoORaYs are linearly related to the size of dataset, which makes



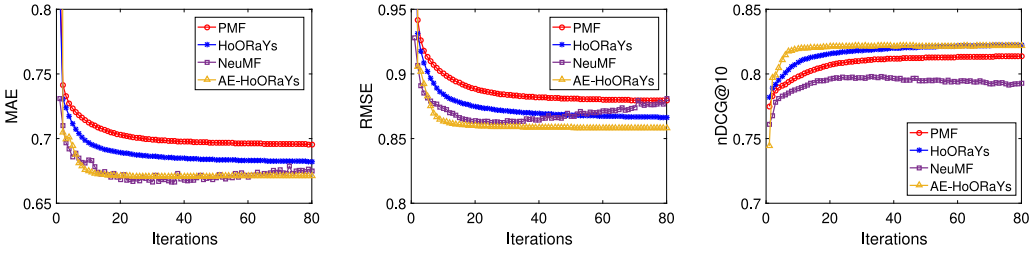
(a) *Variation of MAE* (b) *Variation of RMSE* (c) *Variation of nDCG@10*

Fig. 7. Convergence curves for *MAE*, *RMSE* and *nDCG*@10 on MovieLens1m.



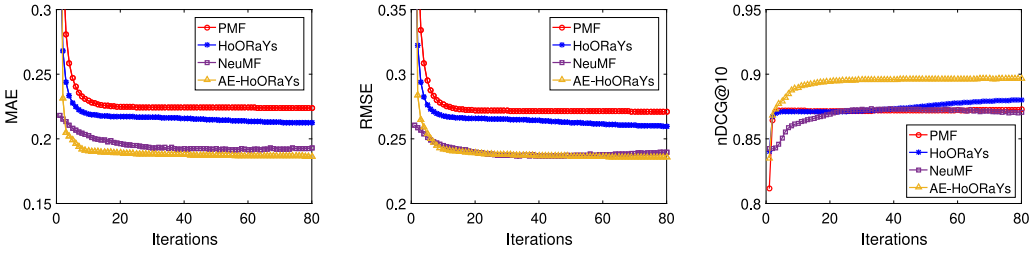
(a) *Variation of MAE* (b) *Variation of RMSE* (c) *Variation of nDCG@10*

Fig. 8. Convergence curves for *MAE*, *RMSE* and *nDCG*@10 on Flixster.



(a) *Variation of MAE* (b) *Variation of RMSE* (c) *Variation of nDCG@10*

Fig. 9. Convergence curves for *MAE*, *RMSE* and *nDCG*@10 on Netflix.



(a) *Variation of MAE* (b) *Variation of RMSE* (c) *Variation of nDCG@10*

Fig. 10. Convergence curves for *MAE*, *RMSE* and *nDCG*@10 on Eachmovie.

it be applied on datasets with any size. Due to the inevitable noise of real-world data, introducing slack variable can omit part of updating process without losing accuracy. Besides, our method uses Newton method instead of stochastic gradient descent can improves the convergence speed, and achieves the optimal prediction performance. It is superior than HoORaYs in terms of accuracy and efficiency. Since our method does not need many manually parameters, it demonstrates its adaptability comparing to HoORaYs and NeuMF.

7. Conclusion

In this paper, we propose an adaptive and efficient high-order rating distance model by introducing slack variable, and utilizes Newton method to achieve the minimization about the first-order rating distance and the second-order rating distance to accelerate the convergence speed. Our model has solid theoretical support and preserves good prediction accuracy. Experimental comparison between our model and the state-of-the-art on four

stable datasets demonstrate the superiority of our method in terms of adaptability and efficiency.

There are several interesting directions for our future work. Firstly, Hashing for collaborative filtering [24–26] has attracted increasing attention, the binary codes can significantly reduce the storage requirement and make similarity calculations efficient. We would like to explore how to find a balance between the recommendation accuracy and efficiency by transferring user–item feature vectors into binary code with less precision loss. Secondly, the diversity-oriented recommendation [27,28] has been recognized as a crucial issue. We would like to improve our AE-HoORaYs model for generating recommendation list both accurate and diverse. Thirdly, auxiliary information has always been used to improve the performance of recommender system [29, 30], such as social-network, review context, etc. We would like to utilize these auxiliary information to enhance the performance of our model in near future.

CRediT authorship contribution statement

Yifan He: Conceptualization, Methodology, Writing - original draft. **Haitao Zou:** Validation, Formal analysis , Writing - review & editing, Data curation. **Hualong Yu:** Visualization , Investigation, Supervision. **Shang Zheng:** Supervision. **Shang Gao:** Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by Postgraduate Research & Practice Innovation Program of Jiangsu Province, China No. KYCX19_1698; Scientific Research Foundation for the introduction of talent of Jiangsu University of Science and Technology, China; Natural Science Foundation of the Higher Education Institutions of Jiangsu Province, China *GrantNo.* 18JKB520011; Primary Research & Development Plan *Social Development* of Zhenjiang City, China *GrantNo.* SH2019021; Natural Science Foundation of Jiangsu Province, China *GrantNo.* BK20191457. Haitao Zou was also supported in part by the High Level Talents Scientific Research Fund in Jiangsu University of Science and Technology under grants 1132921506.

Any opinions, findings, conclusions, and/or recommendations expressed in this material, either expressed or implied, are those of the authors and do not necessarily reflect the views of the sponsors listed above.

References

[1] C.D. Sa, M. Feldman, C. R, K. Olukotun, Understanding and optimizing asynchronous low-precision stochastic gradient descent, ACM SIGARCH Comput. Archit. News 45 (2) (2017) 561–574, http://dx.doi.org/10.1145/3140659.3080248.

[2] J. Oh, W. Han, H. Yu, X. Jiang, Fast and robust parallel sgd matrix factorization, in: Proceedings of the 21th International Conference on Knowledge Discovery and Data Mining, 2015, pp. 865–874, http://dx.doi.org/10.1145/2783258.2783322.

[3] K. Zhou, H. Zha, Learning binary codes for collaborative filtering, in: The 18th ACM International Conference on Knowledge Discovery and Data Mining, 2012, pp. 498–506, http://dx.doi.org/10.1145/2339530.2339611.

[4] Z. Zhang, Q. Wang, L. Ruan, L. Si, Preference preserving hashing for efficient recommendation, in: Proceedings of the 37th International ACM Conference on Research and Development in Information Retrieval, 2014, pp. 183–192, http://dx.doi.org/10.1145/2600428.2609578.

[5] P. Bhargava, T. Phan, J. Zhou, J. Lee, Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 130–140, http://dx.doi.org/10.1145/2736277.2741077.

[6] Z. Cheng, Y. Ding, L. Zhu, M.S. Kankanhalli, Aspect-aware latent factor model: Rating prediction with ratings and reviews, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 639–648, http://dx.doi.org/10.1145/3178876.3186145.

[7] J. Xu, Y. Yao, H. Tong, X. Tao, J. Lu, Hoorays: High-order optimization of rating distance for recommender systems, in: Proceedings of the 23rd ACM International Conference on Knowledge Discovery and Data Mining, 2017, pp. 525–534, http://dx.doi.org/10.1145/3097983.3098019.

[8] H. Zou, Z. Gong, W. Hu, Identifying diverse reviews about products, World Wide Web 20 (2) (2017) 351–369, http://dx.doi.org/10.1007/s11280-016-0391-3.

[9] S. Roy, S.C. Guntuku, Latent factor representations for cold-start video recommendation, in: Proceedings of the 10th ACM Conference on Recommender Systems, 2016, pp. 99–106, http://dx.doi.org/10.1145/2959100.2959172.

[10] J. Cheng, T. Yuan, J. Wang, H. Lu, Group latent factor model for recommendation with multiple user behaviors, in: Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2014, pp. 995–998, http://dx.doi.org/10.1145/2600428.2609493.

[11] J. Zhu, J. Zhang, L. He, Q. Wu, B. Zhou, C. Zhang, P.S. Yu, Broad learning based multi-source collaborative recommendation, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1409–1418, http://dx.doi.org/10.1145/3132847.3132976.

[12] S. Kabbur, X. Ning, G. Karypis, 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, 2013, pp. 659–667, http://dx.doi.org/10.1145/2487575.2487589.

[13] D. Park, J. Neeman, J. Zhang, S. Sanghavi, I.S. Dhillon, Preference completion: Large-scale collaborative ranking from pairwise comparisons, in: Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 1907–1916.

[14] L. Wu, C. Hsieh, J. Sharpnack, Large-scale collaborative ranking in near-linear time, in: Proceedings of the 23rd ACM International Conference on Knowledge Discovery and Data Mining, 2017, pp. 515–524, http://dx.doi.org/10.1145/3097983.3098071.

[15] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Comput. Surv. 52 (1) (2019) 5:1–5:38, http://dx.doi.org/10.1145/3285029.

[16] S. Sedhain, A.K. Menon, S. Sanner, L. Xie, Autorec: autoencoders meet collaborative filtering, in: Proceedings of the 24th International Conference on World Wide Web Companion, 2015, pp. 111–112, http://dx.doi.org/10.1145/2740908.2742726.

[17] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 173–182, http://dx.doi.org/10.1145/3038912.3052569.

[18] J. Hu, P. Li, Decoupled collaborative ranking, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 1321–1329, http://dx.doi.org/10.1145/3038912.3052685.

[19] D. Liang, J. Altosaar, L. Charlin, D.M. Blei, Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence, in: Proceedings of the 10th ACM Conference on Recommender Systems, 2016, pp. 59–66, http://dx.doi.org/10.1145/2959100.2959182.

[20] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.

[21] M. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Natl. Bur. Stand. 49 (6) (1952) 409–436, http://dx.doi.org/10.6028/jres.049.044.

[22] E.Y. Rodin, Nonlinear programming analysis and methods, Comput. Math. Appl. 3 (2) (1977) 151.

[23] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: Proceedings of the 21th Annual Conference on Neural Information Processing Systems, 2007, pp. 1257–1264.

[24] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, T. Chua, Discrete collaborative filtering, in: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2016, pp. 325–334, http://dx.doi.org/10.1145/2911451.2911502.

[25] Y. Li, S. Wang, Q. Pan, H. Peng, T. Yang, E. Cambria, Learning binary codes with neural collaborative filtering for efficient recommendation systems, Knowl. Based Syst. 172 (2019) 64–75, http://dx.doi.org/10.1016/j.knosys.2019.02.012.

[26] G. Guo, E. Yang, L. Shen, X. Yang, X. He, Discrete trust-aware matrix factorization for fast recommendation, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, 2019, pp. 1380–1386, http://dx.doi.org/10.24963/ijcai.2019/191.

[27] P. Cheng, S. Wang, J. Ma, J. Sun, H. Xiong, Learning to recommend accurate and diverse items, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 183–192.
[28] Y. He, H. Zou, H. Yu, Q. Wang, S. Gao, Diversity-aware recommendation by user interest domain coverage maximization, in: 2019 IEEE International Conference on Data Mining, 2019, pp. 1084–1089, http://dx.doi.org/10.1109/ICDM.2019.00128.
[29] C. Liu, X. Wang, T. Lu, W. Zhu, J. Sun, S.C.H. Hoi, Discrete social recommendation, in: The 31th Innovative Applications of Artificial Intelligence Conference, 2019, pp. 208–215, http://dx.doi.org/10.1609/aaai.v33i01.3301208.
[30] J. Ding, G. Yu, Y. Li, X. He, D. Jin, Improving implicit recommender systems with auxiliary data, ACM Trans. Inf. Syst. 38 (1) (2020) 11:1–11:27, http://dx.doi.org/10.1145/3372338.