Journal of Big Data

METHODOLOGY

Open Access

An efficient annealing-assisted differential evolution for multi-parameter adaptive latent factor analysis

Qing Li[1,2], Guansong Pang[3] and Mingsheng Shang[2*]

*Correspondence:
msshang@cigit.ac.cn

[1] College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[2] Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China
[3] School of Computing and Information Systems, Singapore Management University, Singapore 188065, Singapore

Abstract

A high-dimensional and incomplete (HDI) matrix is a typical representation of big data. However, advanced HDI data analysis models tend to have many extra parameters. Manual tuning of these parameters, generally adopting the empirical knowledge, unavoidably leads to additional overhead. Although variable adaptive mechanisms have been proposed, they cannot balance the exploration and exploitation with early convergence. Moreover, learning such multi-parameters brings high computational time, thereby suffering gross accuracy especially when solving a bilinear problem like conducting the commonly used latent factor analysis (LFA) on an HDI matrix. Herein, an efficient annealing-assisted differential evolution for multi-parameter adaptive latent factor analysis (ADMA) is proposed to address these problems. First, a periodic equilibrium mechanism is employed using the physical mechanism annealing, which is embedded in the mutation operation of differential evolution (DE). Then, to further improve its efficiency, we adopt a probabilistic evaluation mechanism consistent with the crossover probability of DE. Experimental results of both adaptive and non-adaptive state-of-the-art methods on industrial HDI datasets illustrate that ADMA achieves a desirable global optimum with reasonable overhead and prevails competing methods in terms of predicting the missing data in HDI matrices.

Keywords: Big data analysis, Latent factor analysis, Simulated annealing, Differential evolution algorithm, Multi-parameter adaptive

Introduction

An increasing number of industrial applications have focused on exploring high-dimensional and incomplete (HDI) matrices in the artificial intelligence era including e-commerce recommendation systems, social networks analysis, biometric feature recognition etc. [1–5]. The missing items in an HDI matrix are probably the most valuable information that can be exploited by latent factor analysis [6], matrix factorization [7], collaborative filtering [8], and neural network-related models [9]. They are constructed to approximate the high-dimensional data with low-dimensional features that can represent the unobserved data in the HDI matrix.

Latent factor analysis (LFA) model is a nontrivial model for big data analysis problems, which represents the HDI matrix with magnitude lower-dimensional latent factor matrices. From the learned features of users and items in the latent matrices, missing items of interested users can be acquired when meeting a goal built as an objective function. To date, there are symmetric, non-negative, dynamic, and deep latent factor models, etc. [10–14]. Song et al. [10] integrate triple-factorization-based symmetric into a nonnegative latent factor model for a lower expenditure of time and storage space. Luo et al. [11] establish an ingenious method of unconstrained non-negative latent factor analysis method through a single-element-dependent mapping function. Bong et al. [12] focus on the cross-correlations among the latent variables in neural recordings, the dynamic high-dimensional time series data, to capture target factors. Wu et al. [14] provide a deep-structured model that successively connects multiple latent factor models with computational complexity linearly related to its layer count.

In practical application, however, extra parameters in these models unavoidably affect their performance. Recently, evolutionary computation-based methods, one of the most suitable and effective adaptive techniques with the swarm intelligence, arose. Differential evolution (DE), innovatively proposed by Storn and Price for solving the Chebyshev polynomials, is a simple yet efficient global numerical optimization evolutionary algorithm [15]. Instead of mimicking biological or social behaviors of ants [16], birds [17], bees [18], or humans [19], DE directly inherits the mechanisms of evolutionary: mutation, crossover, and selection. In fact, these three mechanisms are also included in a genetic algorithm (GA) [20], but DE directly employs the parent individuals to generate the offspring in which the gene representation process is omitted. Mutation, the decisive mechanism of DE, determines the range of generated offspring. For instance, with different selection mechanism of fundamental individual, there are standard DE (DE/rand/1), DE/best/1, DE/cur-to-best/1, DE/best/2, DE/rand/2, DE/rand-to-best/1, and DE/rand-to-best/1 schemes [21]. However, offspring random selection operation in mutation mechanism restricts its exploration and exploitation abilities, thereby inducing search stagnation and premature convergence. Engaged by this problem, many studies propose new models like DEHM [22], GA-KMEANS [23], and NBOLDE [24]. All these methods enhance the performance of convergence more or less, but they did not focus on the internal operation of mutation mechanism.

To solve multi-parameter adaptive latent factor analysis on an HDI matrix with the accuracy and efficiency, this paper presents an efficient annealing-assisted differential evolution for adaptive multi-parameter latent factor analysis (ADMA) utilizing the standard DE to train parameters. To further improve its efficiency, with the probability of the crossover in DE, some unchanged individuals will not participate in the evaluation process, which efficiently reduces the total computational time in DE. In addition, a physical mechanism annealing is further incorporated to help the model overstep its local saddle points, thereby receiving a desirable performance in predicting missing data. Contributions of this study can be summarized as follows:

1) An ADMA algorithm, an adaptive multi-parameter latent factor analysis on an HDI matrix, witch adopts the philosophy of annealing mechanism to improve the exploration and exploitation abilities of DE;

2) Efficient self-adaptive mechanism for multi-parameter estimation in the dynamic training process of the proposed latent factor analysis model;
3) Improved DE for analyzing HDI matrices in real industrial applications.

The rest of this paper is organized as follows. Section II illustrates the preliminaries, Section III introduces the ADMA model, Section IV analyzes the performance of the ADMA algorithm as well as the state-of-the-art, and lastly, the work is conducted in Section V.

Related work

Parameter selection has been revealed as one of the most significant challenges especially in big data analysis tasks [25–28]. Manual fine-tuning of the parameters is inevitably time consuming and cannot achieve high accuracy in capricious industrial applications. Along this line, parameter adaptable algorithms are proposed as a promising approach to address the problem. Various techniques have been developed for parameter adaptive. Pertaining to traditional optimization algorithm, Miao et al. [29] proposed a novel parameter-adaptive variational mode decomposition (VMD) by grasshopper optimization algorithm (GOA) applied to compound-fault diagnosis, which can acquire a competitive decomposition results more efficiently. It can significant decline the overhead triggered by the long-term and harsh operation environment. Moreover, machine learning related methods are also utilized to solve this problem. For example, Xiao et al. [30] employ an unsupervised reinforcement learning to dynamically acquire the parameters in navigation systems of robots.

Among the parameter adaptive algorithms, three main categories of their adaptive mechanisms, which is mentioned in Table 1, can be concluded as follows:

1) Analytical approach-based. The motivation of analytical approach is controlling the updating process by analyzing the objective function in a big data analysis task. El-Naggar et al. [31] firstly propose a simulated annealing approach-based solar cell model parameters estimation method. Na et al. [32] estimate the time-varying parameters from parameter estimation errors that drive several new adaptive laws

Table 1 Classification of parameter adaptive mechanisms

Classifications	Adaptive mechanisms	Defects	HDI data analysis	Multi-parameters
Analytical approach	Simulated annealing approach[31]	Internal assumptions	×	×
	Parameter estimation adjustment[32]	Manual tunning	×	×
Dynamic optimization	NE[33]	Need Priori Knowledge	×	×
	Gradient-based estimation[34]	Sensitive to initialization	×	×
Evolutionary computation	PSO[36]	Premature convergence	√	×
	BSO[37]	time consuming	√	√

ADMA employs evolutionary computation DE as the multi-parameter adaptive mechanism and improve its optimization progress by an annealing technique. Therefore, ADMA inherits the flexibility and adaptively of swarm intelligence, while enjoying the exploration and exploitation merits of the improved DE, which thereby can solve the big data analysis problems well especially conducting an LFA on an HDI matrix

with a faster convergence rate. Noticeable, analytical approach-based mechanisms are efficient and concise, but it is unavoidable to set internal assumptions precisely in advance.

2) Dynamic optimization method-based. For a more intelligent parameter adaptation, some dynamic optimization methods are employed into the parameters training, such as the Newton–Euler (NE) method and the stochastic gradient descent (SGD) method. Yang et al. [33] identify dynamic parameters of robot systems by a robot control scheme based on the NE method accelerating the convergence speed. Kapetina et al. [34] propose a gradient-based parameter estimation technique that can track sufficiently slow changes in process and parameters. Although dynamic optimization method-based methods are intelligent in searching optimal solutions, the searching results are highly sensitive to their initialization and searching knowledge.
3) Evolutionary computation-based. Despite the methods based on dynamic optimization rules, using evolutionary algorithms such as ant colony optimization (ACO) [35], particle swarm optimization (PSO) [36], gene and brain storm optimization (BSO) [37], etc. is an advanced genre in parameter adaptable algorithms. For example, Luo et al. [36] incorporate PSO into the latent factor analysis that makes the big data analysis convergence faster than traditional SGD-based mechanism. The evolutionary computation-based method neither tracks error with bound nor require the knowledge of dynamics. Unfortunately, the evolution process is time consuming and has premature convergence.

Peliminaries

Problem statement

The key problem of finding an LFA on an HDI scoring matrix is stated as follows:

Definition 1. Suppose M and N are entity sets of users and items respectively, each of the entity $m \in M$ grades the entity $n \in N$ with the score $z_{m,n}$ that constitutes the HDI scoring matrix $Z^{|M|\times|N|}$; The entities in an HDI matrix can be divided into two sets Λ and Γ, where the known entry set Λ is far less in size than that of the unknown entry set Γ, namely, $|\Lambda| \ll |\Gamma|$.

Definition 2. Suppose there are $P^{|M|\times D}$ and $Q^{|N|\times D}$ latent factor matrices for M and N respectively, from which a rank-d matrix $Z = PQ^{\mathrm{T}}$ is learned to approximate Z via an LFA model. With these latent factor matrices, the dimensional space of the approximation matrix Z of an HDI matrix is far less than that of entities of users and items, i.e., $D \ll \min\{|M|, |N|\}$.

The objective of the LFA model is to predict unknown entities in Γ via learned latent factor matrices P and Q from the limited known entry set Λ. Along this line, the unknown information in an HDI matrix can be acquired, which is theoretically expressed as solving the following Euclidean distance measure problem [10, 11]:

$$\min_{P,Q} \varepsilon(P,Q) = \sum_{z_{m,n}} \left(z_{m,n} - \sum_{d=1}^{D} p_{m,d} q_{n,d} \right)^2 \tag{1}$$

where $z_{m,n}$ represents a known element in HDI matrix Z, $p_{m,d}$ demotes the m-th row vector of P in the *d-th* dimension, while $q_{n,d}$ denotes the n-th row vector of Q in the *d-th* dimension.

Due to the ill-posed characteristic of solving (1), the commonly used Tikhonov regularization is employed to fight against the disturbance from the model. Along this line, the fundamental Eq. (1) is accordingly extended into:

$$\begin{aligned} \arg\min_{P,Q} \varepsilon(P,Q) &= \sum_{z_{m,n} \in \Lambda} \left(\left(z_{m,n} - \hat{z}_{m,n} \right)^2 + \lambda_P \left\| p_{m,\cdot} \right\|_2^2 + \lambda_Q \left\| q_{n,\cdot} \right\|_2^2 \right) \\ &= \sum_{z_{m,n} \in \Lambda} \left(\left(z_{m,n} - \hat{z}_{m,n} \right)^2 + \lambda_P \sum_{d=1}^{D} p_{m,d}^2 + \lambda_Q \sum_{d=1}^{D} q_{n,d}^2 \right) \end{aligned} \tag{2}$$

where $\hat{z}_{m,n} = \sum D\ d = 1\ pm_{,d} q_{n,d}$, λ_P and λ_Q are set as regularization constants of P and Q respectively, and $||||_2$ denotes L_2norm.

An SGD-based LFA model

An LFA model is commonly addressed by the stochastic gradient descent (SGD) algorithm that shows the stability, flexibility, and advanced feedback control in solving big data analysis issues, especially on HDI data. Based on the SGD algorithm, matrices P and Q in (2) can be updated as follows:

$$\arg\min_{P,Q} \varepsilon(P,Q) \overset{SGD}{\Rightarrow} \forall z_{m,n} \in \Lambda, d \in \{1,2,\ldots,D\} : p_{m,d}^t \leftarrow p_{m,d}^{t-1} - \eta \frac{\partial \varepsilon_{m,n}^{t-1}}{\partial p_{m,d}^{t-1}},\ q_{n,d}^t \leftarrow q_{n,d}^{t-1} - \eta \frac{\partial \varepsilon_{m,n}^{t-1}}{\partial q_{n,d}^{t-1}}. \tag{3}$$

where $\varepsilon_{m,n} = (z_{m,n} - \hat{z}_{m,n})^2 + \lambda_P ||\boldsymbol{p}_m||2\ 2 + \lambda_Q ||\boldsymbol{q}_n||2\ 2$, in which $\boldsymbol{p}_m$demotes the m-th row vector of P and $\boldsymbol{q}_n$ denotes the n-th row vector of Q, η denotes the learning rate in the SGD, and the detailed update process for (2) in (3) is derived as follows:

$$\frac{\partial \varepsilon_{m,n}^{t-1}}{\partial p_{m,d}^{t-1}} = -err_{m,n}^{t-1} \cdot q_{n,d}^{t-1} + \lambda_P \cdot p_{m,d}^{t-1}, \frac{\partial \varepsilon_{m,n}^{t-1}}{\partial q_{n,d}^{t-1}} = -err_{m,n}^{t-1} \cdot p_{m,d}^{t-1} + \lambda_Q \cdot q_{n,d}^{t-1}. \tag{4}$$

where $err_{m,n} = z_{m,n} - \hat{z}_{m,}$ denotes an instant error. As noted above, there are three hyper-parameters that need to be tuned including the learning rate η in SGD and the regularization parameters λ_P and λ_Q. The model can fast converge with impressive performance when these hyper-parameters are properly tuned.

Differential evolution

As a particle swarm optimization algorithm, original differential evolution (DE) conducts its mutation, crossover, and update operations on N_C particles that can be represented as set $X = \{\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,K}] |\ \boldsymbol{x}_i \in \Omega,\ 1 \le \mathrm{i} \le N_C\}$ where $\Omega = R^K$ is the feasible solution space. The mutation operation in t-th iteration generates a new particle $\boldsymbol{v}_i^t = \left(v_{i,1}^t, v_{i,2}^t, \ldots, v_{i,K}^t\right)$. In

the original DE, the new particle is generated based on a random particle$x_{r1,k}^{t-1}$ in X, and a differential value of the other two random particles $x_{r2,k}^{t-1}$ and $x_{r3,k}^{t-1}$:

$$v_{i,k}^{t} = x_{r1,k}^{t-1} + \gamma\left(x_{r2,k}^{t-1} - x_{r3,k}^{t-1}\right) \tag{5}$$

where *r1, r2, r3* are different random constants in the range of [1,N_C], γ is a scaling factor between 0 and 1.

Crossover decides whether the new particle can be inherited. With the probability of P_{select} or when the operation is on the i_{select}-th particle, the particle $\boldsymbol{u}_i^t=(u_{i,1}^t, u_{i,2}^t, ..., u_{i,K}^t)$ is replaced by the mutation particle. Instead, it will inherit the original particles. It is processed as follows:

$$u_{i,k}^{t} = \begin{cases} v_{i,k}^{t}, \textit{ if } rand(0,1) < p_{select} \textit{ or } i = i_{select}, \\ x_{i,k}^{t-1}, \text{else.} \end{cases} \tag{6}$$

where i_{select} is a fixed constant in the range of [1,N_C]. This guarantees that at least one particle will be inherited by the new particle.

Selection operation chooses the better particle as$\boldsymbol{x}_i^t$ in the iteration t from $\boldsymbol{u}_i^t$ and $\boldsymbol{x}_i^{t-1}$ after comparing the evaluation results of them, which is defined as:

$$x_{i,k}^{t} = \begin{cases} u_{i,k}^{t}, \textit{ if } f\left(u_{i,k}^{t}\right) < f\left(x_{i,k}^{t-1}\right), \\ x_{i,k}^{t-1}, \text{else.} \end{cases} \tag{7}$$

where f is the objective function.

After limited iterations for all particles, such as, meeting the maximal iteration count or multiple rounds of updates unchanged, the algorithm meets the termination criteria and acquires a final solution.

Methods

An annealing-assisted differential evolution algorithm

To further improve the performance of DE, the cosine anneal mechanism is incorporated into the original DE algorithm for adjusting the parental particles in mutation operation. Following the definition of the original ED, the cosine annealing mechanism can be defined as a period function within minimal and maximal value of the target x_{min} and x_{max}:

$$x_{anneal}^{t} = x_{\min} + \frac{1}{2}(x_{\max} - x_{\min})\left(1 + \cos\frac{t\pi}{T}\right) \tag{8}$$

where x_{anneal}^{t} is the anneal value at t-th iteration, and the maximal iteration time is T.

In the mutation operation, two of three parents are selected to calculate their differential values between the anneal values, and the equilibrium factor μ is set to balance them and the original differential value between the selected parents, which are defined as:

$$v_{i,k}^{t} = x_{r1,k}^{t-1} + \gamma(1-\mu)\left(x_{r2,k}^{t-1} - x_{r3,k}^{t-1}\right) + \gamma\mu\left(x_{anneal}^{t} - x_{r2,k}^{t-1}\right) + \gamma\mu\left(x_{anneal}^{t} - x_{r3,k}^{t-1}\right) \tag{9}$$

where $\boldsymbol{v}_i^t=(v_{i,1}^t, v_{i,2}^t, ..., v_{i,K}^t)$, *r1, r2, r3* are different random constants in the range of $[1,N_C]$, and γ is the scaling factor in the range of (0,1).

To further improve its efficiency, evaluation operation in HDI data analysis task need to be conducted more concisely, which is consistent with estimating particles. Specifically, after the crossover operation, some particles are unchanged, whose evaluation results can be reserved without further estimation, in contrast, other renewed particles need to be evaluated exactly.

An ADMA model

From the previous information of the improved DE, the flowchart of the proposed ADMA can be illustrated as Fig. 1, where the red boxes represent the improvements. In order to make parameters adaptive in solving the LFA model on an HDI matrix, three-dimensional particles (learning rate η and regularization constants λ_P and λ_Q) need to learn using the improved DE, i.e., the problem dimension $K=3$ in this part which means $X=\{\boldsymbol{x}_i=[x_{i,P},x_{i,2},x_{i,3}]|\ \boldsymbol{x}_i\in\Omega, 1\leq i\leq N_C\}$ where $\Omega=R^3$.

As discussed in Sect. 3.1, the mutation particle $\boldsymbol{v}_i^t=(v_{i,1}^t, v_{i,2}^t, v_{i,3}^t)$ is limited in the lower and upper bounds:

$$v_{i,k}^t=\begin{cases} x_{\max,k}, & v_{i,k}^t > x_{\max,k} \\ x_{\min,k}, & v_{i,k}^t < x_{\min,k} \end{cases} \tag{10}$$

where empirical settings of these parameters' boundaries are $x_{\max,1}=2^{-8}$, $x_{\min,1}=2^{-12}$, and $x_{\max,2}=x_{\max,3}=2^{-3}$, $x_{\min,2}=x_{\min,3}=2^{-7}$, respectively in the vector $\boldsymbol{x}_{min}$ and $\boldsymbol{x}_{max}$.

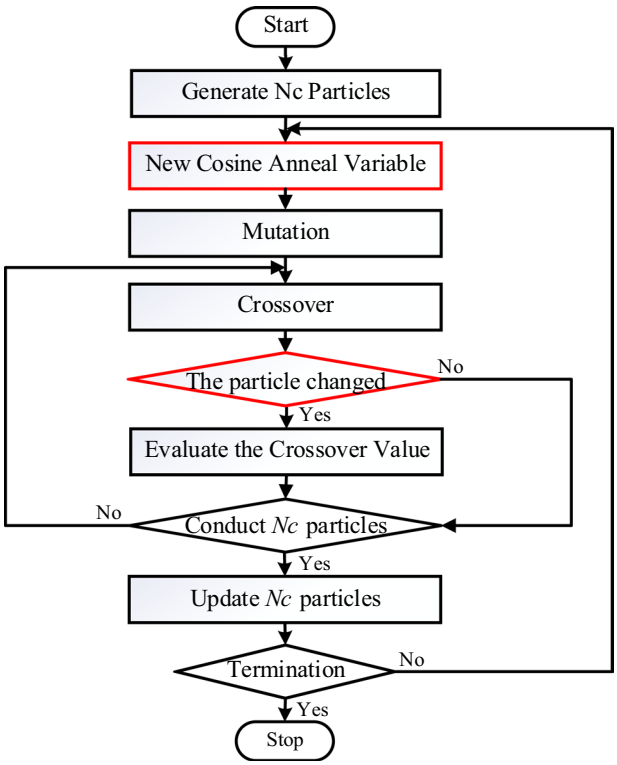

Fig. 1 Flowchart of ADMA

Crossover operation is the same as the original one, which is:

$$u_{i,k}^{t} = \begin{cases} v_{i,k}^{t}, \text{ if } rand(0,1) < p_{select} \text{ or } i = i_{select}, \\ x_{i,k}^{t-1}, \text{else.} \end{cases} \tag{11}$$

where $\boldsymbol{u}_i^t = \left(u_{i,1}^t, u_{i,2}^t, u_{i,3}^t\right)$, i_{select} is in the range of $[1, N_C]$, and P_{select} is the probability of selecting mutation particles.

Based on Eqs. (3) and (4), when the crossover particle is replaced by the mutation particle, the update of P and Q is completed in the evaluation operation as follows:

$$p_{m,d}^{t} \leftarrow p_{m,d}^{t-1} + u_{i,1}^{t-1}\left(err_{m,n}^{t-1} q_{n,d}^{t-1} - u_{i,2}^{t-1} p_{m,d}^{t-1}\right), q_{n,d}^{t} \leftarrow q_{n,d}^{t-1} + u_{i,1}^{t-1}\left(err_{m,n}^{t-1} p_{m,d}^{t-1} - u_{i,3}^{t-1} q_{n,d}^{t-1}\right). \tag{12}$$

where $u_{i,1}^t$ represents the learning rate η on the SGD-based model, $u_{i,2}^t$ and $u_{i,3}^t$ denote the regularization constants λ_P and λ_Q of P and Q respectively.

Selection operation chooses the particle with a better evaluation value between the replaced particle and the original one, which is defined as:

$$x_{i,k}^{t} = \begin{cases} u_{i,k}^{t}, \text{ if the particle changed}, f\left(u_{i,k}^{t}\right) < f\left(x_{i,k}^{t-1}\right), \\ x_{i,k}^{t-1}, \text{else.} \end{cases} \tag{13}$$

where $\boldsymbol{x}_i^t = \left(x_{i,1}^t, x_{i,2}^t, x_{i,3}^t\right)$, f is the RMSE or MAE evaluation function in this method.

In the end, if the model meets the termination criteria that the best evaluation value is unchanged for five iterations or the maximum iteration time 1000 is reached, the latest optimum latent factor matrices are returned as the approximate results of the model.

Algorithm design and analysis

As is shown in Fig. 1, the flowchart of algorithm ADMA demonstrates that there are two additional mechanisms: an anneal mechanism and a screening mechanism of evaluation compared with the traditional DE algorithm. Evaluation for latent factor matrices learned from an HDI matrix in this model is the most time-consuming part, which can also be shown in the Algorithm ADMA.

ALGORITHM ADMA

Input: Λ, Γ, M, N, D	
Operation	**Cost**
Initialize $P^{M\times D}$, $Q^{N\times D}$	$\Theta(\lvert M\rvert+\lvert N\rvert)\times f$
Initialize λ_P, λ_Q, N_C, D, K, γ, μ, $\boldsymbol{x}_{\max}$, $\boldsymbol{x}_{\min}$	$\Theta(1)$
Initialize U^{N_C}, V^{N_C}, X^{N_C}	$\Theta(N_C)$
Initialize t=1, T=Maximum Round Count	$\Theta(1)$
while not converge **and** $t \le T$ **do**	$\times C$
for each $\boldsymbol{x}_i$ in X **do**	$\times N_C$
Update η_{anneal} for each dimension by (8)	$\Theta(K)$
Update v_i by (10)	$\Theta(K)$
Update u_i by (11)	$\Theta(K)$
if crossover particle is replaced by mutation particle	$\Theta(1)$
for each $p_{m,d}$, $q_{n,d} \in \Lambda$	$\times\lvert\Lambda\rvert$
Update $p_{m,d}$ and $q_{n,d}$ by (12)	$\Theta(D)$
end for	--
Obtain evaluation value of x_i	$\Theta(\lvert\Gamma\rvert)$
end if	--
Update x_i by (13)	$\Theta(K)$
end for	--
t=t+1	$\Theta(1)$
end while	--
Output: P, Q	

Time complexity

To train the latent factor matrices in ADMA, the following steps are sequentially performed at each iteration: cosine anneal for each dimension, mutation, crossover, evaluation screening, evaluation, and replace for particles, whose total time complexity is:

$$T_{ADMA} = \Theta(C \times (N_C \times (|\Lambda| \times D + |\Gamma| + D + 3K) + K)) \approx \Theta(|\Lambda| \times N_C \times D \times C). \tag{14}$$

where C is a constant, D represents latent feature dimension, N_C denotes the particles in DE algorithm. From (14) we know that the size of the known entry set $|\Lambda|$ major can affect the overall time consumption, but it is far less than the size of entity size $|M|$ and $|N|$.

Space complexity

Large-scale data processing requires high storage space in most cases. Therefore the space complexity is another significant indicator of the algorithm. As for the ADMA algorithm, there are latent factor matrices and differential evolution particles that take the most space to store. More specifically, the dimension of the target latent factors P and Q, the dimension of the parameters, and the number of the particles are the primary source of storage costs, which is confirmed as follows:

$$S_{ADMA} = (|M| + |N|) \times D + 3N_C + 2D + 12 \approx (|M| + |N|) \times D + N_C. \tag{15}$$

where D is the latent feature dimension and $|M|$ and $|N|$ are the entity size of users and items. The result in (15) indicates that the storage space of ADMA is far less than that of the original rating matrix.

Experimental results and analysis

General settings

Evaluation metrics

Numerical comparison for predicting missing data in HDI matrices commonly performs via two evaluation measures: mean squared error (RMSE) and mean absolute error (MAE) [6, 38], which are:

$$RMSE = \sqrt{\left(\sum_{z_{m,n} \in \Gamma} \left(z_{m,n} - \hat{z}_{m,n}\right)^2\right) \Big/ |\Gamma|}, MAE = \left(\sum_{z_{m,n} \in \Gamma} \left|z_{m,n} - \hat{z}_{m,n}\right|_{abs}\right) \Big/ |\Gamma|;$$

where $\hat{z}_{m,n}$ is the approximation of $z_{m,n} \in \Gamma$. It is worth noting that from the previous illustration of function (13), the evaluation function is consistent with the RMSE and MAE evidencing the affinity of the evaluation function and metrics.

Experimental design

To illuminate the exploration and exploitation abilities of the proposed method, a standard scheme with a fundamental model SGD and its modified one Moment SGD are compared to show the searching ability. And to describe the adaptive searching efficiency, a general adaptive model ALF, and the standard DE adaptive model DELF

Table 2 Statistics of the studied datasets

No.	Name	#User	#Item	#Interaction	Density*
D1	Douban	129,490	58,541	16,830,839	0.22%
D2	MovieLens 20 M	138,493	26,744	20,000,263	0.54%
D3	Flixter	147,612	48,794	8,196,077	0.11%
D4	Yahoo-R2	200,000	136,736	76,344,627	0.076%
D5	Epinion	120,492	775,760	13,668,320	0.015%

* Density denotes the percentage of observed entries in the dataset

are also compared in this part. Datasets from industrial applications with different density under 0.54% are adopted in our experiments. These sparse data form the target high-dimensional and incomplete matrix to be analyzed.

Datasets

As shown in Table 2, five popular datasets of industrial HDI matrices, including Douban, MovieLlens 20 M (ML20M), Flixter, Yahoo-R2, and Epinion [39–42] are used in this paper. Detailed information is described as follows:

1) D1: Douban. It is composed of 16,830,839 ratings from 58,541 items in a large Chinese website Douban where 129,490 users grade the movies, music, books, etc. The data density of Douban is about 0.22% that subjects to an HDI matrix.
2) D2: MovieLens 20 M (ML20M). There are 26,744 movies in the MovieLens system constituted by GroupLens. Within the ML20M dataset collected in the system, 138,493 users grade the movies 20,000,263 times scaling from 0.5 to 5, whose density is 0.54% and size is 20 M.
3) D3: Flixter. The Flixter commercial website dataset has 8,196,077 ratings ranging from 0.5 to 5 on 48,794 movies graded by 147,612 users. The data density of the dataset is around 0.11%.
4) D4: Yahoo-R2. Yahoo-R2 termed Yahoo is a large dataset that contains 200,000 users who rate 136,736 songs with 76,344,627 ratings ranging from 1 to 5. Its density is 0.28%.
5) D5: Epinion. In the website Trustlet, Epinion dataset is collected in which 120,492 users grade 775,760 articles with 13,668,320 known entries. Notably, the density of Epinion is only 0.015%.

Conventional data assignments are also implemented in this paper that randomly divide the data into ten independent parts on average and allocate the train, validation, and test part into a ratio of 7:1:2.

Moreover, each model on these datasets are conducted ten times. Then, the averages of ten experimental results under the same conditions are analyzed in the end, which can dismiss the possible outliers.

Table 3 Summary of the compared models

Parameters	Models	Descriptions
fixed η and λ ($\lambda_P = \lambda_P$) model	M1	SGD. A fundamental SGD-based LFA model that converges very fast with the resilience for the HDI data analysis
	M2	Moment SGD. For a better update of the current one in the process of the SGD-based LFA model, it incorporates the previous updates into the current one in the way of momentum
	M3	Adam. This adaptive moment estimation SGD method further considers exponentially decaying of square average and past stochastic gradients average
η-λ-($\lambda_P = \lambda_P$) adaptive model	M4	ALF. An adaptive SGD-based LFA model that adopts standard PSO to adjust the learning rate in the model
	M5	DELF. Differential evolution-based LF analysis model employs original differential evolution to conduct an SGD-based LFA model adaptively
η-λ-($\lambda_P \neq \lambda_P$) adaptive model	M6	The proposed ADMA. Compared with DELF, the proposed model ADMA improves the original DE algorithm for a more precise prediction in an SGD-based LFA model with limited time

Model settings

We compare our model with five representative state-of-the-art SGD-based LFA models on HDI datasets are compared in the following part. M1 (SGD) and M2 (Moment SGD) are LFA methods based on traditional SGD. M3 (Adam), M4 (ALF), M5 (DELF) [37, 38], and M6 (ADMA) are methods of parameter adaptive SGD-based LFA models. All model descriptions are illustrated in Table 3

The experimental parameter settings in the aforementioned methods are set in advance as follows:

1) For all models, latent factor features in an LFA model are constructed with a fixed dimension $D = 20$, which can both maximize the representation ability and balance the computing effects.
2) In M1–M4, their hyper-parameters are set as the optimal value of each model, for instance, $\lambda_P = \lambda_Q = 0.03$ in M1-M4; $\eta = 0.003$ in M1 and M2; the adaptive boundary of the learning rate η is from 2^{-12} to 2^{-8} in M3 and M4; For PSO mechanism in M4, model parameters are $c_1 = c_2 = 2$, the number of particles is set as 20, and the velocity boundary is $v_{\max} = 1$, $v_{\min} = \text{-}{-}1$ etc.
3) In M5-M6, the learning rate η and regularization constants λ_P and λ_Q are all self-adaptive. Other parameters used in the adaptive learning process are empirically set as$\gamma = \mu = 0.1$, $P_{select} = 0.7$, $N_C = 20$, $i_{select} = N_C/4$, $x_{\max,1} = 2^{-8}$, $x_{\min,1} = 2^{-12}$, and $x_{\max,2} = x_{\max,3} = 2^{-3}$, $x_{\min,2} = x_{\min,3} = 2^{-7}$.
4) Experimental termination criteria is the best evaluation value unchanged in five rounds or the reach of the maximum iteration number—1000.

Performance comparison

Experiments intuitively show the numerical performance of SGD (M1), Moment SGD (M2), Adam (M3), ALF (M4), DELF (M5), and our method GASL (M6) on industrial datasets Douban (D1), ML20M (D2), Flixter (D3), Yahoo (D4), and Epinion (D5). Table 4 and Table 5 report all their results regarding total iterations, total training time

Table 4 Mean RMSE results

Dataset	Method	RMSE and computational efficiency measures			
		RMSE	Iteration number	Total time (s)	Per time (s)
D1	M1	0.7213 ± 1.849E−04	143.9 ± 7.379E-01	**240.8 ± 1.172E + 01**	1.674 ± 8.591E-02
	M2	0.7363 ± 2.298E−04	234.9 ± 3.542E + 00	793.2 ± 2.143E + 01	3.377 ± 6.292E-02
	M3	0.7252 ± 3.741E−03	291 ± 0.000E + 00	1607 ± 2.048E + 03	5.523 ± 2.048E + 00
	M4	0.7232 ± 1.231E−04	7.000 ± 0.000E + 00	303.7 ± 2.755E + 01	43.39 ± 3.936E + 00
	M5	0.7193 ± 2.558E−03	24.30 ± 1.477E + 01	1114 ± 6.864E + 02	45.64 ± 1.383E + 00
	M6	**0.7168 ± 7.518E−04**	24.30 ± 5.926E + 00	1011 ± 2.474E + 02	41.62 ± 1.101E + 00
D2	M1	0.7807 ± 2.030E−04	120.9 ± 1.287E + 00	**221.9 ± 6.055E + 00**	1.835 ± 4.187E-02
	M2	0.7898 ± 2.547E−04	210.3 ± 2.541E + 00	770.7 ± 1.111E + 01	3.665 ± 2.210E-02
	M3	0.7804 ± 1.781E−04	271 ± 0.000E + 00	1900 ± 1.504E + 03	7.010 ± 1.504E + 00
	M4	0.7826 ± 1.292E−04	6.000 ± 0.000E + 00	320.9 ± 2.892E + 01	53.50 ± 4.820E + 00
	M5	0.7796 ± 1.392E−03	21.10 ± 8.006E + 00	1097 ± 4.187E + 02	51.94 ± 1.242E + 00
	M6	**0.7780 ± 1.286E−03**	18.70 ± 5.618E + 00	912.3 ± 2.920E + 02	48.56 ± 2.521E + 00
D3	M1	0.9506 ± 2.439E−04	202.8 ± 1.751E + 00	**149.7 ± 3.050E + 00**	0.7380 ± 1.325E-02
	M2	0.9467 ± 5.613E−04	260.6 ± 2.066E + 00	389.9 ± 1.308E + 01	1.496 ± 5.253E-02
	M3	0.9581 ± 8.206E−04	466 ± 0.000E + 00	1293 ± 7.030E + 01	2.774 ± 7.030E-02
	M4	0.9523 ± 2.134E−03	10.8 ± 3.615E + 00	222.0 ± 7.531E + 01	20.54 ± 5.005E-01
	M5	0.9447 ± 2.557E−03	35.60 ± 6.207E + 01	827.5 ± 1.426E + 03	23.52 ± 5.334E-01
	M6	**0.9398 ± 1.257E−03**	21.60 ± 2.989E + 00	507.6 ± 6.981E + 01	23.51 ± 2.679E-01
D4	M1	1.062 ± 2.223E−04	65.30 ± 6.749E-01	**765.8 ± 2.323E + 01**	11.73 ± 3.386E-01
	M2	1.071 ± 2.933E−04	227.6 ± 1.350E + 00	6353 ± 4.055E + 01	27.91 ± 1.879E-01
	M3	1.061 ± 3.330E−04	93.60 ± 8.600E + 01	7304 ± 6.493 + 01	78.02 ± 4.600E-01
	M4	1.064 ± 2.617E−04	3.700 ± 4.830E-01	1090 ± 1.985E + 02	292.7 ± 2.031E + 01
	M5	1.062 ± 3.834E−03	8.800 ± 2.741E + 00	2889 ± 9.334E + 02	327.6 ± 4.029E + 00
	M6	**1.060 ± 1.257E−03**	10.10 ± 1.729E + 00	3171 ± 5.463E + 02	313.9 ± 1.996E + 00
D5	M1	0.7357 ± 1.072E−04	1000 ± 0.000E + 00	2171 ± 5.503E + 01	2.171 ± 5.503E-02
	M2	0.7376 ± 1.465E−04	684.8 ± 4.341E + 00	2810 ± 3.148E + 01	4.104 ± 4.440E-02
	M3	0.7364 ± 6.200E−05	438.0 ± 4.150E + 00	2027 ± 8.028E + 01	4.627 ± 2.400E-01
	M4	0.7364 ± 8.703E−05	16.90 ± 3.162E-01	**731.5 ± 3.394E + 01**	43.28 ± 1.701E + 00
	M5	0.7358 ± 2.201E−03	41.40 ± 2.899E + 01	1671 ± 1.336E + 03	39.40 ± 3.203E + 00
	M6	**0.7354 ± 5.410E−04**	47.10 ± 6.574E + 00	1775 ± 3.674E + 02	37.59 ± 5.182E + 00

with variance, and the prediction precision evaluated by RMSE and MAE respectively. Noticeabley, the precision and total training time are in bold in each experiments. Their training processes are compared in Fig. 2 and Fig. 3 with RMSE and MAE respectively. The follwing observations can be made based on these results:

1) Our model ADMA predicts the missing data in an HDI matrix more accurately than both adaptive and non-adaptive LFA models**.** Overall, Table 4 illustrates that the RMSE evaluation results of M6 are more accurately than M1-M5 on all five datasets D1-D5. And from Table 5, the MAE evaluation results of our model M6 are also more accurately than M1-M5 on D1-D4, and M2-M3 on D5. Take the non-adaptive model M2 as an example, the prediction accuracy is 2.648%, 1.494%, 0.7289%, 1.027%, and 0.2983% lower than that of our model on D1-D5 respectively in RMSE results. Andcompared with the adaptive model M3, the MAE prediction errors of our model M6 are 3.959%, 3.661%, 3.464%, 3.019%, and 0.7723% more accurately on

Table 5 MEAN MAE RESULTS

Dataset	Method	MAE and computational efficiency measures			
		MAE	Iteration number	Total time (s)	Per time (s)
D1	M1	0.5609 ± 1.084E-04	151.6 ± 6.992E-01	**274.9 ± 5.715E + 00**	1.813 ± 3.958E-02
	M2	0.5686 ± 2.007E-04	238.5 ± 4.720E + 00	777.6 ± 2.052E + 01	3.260 ± 4.937E-02
	M3	0.5791 ± 1.086E-04	303 ± 0.000E + 00	1938 ± 1.816E + 03	6.397 ± 1.816E + 00
	M4	0.5666 ± 1.038E-04	7.000 ± 0.000E + 00	279.6 ± 9.736E + 00	39.95 ± 1.038E-04
	M5	0.5613 ± 2.672E-03	19.20 ± 4.614E + 00	870.4 ± 2.043E + 02	45.44 ± 1.261E + 00
	M6	**0.5579 ± 1.640E-03**	21.40 ± 5.948E + 00	885.7 ± 2.510E + 02	41.32 ± 1.165E + 00
D2	M1	0.5963 ± 9.869E-05	128.4 ± 8.433E-01	**263.0 ± 6.029E + 00**	2.049 ± 4.572E-02
	M2	0.5998 ± 3.005E-04	225.5 ± 3.567E + 00	799.7 ± 1.455E + 01	3.547 ± 4.325E-02
	M3	0.6128 ± 2.573E-04	76.00 ± 0.000E + 00	500.8 ± 1.434E + 03	6.590 ± 1.434E + 00
	M4	0.6011 ± 4.181E-04	6.000 ± 0.000E + 00	273.6 ± 2.009E + 01	45.60 ± 4.181E-04
	M5	0.5967 ± 4.362E-03	17.10 ± 1.135E + 01	894.0 ± 5.889E + 02	52.39 ± 2.867E + 00
	M6	**0.5943 ± 2.424E-03**	15.20 ± 4.872E + 00	740.6 ± 2.421E + 02	48.64 ± 1.740E + 00
D3	M1	0.6711 ± 1.759E-04	259.0 ± 4.761E + 00	**213.7 ± 1.067E + 01**	0.8250 ± 3.851E-02
	M2	0.6689 ± 3.549E-04	299.5 ± 3.308E + 00	449.2 ± 1.688E + 01	1.500 ± 5.042E-02
	M3	0.6871 ± 5.071E-04	453.0 ± 0.000E + 00	1374 ± 3.067E + 02	3.032 ± 3.067E-01
	M4	0.6733 ± 2.933E-03	13.30 ± 4.473E + 00	279.0 ± 1.068E + 02	20.81 ± 1.273E + 00
	M5	0.6659 ± 4.917E-03	29.30 ± 2.028E + 01	687.8 ± 4.763E + 02	23.40 ± 4.113E-01
	M6	**0.6633 ± 3.314E-03**	19.60 ± 3.534E + 00	464.1 ± 8.557E + 01	23.66 ± 2.061E-01
D4	M1	0.8120 ± 1.606E-04	78.00 ± 4.714E-01	**972.0 ± 3.236E + 01**	12.46 ± 4.360E-01
	M2	0.8114 ± 3.939E-04	275.9 ± 2.331E + 00	7656 ± 1.063E + 02	27.75 ± 2.023E-01
	M3	0.8109 ± 2.370E-04	109.6 ± 1.730E + 00	8863 ± 1.626E + 02	80.84 ± 3.300E-01
	M4	0.8158 ± 6.763E-05	4.000 ± 0.000E + 00	1117 ± 6.113E + 01	279.5 ± 1.528E + 01
	M5	0.8115 ± 2.468E-03	9.200 ± 3.327E + 00	2999 ± 1.109E + 03	325.3 ± 4.388E + 00
	M6	**0.8095 ± 1.308E-03**	11.90 ± 2.558E + 00	3741 ± 7.906E + 02	314.6 ± 2.360E + 00
D5	M1	**0.3482 ± 1.278E-04**	471.00 ± 3.801E + 00	945.4 ± 1.824E + 01	2.008 ± 4.811E-02
	M2	0.3587 ± 1.840E-04	658.9 ± 4.954E + 00	2706 ± 4.296E + 01	4.107 ± 6.820E-02
	M3	0.3663 ± 1.180E-04	103.0 ± 2.630E + 00	**461.7 ± 1.361E + 02**	4.482 ± 3.200E-01
	M4	0.3500 ± 1.716E-04	19.40 ± 5.164E-01	773.9 ± 4.180E + 01	39.87 ± 1.439E + 00
	M5	0.3500 ± 2.653E-03	25.20 ± 1.314E + 01	982.4 ± 4.826E + 02	39.66 ± 3.189E + 00
	M6	0.3518 ± 4.571E-03	29.30 ± 5.100E + 00	1075 ± 2.453E + 02	36.60 ± 4.395E + 00

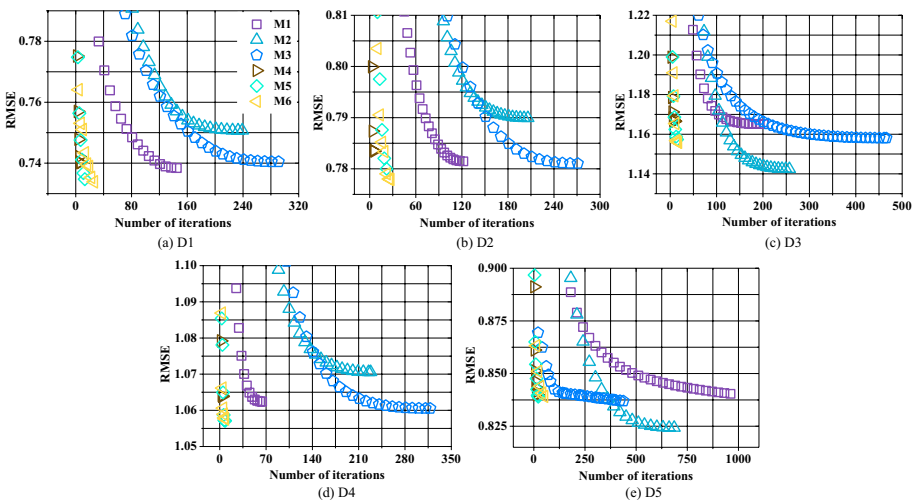

Fig. 2 RMSE Comparison of M1 to M6 on D1 to D5

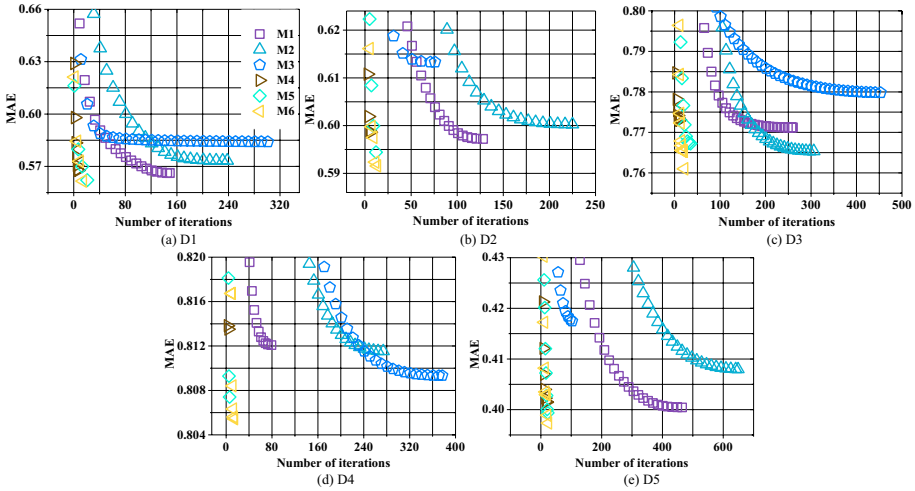

Fig. 3 MAE Comparison of M1 to M6 on D1 to D5

D5, D1, D3, D2, and D4 respectively. Although there is an inferior MAE performace of M6 on D5, its RMSE performace is superior. Resonalble explaination for the precise prediction is that the physical mechanism annealing helps the model overstep its local saddle points, thereby achieving a desirable performance in terms of prediction accuracy.

2) ADMA saves a great deal of computational time without degrading prediction accuracy. From the total training iterations shown in Table 4 and Table 5, the iteration time of ADMA plummets from previous researches, which is neither too much to be trained fully, nor too little to save the computational time. From the average results, the RMSE iteration number of our model M6, as shown in Table 4, is less than 84.53% of M1-M3 on D1-D5 without degrading the performances of prediction accuracy. Similarly, the MAE iteration of our model M6, as shown in Table 5, is less than 80.00% of M1-M3 on D1-D5 except for the iteration time of M3 on D5 that is also less than 71.55% with fast convergence. Although, the iteration of our model M6 is slightly more than that of M4, our model breaks the bottleneck of the premature shortcoming in M4 that makes the model stop early before reaching its optimal value.
3) ADMA is increasing rapidly in convergence with relatively higher computational efficiency. Another advantage of an ADMA is that it converges fastly. From the convergence curves drawn in Fig. 2 and Fig. 3 for RMSE and MAE training process respectively, the yellow triangle convergent curves represents our model M6, which is plummeting with limited number of iterations in a relatively short time on all HDI datasets. On the contrary, M1-M3 colored as purple, light blue, and blue have a slower rate of convergence on D1-D5, where the preset maximun is even reached forcing to stop training untimely and resulting in suboptimal prediction accuracy. The M4 that adapts the learning rate of an LFA model colored as brown in Fig. 2 and Fig. 3 is the fastest converged algorithm. Unfortunately, the accuracy of M4 is lower than our model M6 and other models in a number of cases.

4) Compared with traditional DE algorithm, the improved DE algorithm in ADMA avoids its premature convergence while at the same time requiring less training time. One typical improvement of an algorithm is to avoid premature convergence, thus reducing total training time. Surprisingly, they are both realized in our method M6. From the RMSE results in Table 4, the convergence time of M6 is 16.84% and 38.66% smaller than that of M5 on D2 and D3 respectively, while its prediction error is also 0.5878% and 1.313% on each data. The same situation is also shown in Table 5, the MAE convergence results of M6 is lower 17.16% and 32.52% lower than M5 on D2 and D3 respectively, while its prediction error is lower 1.131% and 1.485% on each data. It shows that the physical mechanism cosine annealing dose help the original DE-based model overstep its local saddle points, and the improved differential evolution (DE) mechanism achieves a desirable prediction efficiency.

Summary

The aforementioned results show that the proposed model ADMA (M6) can balance the exploration and exploitation well, and has better prediction accuracy when solving latent factor analysis (LFA) on HDI matrices. It also helps reduce a great number of iterations without degrading prediction accuracy. Moreover, the convergence of an ADMA is increasing rapidly, thereby maintaining a relatively higher computational efficiency. The model overstep its local optimum effectively with the support of the annealing mechanism. And the improved DE algorithm in ADMA avoids its premature convergence. The resulting ADMA is an efficient parameter self-adaptive HDI data analysis model.

Discussion

On the one hand, ADMA adaptively learn latent factors by the use of DE algorithm firstly. Although the basic offspring random selection operation in mutation mechanism restricts its exploration and exploitation abilities, we innovatively control the internal operation by a periodic equilibrium mechanism thereby improving its search and convergence abilities. Compared with other non-adaptive and adaptive related methods, the proposed method has better properties in terms of predicting the missing data in HDI matrices with reasonable overhead. On the other hand, ADMA estimates parameters basically under the stochastic mechanism. However, the SGD mechanism is easy to fall into the local optimum. Therefore, other learning mechanisms like Newton methods, alternating least squares (ALS), and alternating direction method (ADM) can be employed to dismiss the disadvantage of the mechanism used in ADMA. We plan to explore these alternative mechanisms to further enhance our model ADMA in our future work.

Conclusion

An efficient annealing-assisted differential evolution for multi-parameter adaptive latent factor analysis (ADMA) is proposed in this paper. By incorporating the physical mechanism annealing, it efficiently improves the fundamental differential evolution (DE) mechanism and helps the latent factor analysis (LFA) model analyze high-dimensional

and incomplete (HDI) matrix adaptively. Experiments on five popular industrial HDI datasets show that our proposed model ADMA has better performance in terms of prediction accuracy and convergence speed. Integrating the annealing mechanism, ADMA assists the model overstep its local optimum and the improved DE algorithm effectively, which is increasing rapidly in convergence and reduces a great number of iterations without degrading prediction accuracy. Although on some datasets, ADMA has relatively longer training time, it substantially increases the prediction accuracy without requiring the extremely costly multi-parameter computing. As a result, ADMA outperforms the state-of-the-art in terms of predicting missing data in HDI matrices in real industrial applications.

In the future, there are at least three research directions worth studying. First, with time perspective, this method focus on a periodic equilibrium mechanism. To further improve particles searching accuracy, space related accelerate mechanisms can also be analyzed and incorporated into the method. Second, as regard to adaptive learning efficiency, we employ DE algorithm as the main update strategy. Meanwhile, other state of the art algorithms that have different advantages is worth studying according to different tasks. And last, but certainly not least, fundamental stochastic gradient descent mechanism, generally used in the HDI data analysis methods, could also be improved, and then strengthen the big data analysis field.

Acknowledgements
We thank Chongqing University of Posts and Telecommunications, Singapore Management University, and Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology for financing this study. Then we thank all the staff members of Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology. Finally, I want to thank my parents and friends who always accompany and support me.

Author contributions
QL: Investigation, conceptualization, visualization, formal analysis, methodology, software, validation, writing- original draft preparation, writing-reviewing and editing. GP: writing-reviewing and editing, supervision. MS: resources, data curation, supervision, funding acquisition. All authors read and approved the final manuscript.

Funding
This research is supported in part by grants from the Key Cooperation Project of Chongqing Municipal Education Commission (HZ2021017) and nsfc 62072429. The collaborative fund is from Chongqing University of Posts and Telecommunications, and Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology.

Availability of data and materials
The dataset has no restrictions that all the data can be acquired in the related site.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 7 December 2021 Accepted: 27 June 2022
Published online: 16 July 2022

References

1. Thudumu S, Branch P, Jin J, Singh JJ. A comprehensive survey of anomaly detection techniques for high dimensional big data. J Big Data. 2020;7(1):1–30.

2. Angskun J, Tipprasert S, Angskun T. Big data analytics on social networks for real-time depression detection. J Big Data. 2022;9(1):1–15.
3. Tarus JK, Niu Z, Mustafa G. Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. Artif Intell Rev. 2018;50(1):21–48.
4. Peng S, Wang G, Xie D. Social influence analysis in social networking big data: opportunities and challenges. IEEE Network. 2016;31(1):11–7.
5. Giatrakos N, Alevizos E, Artikis A, Deligiannakis A, Garofalakis M. Complex event recognition in the big data era: a survey. VLDB J. 2020;29(1):313–52.
6. Luo X, Liu H, Gou G, Xia Y, Zhu Q. A parallel matrix factorization based recommender by alternating stochastic gradient decent. Eng Appl Artif Intell. 2012;25(7):1403–12.
7. Bisot V, Serizel R, Essid S, Richard G. Feature learning with matrix factorization applied to acoustic scene classification. IEEE/ACM Trans Audio Speech Lang Process. 2017;25(6):1216–29.
8. Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng. 2005;17(6):734–49.
9. Y. Chen, X. Hu and Y. Hui. Correlation filter network model performance analysis. 2020 International Conference on Computer Network, Electronic and Automation (ICCNEA), 2020. P. 254–258.
10. Song Y, Li M, Luo X, Yang G, Wang C. Improved symmetric and nonnegative matrix factorization models for undirected, sparse and large-scaled networks: a triple factorization-based approach. IEEE Trans Industr Inform. 2020;16(5):3006–17.
11. Luo X, Zhou M, Li S, Wu D, Liu Z, Shang M. Algorithms of unconstrained non-negative latent factor analysis for recommender systems. IEEE Trans Big Data. 2021;7(1):227–40.
12. H. Bong, Z. Liu, Z. M. A. Ren, Smith, V. Ventura, and R. E. Kass. Latent dynamic factor analysis of high-dimensional neural recordings. In: NeurIPS, 2020.
13. Li Q, Xiong DW, Shang MS. Adjusted stochastic gradient descent for latent factor analysis. Inf Sci. 2022;588:196–213.
14. Wu D, Luo X, Shang M, He Y, Wang G, Zhou M. A deep latent factor model for high-dimensional and sparse matrices in recommender systems. IEEE Trans Syst Man Cybern Syst. 2021;51(7):4285–96.
15. Price KV. Differential evolution. Handbook of optimization. Heidelberg: Springer; 2013. p. 187–214.
16. Dorigo M, Birattari M, Stutzle T. Ant colony optimization. IEEE Comput Intell Mag. 2006;1(4):28–39.
17. Poli R, Kennedy J, Blackwell T. Particle swarm optimization. Swarm Intell. 2007;1(1):33–57.
18. DT Pham, A Ghanbarzadeh, E Koç, S Otri, S Rahim, and M Zaidi. The bees algorithm—a novel tool for complex optimization problems. Intelligent production machines and systems. Amsterdam: Elsevier Science Ltd. 2006.
19. Shi YH, Brain storm optimization algorithm in objective space. IEEE Congress on Evolutionalry Computation (CEC). Sendai. 2015;2015:1227–34.
20. Choi K, Jang D, Kang S, Lee J, Chung T, Kim H. Hybrid algorithm combing genetic algorithm with evolution strategy for antenna design. IEEE Trans Magn. 2016;52(3):1–4.
21. Das S, Suganthan PN. Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput. 2011;15(1):4–31.
22. Zhou Z, Abawajy J, Shojafar M, Chowdhury M. DEHM: an improved differential evolution algorithm using hierarchical multistrategy in a cybertwin 6G network. IEEE Trans Industr Inform. 2022;18(7):4944–53.
23. Baldassi C. Recombinator-k-means: an evolutionary algorithm that exploits k-means++ for recombination. IEEE Trans Evol Comput. 2022. https://doi.org/10.1109/TEVC.2022.3144134.
24. Deng W, Shang S, Cai X, Zhao H, Song Y, Xu J. An improved differential evolution algorithm and its application in optimization problem. Soft Comput. 2021;25(7):5277–98.
25. Zhang X, Ge Z. Local parameter optimization of LSSVM for industrial soft sensing with big data and cloud implementation. IEEE Trans Industr Inform. 2020;16(5):2917–28.
26. Zhang Q, Yang LT, Chen Z, Li P, Bu F. An adaptive dropout deep computation model for industrial IoT big data learning with crowdsourcing to cloud computing. IEEE Trans Industr Inform. 2019;15(4):2330–7.
27. Fikri N, Rida M, Abghour N, Moussaid K, El Omri A. An adaptive and real-time based architecture for financial data integration. J Big Data. 2019;6(1):1–25.
28. Taheri S, Goudarzi M, Yoshie O. Learning-based power prediction for geo-distributed Data Centers: weather parameter analysis. J Big Data. 2020;7(1):1–16.
29. Miao Y, Zhao M, Lin J. Identification of mechanical compound-fault based on the improved parameter-adaptive variational mode decomposition. ISA Trans. 2019;84:82–95.
30. Xiao X, Liu B, Warnell G, Fink J, Stone P. Appld: Adaptive planner parameter learning from demonstration. IEEE Robot Autom Lett. 2020;5(3):4541–7.
31. El-Naggar KM, AlRashidi MR, AlHajri MF, Al-Othman AK. Simulated annealing algorithm for photovoltaic parameters identification. Sol Energy. 2012;86(1):266–74.
32. Na J, Xing Y, Costa-Castelló R. Adaptive estimation of time-varying parameters with application to roto-magnet plant. IEEE Trans Syst Man Cybernetics Syst. 2021;51(2):731–41.
33. Yang C, Jiang Y, He W, Na J, Li Z, Xu B. Adaptive parameter estimation and control design for robot manipulators with finite-time convergence. IEEE Trans Industr Elect. 2018;65(10):8112–23.
34. Kapetina MN, Rapaić MR, Pisano A, Jeličić ZD. Adaptive parameter estimation in LTI systems. IEEE Trans Automatic Control. 2019;64(10):4188–95.
35. Kumar A, Kumar A. Adaptive management of multimodal biometrics fusion using ant colony optimization. Inform Fusion. 2016;32:49–63.
36. Luo X, Yuan Y, Chen S, Zeng N, Wang Z. Position-transitional particle swarm optimization-incorporated latent factor analysis. IEEE Trans Knowled Data Eng. 2020. https://doi.org/10.1109/TKDE.2020.3033324.
37. Li Q, Shang M. BALFA: A brain storm optimization-based adaptive latent factor analysis model. Inf Sci. 2021;578:913–29.

38. S. -L Chen, Y. Yuan, and J. Wang. An adaptive latent factor model via particle swarm optimization for high-dimensional and sparse matrices. 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), 2019. p. 1738–1743.
39. M. Shang, Y. Yuan, X. Luo and M. Zhou. An α -β -Divergence-Generalized Recommender for Highly Accurate Predictions of Missing User Preferences. In: IEEE Transactions on Cybernetics, 2021.
40. Luo X, Qin W, Dong A, Sedraoui K, Zhou M. Efficient and high-quality recommendations via momentum-incorporated parallel stochastic gradient descent-based learning". IEEE/CAA J Automatica Sinica. 2020;8(2):403–11.
41. Takács G, Pilászy I, Németh B, Tikky D. Scalable collaborative filtering approaches for large recommender systems. J Mach Learn Res. 2009;10:623–56.
42. Koren Y, Bell R, Volinsky C. Matrix-factorization techniques for recommender systems. IEEE Comput. 2009;42(8):30–7.

Publisher's Note