Neural Comput & Applic
DOI 10.1007/s00521-015-1903-2

ORIGINAL ARTICLE

An integrated chaotic time series prediction model based on efficient extreme learning machine and differential evolution

Wei Guo[1,4] · **Tao Xu**[1,2,3] · **Zonglei Lu**[2,3]

Received: 24 June 2014 / Accepted: 31 March 2015

Abstract In this paper, an integrated model based on efficient extreme learning machine (EELM) and differential evolution (DE) is proposed to predict chaotic time series. In the proposed model, a novel learning algorithm called EELM is presented and used to model the chaotic time series. The EELM inherits the basic idea of extreme learning machine (ELM) in training single hidden layer feedforward networks, but replaces the commonly used singular value decomposition with a reduced complete orthogonal decomposition to calculate the output weights, which can achieve a much faster learning speed than ELM. Moreover, in order to obtain a more accurate and more stable prediction performance for chaotic time series prediction, this model abandons the traditional two-stage modeling approach and adopts an integrated parameter selection strategy which employs a modified DE algorithm to optimize the phase space reconstruction parameters of chaotic time series and the model parameter of EELM simultaneously based on a hybrid validation criterion. Experimental results show that the proposed integrated prediction model can not only provide stable prediction performances with high efficiency but also achieve much more accurate prediction results than its counterparts for chaotic time series prediction.

Keywords Chaotic time series prediction · Efficient extreme learning machine · Differential evolution · Reduced complete orthogonal decomposition · Integrated parameter selection

1 Introduction

Chaotic time series prediction has been an important and challenging issue over the past several decades. Prediction of chaotic time series is a useful method to evaluate the characteristics of dynamical systems and forecast the trend of complex systems. In particular, in recent years, with the development of chaos theory, chaotic time series prediction has been widely applied in the fields of foreign exchange rate prediction [1], wind power prediction [2], traffic flow prediction [3], river basin flow prediction [4], hydrological prediction [5], Sunspot prediction [6] and many others.

Due to the broad applications of chaotic time series, many methods have been proposed to predict chaotic time series. In the early time, classical methods based on mathematical and statistical models such as autoregressive integrated moving average (ARIMA), autoregressive conditional heteroskedasticity (ARCH) were widely used in chaotic time series prediction [7]. For the past few years, there has been a growing interest in employing computational intelligence techniques to predict chaotic time series, such as artificial neural networks (ANNs) [8–16], support vector machine (SVM) [17–19] and evolutionary computation [20]. Out of numerous computational intelligence techniques, ANNs have been playing the dominant roles in

✉ Wei Guo
weiguo031@163.com

1 School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

2 Information Technology Research Base of Civil Aviation Administration of China, Civil Aviation University of China, Tianjin 300300, China

3 School of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China

4 School of Information Science and Technology, Yancheng Teachers University, Yancheng 224002, China

Published online: 11 April 2015

the field of chaotic time series prediction owing to their powerful learning ability and good generalization capability. The well-known ANNs-based predictors can be listed as radial basis function neural networks [14], recurrent neural networks [8, 12], fuzzy neural networks [9, 11], wavelet neural networks [16] and many varieties of them [10, 13, 15]. Nevertheless, as a result of using gradient-based learning algorithms, these predictors also suffer from some apparent drawbacks and limitations, including: (1) slow learning speed, (2) local minima, (3) overfitting, (4) trivial human intervention such as learning rate, learning epochs and stopping criteria.

Recently, an emerging learning algorithm for single hidden layer feedforward networks (SLFNs) named extreme learning machine (ELM) was proposed by Huang et al. [21]. Distinguished from the conventional learning theory, ELM randomly generates the parameters of hidden nodes, and then the output weights are analytically determined by solving a least squares problem. Compared with ANNs and other classical computational intelligence techniques, ELM not only provides better generalization performance at a much faster learning speed but also overcomes the above-mentioned drawbacks and limitations faced by traditional popular gradient-based learning algorithms. Furthermore, the model structure of ELM is quite simple, which is entirely determined just by one parameter (the number of hidden nodes). Owing to these advantages, ELM has been successfully applied in regression problems including many real-world prediction applications, such as sales prediction of fashion retailing [22] and melting points prediction of organic compounds [23]. Chaotic time series prediction can be viewed as a special kind of regression problem in the field of time series analysis, so ELM should be promising to produce good prediction performances in chaotic time series prediction. In this paper, ELM is introduced and employed as the basic model to predict chaotic time series.

The execution efficiency of the prediction model is a very important issue that we should consider in chaotic time series prediction. Although ELM dramatically reduces the training time by avoiding iterative and descent steps, in fact, the learning speed of ELM can be further improved. In ELM, the main computational overhead is the calculation of output weights which is determined by computing a least squares solution of a general linear system. Ordinary ELM solves this problem by calculating the Moore–Penrose generalized inverse using singular value decomposition (SVD). Although SVD is a versatile and numerically stable method, the obvious drawback is that it is computationally expensive, especially when the number of hidden nodes is large, the computational complexity will significantly rise. The large computation complexity of SVD is the critical factor limiting the learning speed of ELM. Thus, seeking a lightweight and reliable alternative to SVD should be promising to improve the computation efficiency of ELM. Complete orthogonal decomposition (COD) [24] is an alternative approach to solve the linear least squares problem that is cheaper than SVD but which provides the same computation reliability and stability. To further improve the learning speed of ELM, we present a reduced complete orthogonal decomposition (RCOD) algorithm to calculate the output weights of ELM, and then an efficient extreme learning machine (EELM) is proposed. Compared with ELM, EELM uses RCOD instead of SVD to calculate the output weights in a cheaper way, which can further reduce the learning time and obtain higher efficiency.

In the process of chaotic time series prediction, there are two key steps: One is to estimate the reconstruction parameters (embedding dimension m and time delay τ) for reconstructing chaotic time series into phase space, the other is to select the optimal model parameters (the number of hidden nodes for EELM) for modeling and forecasting the reconstructed phase points. These two steps are both crucial to the final prediction results. Zhang et al. [25] has further indicated that the reconstruction parameters and model parameters are interrelated and interdependent, which affect the prediction results all together. However, as far as we know, most of the previous studies carried out the above two steps sequentially, to be more specific, they sought the optimal reconstruction parameters and model parameters using phase space reconstruction technologies and model selection methods, respectively. These traditional prediction models separated the organic relationship between reconstruction parameters and model parameters; even if the two were optimal in their respective steps, it could not guarantee that the combination of them was global optimum. To obtain a more accurate prediction result, the reconstruction parameters and model parameters should be optimized simultaneously. In addition, according to the classical embedding theory [26], phase space reconstruction plays a vital role in uncovering the intrinsic characteristics and development laws of the original chaotic system, and which is completely determined by the embedding dimension and time delay, but so far the common ways [27–31] for computing these two parameters are not only complex but also short of a universally accepted standard, different methods may obtain different reconstruction results for the same chaotic time series. In particular, for some real-world chaotic time series with noise, traditional phase space reconstruction methods are difficult to obtain an exact reconstruction result. That is, the accuracy and stability of traditional prediction models are greatly limited by the quality of phase space reconstruction.

In this paper, we discard the traditional two-stage modeling approach and propose an integrated chaotic time series prediction model based on EELM and differential

evolution (DE). In the proposed model, the EELM is used as an efficient and robust learning algorithm to model chaotic time series; meanwhile, a modified DE algorithm is employed to search for the optimal parameter combination of embedding dimension, time delay of chaotic time series and the hidden nodes number of EELM simultaneously based on a hybrid validation criterion. Taking advantage of the powerful global search ability of DE, the proposed model not only guarantees that the selected parameter combination is global optimum or approximate optimum which enhances the prediction performances of the model, but also tactfully bypasses the problem that traditional prediction models depend too much on the phase space reconstruction.

The rest of the paper is organized as follows. A background on chaotic time series and phase space reconstruction is described in Sect. 2. Section 3 presents the proposed EELM algorithm with RCOD. Section 4 presents the proposed integrated chaotic time series prediction model based on EELM and DE. In Sect. 5, the prediction performances of the proposed model on three well-known chaotic time series are studied. Finally, the conclusions and future work are given in Sect. 6.

2 Chaotic time series and phase space reconstruction

Chaotic time series is a chronological sequence of data points which are generated by chaotic systems and measured on a particular variable at successive times. Chaotic time series prediction involves forecasting the system behavior in future based on currently observed values. Takens embedding theorem [26] provides the theoretical foundation for analysis and prediction of chaotic time series. According to Takens embedding theorem, chaotic time series illustrate deterministic characteristics in the reconstructed phase space though they exhibit the behavior of corresponding chaotic systems as random. That is, we can analyze chaotic systems and extract the intrinsic hidden information from apparently random time series.

Given an observed chaotic time series $\{x(t)|\ t = 1,2,\ldots,N\}$, an embedded phase space can be generated in the following form:

$$\begin{aligned} \boldsymbol{X}(t) &= \{x(t), x(t-\tau), \ldots, x(t-(m-1)\tau)\}, \\ & t = (m-1)\tau + 1, (m-1)\tau + 2, \ldots, N, \end{aligned} \tag{1}$$

where m is the embedding dimension, τ is the time delay. If the embedding dimension is large enough, the phase space is homeomorphic to the state space that generates the chaotic time series, in other words, the phase space contains the same information as the original state space. Further, Takens has proved that there exists a smooth map $F : \mathbf{R}^m \rightarrow \mathbf{R}^m$, such that

$$\boldsymbol{X}(t+h) = F(\boldsymbol{X}(t)) \tag{2}$$

where F is called a reconstruction function.

From Eq. (2), we are easy to deduce an equivalent map $f : \mathbf{R}^m \rightarrow \mathrm{R}$, which is

$$x(t+h) = f(x(t), x(t-\tau), \ldots, x(t-(m-1)\tau)) \tag{3}$$

where f is served as a prediction model for chaotic time series prediction, and $h > 0$ is the step of forward prediction. In this paper, we are mainly focus on single-step prediction, that is to say, $h = 1$.

3 Proposed efficient extreme learning machine

In this section, we first provide a brief review of the ELM, next the RCOD method is proposed and used as an alternative to SVD to calculate the output weights of ELM, and then an efficient ELM is proposed.

3.1 Extreme learning machine

ELM [21] is originally developed from the study of SLFNs. For N arbitrary distinct samples $(\boldsymbol{x}_j, \boldsymbol{t}_j) \in \mathbf{R}^d \times \mathbf{R}^m$, SLFNs with n hidden nodes are mathematically modeled as

$$\sum_{i=1}^{n} \boldsymbol{\beta}_i g_i(\boldsymbol{x}_j) = \sum_{i=1}^{n} \boldsymbol{\beta}_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}_j), \quad j = 1,2,\ldots,N, \tag{4}$$

where $\boldsymbol{a}_i$ is the weight vector connecting the ith hidden node and the input nodes, b_i is the threshold of the ith hidden node, and $\boldsymbol{\beta}_i$ is the weight vector connecting the ith hidden node and the output nodes, $g_i(\boldsymbol{x}_j) = G(\boldsymbol{a}_i, b_i, \boldsymbol{x}_j)$ denotes the output function of the ith hidden node.

That SLFNs can approximate these N samples with zero error means that there exist $(\boldsymbol{a}_i, b_i)$ and $\boldsymbol{\beta}_i$ such that

$$\sum_{i=1}^{n} \boldsymbol{\beta}_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}_j) = \boldsymbol{t}_j, \quad j = 1,2,\ldots,N. \tag{5}$$

The above N equations can be written compactly as:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{6}$$

where

$$\mathbf{H} = \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{h}(\boldsymbol{x}_N) \end{bmatrix} = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \ldots & G(\boldsymbol{a}_n, b_n, \boldsymbol{x}_1) \\ \vdots & \ldots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_N) & \ldots & G(\boldsymbol{a}_n, b_n, \boldsymbol{x}_N) \end{bmatrix}_{N \times n}, \tag{7}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\beta}_n^{\mathrm{T}} \end{bmatrix}_{n\times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \boldsymbol{t}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{t}_N^{\mathrm{T}} \end{bmatrix}_{N\times m}. \tag{8}$$

$\mathbf{H}$ is called the hidden layer output matrix of the network; the jth row of $\mathbf{H}$ is the output vector of the hidden layer with respect to input $\boldsymbol{x}_j$; and the jth column of $\mathbf{H}$ is the jth hidden node's output vector with respect to inputs $\boldsymbol{x}_1$, $\boldsymbol{x}_2,\ldots,\boldsymbol{x}_N$.

As rigorously proven in [21], SLFNs with random hidden nodes have the universal approximation capability, and the hidden nodes can be randomly generated independent of the training data and remain fixed, then the hidden layer output matrix $\mathbf{H}$ is a constant matrix. Thus, training an SLFN is simply equivalent to finding a least squares solution $\hat{\boldsymbol{\beta}}$ of the linear system $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$:

$$\left\|\mathbf{H}\hat{\boldsymbol{\beta}} - \mathbf{T}\right\| = \min_{\beta}\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|. \tag{9}$$

In most cases, the number of hidden nodes is much less than the number of distinct training samples, $n \ll N$, $\mathbf{H}$ is a nonsquare matrix, and there may not exist an exact solution such that $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$. In general, the minimum norm least squares solution of the above linear system is adopted:

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T} \tag{10}$$

where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse of matrix $\mathbf{H}$. Several methods can be used to calculate the Moore–Penrose generalized inverse. In view of the universality, SVD has been widely used in ELM and its varieties.

3.2 Reduced complete orthogonal decomposition

As described in Sect. 3.1, the core of the ELM algorithm is to find a least squares solution of the linear system $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$. Ordinary ELM uses the versatile but expensive SVD method to compute the Moore–Penrose generalized inverse of $\mathbf{H}$ and obtains a minimum norm least squares solution $\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$ as output weights. In this study, a new efficient decomposition method called RCOD is proposed to solve this problem instead of using SVD.

We first give the definition of COD [32]. All the following definitions and formulas are pruned according to the actual computing environment of ELM, and all the present notations are tried to be consistent with that of ELM.

Definition 1 Let $\mathbf{H} \in \mathbf{R}^{N\times n}(N \geq n)$ be a matrix with rank $r \leq n$. A complete orthogonal decomposition (COD) of $\mathbf{H}$ is given by

$$\mathbf{H} = UCV^{\mathrm{T}} = U\begin{bmatrix} C_{11} & 0 \\ 0 & 0 \end{bmatrix}_{N\times n} V^{\mathrm{T}} \tag{11}$$

where $U \in \mathbf{R}^{N\times N}$, $V \in \mathbf{R}^{n\times n}$ are orthogonal matrices, and $C_{11} \in \mathbf{R}^{r\times r}$ is nonsingular.

Proposition 1 *Let $\mathbf{H} \in \mathbf{R}^{N\times n}(N \geq n)$ with rank $r \leq n$ have the COD given by Eq. (11). Let the orthogonal matrices U and V be partitioned into*

$$U = [\,U_1 \quad U_2\,] \quad \text{and} \quad V = [\,V_1 \quad V_2\,], \tag{12}$$

where U_1 and V_1 have r columns, then a COD of $\mathbf{H}$ can also be written in a short form

$$\mathbf{H} = U_1 C_{11} V_1^{\mathrm{T}}. \tag{13}$$

Proof Substituting U and V in Eq. (12) into Eq. (11), we have that

$$\begin{aligned} \mathrm{H} &= UCV^{\mathrm{T}} \\ &= [\,U_1 \quad U_2\,]\begin{bmatrix} C_{11} & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} V_1^{\mathrm{T}} \\ V_2^{\mathrm{T}} \end{bmatrix} \\ &= [\,U_1 C_{11} \quad 0\,]\begin{bmatrix} V_1^{\mathrm{T}} \\ V_2^{\mathrm{T}} \end{bmatrix} \\ &= U_1 C_{11} V_1^{\mathrm{T}}. \end{aligned}$$

Since Eq. (13) is an elegant presentation of COD, we call it reduced complete orthogonal decomposition (RCOD).

Notice that the SVD can be viewed as a special example of the COD that has the similar structure of Eq. (11) in which the middle matrix C_{11} is not only nonsingular but also diagonal, and this is what makes the SVD method computationally expensive. In many circumstances, one can sacrifice the diagonal structure and omit the redundancy computation for a more efficient decomposition method, and this is the main idea behind the RCOD. Next, a two-step QR process [24, 32] is used to implement the RCOD, which is described below.

Step 1 The matrix $\mathbf{H}$ is decomposed by Householder QR factorization with column pivoting as

$$\mathbf{H}\pi = UR = U\begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}_{N\times n} \tag{14}$$

where $U \in \mathbf{R}^{N\times N}$ is an orthogonal matrix, $R \in \mathbf{R}^{N\times n}$ is an upper triangular matrix, $\pi \in \mathbf{R}^{n\times n}$ is a permutation matrix, $R_{11} \in \mathbf{R}^{r\times r}$ is an upper triangular and nonsingular matrix, $R_{12} \in \mathbf{R}^{r\times n-r}$ is an $r \times (n-r)$ matrix, $R_{22} \in \mathbf{R}^{N-r\times n-r}$ is an $(N-r)\times(n-r)$ matrix, $r \leq n \leq N$. During the factorization process, the column pivoting strategy attempts to keep R_{11} as well conditional as possible and R_{22} as small as possible. According to [32], QR with column pivoting

discovers rank deficiency if R_{22} is suitably small in norm for some $r < n$; in practice, it is almost always the case that R_{22} is small if $\mathbf{H}$ has rank r. In addition, the column pivoting is chosen and interchanged so that the diagonal elements of the upper triangular matrix R and R_{11} are decreasing. By means of the above two conclusions, it is reasonable to estimate the numerical rank of $\mathbf{H}$ by truncating the R_{22} under a small threshold. Suppose the estimated numerical rank of $\mathbf{H}$ is r, then Eq. (14) can be further written as

$$\begin{aligned}\mathbf{H}\pi = UR &= \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}_{N\times n} \\ &= U_1 \begin{bmatrix} R_{11} & R_{12} \end{bmatrix}_{r\times n} \end{aligned} \tag{15}$$

where $U_1 \in \mathbf{R}^{N\times r}$ is an orthogonal matrix composed of the first r columns of U, $R2 = \begin{bmatrix} R_{11} & R_{12} \end{bmatrix}^{\mathrm{T}} \in \mathbf{R}^{n\times r}$ is a lower trapezoidal matrix.

Step 2 Apply a skinny QR factorization (without pivoting) to $R2$ to get

$$R2 = \begin{bmatrix} R_{11}^{\mathrm{T}} \\ R_{12}^{\mathrm{T}} \end{bmatrix} = QS, \tag{16}$$

where $Q \in \mathbf{R}^{n\times r}$ is an orthogonal matrix, and $S \in \mathbf{R}^{r\times r}$ is an upper triangular matrix.

Based on the above two QR process, the RCOD can be obtained after some transformations. From Eq. (16), we can get

$$\begin{bmatrix} R_{11} & R_{12} \end{bmatrix} = (QS)^{\mathrm{T}} = S^{\mathrm{T}}Q^{\mathrm{T}}. \tag{17}$$

Substitute Eq. (17) into Eq. (15), then

$$\mathbf{H}\pi = U_1 \begin{bmatrix} R_{11} & R_{12} \end{bmatrix}_{r\times n} = U_1 S^{\mathrm{T}} Q^{\mathrm{T}}. \tag{18}$$

Recall that the permutation matrix π is an orthogonal matrix satisfying $\pi * \pi^{\mathrm{T}} = I$. Therefore,

$$\mathbf{H} = U_1 S^{\mathrm{T}} Q^{\mathrm{T}} \pi^{\mathrm{T}} = U_1 S^{\mathrm{T}} (\pi Q)^{\mathrm{T}}, \tag{19}$$

where $U_1 \in \mathbf{R}^{N\times r}$ and $\pi Q \in \mathbf{R}^{n\times r}$ are both orthogonal matrices, and $S^{\mathrm{T}} \in \mathbf{R}^{r\times r}$ is nonsingular. If we set $V_1 = \pi Q$, then Eq. (13) is realized with $C_{11} = S^{\mathrm{T}}$, and the RCOD of $\mathbf{H}$ can be written as $\mathbf{H} = U_1 C_{11} V_1^{\mathrm{T}}$where

$$\begin{aligned} U_1 &= U(:, 1:r), \\ C_{11} &= S^T, \\ V_1 &= \pi Q. \end{aligned} \tag{20}$$

Combining all of the above steps, we obtain the following RCOD algorithm:

Algorithm RCOD: RCOD (H)
Input: H
Output: U_1, C_{11}, V_1.

1. $[N, n] = size\ (\mathrm{H})$;
2. $[U, R, P] = qr\ (\mathrm{H}, 0)$; //Eq. (14)
3. $P = sparse\ (P, 1{:}n, 1)$;
4. $d = abs\ (diag\ (R))$;
5. $\mathrm{tol} = max(N, n) * eps\ (max\ (d))$;
6. $r = sum\ (d > tol)$; //Estimate numerical rank
7. $R2 = R(1{:}r, :)^{\mathrm{T}}$;
8. $[Q, S] = qr\ (R2, 0)$; //Eq. (16)
9. $U_1 = \mathrm{U}\ (:, 1{:}r)$;
10. $C_{11} = S^{\mathrm{T}}$; //Eq. (20)
11. $V_1 = P * Q$;

Remark 1 The RCOD has the following excellent properties:

1. High efficiency. It has established that COD is much cheaper than SVD in computation complexity. As a lightweight implementation of COD, RCOD tends to be more efficient and achieve better performance than SVD. The theoretical and experimental comparisons of computation complexity between RCOD and SVD can be found in Sects. 3.4, 5.2 (the Time column in Table 2), respectively.
2. High stability. As discussed in [24], the QR factorization with column pivoting in the first step generates a matrix $R2$ that is well conditioned up to a scaling of the rows or the columns. Using this, we are able to show that the result of the second step is also well conditioned. Thus, from the implementation process, we have reasons to believe that the proposed RCOD is accurate and stable.
3. High reliability. The RCOD is always robust, no matter the matrix $\mathbf{H}$ has full column rank in which case $r = n$ or $\mathbf{H}$ is rank deficient ($r < n$), which means that the RCOD is also a versatile decomposition method like SVD.

3.3 Proposed efficient extreme learning machine

In view of the excellent properties of the RCOD, we use RCOD instead of SVD to calculate the output weights of ELM, and then a more efficient ELM algorithm is pro-

posed. We first give the theoretical foundation by the following Theorem 1.

Theorem 1 *Let* $\mathbf{H} \in \mathbf{R}^{N\times n}$, $\boldsymbol{\beta} \in \mathbf{R}^{n\times m}$ *and* $\mathbf{T} \in \mathbf{R}^{N\times m}$ *be as in Eqs.* (7) *and* (8), *and assume that* $\mathbf{H}$ *has a RCOD given by Eqs.* (12) *and* (13). *Then*

1. *All least squares solutions* $\boldsymbol{\beta}$ *of Eq.* (9) *are given by*
$$\boldsymbol{\beta} = V_1 C_{11}^{-1} U_1^{\mathrm{T}} \mathbf{T} + V_2 z, \tag{21}$$
where $z \in \mathbf{R}^{(n-r)\times m}$ *is an arbitrary matrix.*
2. *The solution* $\hat{\boldsymbol{\beta}}$ *has minimum norm* $\|\hat{\boldsymbol{\beta}}\|_2$ *precisely when* $z = 0$, *in which case*
$$\hat{\boldsymbol{\beta}} = V_1 C_{11}^{-1} U_1^{\mathrm{T}} \mathbf{T}. \tag{22}$$

Proof Notice that U is orthogonal. According to the definition and properties of orthogonal matrix, we have

$$\begin{aligned}\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_2^2 &= \left\|U^{\mathrm{T}}(\mathbf{H}\boldsymbol{\beta} - \mathbf{T})\right\|_2^2 \\ &= \left\|\begin{bmatrix} U_1^{\mathrm{T}} \\ U_2^{\mathrm{T}} \end{bmatrix}\left(U_1 C_{11} V_1^{\mathrm{T}}\boldsymbol{\beta} - \mathbf{T}\right)\right\|_2^2 \\ &= \left\|\begin{bmatrix} C_{11} V_1^{\mathrm{T}}\boldsymbol{\beta} - U_1^{\mathrm{T}}\mathbf{T} \\ -U_2^{\mathrm{T}}\mathbf{T} \end{bmatrix}\right\|_2^2 \\ &= \left\|C_{11} V_1^{\mathrm{T}}\boldsymbol{\beta} - U_1^{\mathrm{T}}\mathbf{T}\right\|_2^2 + \left\|U_2^{\mathrm{T}}\mathbf{T}\right\|_2^2.\end{aligned}$$

1. $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_2$ is minimized when $C_{11} V_1^{\mathrm{T}}\boldsymbol{\beta} = U_1^{\mathrm{T}}\mathbf{T}$. Recall that C_{11} is nonsingular, V_1 is orthogonal and $V_1^{\mathrm{T}} V_2 z = 0$ for all z, thus all least squares solutions can be written as $\boldsymbol{\beta} = V_1 C_{11}^{-1} U_1^{\mathrm{T}}\mathbf{T} + V_2 z$. In particular, if $r = n$, that is, if $\mathbf{H}$ has full column rank, there is no matrix V_2 and the least squares solution is unique.
2. Since the columns of V_1 and V_2 are mutually orthogonal, applying orthogonal equivalence of the two norm to $\boldsymbol{\beta}$ yields $\|\boldsymbol{\beta}\|_2^2 = \left\|V_1 C_{11}^{-1} U_1^{\mathrm{T}}\mathbf{T}\right\|_2^2 + \|V_2 z\|_2^2$, and this is minimized by choosing $z = 0$. Therefore, the unique minimum norm least squares solution is given by $\hat{\boldsymbol{\beta}} = V_1 C_{11}^{-1} U_1^{\mathrm{T}}\mathbf{T}$.

As analyzed by Bartlett [33], networks tend to have better generalization performance with smaller weights, so we also select the special minimum norm least squares solution $\hat{\boldsymbol{\beta}} = V_1 C_{11}^{-1} U_1^{\mathrm{T}}\mathbf{T}$ as the output weights like ELM, where $V_1 C_{11}^{-1} U_1^{\mathrm{T}}$ is a transformational expression of the RCOD of $\mathbf{H}$.

Remark 2 As discussed above, the obtained solution has three important properties:

1. Minimum training error. $\hat{\boldsymbol{\beta}} = V_1 C_{11}^{-1} U_1^{\mathrm{T}}\mathbf{T}$ is one of the least squares solutions of a general linear system $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$, which means that the smallest training error can be achieved by this special solution:
$$\begin{aligned}\left\|\mathbf{H}\hat{\boldsymbol{\beta}} - \mathbf{T}\right\| &= \left\|U_1 C_{11} V_1^{\mathrm{T}} V_1 C_{11}^{-1} U_1^{\mathrm{T}}\mathbf{T} - \mathbf{T}\right\| \\ &= \min_{\beta}\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|.\end{aligned} \tag{23}$$
2. Smallest norm of weights. $\hat{\boldsymbol{\beta}} = V_1 C_{11}^{-1} U_1^{\mathrm{T}}\mathbf{T}$ is a special solution which has the smallest norm among all the least squares solutions $\boldsymbol{\beta} = V_1 C_{11}^{-1} U_1^{\mathrm{T}}\mathbf{T} + V_2 z$ of $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$:
$$\begin{aligned}&\left\|\hat{\boldsymbol{\beta}}\right\| = \left\|V_1 C_{11}^{-1} U_1^{\mathrm{T}}\mathbf{T}\right\| \le \|\boldsymbol{\beta}\| \\ &\forall\boldsymbol{\beta} \in \left\{\boldsymbol{\beta}|\boldsymbol{\beta} = V_1 C_{11}^{-1} U_1^{\mathrm{T}}\mathbf{T} + V_2 z, \forall z \in \mathbf{R}^{(n-r)\times m}\right\}.\end{aligned} \tag{24}$$
3. The minimum norm least squares solution $\hat{\boldsymbol{\beta}} = V_1 C_{11}^{-1} U_1^{\mathrm{T}}\mathbf{T}$ is unique.

Based on the above analysis, as a lightweight decomposition method, the RCOD can also be successfully used instead of SVD to compute the Moore–Penrose generalized inverse and obtain the minimum norm least squares solution, then an efficient extreme learning machine (EELM) algorithm incorporating RCOD is proposed for chaotic time series prediction, which can be summarized as the following five steps:

Algorithm EELM: EELM ($x(t)$, m, τ, n)
Input: A chaotic time series $\{x(t)|t = 1, 2, \ldots, N\}$, embedding dimension m, time delay τ and hidden nodes number n
Output: output weights β.

1. Reconstruct phase space with Eq. (1)
2. Build the input space and output space of prediction model with reconstructed phase points:
$$\aleph = \{(\boldsymbol{X}(t), x(t+h))|t = (m-1)\tau + 1, \ldots, N - h\}, \tag{25}$$
where $\boldsymbol{X}(t) \in \mathbf{R}^m, x(t+h) \in \mathrm{R}$ are the input and output respectively, and h is the prediction step
3. Randomly generate hidden nodes parameters $(\boldsymbol{a}_i, b_i), i = 1, 2, \ldots, n$.
4. Calculate the hidden layer output matrix $\mathbf{H}$ and target output $\mathbf{T}$ as Eqs. (7) and (8)
5. Calculate the output weights $\boldsymbol{\beta}$ using RCOD:
$$[U_1, C_{11}, V_1] = \mathrm{RCOD}(\mathbf{H}),$$
$$\boldsymbol{\beta} = V_1 C_{11}^{-1} U_1^{\mathrm{T}}\mathbf{T}. \tag{26}$$

Remark 3 Compared with the original ELM algorithm, the most prominent distinction of EELM is that an efficient and reliable RCOD approach [Eq. (26)] is used to calculate the output weights instead of SVD. By means of the excellent properties of the RCOD, the EELM can achieve a much faster learning speed while maintaining the same merits as ELM.

Remark 4 In the proposed EELM algorithm, phase space reconstruction is incorporated as data pre-processing to establish prediction space for chaotic time series prediction. In fact, the above EELM algorithm can also be applied in other general data sets besides time series by cutting out the first two steps.

3.4 Comparison of computation complexity

EELM and ELM share the same idea of extreme learning, that is the hidden nodes parameters are randomly generated and need not be tuned, and the main computational overhead of ELM (EELM) is the calculation of output weights. ELM calculates the output weights by computing Moore–Penrose generalized inverse based on SVD. For matrix $\mathbf{H} \in \mathbf{R}^{N\times n}$, the computation complexity of SVD is $4Nn^2 + 8n^3$ flops [32], so the computation complexity of ELM is approximately $4Nn^2 + 8n^3$ flops.

Instead of using SVD during the calculation of the output weights, RCOD is used in the proposed EELM algorithm, and the work for the two-step QR factorization dominates the work required for the RCOD. For matrix $\mathbf{H} \in \mathbf{R}^{N\times n}$ with estimated numerical rank r, the Householder QR factorization with column pivoting requires $2Nn^2 - 2n^3/3$ flops [32] in step 1. For the lower trapezoidal matrix $R2 \in \mathbf{R}^{n\times r}$ generated by step 1, the complexity of QR factorization without column pivoting in step 2 is $2nr^2 - 2r^3/3$ flops [32]. Therefore, the total computation complexity of the proposed EELM is approximately $2Nn^2 - 2n^3/3 + 2nr^2 - 2r^3/3 = 2Nn^2 + 2nr^2 - 2(n^3 + r^3)/3$.

The comparison of the computation complexity of EELM and ELM is written as follows:

$$\frac{O(EELM)}{O(ELM)} \approx \frac{O(RCOD)}{O(SVD)} = \frac{2Nn^2 + 2\mathrm{nr}^2 - 2(\mathrm{n}^3 + \mathrm{r}^3)/3}{4Nn^2 + 8n^3} = \frac{2 + 2\frac{r^2}{Nn} - \frac{2}{3}\left(\frac{n}{N} + \frac{r^3}{Nn^2}\right)}{4 + 8\frac{n}{N}} \tag{27}$$

Since $0 < r \le n \le N$, we are easy to get $\frac{r^2}{Nn} \le 1$ and $\frac{n}{N} + \frac{r^3}{Nn^2} > 0$, then the right hand of Eq. (27) can be further derived as

$$\frac{2 + 2\frac{r^2}{Nn} - \frac{2}{3}\left(\frac{n}{N} + \frac{r^3}{Nn^2}\right)}{4 + 8\frac{n}{N}} < \frac{4}{4 + 8\frac{n}{N}} < 1 \tag{28}$$

Based on the above equations, we can infer that the computation complexity of EELM is smaller than that of ELM.

4 Proposed integrated prediction model

In this section, we first briefly review the DE algorithm and then follow up with a description of the proposed integrated prediction model in detail.

4.1 Differential evolution

DE [34] is one of the most efficient and powerful evolutionary algorithms for global optimization. Given a set of parameter vectors $\{\boldsymbol{\theta}_{i,G} | i = 1, 2, \ldots, \mathrm{NP}\}$ as a population at each generation G, the general procedures of DE are as follows:

Mutation For each target vector $\boldsymbol{\theta}_{i,G+1}, i = 1, 2, \ldots, \mathrm{NP}$, a mutant vector is generated according to

$$\boldsymbol{v}_{i,G+1} = \boldsymbol{\theta}_{r_1,G} + F \cdot (\boldsymbol{\theta}_{r_2,G} - \boldsymbol{\theta}_{r_3,G}) \tag{29}$$

with random and mutually different index $r_1, r_2, r_3 \in \{1, 2, \ldots, \mathrm{NP}\}$ and F is a real and constant factor within $[0, 2]$ which is used to control the amplification of the differential variation $(\boldsymbol{\theta}_{r_2,G} - \boldsymbol{\theta}_{r_3,G})$.

Crossover To increase the diversity of the perturbed parameter vectors, the D-dimensional trial vector:

$$\boldsymbol{u}_{i,G+1} = (\boldsymbol{u}_{1i,G+1}, \boldsymbol{u}_{2i,G+1}, \ldots, \boldsymbol{u}_{Di,G+1}) \tag{30}$$

is formed, where

$$\boldsymbol{u}_{ji,G+1} = \begin{cases} \boldsymbol{v}_{ji,G+1} & \text{if} \quad \text{rand b(j)} \le \text{CR or j=rnbr(i)} \\ \boldsymbol{\theta}_{ji,G} & \text{if} \quad \text{rand b(j)} > \text{CR or j} \ne \text{rnbr(i)} \end{cases}, \quad j = 1, 2, \ldots, D. \tag{31}$$

In Eq. (31), *rand b*(*j*) is the *j*th evaluation of a uniform random number generator with outcome within $[0, 1]$. *CR* is the crossover rate within $[0, 1]$ which needs to be determined by user. *rnbr*(*i*) is a randomly chosen integer from $[1, D]$ which ensures that $\boldsymbol{u}_{i,G+1}$ obtains at least one parameter from $\boldsymbol{v}_{ji,G+1}$.

Selection If vector $\boldsymbol{u}_{i,G+1}$ yields a smaller cost function than $\boldsymbol{\theta}_{i,G}$, then $\boldsymbol{\theta}_{i,G+1}$ is set to $\boldsymbol{u}_{i,G+1}$. Otherwise, the old value $\boldsymbol{\theta}_{i,G}$ is retained.

4.2 Proposed integrated prediction model

As discussed in the introduction, traditional two-stage chaotic time series prediction models perform phase space reconstruction and model selection asynchronously, the consequence of which is that it is difficult to guarantee the obtained results are global optimum. Besides, the accuracy and stability of traditional prediction models are greatly limited by the quality of phase space reconstruction. In

order to overcome these limitations, we discard the traditional two-stage modeling approach and propose an integrated chaotic time series prediction model based on EELM and DE.

The core idea of the proposed model is to optimize the phase space reconstruction parameters and model parameters synchronously by adopting an integrated parameter selection strategy. To be more specific, we use the proposed EELM algorithm as an efficient modeling tool to model chaotic time series and employ a modified DE algorithm to search for the optimal parameter combination of embedding dimension, time delay of chaotic time series and the hidden nodes number of EELM simultaneously. Taking the advantages of the powerful global search ability of DE and the excellent properties of EELM, the proposed integrated prediction model not only guarantees that the selected parameter combination is global optimum or approximate optimum which enhances the prediction accuracy of the model, but also provides stable prediction performances with high efficiency.

For a given chaotic time series $x(t)$, the main execution process of the integrated prediction model can be summarized as below: First, we separate the time series $x(t)$ into training set $S1$, validation set $S2$ and testing set $S3$; then an integrated parameter selection process incorporating EELM and DE is executed on $S1$ and $S2$ to search for the optimal parameter vector $\boldsymbol{\theta} = (m, \tau, n)$ which achieves the minimum cost function (validation error) with parsimonious model structure; finally, the selected (m, τ) and n are used, respectively, to reconstruct the chaotic time series and to establish the model of EELM for prediction on $S3$.

A two-level hybrid architecture is designed to illustrate the proposed prediction model, which is shown in Fig. 1. From the level of DE, both the reconstruction parameters of time series as well as the model parameter of EELM are evolved together for integrated parameter selection, all the parameter vectors go through four evolutionary stages: initialization, cost function calculation, mutation and crossover, evaluation and selection, and the parameters used in the evolutionary process can be set according to the common criteria of DE algorithm [34]. From the level of EELM, as a "generalized" SLFN, the model structure of EELM can be determined as follows: The hidden nodes number is given directly by parameter n; the input nodes number is equal to the dimensions of reconstructed input phase point ($\boldsymbol{X}(t) \in \mathbf{R}^m$), that is, the embedding dimension m; the output nodes number represents the dimensions of reconstructed output phase point, since this paper is focus on single-step prediction, the output nodes number is set as 1.

The details of the four evolutionary stages will be described below.

Stage 1: *Initialization* A set of NP individual parameter vectors where each one is composed of (m, τ, n) is initialized as the populations by the following equation

$$\boldsymbol{\theta}_{i,G} = \boldsymbol{\theta}_{\min} + rand(0,1) \cdot (\boldsymbol{\theta}_{\max} - \boldsymbol{\theta}_{\min}), i = 1, \cdots, NP \tag{32}$$

where $\boldsymbol{\theta}_{\min} = (m_{\min}, \tau_{\min}, n_{\min})$ and $\boldsymbol{\theta}_{\max} = (m_{\max}, \tau_{\max}, n_{\max})$ are the prescribed minimum and maximum parameter bounds, respectively.

Stage 2: *Cost function calculation* In our study, the widely used validation methodology is adopted for integrated parameter selection, and all the parameter vectors are evaluated by cost function on validation set. For each population vector $\boldsymbol{\theta}_{i,G}$, which is first used as input parameter to train the EELM on training set $S1$, and the

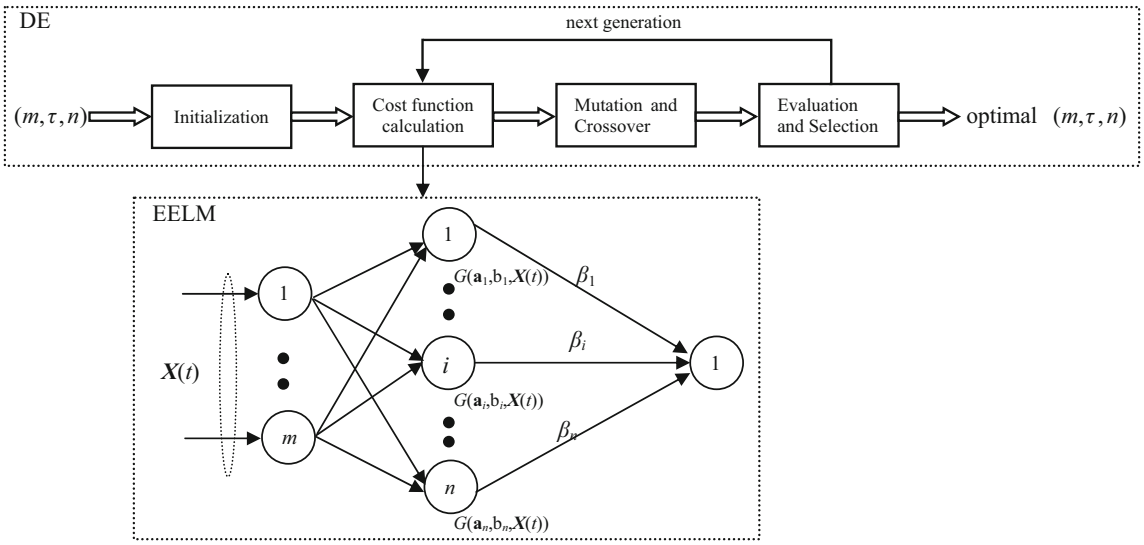

Fig. 1 Structure of the integrated chaotic time series prediction model

corresponding output weights are analytically calculated by calling EELM algorithm [Eq. (33)]; then, the obtained output weights are used to evaluate the fitness of $\boldsymbol{\theta}_{i,G}$ on validation set $S2$, and the corresponding cost function (suppose that the root-mean-squared error (RMSE) is used as cost function) is calculated with Eq. (34)

$$\boldsymbol{\beta} = \mathrm{EELM}(S1, \boldsymbol{\theta}i, G) \tag{33}$$

$$f(\boldsymbol{\theta}i, G) = \mathrm{RMSE}i, G = \sqrt{\frac{\sum_{j=1}^{N} \|\mathbf{H} * \boldsymbol{\beta} - \mathbf{T}\|_2^2}{N}} \tag{34}$$

where $\mathbf{H}$ and $\mathbf{T}$ are the hidden layer output matrix and target output on validation set $S2$ as shown in Eqs. (7) and (8), and N is the length of $S2$.

Stage 3: *Mutation and crossover* The mutation and crossover are performed as original DE with Eqs. (29, 30, 31) by setting $D = 3$. Since the parameter of individual may exceed the bounds of maximum or minimum after the mutation, a subsequent boundary checking and adjustment is added as follows:

$$\begin{aligned} &\boldsymbol{v}_{ji,G+1} = \boldsymbol{\theta}_{\max}(j) + rand(0,1) \cdot (\boldsymbol{\theta}_{rand}(j) - \boldsymbol{\theta}_{\max}(j)) \\ &\quad \text{if } \boldsymbol{v}_{ji,G+1} > \boldsymbol{\theta}_{\max}(j), \quad j = 1,2,3. \\ &\boldsymbol{v}_{ji,G+1} = \boldsymbol{\theta}_{\min}(j) + rand(0,1) \cdot (\boldsymbol{\theta}_{rand}(j) - \boldsymbol{\theta}_{\min}(j)) \\ &\quad \text{if } \boldsymbol{v}_{ji,G+1} < \boldsymbol{\theta}_{\min}(j), \quad j = 1,2,3. \end{aligned} \tag{35}$$

where $\boldsymbol{\theta}_{rand}(j)$ is a random parameter vector.

Stage 4: Evaluation and selection During selection stage in original DE algorithm, the trial vector $\boldsymbol{u}_{i,G+1}$ is evaluated by calculating cost function [Eqs. (33–34)]; then, the result $f(\boldsymbol{u}_{i,G+1})$ is compared with the original one $f(\boldsymbol{\theta}_{i,G})$, and the vector with smaller cost function value is selected into the next generation. However, for the model selection of ELM (EELM), using the cost function (validation RMSE) alone as the selection criterion is not appropriate. As demonstrated in [21], the generalization performance of ELM (EELM) is very stable, which means that we may obtain similar small validation error on a wide range of hidden nodes numbers. But unfavorably, more hidden nodes mean more training time and testing time; what is more, too many hidden nodes may lead to the model overfitting. Therefore, to obtain an exact and parsimonious architecture, one more criteria besides validation RMSE are added into the selection process: the number of hidden nodes n. In our new selection strategy, when the difference of the cost functions between $\boldsymbol{u}_{i,G+1}$ and $\boldsymbol{\theta}_{i,G}$ is less than some threshold, the one with fewer hidden nodes is selected. The determination of new population $\boldsymbol{\theta}_{i,G+1}$ can be described as follows:

$$\theta_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G+1} & \text{if } f(\theta_{i,G}) - f(\boldsymbol{u}_{i,G+1}) > \varepsilon f(\theta_{i,G}) \\ \boldsymbol{u}_{i,G+1} & \text{if } \left|f(\theta_{i,G}) - f(\boldsymbol{u}_{i,G+1})\right| < \varepsilon f(\theta_{i,G}) \\ & \text{and } \boldsymbol{u}_{3i,G+1} < \theta_{3i,G} \\ \theta_{i,G} & \text{else} \end{cases} \tag{36}$$

where ε is a preset tolerance rate. Steps 3 and 4 are repeated until the maximum iterations are accomplished, and the best individual with minimum validation RMSE in the last generation is selected as the final optimal parameter combination.

Remark 5 For a given chaotic time series, the integrated prediction model employs DE to search for the most excellent combination of reconstruction parameters and model parameters simultaneously and automatically, which is prone to obtaining good prediction performances with little human intervention. Meanwhile, by means of the extremely fast learning speed of EELM, the execution efficiency of this model is also very promising.

5 Numerical experiments

To evaluate the performance of the proposed model, it is tested on two benchmark chaotic systems, Mackey–Glass and Lorenz and one real lifetime series, Sunspot time series. The comparisons of experimental results are focus on two parts: First, the efficiency and performance of the integrated prediction model combining DE with EELM (DE-EELM) are compared with that of the model combining DE with ELM (DE-ELM); further, the prediction accuracies of the proposed model are compared with the results reported in the literatures.

5.1 Dataset and experimental set-up

The Mackey–Glass time series has been used as a benchmark problem in chaotic time series prediction due to its chaotic nature. The time series is generated by the following nonlinear differential equation:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{ax(t-\tau)}{1 + x^c(t-\tau)} - bx(t) \tag{37}$$

This time series is chaotic for $\tau > 16.8$. The parameters selected for generating the time series are $a = 0.2$, $b = 0.1$, $c = 10$, $\tau = 17$ according to the literatures [10, 12, 13]. A chaotic time series samples set with length of 10,000 is generated by the Runge–Kutta integration of Eq. (37) with initial value $x(0) = 1.2$. To reduce the transient effect, we omit the initial 8000 values and only keep the last 2000 values for experiment.

The Lorenz time series introduced by Edward Lorenz is a three-dimensional dynamical system that exhibits chaotic flow. The Lorenz equations are written as:

$$\begin{aligned}\frac{dx(t)}{dt} &= \sigma[y(t) - x(t)] \\ \frac{dy(t)}{dt} &= x(t)[r - z(t)] - y(t) \\ \frac{dz(t)}{dt} &= x(t)y(t) - bz(t)\end{aligned} \tag{38}$$

where σ, r, b are the dimensionless parameters, and the typical values for these parameters are $\sigma = 10$, $r = 28$ and $b = 8/3$ [10, 12, 13]. The x-coordinate of the Lorenz time series is considered for prediction, and a time series with a length of 10,000 is generated by the Runge–Kutta integration of Eq. (38) with sampling interval h = 0.02 and initial state $\{x0, y0, z0\} = \{8, 5, 10\}$. To reduce the transient effect, we omit the initial 8000 values and only keep the last 2000 values for experiment.

Forecasting Sunspot time series is a critical topic since the Sunspot time series is a good indication of the solar activities for solar cycles. However, the prediction of solar cycles is very difficult and extremely challenging due to its complexity. The monthly smoothed Sunspot time series has been obtained from the SIDC (World Data Center for the Sunspot Index) [35]. To compare the results with other works published in the literatures, the Sunspot time series from November 1834 to June 2001 (2000 points) is selected as [10, 12, 13].

In every chaotic time series, the data are divided into three parts: The first 1000 samples are used for training, the next 500 samples are used to validate the trained model for integrated parameter selection, and the remaining 500 samples are used for prediction. All the data are scaled between [0, 1].

The default parameters used in the integrated model (DE-EELM and DE-ELM) are listed in Table 1, in which, the generations, population size, mutant factor, crossover rate are preset according to the common criteria of DE algorithm [34]; the ranges of embedding dimension and time delay are selected by the general experiences; the range of hidden nodes numbers is broadly set from one to the number of training samples according to Theorem 2.2 in [21]. In real applications, if we have some priori knowledge of the data sets, the ranges of embedding dimension, time delay and hidden nodes numbers can be preset more parsimoniously, which will be helpful to improve the performances of the prediction model both on training time and prediction accuracy. To show the universality of the proposed model, we use the same parameters for all the experiments without any adjustments.

All the simulations are carried out in MATLAB R2010b environment running on an ordinary PC with 3.4 GHZ CPU and 4 GB RAM. In order to evaluate the prediction performance and compare it with the results reported in the literatures, root-mean-squared error (RMSE), normalized mean squared error (NMSE) and absolute error (Error) are calculated according to Eqs. (39), (40) and (41), respectively.

Table 1 The parameters of DE-EELM (DE-ELM)

Parameter	Value (interval)
Generations (G)	50
Population size (NP)	20
Mutant factor (F)	0.6
Crossover rate (CR)	0.4
Tolerance rate (ε)	0.02
Embedding dimension (m)	[1, 30]
Time delay (τ)	[1, 5]
Hidden nodes number (n)	[1, 1000]

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{39}$$

$$\text{NMSE} = \left(\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}\right) \tag{40}$$

$$\text{Error} = y_i - \hat{y}_i \tag{41}$$

where y_i, $\hat{y}_i$ and $\bar{y}$ are the observed data, predicted data and average of observed data, respectively. N is the number of the observed data.

5.2 Results and discussion

According to the strategy of integrated parameter selection, the integrated model is run first to search for the optimal parameter combination of (m, τ, n) which achieves the minimum validation RMSE with fewer hidden nodes. In order to validate the effectiveness and practicability of the proposed model, 30 independent experimental runs have been conducted for DE-ELM and DE-EELM on the three chaotic time series, respectively, and the 30 selected optimal parameter vectors (m, τ, n) as well as their corresponding validation RMSEs, search time are obtained, respectively. The minimum [and the corresponding (m, τ, n)], maximum[and the corresponding (m, τ, n)], mean, standard deviation (Stdev) of the validation RMSEs and the average search time (CPU time) among 30 experiments for each case are shown in Table 2, and the better results are highlighted in bold.

Comparing the results of DE-EELM with that of DE-ELM, it is easy to see that the DE-EELM can obtain the similar or better average validation RMSE with much less search time, which confirms that the proposed EELM is

Table 2 The validation RMSE and average CPU time for integrated parameter selection

Time series	Model	Validation RMSE				Time (s)
		Minimum (m, τ, n)	Maximum (m, τ, n)	Mean	SD	
Mackey	DE-ELM	1.15E−06 (17, 1, 686)	1.49E−06 (18, 1, 634)	1.33E−06	8.12E−08	1350
	DE-EELM	1.09E−06 (17, 1, 665)	1.43E−06 (17, 1, 670)	1.30E−06	7.67E−08	**737**
Lorenz	DE-ELM	6.97E−08 (9, 1, 120)	1.10E−07 (8, 1, 108)	8.82E−08	7.92E−09	364
	DE-EELM	6.19E−08 (8, 1, 127)	7.87E−08 (8, 1, 132)	**7.00E−08**	3.62E−09	**186**
Sunspot	DE-ELM	5.08E−03 (14, 1, 43)	5.52E−03 (13, 1, 33)	5.29E−03	1.08E−04	126
	DE-EELM	4.95E−03 (14, 1, 40)	5.52E−03 (15, 1, 32)	5.26E−03	1.27E−04	**79**

effective to improve the learning speed and maintain good learning ability compared with ELM. In addition, from the minimum, maximum, mean and standard deviation of the validation RMSEs, we can see that the range of the obtained validation errors among 30 independent experiments is very compact with slight fluctuation, which implies that the proposed integrated model combining DE and EELM supplies a quite stable way for integrated parameter selection.

Besides, to describe the convergence of DE-EELM, one more independent experimental run has been conducted on the three chaotic time series, respectively. The minimum validation RMSE of each generation is collected, and the convergence graphics of each case are presented in Figs. 2, 3 and 4. As shown, the proposed DE-EELM model provides good convergence performances with fast convergence speed. In order to further verify the global convergence of DE-EELM, the finally obtained minimum validation RMSE in the last generation should be compared with the global minimum. It is well known that brute-force approach is a time-consuming but most effective way to get the global optimum solution, so the brute-force search method combining EELM (BF-EELM for short) is used to search through all the continuous space of m, τ, n by hierarchical iterations, which is most likely to achieve the global minimum. The finally obtained results of DE-EELM and BF-EELM on validation sets are shown in Table 3. As can be seen from Table 3, the validation RMSE of DE-EELM is comparative with that of BF-EELM considering the random error. Combining the results of 30 independent experiments as shown in Table 2, we can conclude that the proposed DE-EELM model can always provide consistent convergence to the global minimum in consecutive independent trials.

Further, all these 30 selected optimal parameter vectors are applied as construction parameters and model parameter on the testing sets for prediction, respectively. Average results (RMSE, NMSE and CPU time) of 30 trials for each parameter vector are obtained, and finally, the minimum, maximum, mean and standard deviation (Stdev) of prediction RMSEs and prediction NMSEs, as well as the average prediction time, are reported in Table 4.

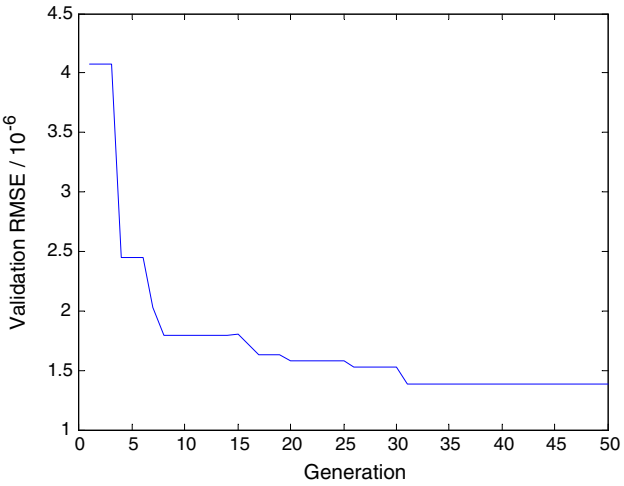

Fig. 2 Convergence graphics of the Mackey–Glass

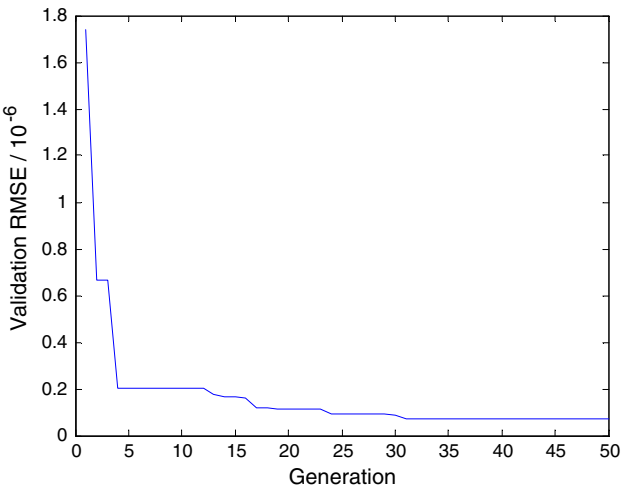

Fig. 3 Convergence graphics of the Lorenz

As shown in Table 4, similar to the validation RMSE, the ranges of the prediction RMSEs and the prediction NMSEs of our proposed DE-EELM are also very compact, which means that our proposed integrated model is quite stable in both parameter selection stage and prediction

stage. Moreover, from the comparisons between DE-ELM and DE-EELM, the prediction results of DE-ELM and DE-EELM are almost the same on Mackey–Glass and Sunspot; while on Lorenz, the DE-EELM achieves a much better and more stable results on both prediction RMSE and prediction NMSE. This indicates that the DE-EELM can provide comparable or even better prediction performances in comparison with the DE-ELM for all the three time series. In addition, the prediction time is mainly depended on the number of hidden nodes, and the number of hidden nodes used in DE-ELM and DE-EELM is about the same, so we can see that the prediction time of the DE-ELM and the DE-EELM is approximately equivalent and extremely short.

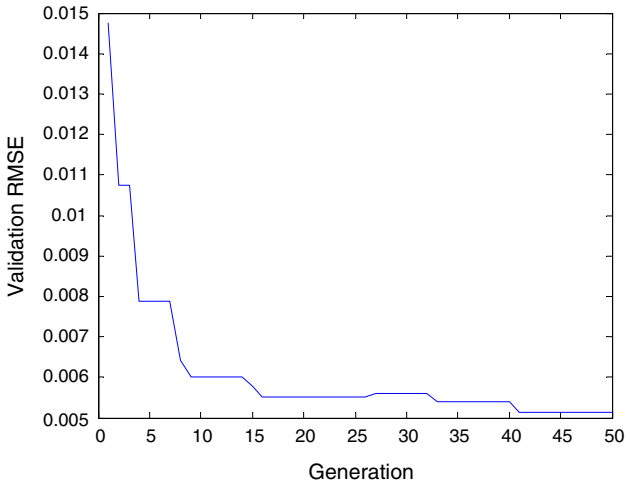

Fig. 4 Convergence graphics of the Sunspot

Table 3 The search results of DE-EELM and BF-EELM on validation sets

Time series	Model	(m, τ, n)	Validation RMSE	Time (s)
Mackey	BF-EELM	(17,1,682)	1.27E−06	88,141
	DE-EELM	(18,1,694)	1.38E−06	802
Lorenz	BF-EELM	(8,1,126)	6.79E−08	84,920
	DE-EELM	(9,1,122)	6.97E−08	180
Sunspot	BF-EELM	(14,1,41)	5.00E−03	80,852
	DE-EELM	(14,1,38)	5.14E−03	73

Figures 5, 6, 7, 8, 9 and 10 demonstrate that the DE-EELM has been able to provide good prediction performances on Mackey–Glass, Lorenz and Sunspot. Figures 5, 7, 9 compare the original values with the forecast values of the three problems, respectively, which illustrate that the forecast values are almost entirely consistent with the original values. To go along with this, the corresponding prediction errors (absolute errors) are given in Figs. 6, 8, 10, respectively. The results show that the absolute errors of Mackey–Glass, Lorenz and Sunspot can reach the level of 10^{-5}, 10^{-7} and 10^{-2}, respectively. Notice that the Sunspot time series is a real-world problem which contains noise, it follows that the proposed DE-EELM is effective for both noise-free and noisy chaotic time series.

To validate the superiority of the proposed model, the prediction accuracies of DE-EELM as well as DE-ELM on the three time series are further compared with some of the latest and the best results published in the literatures, which are shown in Tables 5, 6 and 7. For comparison, the best results in the previous literatures are highlighted in bold, and the whole intervals between the best value (minimum error) and the worst value (maximum error) of our results as shown in Table 4 are used to compare with the best results from literatures directly.

Table 5 presents the comparison of RMSE and NMSE reported in the literatures with that of the proposed model on Mackey–Glass. It is easy to see that the proposed DE-ELM and DE-EELM both perform better in prediction of Mackey–Glass chaotic time series when compared with other models reported in the literatures.

In Table 6, the proposed DE-ELM and DE-EELM achieve smaller prediction error on both RMSE and NMSE than other reported models for forecasting Lorenz chaotic time series. Although the NMSE of DE-ELM fluctuates within a relatively large scale, the worst prediction NMSE

Table 4 The prediction performances of all the selected optimal parameters on testing sets

Time series	Model	Prediction RMSE				Prediction NMSE				Time(s)
		Minimum	Maximum	Mean	SD	Minimum	Maximum	Mean	SD	
Mackey	DE-ELM	2.29E−06	2.68E−06	2.40E−06	8.94E−08	9.00E−11	1.22E−10	9.87E−11	7.42E−12	3.40E−02
	DE-EELM	2.30E−06	2.66E−06	2.46E−06	1.05E−07	9.07E−11	1.24E−10	1.05E−10	1.01E−11	3.59E−02
Lorenz	DE-ELM	1.14E−07	5.65E−07	1.91E−07	1.03E−07	3.70E−13	1.28E−11	1.58E−12	2.57E−12	6.35E−03
	DE-EELM	**6.58E−08**	**9.61E−08**	**7.67E−08**	**6.01E−09**	**1.10E−13**	**2.41E−13**	**1.49E−13**	**2.55E−14**	6.42E−03
Sunspot	DE-ELM	5.38E−03	7.81E−03	5.76E−03	4.38E−04	5.26E−04	1.14E−03	6.08E−04	1.10E−04	3.38E−03
	DE-EELM	5.29E−03	7.23E−03	5.67E−03	3.51E−04	5.08E−04	9.68E−04	5.88E−04	8.24E−05	3.54E−03

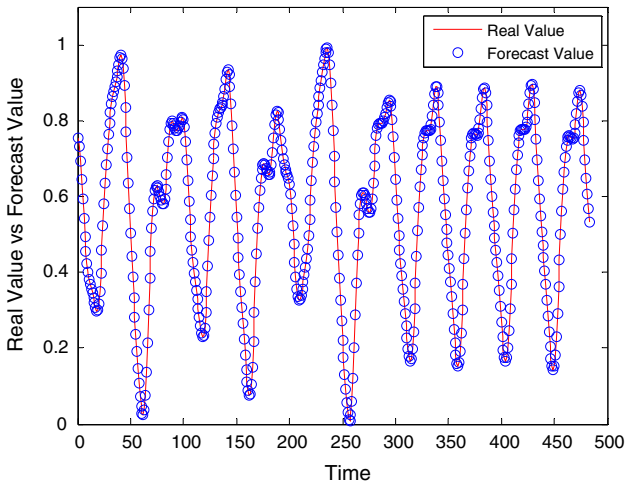

Fig. 5 Typical prediction results of the Mackey–Glass

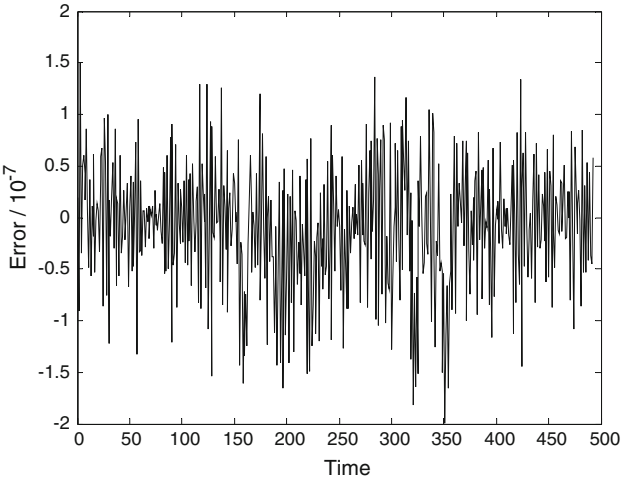

Fig. 8 Prediction errors of the Lorenz

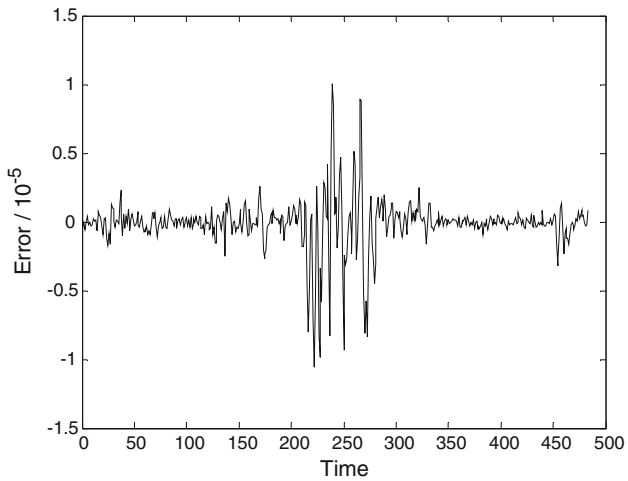

Fig. 6 Prediction errors of the Mackey–Glass

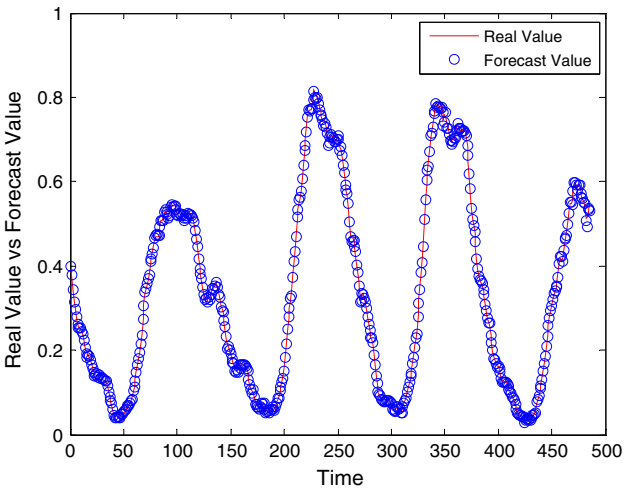

Fig. 9 Typical prediction results of the Sunspot

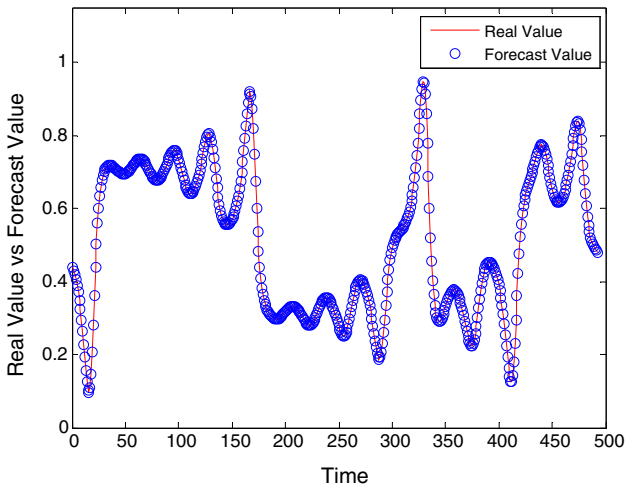

Fig. 7 Typical prediction results of the Lorenz

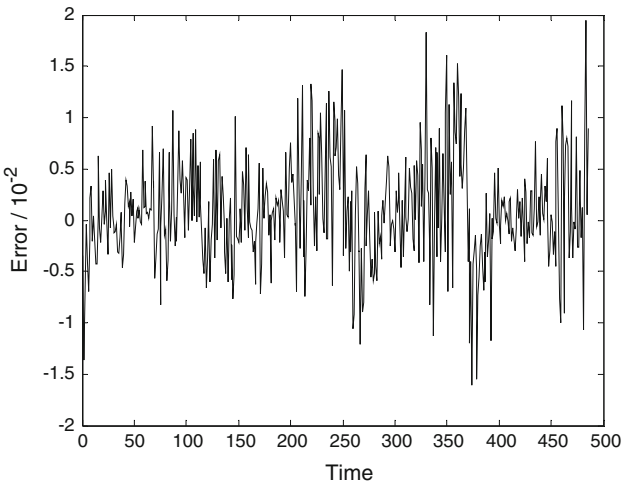

Fig. 10 Prediction errors of the Sunspot

Table 5 A comparison with the results from literatures on Mackey–Glass

Prediction model	RMSE	NMSE
CCPSO (2009) [9]	8.42E−03	
SL-CCRNN (2012) [12]	6.33E−03	2.79E−04
ARMA-ANN (2008) [36]	2.50E−03	
BPNN-GA-RA (2011) [37]	1.30E−03	
RBF-OLS (2006) [6]	1.02E−03	
LLNF-LoLiMot (2006) [6]	9.61E−04	
ERNN (2007) [8]	4.20E−05	3.15E−08
HENNN-RA (2010) [10]	3.72E−05	2.70E−08
Taguchi's DoE (2013) [13]	**2.87E−05**	**1.59E−08**
Proposed DE-ELM	[2.29, 2.68]E−06	[0.90, 1.22]E−10
Proposed DE-EELM	[2.30, 2.66]E−06	[0.91, 1.24]E−10

Table 6 A comparison with the results from literatures on Lorenz

Prediction model	RMSE	NMSE
ARMA-ANN (2008) [36]	8.76E−02	
BPNN-GA-RA (2011) [37]	2.96E−02	
SL-CCRNN (2012) [12]	6.36E−03	7.72E−04
RBF-OLS (2006) [6]		1.41E−09
ERNN (2007) [8]	**8.79E−06**	9.90E−10
LLNF-LoLiMot (2006) [6]		9.80E−10
BELRFS (2013) [38]		4.85E−10
HENNN-RA (2010) [10]	1.08E−04	1.98E−10
Taguchi's DoE (2013) [13]	4.41E−05	**3.29E−11**
Proposed DE-ELM	[1.14, 5.65]E−07	[0.04, 1.28]E−11
Proposed DE-EELM	[6.58, 9.61]E−08	[1.10, 2.41]E−13

is still better than the best one reported in the literatures. Relatively, our proposed DE-EELM achieves much more accurate prediction results with high stability.

Table 7 reports the comparison of prediction errors on Sunspot. The results show that the proposed DE-ELM and DE-EELM achieve best RMSE, while the best NMSE is achieved by Ref. [13], which is 5.04E−04. Although the whole NMSE interval ([5.08, 9.68]E−04) of our proposed DE-EELM is slightly larger than 5.04E−04, in fact, among our 30 independent experiments, the prediction NMSEs of 29 tests are within [5.08, 6.65]E−04 except one test getting 9.68E−04, that is to say, the prediction NMSEs obtained by our proposed DE-ELM are approximate to the best one reported in the literatures in most cases. Taken RMSE and NMSE together, we have reasons to say that the proposed DE-EELM can always provide better or comparable prediction accuracy in comparison with the existing most excellent models for predicting Sunspot chaotic time series.

Table 7 A comparison with the results from literatures on Sunspot

Prediction model	RMSE	NMSE
RBF-OLS (2006) [6]		4.60E−02
LLNF-LoLiMot (2006) [6]		3.20E−02
ERNN (2007) [8]	1.29E−02	2.80E−03
SL-CCRNN (2012) [12]	1.66E−02	1.47E−03
HENNN-RA (2010) [10]	1.19E−02	5.90E−04
Taguchi's DoE (2013) [13]	**1.10E−02**	**5.04E−04**
Proposed DE-ELM	[5.38, 7.81]E−03	[0.53, 1.14]E−03
Proposed DE-EELM	[5.29, 7.23]E−03	[5.08, 9.68]E−04

6 Conclusions and future work

This paper presents an integrated model for chaotic time series prediction. In the proposed model, an efficient and reliable EELM algorithm is proposed to model and forecast the chaotic time series. In addition, distinguished from the traditional models, the proposed model performs the phase space reconstruction and model selection synchronously, and a modified DE algorithm is employed to search for the optimal parameter combination which achieves the minimum validation error with parsimonious model structure. The effectiveness and superiority of the proposed model are confirmed through three benchmark examples.

Some characteristics of the proposed model can be concluded as follows:

1. *High accuracy* Compared with the existing models, the proposed model can achieve better prediction accuracy in both synthetic and real-world chaotic time series.
2. *High stability* Although using a random search algorithm (DE) for integrated parameter selection, experimental results show that our proposed model can always obtain a compact error range among multiple trials on both validation error and prediction error.
3. *High efficiency* The high efficiency of the embedded EELM algorithm guarantees that the proposed model has faster learning speed and higher prediction efficiency than homogeneous models.
4. *High automation* The proposed model can choose the reconstruction parameters and the model parameter synchronously and automatically by using a modified DE algorithm, and the preset parameters of DE can be simply determined based on common criteria and remain fixed for all applications, and little human intervention is needed during the whole training and prediction processes.

Above conclusions indicate that the proposed integrated prediction model has very promising application prospects in chaotic time series prediction.

Although the proposed model has shown satisfactory results, it still can most probably be improved. For example, the execution efficiency (search time) of the proposed model is greatly affected by the search range of the hidden nodes numbers, while in our study, the upper bound of the hidden node number is broadly set as the number of training samples, when the number of training samples is very large, the model needs to search in a broad range, and the required search time will increase quickly, so for large-scale applications, how to obtain usable results in a reasonable amount of time remains open. To solve this problem, heuristic approaches such as constructive methods (or growing methods) and destructive methods (or pruning methods) may be promising ways to determine a more compact search range of the hidden nodes numbers, which will be helpful to shorten the search time and improve the availability of the model further, especially for large-scale applications.

In addition, as future work, we will consider the application of the proposed model in a practical airport noise prediction problem. Moreover, this paper is mainly focused on single-step prediction (or short-term prediction), and transferring the proposed model to the multi-step prediction (or long-term prediction) situations is another research topic in the future.

Acknowledgments This work is supported by the Key Program of the National Natural Science Foundation of China (Grant No. 61139002), the National High Technology Research and Development Program of China (Grant No. 2012AA063301), the National Key Technology Research and Development Program of the Ministry of Science and Technology of China (Grant No.2014BAJ04B02), the Fundamental Research Funds for the Central Universities of Ministry of Education of China (Grant Nos. 3122014D032, 3122013P013), the Open Project Foundation of Information Technology Research Base of Civil Aviation Administration of China (Grant No. CAAC-ITRB-201401). All of these supports are appreciated.

References

1. Das A, Das P (2007) Chaotic analysis of the foreign exchange rates. Appl Math Comput 185(1):388–396
2. Bao Y, Wang H, Wang BN (2013) Short-term wind power prediction using differential EMD and relevance vector machine. Neural Comput Appl 23(3):1–7
3. Abdi J, Moshiri B, Abdulhai B, Sedigh AK (2013) Short-term traffic flow forecasting: parametric and nonparametric approaches via emotional temporal difference learning. Neural Comput Appl 23(1):141–159
4. Sivapragasam C, Vanitha S, Muttil N, Suganya K, Suji S, Selvi MT, Selvi R, Sudha SJ (2014) Monthly flow forecast for Mississippi River basin using artificial neural networks. Neural Comput Appl 24(7):1785–1793
5. Yang XH, Mei Y, She DX, Li JQ (2011) Chaotic Bayesian optimal prediction method and its application in hydrological time series. Comput Math Appl 61(8):1975–1978
6. Gholipour A, Araabi BN, Lucas C (2006) Predicting chaotic time series using neural and neurofuzzy models: a comparative study. Neural Process Lett 24(3):217–239
7. De Gooijer JG, Hyndman RJ (2006) 25 years of time series forecasting. Int J Forecast 22(3):443–473
8. Ma QL, Zheng QL, Peng H, Zhong TW, Xu LQ (2007) Chaotic time series prediction based on evolving recurrent neural networks. In: 2007 International conference on machine learning and cybernetics, Hong Kong, 2007. IEEE, pp 3496–3500
9. Lin CJ, Chen CH, Lin CT (2009) A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications. IEEE Trans Syst Man Cybern Part C Appl Rev 39(1):55–68
10. Ardalani-Farsa M, Zolfaghari S (2010) Chaotic time series prediction with residual analysis method using hybrid Elman-NARX neural networks. Neurocomputing 73(13):2540–2553
11. Castro JR, Castillo O, Melin P, Mendoza O, Rodríguez-Díaz A (2011) An interval type-2 fuzzy neural network for chaotic time series prediction with cross-validation and Akaike test. In: Castillo O, Kacprzyk J, Pedrycz W (eds) Soft computing for intelligent control and mobile robotics. Berlin Heidelberg, Springer, pp 269–285
12. Chandra R, Zhang MJ (2012) Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction. Neurocomputing 86(1):116–123
13. Ardalani-Farsa M, Zolfaghari S (2013) Taguchi's design of experiment in combination selection for a chaotic time series forecasting method using ensemble artificial neural networks. Cybern Syst 44(4):351–377
14. Chen DY, Han WT (2013) Prediction of multivariate chaotic time series via radial basis function neural network. Complexity 18(4):55–66
15. Marzban F, Ayanzadeh R, Marzban P (2014) Discrete time dynamic neural networks for predicting chaotic time series. J Artif Intell 7(1):24–34
16. Abiyev RH (2011) Fuzzy wavelet neural network based on fuzzy clustering and gradient techniques for time series prediction. Neural Comput Appl 20(2):249–259
17. Miranian A, Abdollahzade M (2013) Developing a local least-squares support vector machines-based neuro-fuzzy model for nonlinear and chaotic time series prediction. IEEE Trans Neural Netw Learn Syst 24(2):207–218
18. Wu Q (2010) The hybrid forecasting model based on chaotic mapping, genetic algorithm and support vector machine. Expert Syst Appl 37(2):1776–1783
19. Wang B, Huang H, Wang X (2013) A support vector machine based MSM model for financial short-term volatility forecasting. Neural Comput Appl 22(1):21–28
20. Donate JP, Li X, Sánchez GG, de Miguel AS (2013) Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm. Neural Comput Appl 22(1):11–20
21. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1):489–501
22. Sun ZL, Choi TM, Au KF, Yu Y (2008) Sales forecasting using extreme learning machine with applications in fashion retailing. Decis Support Syst 46(1):411–419
23. Bhat AU, Merchant SS, Bhagwat SS (2008) Prediction of melting points of organic compounds using extreme learning machines. Ind Eng Chem Res 47(3):920–925
24. Hough PD, Vavasis SA (1997) Complete orthogonal decomposition for weighted least squares. SIAM J Matrix Anal Appl 18(2):369–392
25. Zhang WZ, Long W, Jiao JJ (2012) Parameters determination based on composite evolutionary algorithm for reconstructing phase-space in chaos time series. Acta Phys Sin 61(22):1–7
26. Takens F (1981) Detecting strange attractors in turbulence. In: Rand D, Young L-S (eds) Dynamical systems and turbulence, Warwick 1980. Berlin Heidelberg, Springer, pp 366–381

27. Albano AM, Muench J, Schwartz C, Mees A, Rapp P (1988) Singular-value decomposition and the Grassberger–Procaccia algorithm. Phys Rev A 38(6):3017–3026
28. Fraser AM (1989) Information and entropy in strange attractors. IEEE Trans Inf Theory 35(2):245–262
29. Kugiumtzis D (1996) State space reconstruction parameters in the analysis of chaotic time series—the role of the time window length. Physica D 95(1):13–28
30. Cao L (1997) Practical method for determining the minimum embedding dimension of a scalar time series. Physica D 110(1):43–50
31. Kim HS, Eykholt R, Salas J (1999) Nonlinear dynamics, delay times, and embedding windows. Physica D 127(1):48–60
32. Golub GH, Van Loan CF (2012) Matrix computations, vol 3. JHU Press, Baltimore
33. Bartlett PL (1998) The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Trans Inf Theory 44(2):525–536
34. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359
35. SIDC (World Data Center for the Sunspot Index) (2014). http://sidc.oma.be/
36. Rojas I, Valenzuela O, Rojas F, Guillén A, Herrera LJ, Pomares H, Marquez L, Pasadas M (2008) Soft-computing techniques and ARMA model for time series prediction. Neurocomputing 71(4):519–537
37. Ardalani-Farsa M, Zolfaghari S (2011) Residual analysis and combination of embedding theorem and artificial intelligence in chaotic time series forecasting. Appl Artif Intell 25(1):45–73
38. Parsapoor M, Bilstrup U (2013) Chaotic time series prediction using brain emotional learning-based recurrent fuzzy system (BELRFS). Int J Reason Based Intell Syst 5(2):113–126